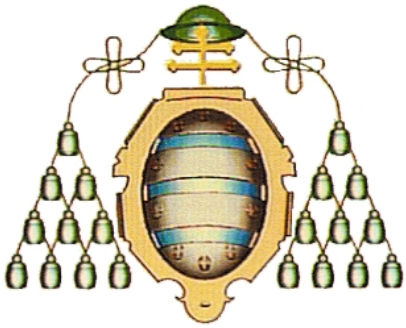
UNIVERSIDAD DE OVIEDO

DEPARTAMENTO DE INFORMÁTICA



TESIS DOCTORAL

GADEA: Sistema de Gestión de Interfaces de Usuario Auto-adaptables basado en Componentes, Tecnología de Objetos y Agentes Analizadores de Patrones de Comportamiento

Presentada por

D. Bernardo-Martín González Rodríguez

Para la obtención del título de Doctor por la Universidad de Oviedo

Dirigida por el

Catedrático Doctor D. Juan Manuel Cueva Lovelle

Oviedo, Julio de 2001

G A D E A

Sistema de Gestión de Interfaces de Usuario Auto-Adaptables basado en Componentes, Tecnología de Objetos y Agentes Analizadores de Patrones de Comportamiento

Desarrollado por
Martín González Rodríguez

Dirigido por
Dr. Juan Manuel Cueva Lovelle

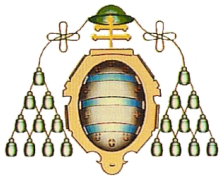
CINEMEDIA ASTUR ©

INXENA©

ANTS©

TIRSUS©

T
T I T
T I C I T
T I C E C I T
T I C E F E C I T
T I C E F N F E C I T
T I C E F N Í N F E C I T
T I C E F N Í T Í N F E C I T
T I C E F N Í T R T Í N F E C I T
T I C E F N Í T R A R T Í N F E C I T
T I C E F N Í T R A M A R T Í N F E C I T
T I C E F N Í T R A R T Í N F E C I T
T I C E F N Í T R T Í N F E C I T
T I C E F N Í T Í N F E C I T
T I C E F N Í N F E C I T
T I C E F N F E C I T
T I C E F E C I T
T I C E C I T
T I C I T
T I T
T



UNIVERSIDAD DE OVIEDO

DEPARTAMENTO DE INFORMÁTICA

Dr. D. JUAN MANUEL CUEVA LOVELLE, Catedrático de Escuela Universitaria del Departamento de Informática de la Universidad de Oviedo.

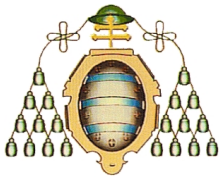
AUTORIZA:

A Don Bernardo-Martín González Rodríguez para que presente la tesis doctoral titulada *GADEA: Sistema de Gestión de Interfaces de Usuario Auto-adaptables basado en Componentes, Tecnología de Objetos y Agentes Analizadores de Patrones de Comportamiento.*

Lo que manifiesta en su calidad de director de la misma, en cumplimiento de las normas vigentes en esta Universidad para la presentación de tesis doctorales.

Oviedo, 3 de mayo de 2001

Fdo: D. Juan Manuel Cueva Lovelle



UNIVERSIDAD DE OVIEDO

DEPARTAMENTO DE INFORMÁTICA

Dr. D. JOSÉ ANTONIO LÓPEZ BRUGOS, Director del Departamento de Informática de la Universidad de Oviedo, en conformidad con las normas vigentes en esta Universidad para la presentación de tesis doctorales.

AUTORIZA:

A Don Bernardo-Martín González Rodríguez para que presente la tesis doctoral titulada *GADEA: Sistema de Gestión de Interfaces de Usuario Auto-adaptables basado en Componentes, Tecnología de Objetos y Agentes Analizadores de Patrones de Comportamiento* para la obtención del grado de Doctor.

Oviedo, 3 de mayo de 2001

Fdo: José Antonio López Brugos

A la mi güela Josefa
quien entamó a emplegar el nome sacro
abondo enantes de que fora un suaño

Redde Caesari Quae Sunt Caesaris

La exclusividad del mérito de cualquier trabajo de investigación, por modesto que éste sea, no pertenece en modo alguno a su autor, dado que por un lado, éste desarrolla su actividad investigadora gracias al soporte proporcionado por las personas que le rodean, y por el otro lado, la propia naturaleza del proceso de investigación, implica sustentar cualquiera de sus hipótesis en un conjunto de hechos demostrados previamente por otros autores. En este sentido, cada autor sigue la estela de autores previos, retomando y combinando ideas antiguas para que, a través de distintos enfoques, se puedan alcanzar ideas novedosas.

Partiendo de esta premisa, quisiera adjudicar el mérito de esta investigación a quien verdaderamente lo merece. En primer lugar, gran parte de este mérito corresponde a todos aquellos autores cuyos trabajos han formado parte de la fuente de información de esta tesis y que han servido como punto de partida para esta investigación, en especial a aquellos autores cuyo trabajo ha servido para establecer los pilares de la ingeniería de factores humanos, para desarrollar la psicología cognitiva a un elevado nivel actual y para elevar el estudio de la interacción y comunicación entre humanos al grado de disciplina.

En segundo lugar, quisiera adjudicar parte del mérito de esta investigación a todas aquellas personas que han aportado su granito de arena para permitir alcanzar los objetivos trazados al principio de la misma. A **Jesús Arturo Pérez Díaz,** por marcarme el camino a seguir con su profesionalidad y con sus contribuciones siempre novedosas, las cuales me han servido para crear una nueva generación de agentes. A **Agueda Vidau Navarro**, por los novedosos enfoques aportados a las tecnologías Tirsus y ANTS, especialmente por su contribución a la metáfora de las *migas*. A **Nuno Correia** y a **Teresa Chambel**, por darme la oportunidad de someter los primeros prototipos de esta tecnología, a la crítica y revisión de los expertos en multimedia. Al equipo multidisciplinar de ingeniería web liderado por **San Murugesan** y **Yogesh Despande**, permitirme introducir a GADEA al mundo del comercio electrónico. A **Juan De Lara**, cuya aportación estableció por fin la frontera entre la sintaxis y la semántica. A **Diego Díez Redondo**, a **Alfonso Alonso Blanco**, a **Pablo Manrubia Díez**, a **Roberto Aparicio Ferreras**, a **Roberto Alvarez Blanco**, a **Patricia Cordero Noval**, **a José Manuel Redondo López**, a **Antonio Sánchez Padial**, a **Ana García Fernández**, a **Santiago García Castro**, a **Tomás Martín Santos** y demás miembros del *Marteam*, sin cuya profesionalidad y saber hacer hubiese sido del todo imposible plasmar las ideas que contiene esta memoria en realidades concretas. A todos los miembros del grupo de investigación en interacción y comunicación humana de la Universidad de Oviedo, en especial a **María del Puerto Paule Ruíz** y a **Juan Ramón Pérez Pérez**, por sus diversas aportaciones al programa Tirsus. A **Juan Manuel Cueva Lovelle**, mi director de tesis, cuyos acertados consejos me han permitido organizar esta memoria de una forma coherente, así como lograr la consecución de todos los objetivos planteados por esta

tesis. A los **cientos de personas** que participaron en las pruebas de GADEA, CineMedia Astur, Inxena, Tirsus y ANTS.

En último lugar, quisiera asignar la cuota de mérito que les corresponde a todas aquellas personas que, de un modo u otro, me han proporcionado el apoyo logístico necesario para llevar a buen término esta larga investigación. A mi mujer, **Agueda Vidau Navarro,** quien con su infinita paciencia, comprensión y altas dosis de cariño, supo mantener mi moral en alto aún en los peores momentos. A mi abuela, **Josefa Sánchez Turanzas**, quien –no se por qué– tenía más confianza en mis posibilidades que yo mismo. A mis padres**, Estrella Rodríguez Sánchez** y **Juan Bernardo González Pelíz**. quienes me dieron todo el apoyo del mundo y me alentaron siempre a continuar. A mi hermano, **Juan Miguel González Rodríguez**, por ayudar a que mis huesos no se anquilosaran a lo largo de todos estos largos meses por medio de su bendito béisbol. A **Jesús Arturo Pérez Díaz** de nuevo, quien más que darme ánimos para continuar con la investigación, prácticamente me obligó a ello. A **María Nery Correoso González**, bibliotecaria quien siempre ha hecho lo posible y lo imposible por conseguirme todos los fondos bibliográficos que he requerido de la biblioteca de la univeridad a lo largo de estos años. A **María Jesús González Sanz** y a **María Rosario Armengol González** por su esmero y cuidado en la tramitación de toda la documentación generada por esta tesis. Al doctor **Amadeo Oria**, por poner mi cuerpo a punto cada vez que me he encargado de estropearlo. A **Raúl Vázquez Berroeta**, por ayudarme con la logística de mi viaje a Limerick con motivo del *ICSE'2000*, así como por aquellos ratos tan agradables en Dublín. A **Carmen Díaz de Berio**, por brindarme su hogar y su apoyo durante mi estancia en San Sebastian con motivo de las *V Jornadas de la Calidad del Software*. A ese delicioso canal de comunicación llamado **Internet**, sin el cual no hubiese tenido acceso ni a la mitad de las referencias bibliográficas empleadas a lo largo de esta investigación. Finalmente, a **mis enemigos**, quienes nunca escatimaron esfuerzos por mantenerme en forma.

A todos y a todas, llegue mi reconocimiento y mi más sincero agradecimiento.

Resumen

El diseño flexible del nivel conceptual y semántico de una interfaz de usuario moderna, permite un acceso rápido y eficaz a la información proporcionada por la aplicación, facilitando su distribución entre diferentes comunidades de usuarios, caracterizadas generalmente por poseer diferentes necesidades de información. Sin embargo, por regla general, un buen diseño para el nivel conceptual y semántico no garantiza un acceso universal a la información almacenada en la base de conocimiento de la aplicación, dado que los niveles léxico y sintáctico del diseño de una interfaz juegan un papel fundamental al respecto. Se debe recordar que aquellos usuarios que sufren cual algún quier tipo de incapacidad han de acceder a la interfaz –y a su base de conocimiento asociada– por medio de técnicas de interacción diseñadas especialmente para superar sus incapacidades. Naturalmente, el lugar en el que se deben implementar estas técnicas es precisamente el nivel léxico y sintáctico de la interfaz.

Sin embargo, estos niveles suelen estar diseñados para satisfacer los requisitos del llamado usuario típico, el cual representa una generalización abstracta de cada usuario de una aplicación. Las técnicas clásicas para el diseño de interfaces basadas en esta idea suelen conducir a una generalización abstracta de la interfaz, la cual, no solo subestima la naturaleza real de sus usuarios, sino que además tampoco tiene en cuenta la individualidad y diversidad humana. Como consecuencia, los sistemas interactivos obtenidos suelen presentar graves defectos en términos de facilidad de uso, dado que aquellos usuarios cuyos requisitos de interacción no se asemejan a aquellos previstos para el usuario típico suelen experimentar problemas y frustración cuando intentan acceder a determinadas funcionalidades de la interfaz.

El enfoque adoptado para resolver este problema pasa por el diseño de diferentes versiones del nivel léxico y sintáctico de la interfaz, en donde cada versión es adaptada especialmente para satisfacer las características cognitivas, perceptivas y motrices de un tipo específico de usuario. Dado que el diseño estático de múltiples versiones de una misma interfaz es demasiado costoso como para resultar rentable a la industria del software, en la práctica se fabrican muy pocas versiones de la interfaz –si es que llega a fabricarse alguna.

Para evitar estos problemas, las diferentes versiones del nivel léxico y sintáctico de una interfaz deberían ser generadas dinámicamente en función de los requisitos de interacción del usuario final. Este es el enfoque adoptado por GADEA, un sistema de gestión de interfaces de usuario inteligente, capaz de diseñar y de construir diálogos interactivos en tiempo real, los cuales se adaptan especialmente a las características cognitivas, perceptivas y motrices del usuario activo. GADEA constituye una marco conceptual general para el desarrollo de aplicaciones adaptables, el cual resulta fácil de usar por parte de los programadores, a la vez que resulta suficientemente flexible como para

garantizar un acceso universal al nivel léxico y sintáctico de las interfaces generadas. Este sistema permite un acceso efectivo y preciso a la interfaz para un amplio rango de usuarios, incluyendo aquellos que sufren algún tipo de discapacidad física.

GADEA separa la funcionalidad de una aplicación de su interfaz por medio de un mecanismo automático basado en reflexión estructural. Por medio de un examen cuidadoso del código binario de cualquier aplicación compatible con el marco conceptual para el desarrollo de interfaces de usuario planteado por GADEA, este sistema es capaz de analizar el nivel semántico de una aplicación, extrayendo información acerca de los procesos de usuario definidos por el programador en tiempo de edición. Esta información es empleada para definir los requisitos de la aplicación en un marco libre de contexto, los cuales son convertidos dinámicamente en diálogos interactivos.

Por medio de un motor de inferencias basado en lógica difusa, los agentes interactivos inteligentes incluidos en el corazón de GADEA son capaces de definir el formato de los diálogos interactivos en tiempo real, determinando cuales son los *widgets* más adecuados para los mismos dependiendo de las características del usuario actual. Una vez seleccionados, los *widgets* son configurados dependiendo también de las preferencias y requisitos del usuario, determinando de este modo su tamaño, color, etc. Algunos de los factores tomados en cuenta por los agentes inteligentes para diseñar los diálogos interactivos son la edad del usuario, su sexo y lateralidad así como su grado de precisión visual, auditiva y motriz. GADEA almacena esta información en un modelo de usuario central, el cual es actualizado por medio de un pequeño ejército de agentes, los cuales analizan constantemente los estilos de interacción empleados por el usuario, midiendo su eficiencia en términos cognitivos.

Palabras Clave

Interacción Basada en Agentes, Interacción Multimodal, Interfaces Inteligentes, Interfaces Adaptables, Pruebas de Navegación Remotas, Modelos de Navegación, Pruebas de Usabilidad, Tecnología Cognitiva, Navegación Multidimensional, Hipermedia.

Abstract

The flexible design of the conceptual and semantic levels of a modern user interface, allows an efficient and quick access to the information provided by the application, facilitating its distribution among different communities of users, usually characterised by different information requirements. However, a good design for the conceptual and semantic level of an user interface, usually is not a warranty for a universal access to the information contained in the application's knowledge base, as the syntactic and lexical levels of the interface's design also play a crucial role in this proccess. We must remember that users, who suffer from any kind of physical disability, must access the interface –and its associated knowledge base– by mean of interaction techniques specially adapted to their degree of disability. Naturally, those interaction techniques must be included in the lexical and syntactical level of the interface.

However, those levels are commonly designed to match the requirements of the so-called typical user, an abstract generalisation of each user of a specific application. Classic user interface design techniques based on this idea, usually conduct to an abstract generalisation of the interface, which not only misconceive the real nature of users, but neither comply with the human individuality and diversity. As a consequence, those interactive systems have important drawbacks in terms of usability as users whose interaction requirements don't match with those of the typical user, usually experience frustration and problems when they try certain features of the interface.

The approach generally adopted to solve this problem is the design of different versions of the lexical and syntactic level of a user interface, where each version is specially tailored to comply with the cognitive, perceptive and motor characteristics of a specific kind of user. As the static design of multiple versions of an interface is too expensive to be attractive to the software industry, only a few versions –if any– are finally produced.

In order to avoid those problems, the different versions of the lexical and syntactical level of the user interface should be generated dynamically, tailoring them to the final user interaction needs. This is the approach followed by GADEA, an intelligent user interface management system able to design interactive dialogues in real time, which are specially adapted to the specific cognitive, perceptive and motor characteristics of the final user. GADEA represents a general framework for the development of adaptive applications, which is both easy to use and flexible enough to guarantee a universal access to the lexical and syntactical level of the generated interfaces. This system allows an effective and accurate access to the interface by a wide range of users, including those who suffer from physical disabilities.

GADEA separates the functionality of an application from its user interface by mean of an automatic mechanism, which is based on

computational reflection. Examining the binary code of any application compatible with the user interface development's framework of GADEA, this system is able to analyse the semantic level of the application, extracting information about the user processes defined by the programmer at compiling time. This information is used to define the application's context-free information requirements, which are dynamically converted into interactive dialogues.

Based on a fuzzy logic engine, the intelligent interactive agents inside GADEA define the layout of the interactive dialogues on real time, determining which are the most suitable widgets to be employed, depending on the characteristics of the current user. Once selected, the widgets are configured depending on the user's preferences and needs, determining its size, colour, and many other parameters. Some of the facts employed by the intelligent agents to design the interactive dialogues are the age of the user, his sex, his laterality and his visual, auditory and motor accuracy. GADEA stores this information in a central user model who is kept updated by mean of a small army of agents that constantly analyse the user interaction styles employed by the user, measuring his accuracy in terms of cognition.

Keywords

Agent Based-Interaction, Multi-modal Interaction, Intelligent Interfaces, Adaptive Interfaces, Remote Navigability Testing, Navigation Models, Usability Testing, Cognitive Technology, Multidimensional Navigation, Hypermedia.

Tabla de Contenidos

Código

Figuras

Tablas

0

0 Introducción

Lo último que se descubre al escribir un libro es lo que hay que poner primero

Blaise Pascal

0.1 Contexto

El proceso de desarrollo de los mecanismos de interacción de toda herramienta informática comprende la identificación de las características particulares del entorno cultural al que pertenecen dichos usuarios [Roast (1997)]. Sin embargo, la naturaleza de cierto tipo de aplicaciones, dirigidas a una muy amplia variedad de usuarios distintos, tales como los quioscos multimedia, las aplicaciones educativas o los sitios web, implica la necesidad de un diseño que abarque las expectativas de diferentes entornos culturales de manera simultánea [Yeo (1996)].

La complejidad de este proceso de diseño aumenta más aún si consideramos que diseñar para mas de un entorno cultural no es suficiente dado que en ocasiones la diversidad existente entre los miembros de una determinada sociedad es muy marcada. Hemos de recordar que la diversidad en la adquisición de conocimientos descansa

fundamentalmente sobre la variedad de inteligencias y los diferentes niveles y formas que presentan las personas a la hora de aprender [Álvarez y Soler (1999)]. Esta máxima se aplica directamente al manejo de una herramienta así como al conocimiento que se puede obtener mediante el uso de la misma.

Es por ello que al final de la fase de diseño de una interfaz es frecuente considerar el empleo de más de un mecanismo de interacción, dependiendo de los distintos enfoques empleados para estructurar la base de conocimiento de la aplicación; así como también de los usuarios que van a acceder a ella. Hablamos entonces de interfaces multimodales [Myers (1998)] en los que una misma acción se puede ejecutar mediante diversos métodos como por ejemplo, impartiendo órdenes orales o escritas y / o moviendo un puntero por la pantalla de un ordenador.

0.1.1. Adaptabilidad

Con un diseño adaptado al máximo a las características de los usuarios, los individuos pertenecientes al entorno destinatario pueden obtener el mayor rendimiento y satisfacción posible ofrecido herramienta. Sin embargo, la adaptación de aplicaciones informáticas a las necesidades cognitivas, perceptivas o motrices concretas de determinados grupos de usuarios resulta en la mayoría de los casos excesivamente costosa y poco rentable como para que resulte atractiva a la industria del software. La amplia diversidad cognitiva, perceptiva, motriz o cultural de los usuarios de una aplicación hacen muy difícil y costoso el diseño de interfaces estáticas efectivas, ya que es prácticamente imposible que un mecanismo de interacción sea aprovechado con la misma eficacia por dos usuarios dotados de modelos cognitivos, perceptivos o motrices distintos. Con frecuencia se llega a dar el caso en el que una aplicación sólo puede ser empleada por ciertos usuarios dotados de unas características particulares. Este es el caso por ejemplo de los exploradores de Internet estándar, los cuales resultan ser aplicaciones totalmente prohibidas para usuarios con problemas de visión grave o con ceguera.

Aquellos casos en los que para que un determinado colectivo de usuarios pueda hacer uso de aplicaciones cuya interfaz no ha sido diseñada para contemplar un acceso universal, será necesario una modificación total o parcial del código de la misma, llegando a modificar notablemente su diseño original. En todo caso, estas modificaciones requieren la creación de una versión especial (estática) de la aplicación, es decir, creando una versión diseñada ex profeso para cada colectivo de usuario, con los costes de diseño que ello conlleva y los conocidos problemas de mantenimiento inherentes de disponer múltiples versiones de un mismo producto en la línea de producción. Así, en el caso del ejemplo de los exploradores de Internet por parte de usuarios discapacitados visualmente, será necesaria la adaptación de la interfaz para hacer uso de las metáforas de navegación existentes para este tipo de usuarios [Savidis y Stephanidis (1995)] o en todo caso, será necesario modificar sensiblemente el contenido de la información incluida en la base de conocimiento de la aplicación para que pueda ser

percibida por dichos usuarios a través de un explorador de Internet estándar, opción esta última adoptada por la Organización Nacional de Ciegos Españoles (ONCE) en su portal de Internet (2001).

Los dos posibles enfoques planteados en el ejemplo anterior presentan en común su elevado coste de desarrollo y de mantenimiento, presentándose la paradoja de que ámbos enfoques la mayor parte de los mecanismos internos de la aplicación a adaptar (motor de descarga de páginas, configuración de proxies, gestor de listas de favoritos, etc.) permanecen completamente inalterados, siendo tan solo la interfaz el módulo software que requiere de modificaciones importantes. Aún así, será el coste de diseñar un mecanismo de interacción para cada tipo de usuario la actividad lo que encarezca el proceso, haciéndolo prohibitivo para la mayoría de los proyectos.

La solución planteada en esta tesis pasa por el empleo de mecanismos explícitos e implícitos para la detección del tipo de usuario de una aplicación, los cuales facilitan una posterior adaptación dinámica y automática de la interfaz, seleccionado para ello los mecanismos de interacción a aplicar en función de las características cognitivas, perceptivas y motrices de sus usuarios. Todo ello sin necesidad de modificar el código funcional de las aplicaciones, puesto que un diseño racional del mismo, en base a su integración con un generador de interfaces dinámico, permite que éste último componente pueda explorar el código binario de cualquier aplicación en tiempo real, con el objeto de obtener toda la información necesaria para confeccionar dichas interfaces.

0.1.2. GADEA

El objetivo primordial de esta tesis es el diseño de un modelo adaptable para el desarrollo de aplicaciones genéricas, fácil de usar por parte de los programadores y capaz de adaptar la sintaxis de la interfaz de usuario de una aplicación de forma automática. Este modelo, bautizado con el nombre de GADEA, emplea técnicas de inteligencia artificial basadas en lógica difusa, agentes inteligentes interactivos, así como escuadrones de agentes software capaces de analizar constantemente los diálogos interactivos empleados por el usuario, intercambiando información con las aplicaciones con el objeto de crear un perfil individual para cada usuario, el cual es almacenado en un modelo de usuario.

GADEA, como sistema de gestión de interfaces de usuario (UIMS o *User Interface Management System*) dotado de inteligencia, parte de una base de conocimiento establecida por el dominio del problema y es capaz de establecer con el usuario mecanismos de diálogo adaptados a sus capacidades cognitivas, perceptivas y motrices, todo ello de forma automática y transparente, tanto para el usuario final de la aplicación como para los programadores y diseñadores de la misma. Para ello, GADEA se basa en el paradigma clásico de los sistemas expertos, los cuales son capaces de emular el comportamiento humano en una determinada área de conocimiento o disciplina. En el presente caso, nuestro sistema de gestión de interfaces de usuario inteligente simula el

comportamiento de un experto diseñador de interfaces, capaz de confeccionar un curriculum adaptado a la diversidad en tiempo real, es decir, un agente interactivo e inteligente capaz de adaptar su discurso a las necesidades concretas del receptor de la información.

En la teoría clásica de los sistemas expertos [Velarde (1991)], un experto alimenta el sistema con sus conocimientos acerca del dominio del problema incluyendo los hechos y reglas que servirán de materia prima para que el motor de inferencias y el módulo de adquisición de conocimiento del sistema pueda inferir nuevo conocimiento a partir del original. En el enfoque planteado en GADEA, el esquema clásico es mejorado para introducir nueva información acerca del dominio del problema pero desde el punto de vista del usuario, es decir, el usuario alimenta también la base de conocimiento con nuevos hechos y reglas. La información proporcionada por éste ultimo agente se introduce en el sistema, bien de forma directa (con reglas y hechos explícitos del tipo de *soy zurdo* o *la acción de imprimir la identifico con este icono*) o bien de forma indirecta, a través de las distintas acciones realizadas sobre la aplicación, las cuales son observadas y analizadas por GADEA.

De este modo, se crea un modelo particular para cada usuario de la aplicación en el cual se plasma la visión que éste tiene acerca de la base de conocimiento. Visión que es adaptada y transformada en una serie de parámetros que permiten convertir al usuario en un entidad única y especial dentro del colectivo al que éste pertenece. En el estado actual de la investigación y con los prototipos de GADEA desarrollados hasta el momento, los parámetros que permiten definir unívocamente a un usuario incluyen entre otros la edad, el sexo, el estado de la percepción auditiva y visual, la habilidad motriz, la velocidad de reacción en la toma de decisiones simples, etc.

A partir de esta información es posible crear un modelo de usuario capaz de reflejar de forma bastante aproximada el estado cognitivo, perceptivo y motriz del usuario real. Es precisamente con este modelo de usuario específico con el que trabaja GADEA a la hora de seleccionar los mecanismos de interacción necesarios para confeccionar los distintos diálogos interactivos requeridos para que una aplicación pueda intercambiar información con el usuario.

GADEA permite una adaptación sencilla y eficaz de los aspectos de bajo nivel (sistema perceptivo y motor) de la interfaz de usuario, sin que ello represente una carga especial para los diseñadores y programadores de la aplicación. De hecho, este proceso de adaptación es completamente transparente para ellos. Esto permite liberar recursos de tal modo que los diseñadores puedan centrarse en el diseño del nivel semántico y conceptual de la interfaz, proporcionando a estos diseñadores dos primitivas básicas con las que pueden construir sus modelos. Estas primitivas se incluyen en dos niveles:

- **Procesos de Usuario**. Proporciona un mecanismo de separación básico entre la funcionalidad de una aplicación y su interfaz, aplicando para ello mecanismos de reflexión estructural que permiten examinar el código binario de las aplicaciones

compatibles con este modelo en busca de procesos de usuario, diálogos interactivos y otros elementos constituyentes de una interfaz. Todo ello de forma automática o semiautomática.

- **Diálogo Interactivo**. Permite especificar los procesos de interacción en base a componentes orientados a objetos que contienen primitivas de datos básicas (entero, cadena de caracteres, fecha, etc.) en lugar de en base a *widgets* (botones, cuadros de diálogo, ventanas, etc.), como viene siendo la práctica habitual en el diseño de interfaces de usuario estáticas. Esto permite aproximar los enfoques metodológicos empleados para el diseño de modelos de datos y de procesos de una aplicación con los respectivos enfoques usados para diseñar el modelo de interacción, dado que se centra la atención de los diseñadores en la naturaleza de los datos empleados por la aplicación en lugar de centrarla en la representación de los mismos.

Con todo ello se pretende liberar a los programadores y diseñadores de una gran carga, permitiéndoles concentrar sus esfuerzos en el diseño de las metáforas de interacción a un nivel cognitivo (adaptación de alto nivel) y en la mejora de los procesos internos de ingeniería del software de su código.

Dado que GADEA emula el trabajo realizado por el diseñador el proceso de toma de decisiones queda centralizado en un único ente, permitiendo garantizar la homogeneidad de todas las aplicaciones que se ejecutan sobre bajo el control de este UIMS. Gracias a que el proceso de toma de decisiones se centraliza en los agentes inteligentes incluidos en GADEA, es posible garantizar la coherencia en el aspecto de la interfaz de todas las aplicaciones desarrolladas por una organización. Este principio de diseño ha sido hasta el momento muy difícil de alcanzar, dadas las lagunas y las ambigüedades existentes en las recomendaciones de diseño para interfaz de usuario de algunas plataformas de desarrollo, como es el caso de la plataforma Macintosh (1992) o de la plataforma Windows [IBM (1989)]. Con el proceso de toma de decisiones centralizado, las ambigüedades pueden ser resueltas de forma homogénea con la incorporación y/o refinamiento de las reglas de inferencia incluidas en la base de conocimiento de GADEA.

0.1.3. Principales Aportaciones

GADEA representa no solo una arquitectura para un sistema de gestión de interfaces de usuario inteligente sino también un novedoso modelo para el desarrollo rápido y eficaz de interfaces de usuario de tipo WIMP especialmente diseñadas para aplicaciones de comercio electrónico, hipermedia y de gestión. Este modelo permite diseñar, implementar y perfeccionar prototipos de modelos de interacción de alto nivel (conceptual y semántico) de forma rápida y sencilla, puesto que la construcción de estos modelos a nivel léxico y sintáctico es realizado íntegramente por GADEA.

Este UIMS permite la adaptación del nivel léxico y sintáctico de una interfaz usando para ello una arquitectura flexible basada en el empleo de agentes interactivos inteligentes, los cuales se corresponden con diversos componentes del modelo cognitivo humano. Este modelo ha sido desarrollado, probado y refinado a lo largo de los últimos años tanto por la psicología cognitiva, como por la ingeniería de factores humanos; demostrando que las inferencias sobre el comportamiento humano que se pueden alcanzar por medio de su uso son fiables y efectivas. Gracias a esta estructura flexible, la arquitectura de GADEA puede ser ampliada, refinada y mejorada mediante la adición, substitución y mejora de nuevos agentes interactivos, permitiendo así que la eficacia del proceso de adaptación aportado por este UIMS evolucione en un futuro a la vez que lo hace el modelo cognitivo sobre el que se fundamenta.

Aunque las principales aportaciones de esta tesis serán descritas y analizadas en profundidad en las correspondientes conclusiones recogidas en esta memoria a partir de la página 395, se incluyen a continuación algunas de las más significativas:

- **Nuevo Paradigma de Desarrollo de Interfaces de Usuario**. Este paradigma devuelve el papel de protagonista al modelo de interacción en el análisis y el diseño de la aplicación, favoreciendo el desarrollo de interfaces de usuario de gran calidad al centrar el proceso en la identificación de procesos y datos de usuario.

- **Mecanismo Automático de Separación entre Funcionalidad e interfaz**. Mediante el empleo de las técnicas de reflexión que el estado de la tecnología permite aplicar en el momento de escribir estas líneas, GADEA es capaz de inspeccionar el código binario de una aplicación en busca de sus procesos de usuario así como de las *precondiciones* y *postcondiciones* asociadas con el objeto de generar la interfaz de una aplicación de forma automática. La transmisión de eventos entre la interfaz generada y el código funcional de la aplicación es transparente para los programadores, reduciendo de este modo la complejidad del proceso de desarrollo.

- **Verificación y Validación Automática de Entradas a Nivel Semántico**. El alto grado de separación entre la interfaz de una aplicación y su código funcional, permite la definición del modelo de datos para la interacción como del modelo de datos de la aplicación, empleando para ello primitivas de datos orientadas a objeto, en lugar de recurrir a una combinación entre estructuras de datos tradicionales (enteros, vectores, etc.) y *widgets* (botones, listas desplegables, etc.), tal y como ocurre por ejemplo en el caso de la programación visual. Este enfoque permite simplificar el proceso de desarrollo de una interfaz al poder especificar los requisitos de información de la misma utilizando las mismas técnicas que emplea la ingeniería del software para especificar los requisitos de su

modelo de datos. Esta información es empleada por GADEA para determinar el dominio de la información asignado a un determinado *widget* de un diálogo interactivo, de tal modo que la interfaz es capaz de realizar comprobaciones automáticas sobre la pertenencia de los datos introducidos por el usuario con respecto a dicho dominio, todo ello sin necesidad de incluir código extra que realice ese tipo de comprobaciones en la parte funcional de la aplicación.

- **Adaptación Automática de Contenidos**. GADEA extiende los mecanismos de internacionalización existentes para adaptar el contenido de las interfaces generadas de forma dinámica sin intervención explícita del programador de la aplicación. El mecanismo prevé la adaptación de contenidos en todas las variedades posibles (lenguaje escrito, lenguaje hablado, representación iconográficas, variaciones especiales para usuarios disminuidos físicamente, etc.).

- **Modelado de Interfaces por medio de la Teoría General de Sistemas (TGS).** Aunque la TGS ha sido empleada con anterioridad para expresar formalmente ciertos requisitos de una interfaz, el enfoque planteado en esta tesis permite modelar cualquier tipo de interfaz a cualquier escala, representando tanto a las aplicaciones como a sus usuarios como componentes de un sistema de comunicación provisto de una jerarquía de objetivos que éste ha de cumplir. El enfoque contempla la identificación de las características especiales de los componentes empleados (hardware, software y humanos) para combinarlos entre sí de tal modo que se puedan maximizar los beneficios obtenidos.

- **Generación Dinámica de Diálogos Interactivos**. Los agentes inteligentes son capaces de generar diálogos interactivos en tiempo real en función de la especificación de contenidos proporcionada por los diseñadores de la aplicación cuando éstos hacen uso del mecanismo de separación entre interfaz y funcionalidad proporcionado por GADEA. La generación de éstos diálogos se hace en función tanto de las normas de uso general, como de aspectos específicos e individuales requeridos por cada usuario. En base a éste último factor, la selección de los *widgets* a emplear en un determinado diálogo interactivo dependerá exclusivamente de las características cognitivas, perceptivas y motrices de cada usuario.

- **Adaptación de Diálogos Interactivos a Nivel Sintáctico**. El motor de inferencias incluido en el corazón de los agentes interactivos permite emplear todo el poder de la lógica difusa para adaptar la configuración de cada *widget* empleado en un diálogo interactivo, así como las características del propio diálogo en función de los requisitos individuales de los usuarios, dejando la puerta abierta

para que en un futuro próximo este proceso de adaptación tenga en cuenta además las características físicas del entorno de trabajo de los mencionados usuarios.

- **Modelo de Señuelos y Predadores basado en *Chunks***. Este modelo simula el mecanismo de asociación de las memorias episódicas y semánticas para permitir el establecimiento de diferentes grados de prioridad para la información a transmitir en un diálogo interactivo, de tal modo sea posible realizar un ajuste preciso del grado de detalle de la información enviada, adaptándola tanto a la capacidad del propio canal de comunicación como a la capacidad de procesamiento del receptor de la misma.

- **Gestión Automática de la Carga Cognitiva de un Proceso de Usuario**. Gracias a la información almacenada en el modelo de usuario de GADEA, los agentes inteligentes pueden verificar que la cantidad de información desplegada en todo momento por un diálogo interactivo, no supere en ningún momento la capacidad cognitiva del usuario destino, capacidad que dependerá de las características cognitivas y perceptivas del usuario y que podrá fluctuar con el tiempo incluso dentro de una misma sesión de trabajo. En caso necesario, estos agentes disponen de la necesaria potestad como para clasificar la información a distribuir y repartirla en varias fases de transmisión.

- **Análisis Continuo del Comportamiento Humano**. Además de los agentes interactivos, GADEA está dotado con un pequeño ejército de agentes con gran capacidad de movilidad, capaces de observar y analizar de forma continua y automática el cúmulo de acciones de bajo nivel realizadas por un usuario dentro del ámbito de un diálogo interactivo, pudiendo determinar de forma implícita el grado de habilidad de un usuario en la realización de tareas relacionadas con aspectos cognitivos, perceptivos y motrices. Esta información es almacenada en el modelo de usuario y podrá ser empleada tanto por los agentes interactivos inteligentes –en la creación dinámica de los diálogos interactivos de las aplicaciones– como por agentes humanos en la mejora y control de calidad de los procesos superiores de interacción de dichas aplicaciones.

0.1.4. Organización de esta Memoria

La presente memoria, que constituye el resumen del trabajo de investigación que ha conducido a la resolución de la tesis planteada, se encuentra integrada por siete partes claramente diferenciadas.

En la **parte I** (capítulos 1: *UIMS*: y 2 *Interfaces Inteligentes*) se recoge la descripción de los elementos básicos constitutivos de las interfaces de usuario avanzadas, incluyendo una breve historia, descripción y clasificación de los sistemas de gestión de interfaces de usuario (UIMS),

analizando especialmente las características de estos sistemas, la definición formal de interfaces por medio de este tipo de herramientas y las diferentes técnicas empleadas para separar la funcionalidad de una aplicación de su interfaz. Para ello se incluyen diversas referencias a prototipos y/o UIMS operativo, tanto académicos como comerciales, destacando sus puntos fuertes y débiles, sus aportaciones a la industria, así como sus posibles carencias. En esta primera parte se analizan además diversos proyectos, prototipos y sistemas dotados de interfaces inteligentes, tanto en aplicaciones como en herramientas de desarrollo y UIMS (capítulo 2: *Interfaces Inteligentes*). En todo este proceso de análisis sobre el estado del arte se destacan especialmente las características de adaptabilidad de los sistemas descritos, así como la separación entre funcionalidad e interfaz.

La **parte II** comprende los capítulos 3 (*La Individualidad*) y 4 (*La Comunicación*) y está dedicada a describir los objetivos planteados en esta investigación, así como a realizar una breve descripción del diseño de GADEA, realizando la descomposición de éste en los tres componentes básicos que lo integran. Estos módulos son: CodeX, DEVA y ANTS. El módulo CodeX es el encargado de separar la interfaz de la funcionalidad de las aplicaciones que se ejecutan en este UIMS. El módulo DEVA será el responsable de la generación dinámica de diálogos interactivos adaptados a las características cognitivas, perceptivas y motrices del usuario. Por último, el módulo ANTS será el encargado de mantener actualizado cada registro individual del modelo de usuario. Mientras que en el capítulo 3 (*La Individualidad*) se realiza una crítica general de los problemas que conlleva el uso de interfaces genéricas, a la vez que se identifican los objetivos y componentes de GADEA, en el capítulo 4 (*La Comunicación*) se diseña un enfoque conceptual para el modelado de interfaces de usuario basado en la Teoría General de Sistemas (TGS), definiendo en este capítulo varios de los conceptos fundamentales que se emplearán en los capítulos posteriores, tales como el concepto de canal de comunicación, y el concepto de ruido.

La **parte III** está dedicada a CodeX y en ella se revisan los enfoques conceptuales empleados para el desarrollo del software en términos del diseño de las interfaces de usuario (capítulo 5: *Diseño Centrado en el Usuario*), se descompone el modelo de interacción empleando estos enfoques conceptuales avanzados de acuerdo con el peso que tienen los datos o los procesos en el mismo, planteando la relación semántica que existe entre el modelo de datos y un diálogo interactivo (capítulo 6: *El Modelo de Interacción*), se identifican los puntos de arranque de un proceso de usuario en un modelo de interacción, describiendo el mecanismo empleado por CodeX para la inspección automática del código binario en busca de estos puntos de entrada (Capítulo 7: *Aplicaciones y Procesos de Usuario*), se define el diálogo interactivo como puente para el intercambio de información entre el usuario y las aplicaciones, diseñando las primitivas básicas de datos que se pueden incluir dentro de éste (capítulo 8: *El Diálogo*), se define el lenguaje ACML como mecanismo para la especificación de estos diálogos, así como clasificación de las unidades de conocimiento contenido en ellos mediante *chunks* (Capítulo 9: *La Semántica*), se describen las

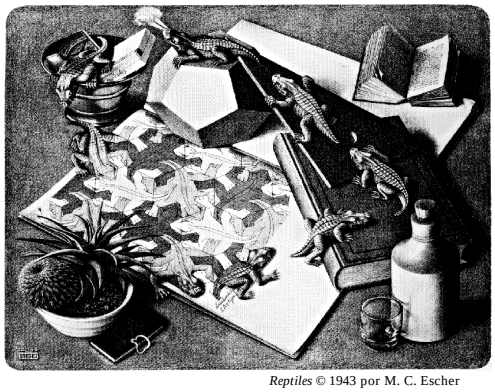
características cognitivas del lenguaje y su importancia en los procesos de adaptación de alto nivel, introduciendo al *Language Manager* de CodeX; el ente encargado de realizar este tipo de adaptación (Capítulo 10: *El Lenguaje*), finalizando con la descripción del *Application Explorer*, la aplicación nativa de CodeX encargada de la gestión de aplicaciones GADEA (capítulo 11: *El Application Explorer*).

En la **parte IV**, dedicada a DEVA, se plantea la limitación de la capacidad humana para el procesamiento de la información, determinando como puede medirse el grado de veteranía de un usuario (capítulo 12: *El Modelo Cognitivo*), se describe el modelo de procesador humano, los diferentes procesadores y memorias que lo componen, como se estructura la información en *chunks* y los mecanismos empleados por DEVA para repartir la cantidad de información a transmitir en bloques del tamaño adecuado (capítulo 13: *El Procesador Humano*). En el capítulo 14 (*La Atención*) se introducen los conceptos básicos relacionados con la atención humana, se identifican los distintos niveles de procesamiento de la información y como el conocimiento de estos niveles sirve a DEVA para dar soporte a tareas de vigilancia en determinados tipos de interfaces. En el capítulo 15 (*El Procesador Perceptivo*) se estudia el proceso de separación entre formas y fondos, introduciendo el modelo de objetos señuelo y objetos predador, que permitirá agrupar y separar componentes para garantizar una percepción eficaz de los contenidos de un diálogo interactivo. En el capitulo 16 (*Un Modelo de Evaluación*) se introduce una importante mejora al modelo de predicción GOMS que permitirá seleccionar el mejor modo de interacción posible para un usuario concreto en función de sus características cognitivas, perceptivas y motrices, mediante la aplicación de funciones de preferencia evaluadas en tiempo real. En el capítulo 17 (*La Diversidad Humana*) se describen diversos factores que afectan a la percepción de la *weltanshaung* en función de parámetros como la edad o el sexo del individuo, así como las ventajas que aporta la lógica difusa para salvar estas diferencias y lograr constantes perceptivas. En este capítulo se describen además las metáforas de distribución visual empleadas por DEVA para repartir el ancho de banda de un canal de comunicación entre todos y cada uno de los objetos transmitidos en él. Por último, en el capítulo 18 (*Adaptación Cromática*) se recogen diversos aspectos relacionados con la transmisión de señales por medio de variaciones cromáticas y como el mensaje emitido puede ser adaptado –antes de su difusión– para lograr una constante perceptiva cromática, con independencia de factores tales como la edad, el sexo o el grado de percepción visual.

La **parte V** está dedicada a ANTS y en ella se describe el diseño del sistema de agentes empleado para la actualización automática y continua del modelo de usuario empleado por GADEA, definiendo el tipo de información obtenida e incluyendo ejemplos acerca de cómo ésta es procesada antes de poder obtener resultados fiables (capítulo 19: *Espías*). En esta parte de incluye además una descripción de la utilidad de ANTS como herramienta auxiliar para la realización de pruebas de usabilidad sobre los niveles superiores (conceptuales y semánticos) de los modelos de interacción, sea esta compatible con el modelo de desarrollo planteado por GADEA o no (capítulo 20: *Usabilidad*).

La **parte VI** está dedicada a las conclusiones. En el capítulo 21 (*Un Nuevo Modelo*) se hace un análisis de la reducción de costes de producción obtenido mediante la aplicación del paradigma de desarrollo formulado por GADEA, codificando un sencillo ejemplo de interfaz empleando para ello los mecanismos clásicos de la programación, así como la propuesta de este UIMS. En este capítulo se identifican las tareas implicadas en los dos paradigmas y se hace un estudio de la complejidad de desarrollo, tanto desde el punto de vista del número de operaciones a realizar, como del conocimiento necesario para ello. En el capítulo 22 (*La Adaptabilidad*) se describe la naturaleza de los experimentos realizados para determinar el grado de usabilidad alcanzado por los prototipos codificados para GADEA, así como los resultados obtenidos en los mismos. El campo de estudio de estos experimentos comprende tanto la prueba de los nuevos mecanismos de interacción aportados por GADEA como la calidad del proceso de adaptación automática realizado por los agentes interactivos inteligentes. En el último capítulo de esta parte (capítulo 23 : *La Tecnología*) se incluye una compilación de los diferentes aspectos que GADEA consigue adaptar en una aplicación, describiendo los ámbitos de uso de esta tecnología, así como las líneas de investigación puestas en marcha para la ampliación y mejora de la calidad de este sistema de gestión de interfaces de usuario

Por último, en la **parte VII** se recogen las imágenes en color de esta memoria en un cuadernillo especial que hace las veces de capítulo 24 (*Imágenes en Color*), se incluye la bibliografía empleada en esta investigación, tanto en medio impreso como en medio electrónico (capítulo 25 : *Reseñas Bibliográficas*), un extenso glosario de términos en el que se recoge la nomenclatura empleada en esta memoria, así como los significados concretos dados en esta investigación a determinados términos de uso común de significado ambiguo (capítulo 26: *Glosario y Acrónimos*), así como un índice analítico en el que se recogen los términos y autores citados en esta memoria (capítulo 27: *Índice Analítico*).



Reptiles © 1943 por M. C. Escher

P A R T E I

Sistemas de Gestión de Interfaces de Usuario Inteligentes

2 Interfaces Inteligentes

Agentes interactivos inteligentes. Las interfaces multimodales. Mecanismos de adaptación de una interfaz en base a las necesidades de un usuario concreto. Definición de las caracerísticas de un usuario a través de los modelos de usuario. Análisis y aportaciones de algunos de los sistemas inteligentes más representativos: APEX, GRID, AUI, CHORIS, CUBRICON, MASTERMIND, PERSONA y VWPC.

p. 69

1

1 UIMS

Como ocurre con la mayoría de los sistemas basados en el Modelo Seeheim, el diseñador acaba construyendo toda la interfaz de usuario dentro de la cajita con la interrogante

Brad Myers

1.1 Sistemas de Gestión de Interfaces de Usuario

El estudio de la interacción y comunicación humana comprende el análisis de cómo los individuos diseñan, construyen y emplean sistemas interactivos, así como el análisis de los efectos que tienen estos sistemas sobre los individuos, las organizaciones y en definitiva, la sociedad. Esta área de investigación aumenta su relevancia día a día en cualquier actividad humana en donde sea necesario diseñar un sistema que vaya a ser empleado por seres humanos. Y la informática no es desde luego la excepción. De hecho, los seguidores de esta disciplina son cada día más conscientes de la necesidad de extender el alcance de la informática más allá de la resolución de problemas relacionados con el hardware o con la ingeniería del software, en un intento por comprender como la tecnología desarrollada por éstos puede mejorar la calidad de vida de los usuarios y ayudarles a alcanzar sus objetivos.

Los mencionados usuarios de los productos que genera la tecnología informática esperan encontrar interfaces fáciles de utilizar y de aprender. El desarrollo de una interfaz de usuario efectiva, útil, segura y agradable son factores clave para el éxito de un producto en el mercado y no cabe ninguna duda de que sus diseñadores son conscientes de ello. De hecho, las encuestas muestran que al menos el 50% del esfuerzo de diseño y de programación de un proyecto informático está dedicado a la parte que compete a su interfaz de usuario [Myers y Rosson (1992)]. Este es un esfuerzo que se ve compensado económicamente. Nielsen desarrolló un modelo matemático en 1993 que sugería que el empleo adecuado de técnicas y herramientas de ingeniería de usabilidad podría ahorrar unos 43.320 euros para un proyecto pequeño, unos 681.120 euros para un proyecto de tamaño moderado y unos 9.111.020 euros para un proyecto de gran envergadura. Este ahorro se atribuye a una considerable reducción en el tiempo necesario para ejecutar una tarea, a una fuerte reducción en la distracción del usuario, a la disminución del número de errores producidos, a la reducción del personal de soporte técnico, a la eliminación de las tareas de entrenamiento y a la disminución del número cambios en el software una vez distribuido.

Entre las herramientas más potentes con las que cuenta esta disciplina se encuentran los sistemas de gestión de interfaces de usuario o UIMS (*User Interface Management Systems*) así como sus *toolkits* asociados. De hecho, una exhaustiva revisión del estado del arte realizada en 1996 por Myers et al colocaba la invención de estos sistemas entre los cuatro hitos más importantes de la disciplina, al mismo nivel que la invención de las ventanas, la manipulación directa de objetos gráficos y el hipertexto.

1.1.1. UIMS

Un sistema de gestión de interfaces de usuario (UIMS) es un componente software que se encarga de gestionar todas las interacciones con el usuario de una aplicación. Se trata de sistemas integrales para el desarrollo de interfaces de usuario formados por un conjunto de herramientas para la creación y gestión de todos los elementos constitutivos de una interfaz. De este modo es posible liberar al código funcional de la aplicación de la realización de tareas relacionadas con la gestión de la interfaz, permitiendo a sus diseñadores y programadores concentrar la mayor parte de su tiempo y esfuerzos en el desarrollo de la funcionalidad intrínseca de la aplicación.

El empleo de este componente en el diseño de las interfaces de usuario de todas las aplicaciones de una organización permite alcanzar un alto grado de consistencia entre los elementos constitutivos de las mismas, facilitando así el aprendizaje de su uso por parte de los usuarios. Además, la reutilización de recursos entre las distintas aplicaciones, así como la distribución de componentes interactivos entre todas ellas, permite amortizar los recursos empleados en el diseño de las técnicas de interacción de las interfaces, ya que éstos diseños pueden

ser compartidos por más de un proyecto. Los UIMS suelen estar diseñados específicamente para personal especializado en interacción y comunicación humana y no para programadores. Estas herramientas de apoyo al diseño de modelos de interacción producen mejores interfaces al facilitar la construcción de prototipos de forma rápida y eficaz, permitiendo además una modificación sencilla de las interfaces generadas, lo cual facilita su mantenimiento. Además, es posible crear varias versiones de una interfaz para una o varias aplicaciones, facilitando la reutilización del código y permitiendo la creación de interfaces consistentes, tanto interna como externamente.

Para ello, todo UIMS necesita un medio para que el diseñador pueda describir las interfaces deseadas y especificar sus características. La mayoría de los UIMS fundamentan este mecanismo de descripción en un lenguaje específico conocido como *lenguaje de descripción de interfaces de usuario* o UIDL (*User Interface Description Language*). Los UIDL pueden ser de muy variadas tipologías en base al enfoque adoptado en su diseño. Así, estos lenguajes pueden estar basados en el uso de elementos gráficos propios de la programación visual (implementados mediante lenguajes intermedios basados generalmente en gramáticas libres de contexto) o en diagramas de transición de estados, gestores de eventos, formas de Backus-Naur (BNF), etc. No obstante, casi todos estos lenguajes tienen en común el hecho de que suelen describir el comportamiento perceptible de la interfaz de usuario pero no su construcción. Por medio de esta descripción de alto nivel, el diseñador puede explorar y evaluar diversas alternativas para su interfaz sin necesidad de llegar a codificarlas. De este modo, es posible crear diversos prototipos de la interfaz de forma rápida, para luego evaluarlos a través de un proceso iterativo de refinamiento sucesivo que concluye con la elección de aquel prototipo que cumple con todos los requisitos de interacción identificados durante la fase de análisis del sistema. Obviamente, este prototipo ideal será el único que llegará a ser codificado realmente.

Para la construcción de los prototipos definidos mediante los UIDL, los UIMS suelen emplear directamente los *toolkits* proporcionados por el sistema operativo destino. Dichos *toolkits* consisten en bibliotecas de rutinas especializadas en la creación y gestión de todos los *widgets* disponibles en la una interfaz de usuario estándar de una determinada plataforma. Estos *widgets* pueden comprender objetos gráficos como botones, campos de texto, menús, etc. y objetos no visibles como traductores de texto, sistemas de lectura en voz alta y similares. Naturalmente, estos *toolkits* dependen exclusivamente de la plataforma destino y se pueden citar como ejemplo el *Toolbox* de la plataforma Macintosh o el *X11* de Motif.

El código generado por un UIMS suele ser más estructurado y modular que el creado por un programador haciendo uso directo de las rutinas incluidas en los *toolkits*, aunque naturalmente, el código puede resultar menos legible dado el uso de variables y objetos privados cuya funcionalidad puede no estar clara para el operador humano. Aún así, dado que el código es generado a partir de un lenguaje de programación de alto nivel, bastará con cambiar la especificación

incluida en un UIDL para modificar automáticamente la versión del código de bajo nivel basado en *toolkits*. Es por ello que el modelo de desarrollo basado en UIMS es más es más fiable, flexible y por lo tanto los productos generados con él son más fáciles de mantener. Gracias también al empleo de este lenguaje de alto nivel, el código obtenido resulta portátil, ya que el UIDL es independiente de una plataforma concreta. De este modo, a sus muchas virtudes, los UIMS añaden la de la portabilidad de las interfaces generadas, dado que a partir de una misma definición para una interfaz un sistema de gestión de interfaces de usuario es capaz de generar código para distintas plataformas.

1.1.2. Niveles de Diseño

Para facilitar la tarea de desarrollo de una interfaz de usuario, ésta suele ser diseñada empleando varios niveles de abstracción distintos, los cuales suelen estar reflejados en los procesos de usuario proporcionados por los UIMS. Esta abstracción facilita la tarea de desarrollo al dividir un problema complejo en varios problemas más pequeños, siguiendo la efectiva estrategia del *divide y vencerás*.

El diseño de una interfaz se suele dividir en un nivel conceptual, un nivel semántico, un nivel sintáctico y un nivel léxico. El nivel conceptual se encarga de describir los conceptos, objetos y demás entidades básicas que percibirá el usuario al emplear la interfaz, así como el conjunto de acciones que puede realizar sobre estas entidades. Se trata en suma de modelar el dominio del problema que la aplicación intenta resolver, pero en términos de relaciones conceptuales. El nivel semántico suele encargarse de describir las funciones realizadas por el sistema, así como de establecer los requisitos funcionales del mismo, describiendo los procesos de usuario involucrados, así como las condiciones necesarias para que éstos puedan ser invocados. Por su parte, el nivel sintáctico se suele emplear para describir las secuencias de entradas y salidas necesarias para invocar los procesos de usuario identificados en los niveles anteriores, así como la situación, configuración y apariencia de los objetos definidos en el nivel conceptual. Por último, el nivel léxico determina como se crearán y agruparán las acciones básicas realizadas con los dispositivos de interacción (ratón, joystick, teclado, etc.) para crear las entradas y salidas requeridas por el nivel sintáctico.

Así por ejemplo, si se desea diseñar una aplicación para la creación de ilustraciones vectoriales, la definición de las primitivas básicas de dibujo (elipses, rectángulos, líneas, etc.) quedaría encuadrada en el nivel conceptual de la interfaz. En este nivel se incluiría además el tipo de operaciones que se pueden realizar sobre estas primitivas básicas (mover, girar, ampliar, reducir, etc.). Una vez definido este modelo conceptual, en el nivel semántico se incluirían los procesos de usuario respectivos y la secuencia de tareas necesarias para llevarlos a cabo. Por ejemplo, para poder ejecutar el proceso de usuario *mover*, se podría establecer que la secuencia de operaciones necesarias debería pasar por *seleccionar* el objeto a mover, *señalar* la nueva posición para el objeto y *confirmar* la operación. Nótese como para la ejecución de estas tres

operaciones básicas es necesario definir además una serie de datos de entrada, los cuales se corresponden con la selección del objeto a mover, las coordenadas de la nueva posición y la confirmación o negación de la operación. En el nivel sintáctico se definirá la sintaxis –valga la redundancia– de los procesos de usuario definidos, determinando los comandos para señalar un objeto, especificar loa correspondientes datos numéricos para las nuevas coordenadas, indicar una respuesta afirmativa o negativa para una confirmación, etc. así como los demás tipos de operaciones básicas definidas en el modelo. Por último se definirá como serán empleados los dispositivos de entrada para introducir la información definida en el nivel sintáctico. Así por ejemplo, se puede establecer que la selección del objeto a mover en el proceso de usuario homónimo puede ser realizada por medio del ratón y que la confirmación o negación de esta operación puede ser efectuada mediante la emisión de un comando hablado.

Cuando un usuario se enfrenta por primera vez a una interfaz, tendrá que aprender los conceptos establecidos a lo largo de los cuatro niveles. Sin embargo, una vez que éstos son aprendidos, los niveles conceptual y semántico permanecen en la memoria del usuario estables a lo largo del tiempo, con lo que éste puede aprender a usar una aplicación similar con relativa rapidez, teniendo que estudiar tan solo la especificación sintáctica y léxica de esta nueva aplicación. Siguiendo con el ejemplo anterior, si un usuario experto en el uso de una aplicación para el diseño de gráficos vectoriales tiene que aprender a usar otra aplicación del mismo tipo, sólo tendrá que determinar cómo se utilizan las operaciones que ya sabía usar en la antigua aplicación con la sintaxis y el léxico de la nueva. Este usuario sabrá que existe un mecanismo para *mover* un objeto gráfico y aprenderá pronto a utilizarlo, quizás cambiando el modo en el que se especifican los datos en la nueva interfaz.

Llegado a este punto y antes de pasar a describir la arquitectura básica de todo UIMS, es necesario destacar que este tipo de sistemas suelen implementar tan solo los niveles sintáctico y léxico de una interfaz, dejando la responsabilidad de definir el nivel conceptual y semántico al diseñador de la aplicación.

1.1.3. El Modelo Seeheim

El *Modelo Seeheim* fue propuesto por primera vez en un *workshop* en Seeheim, Alemania, en 1983 [Green (1985a)] y su intención era la definición de la arquitectura que debería estructurar las relaciones entre los distintos niveles que forman parte del interfaz de usuario de una aplicación, es decir, la definición de los componentes o modelos que debían formar parte de todo UIMS (ver Figura 1).

La definición original del Modelo Seeheim hereda gran parte de la analogía empleada para describir los primitivos sistemas interactivos *conversacionales*, disponiendo de una capa de *Presentación*, encargada de la gestión de los aspectos léxicos de la interfaz, una capa de *Control de Diálogos* encargada de gestionar la sintaxis y una capa de *Interfaz de Aplicación* encargada de proporcionar acceso a la semántica de las

aplicaciones gestionadas por el UIMS, es decir, a la funcionalidad de dichas aplicaciones. Dada su posición estratégica en el conjunto del modelo, el *Control de Diálogos* se encargará además de gestionar la comunicación entre los objetos interactivos de la capa de *Presentación* y el código funcional de las aplicaciones (sus datos y procesos).

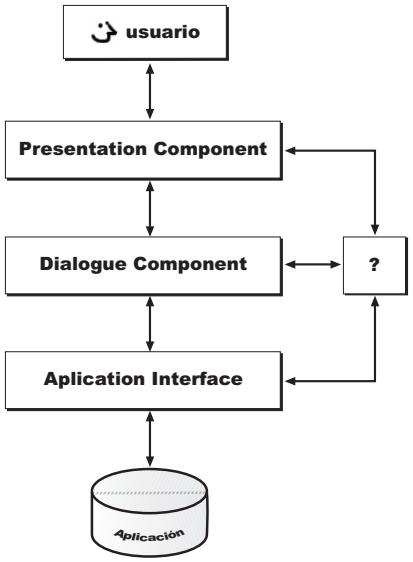


Figura 1. Modelo Seeheim para el diseño de sistemas de gestión de interfaces de usuario.

La arquitectura propuesta por el modelo Seeheim es completamente flexible ya que permite el diseño y construcción de diversas versiones de todas y cada una de las capas de las que está compuesta. De hecho, la sincronización y gestión del intercambio de datos y control entre cada una de las cajas es una interrogante manifiesta en el propio modelo. Desde su mismo inicio ha sido representada mediante una pequeña caja provista de un signo de interrogación (ver Figura 1). La simplicidad del modelo, así como las múltiples variantes del mismo que pueden surgir de una especificación tan flexible, amén de las muchas lagunas manifiestas en las distintas capas de las que está formado, han constituido las principales críticas de este modelo. En concreto, el Modelo Seeheim no define en ningún momento como se debe aplicar a un entorno de desarrollo ni su relación con las herramientas necesarias para implementarlo. La especificación original del Modelo Seeheim tampoco incluye ningún estudio de los requisitos de rendimiento necesarios para construir un sistema basado en él. Esta especificación tampoco proporciona ningún tipo de soporte acerca de cómo ha de realizarse la transmisión de contenidos semánticos entre su emisor original (la capa *de Interfaz de Aplicación*) y su receptor final (la capa de *Presentación*), así como la transmisión de información en sentido opuesto, es decir, como se gestionan las respuestas y/o peticiones del usuario para hacerlas llegar a las aplicaciones clientes del hipotético UIMS desarrollado en base a este modelo.

Otra de las críticas más importantes que ha recibido este modelo, es su incapacidad manifiesta para gestionar estilos de interacción en entornos multitarea en los que el intercambio de información entre las

dos capas extremas del modelo puede producirse de modo simultáneo. Es decir, en aquellos casos en donde los contenedores de información (propiedades) de una aplicación han de ser accedidos a la vez tanto por la propia aplicación como por el usuario a través de la capa de *Presentación*, de tal modo que cuando un cambio es realizado en un contenedor, todos los agentes implicados han de ser notificados. La solución a este problema puede ser la aplicación de una arquitectura basada en el Modelo Vista Controlador en la que un único modelo de información puede ser representado por medio de diferentes vistas, así como modificado por diferentes controladores [Krasner y Pope (1981)]. Como se verá en el capítulo 8 (*El Diálogo*), GADEA resuelve este problema por medio de la inspección automática por reflexión del código binario de sus aplicaciones, en búsqueda de contenedores de información compartidos, estableciendo a continuación un directorio de canalización de eventos encargado de distribuir el tráfico de los mismos entre las capas de *Presentación* y *de Interfaz de Aplicación* (o sus equivalentes en GADEA: DEVA y CodeX respectivamente).

1.1.4. El Modelo de Arco

Para mejorar las limitaciones del Modelo Seeheim, surgió el *Modelo de Arco*, también en el entorno de otro *workshop* en 1992 [ACM SIGCHI(1992)]. En lugar de analizar la funcionalidad de una aplicación para separar ésta de su interfaz como hace el Modelo Seeheim, el Modelo de Arco examina la naturaleza de los datos que son transmitidos entre la interfaz de usuario y el código funcional de la aplicación. En función de la tipología de estos datos, el modelo define tres componentes básicos y dos adaptadores (ver Figura 2).

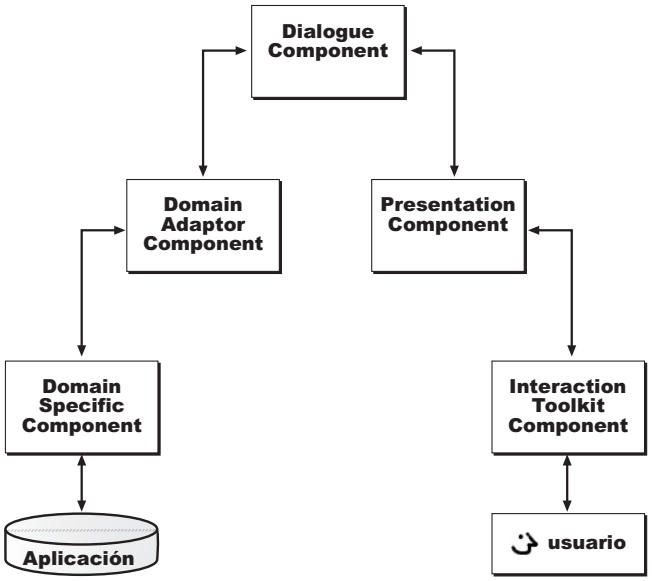


Figura 2. Modelo de Arco.

En los dos extremos del modelo se encuentran el *Interaction Toolkit Component*, encargado de gestionar la interacción directa con el usuario al nivel sintáctico y léxico y el *Domain-Specific Component* encargado de la gestión e intercambio de información con la aplicación. Entre estos

dos componentes se sitúa el *Dialogue Component*, quien se encarga de establecer la secuencia adecuada para la ejecución de las tareas reflejadas en el dominio del problema y de garantizar la consistencia necesaria para todos los diálogos interactivos generados por el UIMS.

Entre los tres componentes citados, se sitúan dos adaptadores que actúan como verdaderos interfaces entre dichos componentes. Por un lado, el *Domain-Adaptor Component*, situado entre el *Domain-Specific Component* y el *Dialogue Component*, se encarga de implementar las tareas dependientes del dominio del problema que son requeridas por usuario de la aplicación, es decir, los puntos de entrada a todos aquellos procesos de usuario del nivel conceptual y semántico. Este componente, que actúa como verdadero colchón entre la aplicación y el generador de diálogos interactivos (*Dialogue Component*) se encarga de arrancar aquellos procesos de usuario iniciados por la propia aplicación, reorganizar y clasificar la información proveniente del usuario y detectar y mostrar errores de tipo semántico. Por el otro lado, *el Presentation Component* situado entre el *Dialogue Component* y el *Interaction Toolkit Component* (ver Figura 2), se encarga de proporcionar al *Dialogue Component* todos aquellos *widgets* disponibles en el *toolkit* de la plataforma destino, así como de configurar la apariencia y funcionalidad de dichos *widgets*.

La flexibilidad impuesta a los adaptadores permite minimizar el efecto futuro de la evolución en la tecnología disponible para implementar el UIMS, dado que es posible desplazar la funcionalidad de estos adaptadores en un sentido u otro del arco en función de las necesidades del momento. Como se puede apreciar, este modelo es lo suficientemente flexible como para generar múltiples arquitectura distintas para un UIMS en función de los objetivos marcados en el diseño del mismo.

Aunque este modelo es suficientemente potente y flexible como para implementar cualquier tipo de UIMS basado en la creación de diálogos interactivos estáticos (es decir, en tiempo de compilación), el modelo se queda cojo cuando se le pretende usar en UIMS inteligentes (ver capítulo 2: *Interfaces Inteligentes*) destinados a la creación de diálogos interactivos dinámicos (es decir, en tiempo de ejecución). Esta deficiencia se debe a la ausencia total de mecanismos para almacenar o inferir algún tipo de conocimiento acerca del usuario real de las interfaces generadas por este modelo. Precisamente, para facilitar la creación de una arquitectura para UIMS capaz de proporcionar servicios propios de sistemas inteligentes adaptables pero manteniendo la eficacia del Modelo de Arco, surgió en 1993 el *Modelo de Arco para Interfaces Inteligentes Adaptables*.

1.1.5. El Modelo de Arco para Interfaces Inteligentes Adaptables

En este modelo, desarrollado por Hefley y Murray (1993), la parte inteligente y adaptable de la interfaz es definida como una instancia del *Domain-Adaptor Component* del Modelo de Arco original (ver Figura 3). Es en este punto del modelo donde el diseñador ha de incluir los mecanismos de actualización y obtención de información acerca de las

características del usuario, así como las estrategias que permitan una adaptación inteligente de la interfaz (ver capítulo 2: *Interfaces Inteligentes*)

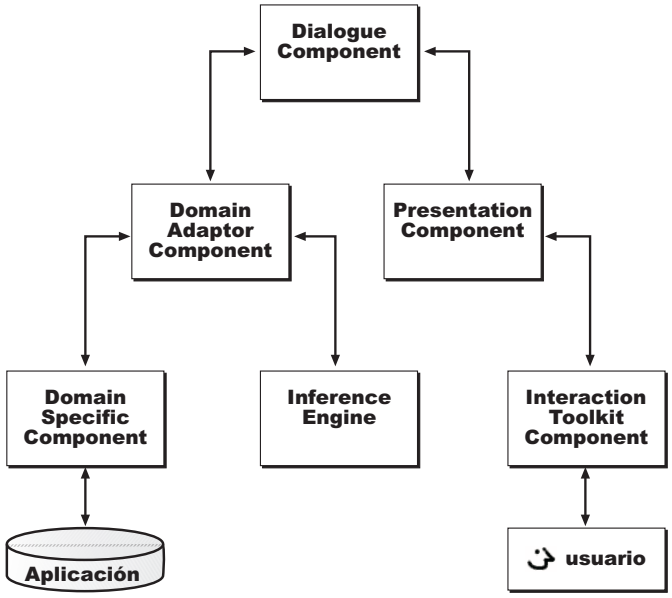


Figura 3. Modelo de Arco para Interfaces Inteligentes Adaptables.

El *Inference Engine* localizado en el *Domain-Adaptor Component* puede modificar el comportamiento del *Dialogue-Component* afectando el cambio a la secuencia en la que se desarrollan las tareas efectuadas por el usuario en función de las características de éste. A su vez, otras capas más cercanas a la interfaz de usuario como el *Presentation Component* y el *Interaction Toolkit Component* pueden verse influidas por las decisiones tomadas por este *Inference Engine* permitiendo la adaptación de la interfaz. A la vez, éstos componentes pueden servir como fuentes de información para mantener actualizada la imagen que se tiene del usuario, infiriendo por ejemplo el tipo de tareas que éste realiza frecuentemente, así como sus posibles intenciones en un contexto determinado.

1.2 UIMS en Acción

Una vez vistos los conceptos y modelos básicos que rigen el diseño y desarrollo de sistemas de gestión de interfaces de usuario, es necesario realizar un breve análisis de algunos de los ejemplos más significativos de este tipo de herramientas. En la siguiente relación se recogen las características más relevantes de los UIMS más novedosos, analizando la naturaleza de la solución aportada por los mismos al problema de la separación entre interfaz y funcionalidad de las aplicaciones, así como el mecanismo empleado para describir los modelos de interacción de sus interfaces.

1.2.1. ALPHA

Este sistema ha sido desarrollado por la compañía norteamericana LoneWolf (2001) y comprende un entorno de desarrollo y ejecución de aplicaciones multiplataforma que, a diferencia de otros sistemas, está orientado a ayudar a diseñadores con cierto grado de experiencia en el desarrollo de interfaces de usuario. El UIMS ALPHA está basado íntegramente en tecnologías orientadas a objetos y según sus autores, reduce el número de líneas de código que el programador debe generar en un rango que va del 40 al 90%, aunque estos valores deben tratarse con precaución ya que no se debe olvidar que se trata de una aplicación comercial. En el momento de escribir estas líneas, existen versiones de ALPHA para prácticamente cualquier versión de la plataforma Unix y derivados.

Este UIMS se basa fundamentalmente en el Modelo de Arco definiendo tres tipos de componentes o procesos independientes denominados *agentes* en su propia jerga. El *agente tecnológico* representa al *Interaction Toolkit Component* del Modelo de Arco, encargándose de la gestión de la pantalla, el ratón, el teclado, el micrófono, los altavoces y cualquier otro componente hardware del sistema. El *agente aplicación* equivale al *Domain-Specific Component* encargándose de la gestión del código funcional de la aplicación. Por último, el *agente diálogo* realiza las mismas funciones encomendadas al *Dialogue Component* del Modelo Arco, permitiendo definir la relaciones entre los diferentes agentes a través de un conjunto de dependencias funcionales. De este modo es posible escribir una aplicación de forma independiente de la interfaz mediante la definición de dependencias entre este agente y los agentes tecnológicos. El empleo de estos últimos agentes permite además el diseño de interfaces independientes de una tecnología concreta, puesto que basta con reemplazar un agente tecnológico por otro equivalente para que la interfaz pueda ejecutarse sobre una tecnología distinta.

En este sentido, los diseñadores de ALPHA ponen el ejemplo de la representación de la hora actual por medio de diferentes relojes, uno analógico y otro digital. En ALPHA se definiría un agente aplicación encargado únicamente de mantener actualizada la hora del día (nivel conceptual y semántico). Nótese como este agente realizará su trabajo de forma independiente del tipo de reloj que empleará el usuario para leer la hora. Serán dos agentes tecnológicos los encargados de mostrar la hora al usuario, uno empleando tecnología digital y otro usando tecnología analógica. Este modelo flexible permite además emplear una misma tecnología por medio de diferentes agentes tecnológicos. Así, podría existir un agente tecnológico para mostrar la hora en modo digital mediante un LED y otro para mostrarla en el mismo modo pero simulando una pantalla LCD. En todo caso, el agente diálogo empleará siempre un agente aplicación para actualizar la hora y un agente tecnológico para mostrarla. En el caso de un reloj analógico, el agente diálogo conectaría el valor de la propiedad *minutos* del *agente aplicación* con la manecilla de los minutos del *agente tecnológico*. En el caso del reloj digital, la conexión se haría con la etiqueta *minutos* del agente tecnológico. Como se puede observar, el *agente aplicación* desconoce por

completo como está implementado el nivel sintáctico y/o léxico de la aplicación.

De este modo, la interfaz de la aplicación puede ser transferida a otra plataforma sin demasiados problemas, substituyendo tan solo los componentes de bajo nivel por aquellos equivalentes de la plataforma destino. Esta técnica permite además que una misma aplicación pueda disponer de varias interfaces de usuario, modificando tan solo aquellos agentes diálogo implicados en el modelo de interacción de la aplicación. Con ello es posible disponer por ejemplo, de una interfaz básica para usuarios novatos y de otra interfaz avanzada para usuarios expertos, sin que ello represente un coste de producción excesivo.

1.2.2. CHARADE

CHARADE es el resultado de un proyecto Esprit que tiene como objetivo el desarrollo de un sistema integrado de apoyo a los equipos de rescate participantes en catástrofes medioambientales y su nombre proviene del acrónimo *Combining Human Assessment and Reasoning Aids for Decision making in environmental Emergencies* [Martí (1996)]. Aunque técnicamente CHARADE no constituye un UIMS, son muchos y muy importantes los elementos empleados en su modelo de interacción que podrían ser extrapolados para su utilización en el diseño de sistemas de gestión de interfaces de usuario. Es por ello que se ha considerado relevante la inclusión de CHARADE en esta relación.

CHARADE está basado en un diseño centrado en el usuario, realizado en términos de las tareas que éste puede realizar en el sistema que se pretende modelar y automatizar. La novedad de CHARADE radica en la construcción de un enfoque metodológico para el diseño, desarrollo y evaluación de los sistemas interactivos basado en un proceso estructurado de análisis y modelado de las tareas realizadas por el usuario. Este enfoque permite el desarrollo rápido de prototipos, reduciendo considerablemente los ciclos de diseño y evaluación. Como sucede con la estrategia de diseño adoptada en otros sistemas interactivos, este enfoque metodológico se sustenta a su vez la dualidad funcional (aplicación) e interactiva (interfaz).

El enfoque funcional concierne al soporte que todo sistema interactivo debe proporcionar a las tareas del usuario. En este sentido, CHARADE destaca la influencia que las tareas originales del sistema tiene sobre los nuevos artefactos desarrollados a partir de dicho sistema y como estos nuevos artefactos, o mejor dicho, la naturaleza de las tareas que se pueden realizar y que pueden modificar el comportamiento del usuario. El análisis de las tareas en el que se fundamenta CHARADE proporciona métodos para comprender la naturaleza de este proceso cíclico, analizar en detalle la naturaleza de las tareas realizadas por el usuario y definir los diferentes aspectos de la funcionalidad del sistema que permiten dar soporte a dichas tareas. Para ello, CHARADE representa las tareas del sistema mediante una notación gráfica en forma de grafo, cuyos nodos representan las tareas que se pueden ejecutar con el sistema. Con esta notación es posible restringir lógica y temporalmente cada de estas tareas, situándolas en

un contexto determinado. Esta notación también permite representar las relaciones entre tareas, tales como condiciones de partida, bucles, invocaciones controladas por eventos, etc. El grado de detalle con el que las tareas son representadas permite descomponer toda tarea en las subtareas que la componen. Esta notación permite recoger además toda la información disponible acerca de los requisitos no funcionales de las tareas (duración, frecuencia, reversibilidad, etc.) por medio de comentarios escritos en lenguaje natural.

La flexibilidad de esta representación permite incorporar la descripción de los objetos empleados por las tareas, así como las relaciones semánticas que existen entre ellos. También es posible incorporar la descripción de los roles de cada usuario, los cuales son representados como el conjunto de tareas asignados a cada uno de ellos. Tanto los roles como los objetos pueden ser descritos por medio de plantillas, las cuales contienen información acerca de las relaciones existentes entre roles, objetos y atributos, así como de las acciones que se pueden aplicar sobre cada objeto. Este enfoque del análisis de tareas deja la puerta abierta a nuevos métodos de análisis orientados a objetos y de la ingeniería del software.

Por su parte, el enfoque interactivo concierte a la apariencia visual de la interfaz y a la generación de diálogos interactivos. En CHARADE el proceso de diseño está basado en el paradigma evolutivo del refinamiento de prototipos [Redmon-Pyle y Moore (1995)], es decir, en la creación de un prototipo inicial que se va refinando tras sucesivas evaluaciones de usabilidad. En cualquier caso, la organización de la interfaz es diseñada para reflejar la estructura del diagrama de tareas descrito. Así, los diferentes procesos de usuario disponibles en la aplicación son codificados como entradas en los diferentes menús de opciones de la misma, mientras que para las áreas conceptuales en las que se realizan las tareas se diseñan amplios espacios visuales asignados a la porción de pantalla ocupada por la aplicación. En estas áreas visuales se incluyen todos los datos necesarios para realizar las tareas, mostrando solo la representación de aquellos objetos que son pertinentes para la ejecución de la tarea activa. La agrupación de los datos mostrados depende de las relaciones existentes entre las tareas a las que dichos datos pertenecen. Otros aspectos del diseño de la interfaz, como el grado de paralelismo entre las tareas o la posible interrupción de la ejecución de algunas de ellas, proviene de una traducción directa del modelo de tareas. Los mensajes proporcionados por el sistema como respuesta a una acción particular del usuario, también provienen de la propia estructura del modelo de tareas, el cual se utiliza además para poner en contexto dichos mensajes y adaptarlos a la semántica de la tarea concreta en la cual se generan.

Sin lugar a dudas, la gran novedad de CHARADE es la formalización del diseño de la funcionalidad de una aplicación mediante el uso de un diseño centrado en tareas, el cual se refleja directa y unívocamente sobre la interfaz de dicha aplicación.

1.2.3. CLIM

CLIM significa *Common Lisp Interface Manager* y ha representado uno de los sistemas de desarrollo de interfaces de usuario más importantes para Common Lisp y; junto a SmallTalk [Krasner y Pope (1981)] puede ser considerado como uno los paradigmas clásicos de UIMS de la segunda década de los noventa. Este UIMS hunde sus raíces en los UIMS HUMANOID y UIDE (ver apartado 2.2.5: *MASTERMIND (HUMANOID* y UIDE)). CLIM permite la gestión de toda la interacción gráfica de una aplicación, proporcionando soporte para diferentes tipos de gráficos, salida multimodal y procesamiento de comandos escritos sensibles al contexto. El alto grado de estructuración de los programas requerido para trabajar con CLIM permite trasladar dichos programas a otras plataformas sin mayores complicaciones. Esta estructuración se realiza mediante una jerarquía de ventanas y paneles en los que pueden ser introducidos los *widgets* [Mckay (1991)].

La gestión de procesos de usuario se realiza en CLIM mediante el comando, el cual es un objeto generado para gestionar una acción realizada por el usuario en un contexto dado y representa una acción del tipo de seleccionar una opción de un menú, operar sobre un objeto gráfico o pulsar una tecla, entre otras. Los comandos se definen por sus nombre y sus parámetros siendo almacenados en una tabla jerárquica para poder acceder a los mismos en el momento en el que el usuario desea invocarlos. Cuando un comando o una tabla de comandos es desactivada en un contexto específico, CLIM deshabita automáticamente la invocación de comandos desde la interfaz gráfica y su procesador de comandos deniega la introducción y reconocimiento de nuevos comandos escritos

Una de las novedades más interesantes planteadas en su día por CLIM es la flexibilidad con la que éste sistema compone el espacio visual de objetos en unos paneles y ventanas en donde el tamaño de los objetos no es fijo, sino que depende del número, posición y tamaño del resto de los objetos que comparten el mismo panel [Mckay y York (1993)]. Esta idea fue adoptada posteriormente por otros sistemas de gestión de interfaces de usuario entre los que destaca el de la plataforma Java [Eckel (2001)].

Sin embargo, el aspecto más relevante de este diseño flexible se encuentra en la selección en tiempo real del *widget* a emplear en función del espacio visual disponible, ya que en este sistema es posible especificar los componentes de un panel en función de su utilidad –en términos de interacción– dejando la responsabilidad de la selección final del *widget* al propio CLIM. Así por ejemplo, cuando sea necesario mostrar varias opciones al usuario para qué éste seleccione una y solo una, el programador puede optar por hacer el trabajo sucio de especificar directamente un conjunto de botones de radio o dejar esta tarea a CLIM, especificando tan solo que es necesario emplear una lista de selección única. En tiempo real, CLIM seleccionará un conjunto de botones de radio para la lista si esto es posible en función del espacio disponible, y si no, empleará en su lugar un menú desplegable u otro *widget* equivalente. No obstante, este proceso de asociación entre

componentes genéricos (lista de selección única) y componentes específicos (grupo de botones de radio o menú desplegable) puede ser controlado por los diseñadores de la aplicación especificando el orden de prioridad que se debe establecer para esta asociación. Así, en el ejemplo anterior, el programador puede indicar a CLIM que invierta el orden de selección, empleando el menú desplegable siempre que sea posible, dejando el grupo de botones de radio en el segundo nivel de prioridad para este proceso de asociación [Möller (1996)].

1.2.4. GUIPW

La etimología de GUIPW proviene del acrónimo *Generic User Interface Package for Windows*. Basado en el Modelo Seeheim, este UIMS emplaza su mecanismo de separación entre interfaz y funcionalidad en un modelo basado en un *toolkit* de componentes reutilizables y en un gestor de eventos que permite dirigir el tráfico de información entre el usuario y la aplicación.

GUIPW está formado por una biblioteca de objetos que se emplean para representar y manipular los elementos del *toolkit* proporcionado por Windows. Estos objetos no representan simples abstracciones de los elementos de la interfaz, ya que se trata de objetos que conectan el nivel conceptual y semántico de la aplicación con el nivel sintáctico y léxico representado por la interfaz. Dichos objetos son empleados para manipular los elementos de la interfaz gráfica. En este sentido, el programador nunca actúa directamente sobre la interfaz, pero permite a estos objetos que gestionen los detalles internos de la de la misma. De este modo, asociando diferentes objetos con el mismo tipo de elemento de la interfaz, es posible cambiar el comportamiento y funcionalidad de ésta última.

Para alcanzar este objetivo, la biblioteca de clases proporcionada por GUIPW hace un uso extensivo del mecanismo de invocación de métodos virtuales, modificando la funcionalidad de los métodos claves de un objeto en instancias heredadas de una clase. Estos métodos virtuales son definidos en los objetos básicos y se encargan de gestionar diferentes aspectos de la funcionalidad de la interfaz, tales como la validación de parámetros, la inclusión de los elementos de una lista de selección, la representación iconográfica de un objeto interactivo, etc. Siempre que sea necesario modificar la funcionalidad o añadir conocimiento proveniente de la aplicación, estos métodos virtuales son definidos de nuevo en los objetos descendientes, asociando dicho objeto con el elemento de la interfaz que se pretende gestionar. De este modo es posible gestionar el tratamiento de errores y el intercambio de información entre usuario y aplicación. Así por ejemplo, dado que la validación de errores está concentrada en el método *parameterOk* de todo elemento de interfaz capaz de recibir información del usuario, tan solo es necesario redefinir la funcionalidad de este método para realizar una verificación efectiva de la información proporcionada por el mencionado usuario. Cuando éste introduce un parámetro en el *widget* asociado al objeto, la función *parameterOk* verifica su validez. Todo parámetro inválido no será aceptado, informando de esta situación al

usuario e invitándole a volver a introducir información válida. Este mecanismo básico de separación es empleado además para indicar cuando un proceso de usuario o un determinado *widget* está disponible para su uso o no en función del estado interno de la aplicación. Para ello, GUIPW proporciona una función denominada *isAvail* que devuelve un valor de verdadero o falso, una vez invocada en tiempo real. Para emplearla, el programador de la aplicación deberá definir su funcionalidad en todos los objetos que pueden no estar disponibles en un momento determinando, especificando en dicha función la situación en la que el *widget* debe quedar fuera del entorno de interacción [Nicholls (1994)].

Este UIMS oculta la naturaleza dirigida por eventos del *toolkit* de Windows para gestionar los mensajes y eventos en un nivel de abstracción superior. Así, en lugar de gestionar los típicos eventos de ratón de esta plataforma (*mouseClicked*, *mouseEntered*, etc.), GUIPW trabaja con eventos del tipo de *objeto seleccionado*, *objeto modificado*, etc. Este enfoque permite reducir la complejidad del nivel léxico de las interfaces así desarrolladas. Aunque Windows proporciona un nivel de separación básica entre los dispositivos de entrada y de salida hardware, esta distinción es eliminada empleando un nivel de abstracción superior que permite ocultar el excesivo nivel de detalle de los eventos relacionados con estos dispositivos. Centrando las operaciones de GUIPW en este nivel de abstracción superior se facilita el diseño de la interfaz en términos de las técnicas de interacción requeridas en lugar de hacerlo en base a primitivas hardware. Para GUIPW, una técnica de interacción será un módulo software que emplea dispositivos de interacción físicos para permitir que los usuarios puedan proporcionar o recibir un determinado tipo de información.

GUIPW proporciona una serie de herramientas que permiten diseñar la interfaz de una forma controlada por medio de un mecanismo adaptable al grado de conocimiento y de destreza del diseñador. Estas herramientas, basadas en un sistema experto, restringen el tipo y la naturaleza de las opciones, comandos y procesos de usuario disponibles en las herramientas de diseño, en función de su usuario, reduciendo así la complejidad del conocimiento que los diseñadores han de manejar para diseñar una interfaz. Esta funcionalidad es apropiada sobre todo en aquellos casos en el que el entorno de diseño contiene componentes de propósito especial que rara vez son empleados en todas las interfaces diseñadas.

Para diseñar una interfaz empleando GUIPW, el programador ha de disecrear el prototipo deseado empleando el sistema experto referido, el cual le guiará en la selección de los componentes apropiados en función del modelo de interacción requerido. Una vez seleccionados estos componentes, el programador los incluirá en el diálogo interactivo correspondiente, situándolos sobre el espacio visual que constituirá el cuadro de diálogo, cuadro de alerta o ventana empleando para implementar el referido diálogo interactivo. Estos diálogos interactivos son almacenados estáticamente en ficheros, optimizando su empleo en tiempo real, pero dinamitando cualquier posibilidad de realizar algún tipo de adaptación en tiempo real. Aunque es posible activar y

desactivar los componentes que lo constituyen, esta estrategia no permite cambiar por ejemplo la distribución física de dichos componentes.

El siguiente paso previsto por el modelo de desarrollo de GUIPW es la definición de las estructuras de datos necesarias para dar soporte a la interacción. Estas estructuras se incluyen en la propia herramienta de desarrollo, en la que se especifican los requisitos de validación e inicialización de los objetos internos de la aplicación relacionados con los elementos de la interfaz, generándose los métodos virtuales en caso necesario. Esta misma herramienta es empleada para generar el código necesario para recrear el menú de opciones desde el cual se invocará –como procesos de usuario– a todos y a cada uno de los diálogos interactivos generados, estableciéndose además el enlace entre los elementos de la interfaz y los objetos de la aplicación, así como la ruta que seguirán los datos introducidos por el usuario desde los primeros elementos hasta los últimos.

1.2.5. KLIDE

Este sistema propuesto por Xing y Li (1992), atiende a las siglas de *Knowledge-based Layered Interface Development Environment* y es un prototipo operativo que implementa un modelo abierto para el diseño de dispositivos basados en técnicas interacción y comunicación humana. Este modelo abierto recompone la estructura del modelo Seeheim creando cinco capas diseñadas especialmente para estructurar el conocimiento existente sobre el dominio del problema a través de un medio estándar que facilita el intercambio de información y del flujo de control entre las capas.

La primera capa es la de más bajo nivel y se responsabiliza de gestionar los dispositivos de entrada (ratón, teclado, joystick, etc.) con el objetivo de proporcionar un medio estándar para reconocer las tareas que el usuario realiza sobre estos dispositivos. Esta información es enviada a la siguiente capa, la cual se encarga de clasificar las tareas realizadas por el usuario de acuerdo con sus posibles intenciones, agrupando estas tareas en conceptos del tipo de *abrir documento*, *imprimir imagen*, etc. Nótese como estos conceptos pueden ser invocados por el usuario mediante diferentes acciones realizadas con los dispositivos de entrada (con el ratón, tecleando el comando, etc.). Una vez extraídos los conceptos a partir de las acciones, la siguiente capa se encargará de agrupar estos conceptos en secuencias con el objetivo de determinar la tarea global que el usuario está intentando realizar, ejecutando un análisis semántico de esta secuencia de conceptos. En la cuarta capa, la intención del usuario es puesta en el contexto del dominio de la aplicación alcanzando así un conocimiento completo sobre la intención del usuario. Una vez determinado el contexto y la intención del usuario, la última capa selecciona el modo y estilo de interacción más adecuado para alcanzar el objetivo buscado [Li et al (1992)].

A diferencia de otros UIMS en los que la función que implementa un determinado proceso de usuario está estrechamente ligada con el *widget*

que le representa en la interfaz (un menú, por ejemplo), en KLIDE, las funciones son invocadas en base a conceptos. De este modo, una función será invocada solo si se ha podido determinar y comprender la secuencia de conceptos que implican su activación. Para poder determinar dichos conceptos, es necesario retroceder al momento en el que el usuario define éstos, todo ello a través de un análisis de las actividades que éste realiza sobre la interfaz empleando los dispositivos básicos de interacción (Li et al (1995)].

Basado en estas cinco capas e implementado en Common Lisp para Unix, KLIDE está formado por dos componentes principales: una interfaz para Unix y un editor que permiten modificar la interfaz por medio de cambios en la base de conocimiento. Mientras que la interfaz integra las cinco capas mencionadas, el editor proporciona un entorno de trabajo para editar el aspecto exterior de la interfaz.

1.2.6. X-CHART

X-CHART es un entorno de desarrollo de interfaces de usuario compuesto por un conjunto de herramientas que permiten editar y definir las especificaciones de una interfaz, así como generar el código que la implementa. Parte de estas herramientas proporcionan soporte en tiempo real al código generado a partir de dichas especificaciones, empleando para ello un interprete. Este UIMS se basa en un lenguaje visual, el cual es empleado para describir los componentes software que controlan la sintaxis de los diálogos interactivos establecidos entre la aplicación y sus usuarios. Por medio de este lenguaje, X-CHART puede gestionar varios diálogos interactivos que se ejecutan concurrentemente, disponiendo de capacidad potencial para que dichos diálogos puedan ser distribuidos en entornos cooperativos.

El lenguaje empleado por X-CHART para definir los diferentes aspectos de la interfaz está basado en diversas mejoras de los diagramas de transición de estados, mejoras que permiten especificar múltiples clientes de una interfaz, los cuales se pueden crear y distribuir dinámicamente con la posibilidad de compartir memoria y señales en entornos distribuidos (aunque ésta funcionalidad aún no está implementada). A diferencia de otros sistemas, en X-CHART, los eventos son instancias de una clase (en lugar de tratarse de primitivas de datos). Estas clases son organizadas en una jerarquía que permite clasificar los tipos de eventos generados por un objeto. Gracias a estas clases, es posible añadir información a una instancia de clase, de tal modo que cada vez que un evento es visible, su información asociada también lo es. Este mecanismo es empleado por X-CHART para intercambiar información entre instancias de una clase cuando se produce un evento. Esta técnica, de notable éxito, ha sido adoptado por los *toolkits* de otras plataformas de desarrollo de interfaces, como es el caso de la AWT (*Abstract Window Toolkit*) de Java [Eckel (2001)].

Los eventos referidos pueden combinarse con variables centinela (propiedades de un objeto que esperan su actualización con un determinado valor) para disparar automáticamente la ejecución de una determinada acción creando así un activador. Los activadores y sus

acciones asociadas pueden ser empleados como etiquetas para las transiciones entre estados o para realizar transiciones dentro de un mismo estado (bucles). En todo caso, las variables centinela pueden asociarse con valores temporales para la ejecución de determinados procesos de usuario en momentos específicos del ciclo de vida de la aplicación. Para evitar los posibles modelos de interacción no deterministas a los que podría conducir el uso de este tipo de variables, X-CHART permite asignar valores de prioridad a cada una de las relaciones que constituyen un modelo de interacción. Sin embargo, si las prioridades son asignadas de forma correcta, la activación simultánea de varias transiciones acarreará un comportamiento no determinista para la interfaz [De Lucena y Liesenberg (1995)].

El empleo de un UIDL basado en diagramas de estado permite el uso de este UIMS en sistemas interactivos en donde el usuario debe ejecutar varias tareas concurrentemente o en donde el UIMS tiene que gestionar múltiples representaciones de la misma información. Basta con diseñar diferentes autómatas para definir el comportamiento de cada uno de los modelos de interacción a emplear y representarlos mediante su correspondiente diagrama de estado. De este modo, la respuesta del sistema puede adecuarse al estado actual en el que se encuentra la aplicación. El empleo de variables centinela permite una interacción continua con el usuario, mientras la aplicación subyacente se ejecuta en paralelo, proporcionando los datos requeridos por el mencionado proceso de interacción. Así por ejemplo, una aplicación de control de un proceso continuo podrá mostrar al usuario los parámetros bajo control en todo momento, incluso tras la modificación de estos valores, bien por parte del propio proceso o bien por medio de una modificación explícita de los valores de control por parte del usuario.

Los sistemas interactivos en X-CHART son definidos en términos de una serie de pequeños componentes denominados *clientes*. Un cliente es la primitiva básica de la construcción de una interfaz en este UIMS y constituye un objeto dotado de un determinado *comportamiento*. Este comportamiento es definido en términos del UIDL de X-CHART, aunque también es posible definir parte de la funcionalidad del cliente por medio de este lenguaje, siempre y cuando la mencionada funcionalidad se encuentre dirigida por eventos. En todo caso, los clientes implementan la funcionalidad asociada a la parte de la aplicación que representan en la interfaz por medio de objetos incluidos en la aplicación. Para este UIMS, una interfaz está compuesta por una colección de clientes y cada uno de ellos gestiona el comportamiento de uno o varios objetos interactivos. La interacción entre clientes se define por medio del UIDL, siendo posible la comunicación y sincronización entre clientes en tiempo real. El objetivo básico de estos clientes es el de detectar las peticiones de información del usuario y activar los respectivos objetos encargados de su confección en el lado de la aplicación, operación que se realiza mediante llamadas a funciones, las cuales pueden ser codificadas en lenguaje C o en C++. Por medio de esta estrategia, el comportamiento de una interfaz se define en un cliente, mientras que la semántica de dicho comportamiento (la funcionalidad) queda aislada por medio de las rutinas internas de los objetos interactivos de la aplicación [De Lucena y Liesenberg (1995a)].

2

2 Interfaces Inteligentes

Si bien hoy en día es el usuario el que hace que una interfaz sea inteligente, en el futuro será la interfaz la que hará inteligente al usuario

William Mark

2.1 Características de una Interfaz Inteligente

El desarrollo inicial de los estudios sobre interfaces inteligentes estuvo dominado en los primeros años por algoritmos capaces de codificar y decodificar el lenguaje natural –con mayor o menor grado de éxito– fundamentando el modelo de interacción en la metáfora del teletipo. Hay que recordar que en los inicios de la informática, la interacción con las aplicaciones se realizaba en diferido, introduciendo los datos que alimentaban los procesos de usuario por medio de un teclado o por bien medio de una serie de tarjetas perforadas, obteniendo la respuesta escrita en una cinta de papel, como si de un teletipo se tratase. De acuerdo con esta metáfora, el usuario realiza preguntas al ordenador empleando el lenguaje natural y éste le responde empleado el mismo lenguaje. De este modo, el usuario humano establecía una comunicación directa con un agente software interactivo, el cual imita el comportamiento humano escondiendo su lógica interna tras la

fachada del lenguaje natural empleado para confeccionar la respuesta. Esto es en esencia el paradigma de la inteligencia artificial en donde se determina el grado de inteligencia de un agente software en base a la calidad de las repuestas confeccionadas en lenguaje natural. De acuerdo con este test, un agente será inteligente si logra engañar con sus respuestas a su contertulio humano.

Con la llegada de las interfaces gráficas y el desarrollo de estilos de interacción basados en nuevas metáforas, como es el caso de la metáfora del escritorio (empleada masivamente a partir de la década de los ochenta) el campo de aplicación de los agentes inteligentes se ha desplazado desde el análisis del lenguaje natural hacia la interpretación de las distintas acciones realizadas por un usuario dentro del ámbito de las nuevas interfaces. Todo ello condujo a una simplificación de la lógica interna de los agentes, a reducir su tamaño (haciéndolos portátiles) y a aumentar su potencialidad, ya que resulta mucho más sencillo determinar el significado de las acciones realizadas por un usuario sobre una colección de objetos gráficos que interpretar el significado de una serie de órdenes y comandos expresados en lenguaje natural. Ello se debe a que toda interfaz gráfica suele construirse modelando el dominio del problema, proporcionando herramientas que permiten modificar dicho modelo y situándolo en un contexto concreto del cual los agentes pueden inferir conclusiones válidas a partir de las acciones realizadas por el usuario.

2.1.1. Semántica *versus* Sintaxis

El problema al que se enfrentan los diseñadores de las interfaces inteligentes actuales sigue siendo el mismo que aquel al que se enfrentaban los diseñadores de las interfaces inteligentes basadas en el lenguaje natural. Este problema no es otro que analizar la eficacia con la que una interfaz o un determinado modelo de interacción se adecua al dominio del problema, y para establecer la carga semántica adecuada para el conjunto de tareas que se pueden realizar sobre el mismo, así como determinar el grado de adecuación de dicha interfaz y del modelo de interacción al conocimiento que el usuario tiene del problema.

Este análisis está basado fundamentalmente en el aspecto semántico de la información manejada, tanto por el diseñador de la aplicación en su modelo de interacción, como por el aspecto semántico de la información percibida por el usuario cuando éste se enfrenta a un determinado modelo de interacción. Es precisamente en este aspecto en donde destacan las interfaces inteligentes, al poder extraer conocimiento acerca de las acciones realizadas por el usuario, así como del contexto en que dichas acciones son realizadas, examinando para ello la carga semántica de la colección de objetos manipulada por el usuario. De este modo, cada interfaz inteligente o cada agente interactivo puede efectuar por su cuenta el proceso de análisis mencionado, reaccionando y adaptando la representación del dominio del problema en un proceso de evolución constante y continuo.

Sin embargo, incluso si el proceso de análisis de la semántica de la información manejada se realiza de forma adecuada, es posible que la

interfaz inteligente no pase de resultados mediocres si no se toma en cuenta su aspecto sintáctico, es decir, la forma en la que la información es representada. Aún cuando un agente inteligente sea capaz de inferir de forma automática el tipo de acción requerida por un usuario y obtener la respuesta correcta para el problema a resolver, ésta puede ser del todo inútil para el usuario si esta respuesta no le es comunicada de forma adecuada.

Para confeccionar una representación sintáctica adecuada, el agente inteligente debería tener en cuenta los aspectos cognitivos, perceptivos e incluso motrices del usuario concreto al que está dirigida la respuesta. Para ello, una interfaz inteligente deberá aumentar las habilidades innatas del usuario a la vez que se intentan reducir sus limitaciones físicas, analizando los puntos fuertes y débiles de la comunicación entre los humanos y los ordenadores conduciendo la tecnología a la superación de los puntos débiles.

2.1.2. Comunicación Multimodal

Toda estrategia diseñada para reducir las consabidas limitaciones humanas y alcanzar un aumento considerable en el uso potencial de cualquier herramienta informática debería pasar por la creación de un amplio abanico de canales de comunicación a través de los cuales el humano y la máquina pueden intercambiar información. Mediante el empleo de mecanismos de interacción en diversas modalidades de comunicación, las limitaciones de un determinado usuario con respecto a un canal de comunicación específico pueden ser superadas por los benéficos obtenidos del empleo de canales alternativos. Así por ejemplo, la limitación que tiene un canal de comunicación visual para un usuario ciego es superada ampliamente por el uso de un canal de comunicación auditivo, a través del cual se transmite –en la medida de lo posible– toda la información incluida en el canal de comunicación visual.

El uso de varios métodos de comunicación dentro de una misma interfaz es conocido como comunicación multimodal. Esta estrategia no se limita al aprovechamiento de las características físicas del medio de comunicación, como ocurría en el ejemplo anterior sino que se extiende a todo el proceso de intercambio de información entre dos entes, incluyendo el lenguaje o protocolo necesario para codificar y decodificar los datos transmitidos. Así, otros ejemplos de comunicación multimodal incluyen el uso del lenguaje natural (a través del habla o por medio del uso de un teclado), el uso de voz generada por ordenador o la habilidad de una determinada interfaz para permitir que el usuario pulse una determinada área de la pantalla y emita un comando hablado del tipo de *dibuja un triángulo aquí* o *elimíname este objeto*.

El empleo de sistemas de comunicación multimodal proporciona mayor libertad de acción al usuario al no restringir sus movimientos a un esquema prefijado, además de permitir un uso más intuitivo de la interfaz al proporcionar soporte para el empleo de diversas metáforas y modelos mentales. Estos sistemas se muestran especialmente eficaces en

situaciones en las que un usuario concreto no puede emplear determinados organos sensoriales y motores en la ejecución de una determinada tarea, situación que no solo se presenta en el caso de usuarios con determinadas discapacidades –tal y como se ilustró en el ejemplo anterior– sino que también es habitual en determinados entornos de trabajo. Así por ejemplo, en un entorno de vigilancia (para más información véase apartado 14.3: *Atención Continuada*), como es el caso de una estación de radar o del módulo de control de procesos de un tren de galvanizado, es muy posible que el usuario tenga que mantener un estrecho contacto visual con un determinado objeto, como es el caso la posición de un avión en un radar o de la situación de un objeto en el tren de galvanizado y que no pueda prestar atención a determinados mensajes transmitidos por medio de dicho canal, de tal modo que el empleo de un canal de comunicación visual resulta poco efectivo. En su lugar, la interfaz podría leer la respuesta obtenida por el sistema en voz alta, de tal modo que el usuario pueda examinar dicha respuesta sin necesidad de poner atención a la salida visual proporcionada por la aplicación.

En todo caso el agente inteligente podrá decidir cual o cuales de los medios de comunicación deberán ser empleados en función de la situación, decidiendo cuales estarán disponibles, así como los distintos parámetros que formarán parte de su configuración, favoreciendo de este modo el empleo de una comunicación eficaz entre el usuario y la aplicación.

2.1.3. Adaptabilidad

En función del tipo de usuario al que ofrece sus servicios, un agente interactivo inteligente puede adaptar no solo el tipo de canales de comunicación a emplear, sino también la forma en la que la información es percibida por el usuario. En este proceso de adaptación interviene tanto el conocimiento que el agente dispone acerca de las características del usuario como el contexto en que el agente actúa.

Debido a la propia diversidad de la especie humana, cada individuo de una colectividad presenta características únicas a nivel cognitivo, perceptivo y motriz las cuales condicionan la percepción de los objetos que le rodean. En base a esta diversidad, cada usuario de un sistema percibe los componentes con los que se relaciona de modo distinto, por lo que una representación única de la información puede no ser aprovechada con el mismo grado de efectividad por todos los usuarios de un sistema. Para garantizar un máximo de efectividad, la información debería mostrarse empleando diferentes métodos en función del usuario destino. Para ello, los agentes interactivos inteligentes deben ser capaces de examinar las características cognitivas, perceptivas y motrices del colectivo de usuarios al que sirven, permitiendo la generación de información adaptada a las necesidades concretas de cada uno de los usuarios.

El proceso de adaptación de la información regula la cantidad de información a mostrar en todo momento, permitiendo que el usuario no sea sobrecargado con demasiada información ni tampoco sea

desinformado ante la ausencia de información relevante para sus necesidades. Un ejemplo de este tipo de interfaces podría ser el sistema de información de un gran aeropuerto en donde la información acerca de los vuelos es clasificada, estructurada y refinada en función del tipo de usuario. Así, los pasajeros reciben la información que necesitan para acceder a las puertas de embarque, ignorando por completo toda información acerca de aspectos como el peso de la carga que llevará su avión y la cantidad de combustible necesaria para realizar el vuelo, información que será mostrada exclusivamente a los operarios de mantenimiento y a los encargados del balancear la carga en las bodegas del avión. Del mismo modo, toda la información referente al estado de las condiciones atmosféricas, así como la pista a emplear y su tipo serán mostradas exclusivamente a los pilotos y controladores aéreos.

El sistema empleado en el ejemplo del aeropuerto podría considerarse adaptable si los distintos diálogos interactivos de su interfaz no fuesen generados de forma estática ni fuesen asignados a un punto de información concreto, dado que en un sistema adaptable inteligente es el propio agente interactivo quien decide –en tiempo de ejecución– que información debe ser mostrada en función del usuario individual que la solicita. En el ejemplo descrito pueden existir pasajeros con inquietudes acerca del tipo de avión en el que viajarán o de las condiciones atmosféricas en las que se desarrollará su vuelo y a los que un agente interactivo debería satisfacer, mostrando este tipo de información de forma automática.

Aunque la mayoría de las interfaces inteligentes concentran su poder de adaptación en los niveles conceptual y semántico –decidiendo qué contenidos han de ser mostrados y en que momento– el concepto de adaptación se puede extender además a los niveles léxico y sintáctico de una aplicación, determinando de forma automática la apariencia que debe mostrar la información para que ésta alcance el mayor grado de accesibilidad posible ante un usuario determinado. En el ejemplo anterior, una interfaz adaptable a nivel sintáctico sería capaz de variar aspectos como el tamaño de las fuentes empleadas, su color y disposición en función del tipo de usuario destinatario de la información, además de presentar la información de un modo inteligente, desplegando ésta en aquel formato que garantice el mayor grado de entendimiento posible. Así, para mostrar información acerca de los vuelos en curso, un agente inteligente seleccionaría una tabla, mientras que para mostrar la evolución en la velocidad del viento sobre las pistas de aterrizaje el mismo agente debería ser capaz de seleccionar un gráfico de barras o de líneas.

En este nivel de adaptación léxico-sintáctico, los algoritmos empleados deberían ser capaces de reflejar también las preferencias del usuario sobre un tipo determinado de interfaz o de adaptar los modelos de interacción para mejorar este proceso en función de las características motrices del usuario, ayudándole en sus acciones sin que éste lo sepa. Así por ejemplo, en algunos sistemas como CUBRICON (ver apartado 2.2.4), cuando el usuario pulsa cerca de un botón pero no sobre él, el sistema puede expandir el área de interacción del botón para aceptar la pulsación como válida. En este sentido, el proceso de

adaptación incluye un análisis de las intenciones, habilidades y capacidades del usuario, de una forma inteligente, dinámica y automática generando así diálogos interactivos adaptados mediante los cuales éste pueda alcanzar el mayor nivel de productividad y de satisfacción posible.

Al igual que ocurre con la comunicación multimodal, si el proceso de adaptación se realiza de la forma adecuada, esta característica de los agentes interactivos permitirá que la información transmitida por una aplicación pueda llegar al mayor número de usuarios posible con el mayor grado de efectividad disponible, siendo éste el talón de Aquiles de la mayoría de las interfaces inteligentes.

2.1.4. Modelo de Usuario

Como acabamos de ver, para llevar a cabo un proceso de adaptación eficaz es necesario que el agente interactivo tenga conocimiento de aquellas características del usuario que lo individualizan dentro de una colectividad. Al compendio de estas características se le conoce como modelo de usuario y representa uno de los componentes más importantes de una interfaz inteligente, ya que sin este componente, el agente interactivo no tendría ninguna base para diseñar interfaces individualizados. Por medio de la información almacenada en este modelo, el agente podrá determinar como presentar la información, que tipo de sistema de ayuda debe ser empleado y como se han de definir los modelos de interacción con el usuario.

El modelo de usuario puede implementarse por medio de una tabla de una base de datos en donde cada entrada representa a un usuario, pudiendo ser la información almacenada de naturaleza muy diversa. Es precisamente el tipo de información almacenada en el modelo de usuario, lo que permite caracterizar este tipo de componentes. Así, existen modelos de usuario especializados en obtener información del usuario (empleados para decidir como realizar las preguntas y el lenguaje a emplear), modelos para proporcionar ayuda y consejos (usados para evaluar la relevancia que la ayuda pueda tener en un contexto dado o para detectar mal interpretaciones por parte del usuario), modelos para analizar el comportamiento del usuario (empleados para identificar los objetivos del usuario y los planes trazados por el mismo para alcanzarlos), así como modelos especializados en proporcionar información al usuario (empleados para decidir qué mostrar y cómo mostrarlo). En todos los casos, el modelo de usuario estará compuesto por una colección de hechos conocidos o inferidos acerca de alguna característica del usuario. En función del sistema concreto, estos hechos podrán incluir las preferencias del usuario, su grado de familiaridad con las aplicaciones informáticas, el entorno cultural en el que éste se mueve, etc.).

2.2 Interfaces Inteligentes en Acción

Una vez definidas las características básicas de una interfaz inteligente e identificados los componentes básicos de una agente interactivo de este tipo, es necesario realizar un breve pero detallado repaso por todos aquellos sistemas y aplicaciones que representan el estado actual de la tecnología en este apartado al momento de escribir estas líneas. Dado que sería imposible citar todas aquellas interfaces inteligentes con el grado de detalle ideal, se ha optado por hacer una selección de aquellas más significativas dedicándoles mayor espacio en esta memoria.

En las siguientes páginas se incluyen referencias a algunas de las aplicaciones más relevantes en este campo, indicando sus características más representativas, así como sus carencias y ámbito de aplicación, destacando especialmente los aspectos técnicos e innovadores de las mismas que más se aproximan al campo de estudio de esta investigación, especialmente, las técnicas empleadas para separar la interfaz del la funcionalidad del código, el mecanismo de toma de decisiones por parte de los agentes inteligentes, el grado de adaptación alcanzado por las interfaces (siempre a un nivel léxico y sintáctico) y los medios empleados para mantener actualizado su modelo de usuario.

Es necesario destacar que parte de estas aplicaciones son prototipos que nunca pasaron de este estado, pero que sin embargo representaron un hito en este campo por la innovación aportada en alguno de sus componentes. También es necesario poner sobre el tapete el hecho de que la mayoría de estas aplicaciones se han desarrollado dentro del ámbito académico, siendo muy pocas las que han alcanzado amplia difusión en el mundo comercial y cuando algunas lo han hecho, lo hicieron generalmente de la mano de grandes instituciones estatales para usos privados, en general muy restrictivos.

2.2.1. APEX y GRIDS

Este par de sistemas constituyen en realidad los prototipos de pruebas de una modificación parcial del Modelo Seeheim para la creación dinámica de interfaces de usuario en función del contexto. En concreto, APEX es una aplicación para el diseño de secuencias de imágenes tridimensionales mientras que GRIDS es una aplicación para crear diálogos capaces de contener texto e imágenes.

Esta arquitectura, propuesta por Feiner (1991) intenta eliminar los problemas generados en la creación de interfaces de usuario de modo estático. Este problema es derivado del hecho de que una colección estáticas de pantallas y diálogos prediseñados no es suficiente para copar con el tipo de información a ser desplegada por una interfaz en un momento dado ni tampoco para garantizar que esa información se adapte a las necesidades de todos los usuarios. Incluso si la interfaz fuese usada por un único usuario, sus necesidades podrían cambiar a lo largo de una o más sesiones interactivas y lo mismo ocurre con la información requerida o introducida a lo largo de dichas sesiones. La

idea es aumentar la calidad de la interfaz generada, creando diálogos interactivos adaptados a las necesidades de un usuario dado y en un momento muy concreto de la computación.

Aunque esta arquitectura se basa en el Modelo Seeheim, ésta introduce cambios y aportaciones novedosas que facilitan la creación automática y dinámica de los diálogos interactivos. La cadena de producción de interfaces de usuario dinámicas es iniciada por el módulo encargado de definir el conjunto de objetivos que la interfaz debe conseguir. Estos objetivos son transferidos a un agente diseñador de diálogos, el cual planifica como comunicar la información aportada por la aplicación al usuario y obtener de éste los datos requeridos para ello, teniendo en cuenta el cumplimiento de los objetivos previstos. Una vez definidas las partes que ha de tener el diálogo interactivo, éstas son presentadas a un agente coordinador, el cual supervisa la construcción, delegando la tarea en una serie de agentes expertos en el uso de diferentes tipos de *widgets*. Cada uno de estos expertos determinará como se representará la información empleando textos, gráficos, sonidos, vídeos, animaciones, etc. Una vez definida la apariencia de cada una de las piezas que formarán parte del diálogo, éstas son distribuidas en el espacio visual por agente diseñador el cual determinará además el tamaño y posición del material generado por los agentes expertos. Esta parte del sistema se basa en técnicas que hunden sus raices en el diseño gráfico y en la ingeniería de factores humanos, las cuales permiten determinar el tamaño y posición de cada elemento en base al tamaño y posición de los demás [Zhou y Feiner (1997)].

Las decisiones tomadas por los agentes expertos dependen en parte del tipo de dispositivo empleado para visualizar los diseños y en parte del espacio disponible para éstos. En el caso de una imagen por ejemplo, estos factores determinarán el tamaño de la misma y por lo tanto su legibilidad. Así por ejemplo, una imagen que representa satisfactoriamente a un objeto a una determinada escala y distancia, puede resultar en un imagen completamente inútil si esa escala es reducida considerablemente. En estos casos, la imagen reducida puede ser reemplazada por otro tipo de información (texto, por ejemplo) [Zhou y Feiner (1998)].

La información fluye a través de todos los componentes de manera continua a lo largo de toda la sesión de trabajo. Sin embargo, cuando el sistema arranca por primera vez es necesario realizar un proceso de sintonización del mismo en el que se procede a generar el conjunto de reglas de diseño de la interfaz que se emplearán posteriormente para la creación dinámica de los diversos diálogos. Las reglas así generadas se basan en el conocimiento que se tiene acerca del usuario, el dominio del problema que modela la aplicación y el tipo de información a representar. Es por ello que esta arquitectura debe tener acceso explícito al conocimiento manejado por la aplicación, así como a las características que definen a sus usuarios y las circunstancias en las que la información ha de ser representada. El gran problema que presenta este enfoque es que las reglas generadas dependen de la existencia de una serie de estereotipos acerca del usuario final de la aplicación y los valores que configuran estos estereotipos pueden no ser los mismos que

aquellos que definen al usuario final, por lo que el nivel de adaptación obtenido mediante esta técnica puede ser realmente pobre.

2.2.2. AUI – GUIDE

AUI es un UIMS adaptable que responde a las iniciales de *Adaptable User Interface*. Este sencillo prototipo de UIMS permite al usuario emplear varios tipos de diálogos interactivos y poder cambiar de un modo de interacción a otro en cualquier momento, incluso durante la ejecución de un proceso de usuario. Así por ejemplo, el usuario puede pasar de un modo de interacción puramente GUI (*Graphic User Interface*), basado en el empleo de menús desplegables, a un modo de interacción CLI (*Command Line Interface*) basado en la introducción de comandos en lenguaje natural. Este UIMS ha sido empleado con éxito en el diseño de un editor de grafos dirigidos, una pequeña aplicación para la especificación de diagramas de entidad-relación y en un editor de diagramas de transición de estados, el cual es empleado por el propio AUI para definir el modelo de interacción de sus aplicaciones.

El objetivo buscado con esta interfaz multimodal es permitir que diferentes tipos de usuarios (novatos y expertos) puedan sacar el máximo rendimiento posible de cada uno de los modos de interacción disponibles. Este rendimiento viene determinado por el grado de adecuación de un modo de interacción concreto al tipo de información requerida por un proceso de usuario. De este modo, todo usuario dispone de diferentes niveles de aprovechamiento con respecto a los diferentes conceptos incluidos en la interfaz de usuario. Así, el usuario experto podrá emplear el teclado para introducir comandos a su velocidad habitual de tecleado (rápida, es de suponer debida a su condición de experto), empleando el modo de interacción basado en menús solo cuando las características de la aplicación (objetos diseñados exclusivamente para ser manipulados directamente) o cuando las preferencias del usuario así lo especifiquen.

Para diseñar las interfaces empleadas por AUI, el sistema dispone de la herramienta GUIDE (*Graphic User Interface Desing Environment*). Esta herramienta está basada en un modelo en el que se especifican los diagramas de transición de estados de la aplicación, los cuales son almacenados en una base de datos [Kantorowitz y Sudarsky (1989)]. GUIDE dispone además de herramientas para facilitar el diseño de interfaces por parte de personal con poco o nulo conocimiento de la programación. Este sistema permite la incorporación de nuevos dispositivos de interacción, realizando para ello pequeños cambios en los niveles léxicos, sintácticos y semánticos del modelo de diálogos en el que se basa AUI.

GUIDE define el modo de interacción GUI de forma extensa, ya que para este sistema un menú de opciones puede llegar a ser toda una pantalla, en la que se incluyen texto, gráficos e imágenes, llegando a representar el equivalente de una estación, escena o nodo en una aplicación multimedia [Schwabe et al (1998)]. El modo CLI es más tradicional ya que solo incluye la entrada de datos mediante cadena de caracteres. El empleo de estos dos modos se basa en la consideración de

que toda interfaz adaptable debería dar soporte a al menos dos modos de interacción, permitiendo la selección de cualquiera de estos modos en cualquier momento, siendo el propio usuario el que determine el nivel de detalle de la información representada en cada uno de ellos. Esta herramienta soporta una transición sencilla y natural entre los modos de interacción facilitando el hecho de que el usuario aprenda a manejar los dos modos a medida que se familiariza con el sistema, siendo ésta la principal aportación de este UIMS.

2.2.3. CHORIS

La etimología de este sistema proviene de *Computer-Human Object-oriented Reasoning Interface System* y éste consiste en una arquitectura genérica para el desarrollo de interfaces inteligentes, producto del trabajo de Centro de Inteligencia Artificial de Lockheed. El objetivo de este sistema es permitir que el mayor rango de usuarios posible trabajar de manera efectiva con varios tipos de aplicaciones complejas [Sherman et al (1991)].

El enfoque adoptado por CHORIS ha sido el diseño de un conjunto de módulos de razonamiento independientes del dominio de la aplicación, los cuales son dirigidos por bases de conocimiento que sí son dependientes del dominio. Dentro de estas bases de conocimiento se incluyen los modelos de usuario, el dominio del problema (objetos, relaciones, comandos y tareas de alto nivel), así como la propia interfaz. Por su parte, los módulos de razonamiento incluyen un gestor de entrada salida, un gestor de planes (encargado de interpretar las acciones realizadas por el usuario para inferir sus intenciones y poder asistirle en la realización de sus tareas), un modulo de adaptación capaz de modificar ciertas características de la interfaz en función del tipo de usuario y de las tareas que éste se encuentra realizando y por último un gestor de presentación encargado de decidir como presentar la información basando esta decisión en la naturaleza de ésta y el uso que se le pretende dar en un momento dado [Tyler y Schlossberg (1992)].

La versión original de CHORIS fue aplicada al desarrollo de un sistema de apoyo al gabinete de crisis de un desastre (terremoto, inundación, incendio, etc.) proporcionando información sobre el área geográfica en la que ocurre la emergencia por medio de diferentes vistas de un mapa sobre el que los usuarios han de tomar sus decisiones. Inicialmente en la pantalla se presentan los comandos que permiten invocar las acciones base del tipo de usuario que emplea la aplicación, dependiendo éstas de la categoría del mismo. Este usuario puede introducir los comandos al sistema por medio de una interfaz WIMP o también por medio de comandos escritos en lenguaje natural.

El modelo de usuario de este sistema se divide en tres niveles, los cuales describen diversos aspectos del usuario. En el nivel general se incluye el modelo de usuario canónico encargado de almacenar las preferencias y habilidades de los usuarios. En el segundo nivel se almacenan los estereotipos de los distintos miembros del gabinete que actúan en una emergencia, tales como el oficial de logística, el oficial de transmisiones, etc. Al clasificar al os usuarios de acuerdo con

estereotipos, en la interfaz puede inferir los valores por defecto de varias de las variables almacenadas en el modelo de usuario sin necesidad de realizar observaciones exhaustivas de su comportamiento, observaciones que necesitan de un período moderadamente largo que generalmente no está disponible en una situación de crisis. Finalmente, en el tercer nivel se ubican los modelos de usuario individuales, los cuales describen las características particulares de cada usuario del sistema. El tipo de variables almacenadas en estos modelos facilitan el almacenamiento del historial de interacción de cada usuario permitiendo almacenar los comandos y tareas ejecutados, los errores cometidos, las solicitudes de ayuda, etc. Por medio de estos conjuntos de variables, las reglas de inferencia empleadas por los agentes inteligentes de CHORIS pueden determinar el conocimiento que el usuario dispone acerca del dominio del problema, así como el de la propia interfaz, pudiendo modificar su comportamiento de forma adecuada.

El proceso de adaptación de la interfaz realizado por CHORIS se centra fundamentalmente en aspectos de nivel semántico, seleccionando en tiempo real el conjunto de procesos de usuario disponibles en función del tipo de usuario activo. Esta adaptación se pone en marcha empleando información combinada entre los datos del estereotipo del usuario activo y de sus propias preferencias e historial. Así, el conjunto de procesos de usuario invocados por un usuario tienen mayor probabilidad de aparecer en la interfaz que otros que nunca han sido invocados. En todo caso, la selección automática realizada por CHORIS no es definitiva, puesto que el usuario siempre dispone de la opción de modificarla a su gusto, añadiendo nuevos comandos en función de la situación. Naturalmente, esta nueva selección de comandos es observada por el agente interactivo y guardada en el modelo de usuario para ser empleada en acciones futuras [Bickmore et al (1998)].

Además de realizar una selección previa del tipo de contenidos y procesos de usuario a incluir en un diálogo interactivo, CHORIS aplica un proceso de adaptación sobre la información mostrada al usuario, modificándola para alcanzar el mayor grado de expresividad y de efectividad que tiene un tipo determinado de *widget* para mostrar dicha información. Las distintas opciones obtenidas son ordenadas en función en función del grado en que cada opción es capaz de dar soporte el requerimiento de información del usuario, luego este proceso de adaptación es realizado en función del contexto en el que éste es invocado. A pesar de su poder de adaptación, es necesario señalar que éste proceso se realiza solo en función del tipo de información a mostrar y de la situación en la que ésta es requerida, sin entrar en juego los requisitos del usuario activo, puesto que las reglas implicadas en el proceso de adaptación y que hacen uso de la información contenida en los modelos de usuario, hacen referencia a hechos generales que es de suponer son satisfechos por todos los usuarios de CHORIS.

2.2.4. CUBRICON

El nombre de este sistema significa literalmente un CUBRC Inteligente y CONversacional, siendo el término CUBRC el nombre dado por la fuerza aérea de los Estados Unidos a sus sistemas para el control y comando aéreo. El proyecto CUBRICON intenta aplicar técnicas de inteligencia artificial a la tecnología de factores humanos incorporando técnicas reconocimiento del habla, lectura en voz alta, lenguaje natural reconocido en fragmentos de textos, gráficos y señalización manual mediante punteros dentro del marco de un sistema de comunicación multimodal.

El objetivo planteado por este prototipo es la simplificación de la comunicación con sistemas complejos haciendo que el proceso de interacción resulte más cercano al realizado entre un humano y otros humanos. Para ello, CUBRICON combina los mecanismos de comunicación visual, comunicación visual y comunicación táctil mediante lo que sus autores denominan como *visión unificada del lenguaje* [Neal y Shapiro (1991)]. Gracias a esta visión unificada, el usuario es capaz de comunicarse con el usuario mediante mensajes hablados o escritos en lenguaje natural combinados con señalizaciones de objetos concretos representados en alguna de las dos pantallas que emplea esta interfaz (una en color para la representación de los mapas en los que se designan los objetivos militares y otra en blanco y negro en la que se incluye información acerca de estos objetivos). De este modo, el usuario puede emitir comandos en voz alta del tipo de *ataca este objetivo con los aviones de esta base* mientras señala en las pantallas a los objetos protagonistas de la acción (el objetivo y la base aérea).

Para alcanzar sus objetivos, CUBRICON dispone de varios componentes, entre los que destacan su modelo de usuario y su base de conocimiento. El primer componente es capaz de almacenar información acerca de la experiencia del usuario en tácticas militares, el papel jugado por el usuario en un momento determinado (estratega militar, técnico en logística, etc.), sus preferencias con respecto al modo de comunicación. En base a la tarea que el usuario se encuentra realizando y en función de la importancia que el usuario asigna a cada uno de los elementos interactivos que forman parte de la interfaz visual de CUBRICON, este sistema, es capaz de determinar que tipo de información es relevante a la hora de confeccionar una respuesta ante una petición de información. Así, si el usuario instruye al sistema para que muestre un área determinada del mapa, éste determinara los objetos que se encuentren dentro de la misma pero mostrará solo aquellos que sean relevantes para un usuario determinado. Así por ejemplo, si el peticionante es un planificador de misiones militares, las posibles granjas que se encuentre en el área solicitada no serán mostradas puesto esta información resulta totalmente irrelevante para este tipo de usuario concreto.

Por otro lado, la novedad de la base de conocimiento de CUBRICON radica en su diseño en tres capas, en función del tipo de inferencias a realizar. Así, existirá una capa encargada de la planificación de las respuestas proporcionadas por el sistema y de la

composición de la salida en función de los distintos modos de comunicación disponibles. Una segunda capa contiene información acerca de conocimientos generales compartidos por todos los usuarios de la aplicación. Por último, la tercera capa contiene información acerca de las distintas tareas que se pueden emplear en la confección de una táctica efectiva para el control aéreo en tiempo real. Como se puede apreciar, las capas se encuentran clasificadas con respecto a la carga semántica de las mismas, partiendo de una capa con carga semántica cero (dedicada únicamente a gestionar aspectos sintácticos de la interfaz) y llegando a una capa en la que la semántica se escapa por completo al dominio de la interfaz, entrando de lleno en el campo de acción de la funcionalidad de la aplicación.

Este sistema dispone de un mecanismo sencillo de adaptación para distribuir el tráfico de información acerca de las diversos modos de comunicación disponibles. Para ello, CUBRICON se basa en el modelo de equivalencia gráfica de Andriole (1986, citado en [Neal y Shapiro (1991)]) en que se usan varias pantallas y / o ventanas para presentar los mismos contenidos en formas distintas con el objeto de ayudar al usuario a comprender mejor el problema al que se enfrenta, mejorando su rendimiento en sistemas de productividad. Para cada ítem a mostrar, el sistema determina la modalidad ideal para su representación basando la toma de decisiones en estrategias preconcebidas en donde se supone que una representación gráfica siempre es el ideal a ser alcanzado, dejando el lenguaje natural como la última opción a realizar. Dado que las prioridades establecidas a cada tipo de representación de información son asignadas de antemano, el resultado de la adaptación es pobre puesto que carece de mecanismos que permitan una ajuste ideal a las características concretas de sus usuarios. Así por ejemplo, un usuario con serias deficiencias visuales vería empeorado su rendimiento al ser obligado a hacer uso en primer lugar de representaciones visuales en las que su rendimiento es pobre. En ningún momento CUBRICON hace uso de información específica sobre las características perceptivas del usuario para tomar decisiones acerca de la estrategia a emplear en procesos de adaptación, dado que toda la información empleada se refiere exclusivamente al tipo de datos a mostrar [Shapiro (2000)].

Otra crítica importante que se hace a este sistema es el alto grado de acoplamiento entre el dominio del problema, la funcionalidad que lo modela y su interfaz, ya que a pesar de reorganizar la base de conocimiento en diferentes niveles según su carga cognitiva (perteneciendo el último de ellos exclusivamente a la funcionalidad de la aplicación) esta base de conocimiento se encuentra estrechamente integrada dentro de los procesos particulares de la interfaz. Se puede decir, que para CUBRICON no existe separación alguna entre interfaz y funcionalidad, aunque en su propio mérito hay que señalar que no es un objetivo de este sistema el lograr esta separación –por mínima que esta fuera– ya que no se trata de un UIMS propiamente dicho.

2.2.5. MASTERMIND (HUMANOID y UIDE)

Los UIMS *HUMANOID* [Szekely et al (1992 y 1993)] y UIDE (*User Interface Design Environment*) [Foley et al (1991)] se fusionaron mediados los noventa en un único UIMS conocido como MASTERMIND, el cual ha sufrido y sigue sufriendo una evolución constante. MASTERMIND es uno de los sistemas de gestión de interfaces de usuario más completos que existen puesto que comprende todas las fases de desarrollo de un modelo de interacción, desde la fase de análisis hasta la propia evaluación del producto, generando automáticamente gran parte de la documentación necesaria para el mantenimiento y posterior evolución de los diseños generados con esta herramienta.

Mediante MASTERMIND y posteriores es posible representar el diseño conceptual de una interfaz de usuario especificando la jerarquía de objetos que modelan el dominio del problema, las propiedades y acciones que se pueden ejecutar sobre estos objetos, las unidades de información realizadas por estas acciones, así como las *precondiciones* y *postcondiciones* para estas acciones, siendo el primer UIMS en incorporar estos últimos dos conceptos por medios formales en el diseño del modelo de interacción de una aplicación. En este apartado se hace necesario destacar que, aunque los conceptos de *precondición* y *poscondición* formaban parte ya de las opciones de un antiguo UIMS diseñado en la Universidad de Alberta [Green (1985)], el papel jugado por estos conceptos era secundario, siendo UIDE el primer UIMS que centra todo el proceso de diseño del modelo de interacción en las *precondiciones* y *postcondiciones*.

Toda esta información es empleada por MASTERMIND para verificar la consistencia del diseño de la interfaz, determinar su desarrollo, realizar transformaciones de alto nivel sobre la base de conocimiento de la interfaz para generar interfaces funcionalmente equivalentes, evaluar el diseño de la interfaz con respecto a su productividad, producir la descripción impresa del diseño y servir como entrada a un módulo independiente denominado SUIMS (*Simple User Interface Management System*) el cual es el encargado de implementar la interfaz en tiempo de ejecución y generar ayuda sensible al contexto de forma inteligente y en tiempo real.

MASTERMIND no es considerado como una interfaz de usuario inteligente propiamente dicha sino como una herramienta inteligente para el diseño de interfaces de usuario. En este sentido, el UIMS representa una ayuda para el diseñador de interfaces de usuario, contemplando un conjunto de reglas que son aplicadas para generar automáticamente buenos diseños o para realizar críticas constructivas sobre diseños previos con el objetivo de detectar posibles fallos, mejorando la calidad de las interfaces así obtenidas.

Para diseñar una interfaz con MASTERMIND, el programador ha de incluir de forma manual una descripción detallada de todos los objetos interactivos de la aplicación, incluyendo las propiedades de cada objeto. Cada una de estas propiedades dispone de un atributo que indica si éste va a ser evaluado continuamente por el usuario y por lo tanto la

interfaz ha de reflejar cada cambio que la aplicación efectúe sobre el mismo.

Además de las propiedades, el programador usuario del entorno MASTERMIND debe especificar todos los procesos de usuario asociados a cada objeto, que en la versión original del sistema recibían el nombre de *esquemas*. Para cada uno de los esquemas es necesario definir *precondiciones* y *postcondiciones*. Las precondiciones son predicados que se deben cumplir como condición previa para que el proceso de usuario representado por el esquema pueda ser activado y esté disponible en la interfaz de usuario. Las *precondiciones* son empleadas además para proporcionar explicaciones al usuario acerca de por qué un comando está desactivado o para proporcionar ayuda al mismo explicándole la secuencia de acciones que debe ejecutar para activar un determinado esquema. Los predicados que representan a las *precondiciones* se codifican en una forma que puede ser evaluada en tiempo de ejecución y generalmente están integrados por variables contextuales. Naturalmente, las *precondiciones* se basan en las dependencias semánticas existentes entre los diversos procesos de usuario que integran la funcionalidad de la aplicación [Castells et al (1997)]. Por otro lado, las *postcondiciones* representa una o más sentencias que asignan valores a las variables empleadas en leas *precondiciones*, siguiendo a la ejecución de la acción con la que éstas se encuentran asociadas. La sentencia que asigna valores a las *postcondiciones* se especifica también por medio de una forma que puede ser evaluada en tiempo de ejecución y son empleadas para cambiar el estado del contexto, explicar la semántica de determinados comandos y para proporcionar ayuda al usuario acerca de los resultados de sus acciones [Frank et al (1995)].

Con toda la información incluida en la base de conocimiento, MASTERMIND puede efectuar los tests de integridad habituales de otras herramientas de desarrollo orientadas a la ingeniería del software, verificando por ejemplo que la jerarquía de clases especificada sea realmente un árbol (no existan ciclos), que la información requerida para cada esquema haya sido introducida correctamente y que las *precondiciones* estén especificadas de la forma adecuada, es decir, que el valor de cada variable empleada en una *precondición* sea asignado al ejecutarse alguna *postcondición*.

Una vez terminada la fase de diseño, la base de conocimiento es transferida al SUIMS quien se encarga de controlar el flujo de acciones e información a través de los distintos procesos de usuario del modelo de interacción así diseñado. Este componente organiza los comandos y sus parámetros en menús, menús jerárquicos y cuadros de diálogo tomando como criterio la información disponible entre las relaciones lógicas existentes entre los comandos (procesos de usuario) y sus parámetros asociados (modelo de datos de usuario).

La parte inteligente de este UIMS se encuentra incluida precisamente en su módulo de generación de interfaces y consiste en la realización de un proceso de evaluación de todas las alternativas disponibles en un momento dado para una interfaz concreta, realizando una predicción del tiempo que tardaría un usuario medio en ejecutar sus tareas con la

mencionada interfaz. Para ello, MASTERMIND se basa en el modelo de predicción de Card et al (1983), el cual define el tiempo necesitado para ejecutar una tarea como la suma de los tiempos necesitados para ejecutar los operadores básicos en los que éste se descompone, tales como operadores de tecleado, operadores de toma de decisiones, operadores de señalización con dispositivos del tipo de ratones, joysticks, etc. Este modelo de predicción se describirá en detalle en el capítulo 16 (*Un Modelo de Evaluación*), puesto que también es empleado por el UIMS GADEA aunque de forma totalmente distinta a la empleada por MASTERMIND. En concreto, MASTERMIND asigna tiempos de ejecución estándar para cada uno de los operadores básicos, calculando el tiempo necesitado por usuario medio para llevar a cabo un proceso de usuario, aplicando este proceso de predicción a las diferentes alternativas que se pueden plantear a la hora de diseñar el mismo, seleccionando así aquella que conduce a la más rápida. Hay que señalar sin embargo que la propia diversidad humana implica una alta variabilidad en los tiempos necesitados por cada usuario concreto para la ejecución de estos operadores básicos por lo que la alternativa seleccionada puede no ser la mejor para un usuario en especial. Dado que la evaluación de las alternativas se realiza en tiempo de diseño y no en tiempo de ejecución, MASTERMIND diseña por tanto para un usuario genérico en lugar de para un usuario específico, del cuya existencia no tiene constancia.

2.2.6. PERSONA

PERSONA (sic) es una propuesta de arquitectura para UIMS adaptables propuesta por Reynolds (1997) en respuesta a la carencia de una módulo de gestión de presentaciones personalizadas en el modelo KBFE. A pesar de tratarse de una arquitectura sólida y bien fundamentada, hasta el momento aún no ha sido implementada con éxito.

Para Reynolds una *persona* es definida como un modelo de usuario especializado que sirve como repositorio para todas las preferencias personales de un determinado usuario en un sistema concreto. Como se puede apreciar, una persona representa la abstracción de las características que definen a un individuo en concreto, separando éstas de cualquier relación con una interfaz concreta. Funcionalmente hablando, una persona actuaría como un registro portátil de todas las opciones que diferencian el entorno de trabajo ideal del individuo del entorno por defecto (el del dominio de una aplicación concreta). Por medio de este concepto, un usuario podría realizar una serie de cambios sobre la apariencia de la interfaz de un sistema operativo y almacenarlos en su *persona*. Si el mencionado usuario desea es trasladado a otra máquina, podrá llevarse su *persona* consigo, de tal modo que los cambios realizados se reflejen de forma automática en la apariencia de las aplicaciones del sistema operativo de la segunda máquina. De este modo, el entorno de trabajo de un usuario no está restringido a una máquina concreta sino a una entidad virtual que puede ser instalada en cualquier máquina.

Para construir el UIMS PERSONA, se sugiere el empleo de agentes sastre, los cuales crearían interfaces a medida. Estos agentes sastre obtendrían información del usuario y de su modelo de usuario (*persona*), empleando un sistema experto para determinar la combinación de opciones de la interfaz de usuario que favorecería más al usuario. Antes de modificar permanentemente el modelo *persona*, el agente sastre verificaría la eficacia de sus decisiones mostrando al usuario la interfaz confeccionada y preguntándole si la mencionada interfaz es de su gusto.

Para la confección de las interfaces, el agente sastre requiere el uso de baterías de componentes. En PERSONA, una batería se define como una colección de componentes (*widgets*) que realizan tareas similares y que han sido diseñados para satisfacer los requisitos de un determinado tipo de usuario. Así una batería de diálogos contendría una variedad de cuadrados de diálogo diseñados para diferentes grupos culturales. Estas baterías contendrían por ejemplo, una serie de diálogos para usuarios novatos y otros para usuarios expertos. Dado que el objetivo del agente sastre es crear una interfaz de usuario que agrade al usuario, aquél iría obteniendo las características del usuario mediante preguntas directas. En el caso del ejemplo anterior, el agente preguntaría al usuario por su grado de experiencia, seleccionando el acuerdo de dialogo a emplear en función de la respuesta. El agente sastre verificaría su elección realizando preguntas periódicas al usuario acerca de la complejidad de uso del cuadro de diálogo seleccionado.

Naturalmente, una especificación tan ambigua deja muchas preguntas en el aire. En primer lugar, no quedan claros los criterios que emplearía el agente sastre para seleccionar un determinado componente entre las decenas o cientos de componentes almacenados en una batería. La respuesta a esta pregunta determinaría el grado de inteligencia que deberían tener los agentes y por ende una aproximación de los tiempos de computación necesarios para elaborar respuestas adecuadas. En segundo lugar es cuestionable el grado de efectividad que tendrían las interfaces generadas mediante este UIMS en términos de usabilidad, dado que el usuario sería interrogado periódicamente y por lo tanto sería interrumpido en sus tareas.

Otra importante carencia de esta especificación es la de cómo se realizaría la separación básica entre la interfaz y la funcionalidad de una aplicación compatible con PERSONA, si tomamos en cuenta que esta separación es necesaria para poder combinar los componentes almacenados en las baterías de una forma efectiva, dado que es precisamente este aspecto el talón de Aquiles de este tipo de UIMS.

2.2.7. VWPC

VWPC es un acrónimo que proviene de *Visual Web Pages Critic* y se refiere a una herramienta desarrollada por personal de Microsoft que tiene como objetivo determinar las zonas de máxima atracción visual de una página web, aunque el algoritmo aplicado puede extenderse a cualquier diálogo interactivo visual. Esta herramienta emplea una sólida base de conocimiento en la que se almacenan reglas de carácter

general acerca del comportamiento del sistema perceptivo humano en tareas de rastreo visual [Faraday y Sutcliffe (1997)]. Con estas reglas y recibiendo como entrada una página web cualquiera escrita en HTML, el VWPC puede determinar el recorrido aproximado que seguirá el ojo del usuario al examinar la página, determinando además el orden en el que las distintas zonas de la misma serán recorridas. VWPC es por tanto una herramienta de desarrollo más que una interfaz de usuario. Sin embargo, el alto grado de precisión con la que realiza su tarea y las técnicas de inteligencia artificial empleadas, así como su novedad han sido los factores que han determinado su inclusión en esta sección.

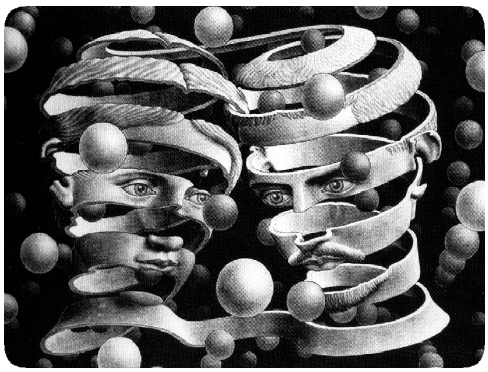
Para realizar su labor, VWPC emula el comportamiento del usuario dividiendo el proceso de lectura de un espacio visual en dos fases, una de búsqueda y otra de recorrido. La primera fase se desarrolla cuando el observador intenta encontrar una zona destacable en la página o punto de atracción. Una vez encontrado este punto, empieza la segunda fase en la que el observador inicia el recorrido extrayendo la mayor cantidad de información posible del espacio visual.

Para cada una de las fases de lectura, la herramienta emplea una jerarquía distinta de elementos a tomar en cuenta. Así, durante la fase de búsqueda, la herramienta buscará primero aquellas zonas del espacio visual en donde haya representado algún tipo de movimiento, generalmente mediante el empleo de animaciones. Después del movimiento, será el tamaño de los objetos el siguiente elemento atrayente, siendo aquellos elementos u áreas de mayor tamaño y extensión las más atrayentes. Las siguientes características atrayentes para VWPC serán las imágenes, el color (en donde los colores brillantes tendrán mayor prioridad), el estilo del texto (dando prioridad al texto en negrita) y en último lugar a la posición en la que se encuentran los objetos, destacando aquellos objetos que se encuentren en la zona superior sobre aquellos situados en la zona inferior y también a aquellos objetos ubicados en la zona izquierda con preferencia sobre aquellos localizados sobre la zona derecha. Una vez ubicado el punto de atracción, la fase de recorrido dará prioridad a los objetos que se encuentren agrupados dentro de una misma área primero y a los objetos próximos o que se encuentren dentro del orden de lectura después (de izquierda a derecha y de arriba abajo) [Faraday y Sutcliffe (1998)].

Un prototipo de esta herramienta ha sido desarrollado empleando Visual C++ y AMZI Prolog para Windows 98 y el componente de edición de Microsoft Explorer 5. Con las reglas codificadas mediante expresiones en Prolog, el prototipo analiza un documento web e identifica el recorrido visual que realizaría el usuario identificando todas las estaciones de parada y asignándoles un número de prioridad [Faraday (1999)].

Esta herramienta es de gran utilidad para identificar los puntos calientes de un espacio visual, permitiendo la obtención rápida de un esquema de visualización para dicho espacio, esquema que puede ser comparado con los objetivos definidos para el diálogo interactivo correspondiente. La gran ventaja de herramientas de este tipo radica en que permiten establecer estrechos lazos entre el nivel semántico de un

diálogo interactivo (el tipo de información a ofrecer o a adquirir) y el nivel sintáctico del mismo (como es distribuida esta información en el espacio físico asignado al diálogo). A partir de este punto, sería posible establecer métricas para la calidad de una interfaz a nivel sintáctico y, por medio de dichas métricas, evaluar todas las alternativas posibles para una interfaz, seleccionando aquella que fuese la adecuada, de modo similar al empleado por MASTERMIND (ver apartado 2.2.5) en su nivel semántico.



Banda sin Fin © 1956 por M. C. Escher

P A R T E I I

GADEA

3 La Individualidad

El mito del usuario típico. Aspectos perceptivos y aspectos motrices implicados en la interacción. Caso de estudio: ¿cuál es la mejor ubicación para una barra de navegación en un portal de Internet? Acceso universal a las interfaces: la adaptación de los niveles léxico y sintáctico. Objetivos de GADEA: hacia la generación dinámica de las interfaces. Marco conceptual para el empleo de agentes interactivos inteligentes. Adaptación del paradigma de los sistemas expertos. El diseño de GADEA: los módulos Codex, DEVA y ANTS.

p. 91

4 La Comunicación

GADEA como un sistema de comunicación. La Teoría General de Sistemas y el modelado de interfaces. Tratamiento de los recursos humanos de un sistema. Jerarquía de objetivos de un sistema. Especificación de componentes y Adaptadores. Características de un canal de comunicación. Canales de

comunicación primarios y secundarios. El papel de la semiótica. El efecto del ruido en la comunicación.

p. 107

3

3 La Individualidad

El problema es que no existe una cosa llamada usuario típico… en realidad, la adopción de un determinado software tiene poco que ver con su facilidad de uso y mucho que ver la adecuación de su funcionalidad a los requisitos de la organización.

R. M. Baeker.

3.1 El Usuario Típico

El diseño tradicional de las interfaces centradas en el usuario está basado en la identificación de su *audiencia tipo*, así como de las características y parámetros que definen a esta audiencia destino. De hecho algunas de las guías de diseño más relevantes en esta disciplina incluyen la identificación y definición de la *audiencia tipo* como uno de los pasos más importantes a la hora de diseñar un producto [Apple (1992), Microsoft (1998)]. La idea subyacente en estas técnicas de diseño radica en que sólo cuando las características de la audiencia tipo de un producto han sido definidas y comprendidas, será posible diseñar una interfaz capaz de satisfacer las necesidades de esta audiencia con un grado óptimo de efectividad.

Sin embargo, la búsqueda del *usuario típico* de una aplicación es una idea contrapuesta a la individualidad y diversidad humana, conceptos

que constituyen gran parte de la identidad de una persona. Si el diseño de los mecanismos de interacción de una herramienta pretende crear interfaces accesibles y atractivos para todos y cada uno de los usuarios, desde luego, lo que no debería hacer es caer en una generalización abstracta de la interfaz.

El proceso erróneo de percepción de una interfaz genérica fue descrito por Norman (1988a) empleando modelos conceptuales Según este autor, cuando un diseñador establece las pautas de su diseño en función de la noción que tiene del usuario típico, lo que hace realmente es proyectar su propio modelo mental sobre el producto a desarrollar. Dado que el modelo mental del usuario final no tiene por qué coincidir con el modelo del usuario genérico y mucho menos con el modelo mental del diseñador, éste se ve obligado a alterar sus modelo hasta hacerlo compatible con el del producto. El grado de desviación existente entre el modelo mental del usuario y el modelo mental del diseñador se corresponde con el grado de frustración del primero al emplear el producto del segundo. En el caso de presentarse una desviación exagerada, puede ocurrir que el usuario sea incapaz de usar la interfaz. Nótese que en cualquier caso es muy difícil que el grado de desviación sea nulo, puesto que dado que cada usuario dispone de su propio modelo mental individualizado, la probabilidad de que su modelo mental coincida a la perfección con la del modelo mental del diseñador es muy remota. Luego la frustración existe siempre.

3.1.1. 4.2 ¿Dónde Colocar una Barra de Navegación?

Para ilustrar este hecho, vamos a intentar responder a la pregunta que da pié a este apartado, circunscribiéndola al entorno de un portal en Internet y mediante un sencillo experimento. Para facilitar este análisis, supondremos que solo existen dos ubicaciones posibles para la barra de navegación, o bien a la izquierda o bien a la derecha del espacio visual. Aunque la pregunta parece sencilla, desde luego su respuesta no lo es tanto. Siguiendo las recomendaciones propuestas por las técnicas de diseño clásicas, el primer paso adoptado por los diseñadores del portal será definir e identificar las características básicas de sus usuarios, tarea de por sí prácticamente imposible dado que ¿cómo puede ser el usuario típico de un portal de Internet visitado por miles de personas diferentes? Con respecto al aspecto de la interfaz que nos atañe y aras de simplificar de nuevo el análisis, procederemos a realizar la primera abstracción suponiendo que se trata de usuarios del entorno cultural occidental. A partir de este supuesto, es posible determinar la mejor ubicación para la barra de navegación tomando en consideración las guías de diseño disponibles en la literatura.

Desde la época en que las publicaciones impresas dominaban el mercado, los diseñadores gráficos consideraron la zona izquierda de un espacio visual como la zona más atractiva a la hora de llamar la atención del posible cliente [Gordon (1994)]. La razón para esta creencia radica en que los lectores de un idioma basado en el sistema de escritura occidental (como es el caso del castellano) esperan encontrar el principio de una oración o frase en la zona izquierda de las páginas.

Este principio de diseño también ha sido trasladado al medio electrónico. Navegando por la red, podemos comprobar como prácticamente todos los sitios web han adoptado este principio, colocando la información más importante (la tabla de contenidos o su barra de navegación) en la zona izquierda del espacio visual disponible (ver figura Figura 4).

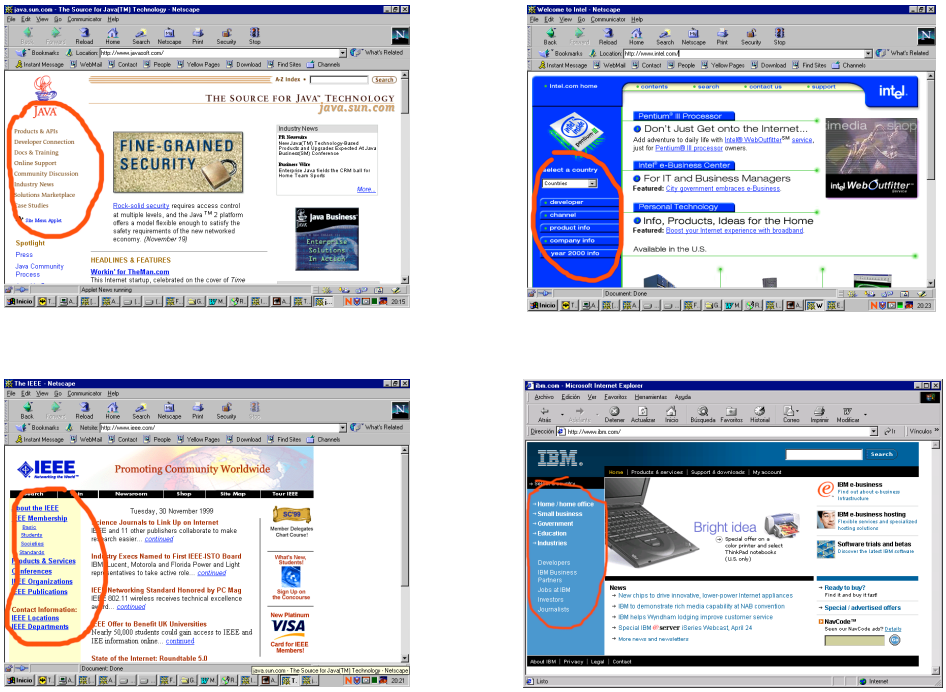


Figura 4. Ejemplo de portales de Internet en donde la barra de navegación (marcada por un círculo rojo) se encuentra situada a la izquierda. De izquierda a derecha y de arriba abajo se puede ver el portal de JavaSoft (http://www.javasoft.com), el portal de Intel (http://www.intel.com) , el portal de IEEE (http://www.ieee.com) y el portal de IBM (http://www.ibm.com), (consultar versión en color en el capítulo 24: *Imágenes en Color*, p. 445).

Desde un punto de vista teórico queda claro que la zona izquierda de un espacio visual es aquella que posee el mayor impacto. Esta conclusión parece quedar respaldada además por el masivo apoyo a la solución izquierda es más dado por decenas de los portales más importantes de Internet. Sin embargo cabe preguntarse la opinión que tienen los usuarios reales de este planteamiento. Para poder cuantificar la influencia que la distribución de objetos visuales sobre un plano horizontal tiene sobre la navegación, se diseño preparado un experimento con usuarios reales, por medio de pruebas de usabilidad remotas, empleando para ello técnicas y herramientas diseñadas en el marco de esta investigación, las cuales serán descritas en el capítulo 19 (*Espías*).

El portal de Internet diseñado para este experimento es realmente sencillo, pues consiste tan solo de un nodo de entrada provisto de un panel de información central y dos barras de navegación idénticas (ver

esquema en Figura 5). Una de ellas se situó a la izquierda y la otra a la derecha, siendo las dos idénticas. Cada una de las barras dispone de un solo enlace (de semántica idéntica en las dos barras) que el usuario ha de seleccionar una vez leída la información incluida en el panel de información central. Obviamente, en dicho panel no se proporciona ninguna pista acerca de cual de los dos enlaces conduce a la información deseada por el usuario. Dado que el objetivo del experimento es obligar a los visitantes del portal a explorar cuidadosamente las dos barras de navegación para seleccionar el enlace más relevante, éstas fueron construidas empleando muy poco contraste cromático. De este modo la lectura resulta difícil, obligando al usuario a concentrar su atención en las áreas visuales ocupadas por las barras de navegación, empleando un esfuerzo mental adicional.

Figura 5. Esquema del nodo de entrada al portal referido en el experimento de las barras de navegación.

Puesto que las dos barras de navegación son idénticas, los usuarios que visitan el portal, tienen que afrontar la disyuntiva de elegir entre dos opciones idénticas [Nickerson (1972)], seleccionando aquella que el usuario considera como la más relevante, evaluándola exclusivamente desde un punto de vista perceptivo y motriz, es decir, realizando una evaluación exclusiva de las mecanismos de interacción provistos por el nivel léxico y sintáctico de la interfaz del portal.

Hay que destacar que la acción de elegir un enlace depende directamente del modelo cognitivo humano e implica consideraciones de tipo ergonómico. Elegir un enlace no es una acción gratuita, pues se necesita algún tiempo para tomar una decisión (determinado por la Ley de Hick [Newell (1991)]), operación seguida por un movimiento mecánico para colocar el puntero pulsar sobre el enlace. El movimiento del puntero desde el lugar original hasta la posición en la que se encuentra el enlace requiere también de cierto lapso de tiempo (determinado por la Ley de Fitt, [Newell (1991)].

En el portal ejemplo [Arqueoastur (2001)] se instalaron agentes capaces de observar las acciones realizadas por los usuarios en el mismo (ver capítulo 19: *Espías*), los cuales fueron empleados para detectar el enlace seleccionado por el usuario (y por ende el área visual de su preferencia), así como para determinar el tiempo de reacción en la operación de selección, es decir, el lapso de tiempo empleado por el usuario desde que llega al nodo de entrada y hasta que pulsa uno cualquiera de los dos enlaces proporcionados. El objetivo de medir el tiempo de reacción en el contexto de este experimento es poder

determinar el tipo de estrategia de observación empleada por el usuario, las cuales podrían ser analíticas, superficiales, aleatorias o deterministas [Ittelson 1960)]. Así, los usuarios que consumen un período de tiempo considerable para tomar la decisión de cuál es el enlace deseado podrían estar empleando una estrategia de tipo analítico, mientras que aquellos que emplean muy poco tiempo, podrían seguir un modelo de decisión compulsivo o superficial. Al medir el tiempo de reacción, se está intentando detectar alguna posible relación entre la zona del campo visual seleccionada y la estrategia de observación empleada.

El número de sesiones de navegación válidas registradas por la herramienta de prueba remota (ANTS) fue de 342 en tres semanas. Los resultados –una vez que los datos recolectados fueron debidamente procesados– se pueden ver en la (Tabla 1). Estos resultados muestran claramente que, aunque los diseñadores gráficos no estaban equivocados cuando afirmaban que la zona izquierda es la preferida por los lectores, la diferencia entre el número de selecciones para las dos zonas no es significativa. Al contrario de la creencia general y a la luz de estos datos, es necesario señalar que el modelo mental empleado por los diseñadores gráficos (que prefieren la zona izquierda sobre la zona derecha) apenas coincide con la representación mental del 52,5% de los usuarios.

Tabla 1. Preferencias de selección de usuarios para las barras de navegación izquierda y derecha. El Tiempo de Reacción para las dos zonas coincide (promedio global = 23,94 segundos).

RESULTADOS GLOBALES	
Universo: 342 sesiones válidas.	**Selección Zona Izquierda:** 179 (52,5%).
Promedio Global para el Tiempo de Reacción: 23,94 segundos.	**Selección Zona Derecha:** 160 (47,5%).

ZONA IZQUIERDA	ZONA DERECHA
Elegida: 179 veces. (52,5%)	Elegida: 160 veces. (47,5%)
Promedio Global para el Tiempo de Reacción: 23.83 segundos.	Promedio Global para el Tiempo de Reacción: 24,08 segundos.

Dado que el tiempo de reacción para las dos zonas es prácticamente el mismo (~23,94 segundos), el experimento ha fracasado en su segundo objetivo de reconocer algún tipo de relación entre la estrategia de observación y la zona del campo visual elegido. Sin embargo, hay que destacar que gracias a éste se puede afirmar que la ubicación de las barras de navegación en el plano horizontal es una decisión que atañe exclusivamente a la facilidad de uso y el grado de intuición del interfaz pero que no tiene demasiados efectos sobre la eficacia con la que el usuario accederá a la interfaz, dado que una ubicación errónea de las barras no afecta al rendimiento del usuario.

Este simple experimento demuestra que en la asignación de grados prioridad a los elementos que componen un espacio visual no solo

influye el sistema perceptivo humano sino que también actúa el sistema motriz. Nótese que en las publicaciones impresas, la tarea planteada en el experimento se reduce a *percibir* la zona más relevante dentro de un área visual, pero que en un entorno interactivo como el presente, la tarea de percepción se completa con la de la interacción, ya que el usuario ha de desplazar el puntero sobre la zona de su preferencia. Se trata pues de dos tareas (percepción e interacción) en lugar de una. Si bien para las publicaciones impresas el área más relevante depende de factores como el sistema de escritura del usuario, en publicaciones interactivas, el área más relevante dependerá además de factores como la lateralidad del usuario.

3.2 Objetivos

En el ejemplo anterior se ha podido apreciar como la generalización de partida (el usuario pertenece al entorno cultural de occidente) puede conducir a resultados erróneos al no tomar en cuenta características particulares del individuo como por ejemplo, su lateralidad, demostrando así que el objetivo de alcanzar una interfaz atractiva para *cada* individuo se contradice totalmente con una interfaz diseñada para *todos* los usuarios [Reynolds (1997)].

Esta misma generalización del usuario típico, es causa de que cientos de miles de portales en Internet estén diseñados para satisfacer plenamente las necesidades de tan solo la mitad de sus usuarios y eso tan solo con respecto a la parte del diseño que se refiere a la ubicación de los objetos interactivos. Si se toma en consideración el hecho de que el lugar de preferencia de una barra de navegación en un sitio web es tan solo una más de las múltiples decisiones los diseñadores han de tomar a la hora de implementar el nivel sintáctico de la interfaz que da soporte a estos dispositivos, puede apreciarse en detalle la magnitud del problema.

En todo caso y aún en el supuesto de que la definición de un usuario típico fuese positiva para el diseño de una interfaz, resulta realmente difícil determinar el *usuario típico* de una aplicación, sobre todo en aquellas aplicaciones dirigidas a un amplísimo número de usuarios, como es el caso de un sistema operativo o el portal de Internet que se ha empleado en el ejemplo. Cabe preguntarse por tanto si es posible describir al *usuario típico* de un portal como Terra o Yahoo que reciben miles de visitas todos los días. La misma pregunta se podría aplicar al Microsoft Office. ¿Es posible describir al usuario típico de esta popular herramienta? Para algunos autores como Reynolds (1997), Baeker et al (1995) o Schneiderman (1987) esta respuesta es negativa, dado que *no existe el usuario típico*. Para este último autor *o se alcanza una solución de compromiso en el diseño de las interfaces o se deben crear múltiples versiones de las mismas*.

La solución de compromiso planteada a la que Schneidermann se refiere no es otra que adoptar una interfaz genérica y dejar que sea el usuario el que se adapte –con mejor o peor fortuna– al modelo mental empleado por el diseñador, con los problemas de usabilidad ya

comentados. La segunda opción consiste naturalmente en crear versiones específicas de la interfaz del producto especialmente diseñadas para determinados tipos de usuario, opción que como ya se ha visto, es la seguida por algunos de los UIMS inteligentes recogidos en el capítulo 2 (*Interfaces Inteligentes*). Sistemas como APEX, GRIDS o MASTERMIND hacen uso de esta estrategia para diseñar múltiples versiones estáticas de una interfaz en función del tipo de usuario destino del producto. Los agentes inteligentes incluidos en el generador de código de estos UIMS permiten generar múltiples versiones de estas interfaces de una forma eficaz, relativamente barata, pero terriblemente restringida, dado que versiones se generan en tiempo de edición y sin tener contacto alguno con el usuario real de la aplicación.

Por su parte, sistemas como CHORIS, CUBRICON, PERSONA y en cierta medida AUI-GUIDE adoptan un planteamiento más agresivo al intentar generar las interfaces en tiempo real, de forma dinámica y en base al conocimiento disponible acerca de las características del usuario activo. Sin embargo, con la excepción de PERSONA, estos sistemas practican sus mecanismos de adaptación en los niveles conceptual y semántico del diseño de la interfaz, es decir, a un nivel puramente cognitivo, dejando de lado los niveles perceptivo y motriz que son causa de un buen número de problemas de usabilidad entre los que se encuentran los recogidos en el experimento que sirvió de introducción a este capítulo. Se necesita por tanto una solución integral que permita dar soporte a mecanismos de adaptación de bajo nivel que cubra principalmente los requerimientos de adaptación de los niveles léxico y sintáctico.

La importancia de un soporte adecuado a estos niveles queda patente si se toma en cuenta que la mayor parte de los aspectos que distinguen a una persona de otra a efectos de interacción se sitúan en el léxico y sintaxis de una interfaz. Factores como la edad, el sexo, la percepción visual, la percepción auditiva, la capacidad motriz o la presencia de algún tipo de discapacidad en los sentidos visual, auditivo y motriz principalmente, son solo algunos de los elementos que contribuyen a crear diferencias notables entre los usuarios de una misma aplicación. De hecho, estas diferencias pueden condicionar seriamente la efectividad con la que un determinado individuo accede a un interfaz no adaptada a sus peculiaridades perceptivas y/o motrices.

Sin embargo, gran parte del esfuerzo dedicado al diseño de una buena interfaz de usuario se encuentra concentrado en la representación adecuada de la información, es decir, en la sintaxis de dicha interfaz. Para llevar a cabo esta tarea de forma eficaz, el diseñador de interfaces combina su experiencia en el diseño de interfaces anteriores, su conocimiento acerca de las reglas básicas del diseño y análisis de las tareas a realizar por la interfaz, organizando la información a desplegar a lo largo de los distintos puntos en los que dicha interfaz a de dialogar o intercambiar información con el usuario.

En prácticamente la totalidad de los UIMS descritos, el diseño como la organización de los aspectos perceptibles de las interfaces se realizan durante la fase de diseño de la misma, codificando sus diálogos interactivos de forma puramente estática, de tal modo que este diseño

es inamovible y no se puede adaptar a las necesidades de un determinado colectivo de usuarios, salvo por deseo expreso de los propios diseñadores y con el condicionante previo de la existencia de ese colectivo en los planes de quien desarrolla. Si la adaptación de una interfaz estática a nivel de colectivo es ya difícil, la misma adaptación a nivel de individuo es prácticamente imposible.

Es necesario destacar que incluso cuando una buena interfaz es producida de forma estática, para cumplir con los requisitos de un determinado tipo de usuario, ésta interfaz será empleada en todas las circunstancias imaginables, con independencia del tipo de tarea a realizar y con independencia de la notable evolución del estado cognitivo, perceptivo y motriz del usuario destino a lo largo del tiempo. Hay que recordar que si bien una interfaz puede ser estática, los usuarios, dada su condición humana, suelen evolucionar con el tiempo, pasando de novatos a expertos, de sanos a enfermos, de relajados a tensos, etc. es decir, modificando los parámetros con los que actúan sus sistemas cognitivo, perceptivo y motriz.

Si se desea lograr una adaptación efectiva de la sintaxis de una interfaz, ésta debe ser generada de forma dinámica y no de forma estática. Los parámetros que determinan la configuración del proceso de generación dinámica de la interfaz han de depender del tipo de usuario de la aplicación, del dominio del problema que ésta desea solucionar y de la tarea que aquel pretende llevar a cabo, siendo evidente la necesidad de emplear agentes interactivos inteligentes capaces de obtener conocimiento acerca de estos tres parámetros en tiempo real. Serán estos agentes los encargados de simular el comportamiento de los diseñadores humanos a la hora de generar interfaces de usuario de una forma dinámica y automática.

Para que un sistema de gestión de interfaces de usuario fuese capaz de resolver el problema descrito, éste debería alcanzar al menos los siguientes objetivos.

3.2.1. Generación Dinámica de los Diálogos Interactivos

A diferencia de los UIMS recogidos en los capítulos 1 (*UIMS*) y 2 (*Interfaces Inteligentes*), un sistema de gestión de interfaces adaptable debería ser capaz de generar todos los elementos que componen su interfaz en tiempo real y en función del usuario actual de la aplicación. Sólo así podrá garantizarse un nivel de adaptación aceptable.

Hay que destacar que esta funcionalidad no garantiza por si sola el éxito del proceso de adaptación, dado que el modelo mental empleado para crear la interfaz sigue correspondiendo al modelo mental del diseñador del UIMS, solo que ahora este modelo mental se encuentra codificado dentro del agente interactivo inteligente encargado del desarrollo de los elementos de la interfaz. Sin embargo es necesario reconocer que este modelo mental del diseñador es mucho más flexible que un modelo estático dado que puede adaptarse y modificarse en base a una serie de parámetros proporcionados por el usuario real de la aplicación y por lo tanto se da pie a que éste usuario proporcione parte

de su visión del dominio del problema, es decir, parte de su modelo mental. Por medio de este enfoque, será posible fusionar –con mayor o menor fortuna– los modelos mentales de los dos entes implicados, tanto del diseñador como del usuario, reduciendo el grado de frustración de éste último, al menos si se compara este grado con el alcanzado al emplear una interfaz estática.

3.2.2. Adaptación Automática de los Niveles Léxico y Sintáctico

Para garantizar que la generación automática y dinámica de los elementos constitutivos de una interfaz se efectiva desde el punto de vista de la usabilidad, será necesario que la interfaz generada se aproxime lo más posible al modelo mental del usuario. Al respecto cabe decir que resulta del todo imposible conseguir que esta aproximación sea exacta en los niveles cognitivos superiores, relacionados con el nivel conceptual y semántico de una interfaz, sin tener un conocimiento profundo del dominio del problema, conocimiento que no es posible modelar de forma genérica para cualquier aplicación. Tal y como se ha visto en los capítulos 1 (*UIMS*) y 2 (*Interfaces Inteligentes*), este conocimiento se puede modelar por medio de diversas técnicas (diagramas de transición de estados, redes conceptuales, etc.) que tienen en común el hecho de necesitar una definición explícita por parte de los diseñadores de la aplicación.

Es por ello que de momento, resulta inapropiado realizar una adaptación a estos niveles dado que el grado de especialización del sistema a modelar por cada aplicación requiere de una solución específica sobre la que los agentes inteligentes generadores del interfaz no pueden trabajar. Sin embargo, dado que los niveles léxico y sintáctico de una interfaz no dependen directamente del dominio del problema, sino de las características intrínsecas del usuario real de la aplicación (al que se supone conocido dado su contacto directo con la misma), es posible diseñar mecanismos que permitan la automatización del proceso de adaptación de estos niveles. Este es precisamente el objetivo primordial de cualquier sistema adaptable.

Como ya se ha comentado a lo largo de este capítulo, un proceso de adaptación efectivo de estos niveles léxico y sintáctico permitiría ampliar acceso a la interfaz de la aplicación aun amplio rango de usuarios entre los que se encuentran todos aquellos con algún grado de discapacidad física, además de aumentar el grado de convergencia del modelo mental de otro tipo de usuarios al modelo mental con el que se diseña la aplicación.

3.2.3. Actualización Continua del Modelo de Usuario

La potencia del proceso de adaptación léxico-sintáctico dependerá por un lado solidez del modelo cognitivo en el que se basan las funciones de adaptación y por el otro lado de la precisión de los parámetros empleados por estas funciones, todos ellos provenientes del propio usuario. Es por ello que resulta fundamental un mecanismo

efectivo para la obtención de información de primera mano acerca de las características cognitivas, perceptivas y motrices del usuario.

Aunque parte de esta información puede de forma explícita mediante preguntas directas al usuario o mediante el empleo de un dossier de preferencias, la mayoría de estos parámetros solo pueden recogerse de forma implícita mediante la observación directa del comportamiento seguido por el usuario en la realización de determinadas tareas. Así por ejemplo, la precisión motriz del usuario al desplazar el puntero por la pantalla o su velocidad de tecleado son parámetros que solo pueden obtenerse de forma explícita ya que pedir estos valores al usuario de forma explícita resulta un completo sin sentido. En todo caso, en aras de alcanzar un buen nivel de usabilidad y para hacer que la interfaz resulta lo más atractiva posible, se debería evitar la obtención de parámetros de forma explícita, haciendo que el proceso de actualización del modelo de usuario resulte transparente al usuario.

Dado que las características perceptivas y motrices de un usuario suelen variar considerablemente con el tiempo, debido a la fatiga o al estrés –entre otros factores– se hace estrictamente necesario que el proceso de observación descrito se efectúe con relativa frecuencia dentro del marco temporal de una sesión de interacción, permitiendo de este modo la adaptación de los elementos de la interfaz a medida que se modifica el estado perceptible de los parámetros empleados en dicho proceso de adaptación.

3.2.4. Separación Efectiva entre Funcionalidad e Interfaz

Si la información necesaria para realizar el proceso de adaptación depende necesariamente del usuario, la información necesaria para confeccionar cada uno de los diálogos interactivos que forman parte de la interfaz, así como de los procesos de usuario que han de ofrecerse al usuario a través de dicha interfaz tienen que provenir necesariamente de la aplicación. Es por ello que resulta imprescindible el diseño de algún mecanismo para intercambiar información entre la aplicación y el usuario empleando al sistema de gestión de interfaces de usuario como intermediario.

Este mecanismo de separación debería obtener de la aplicación la relación de todos aquellos procesos de usuario disponibles para la realización de las tareas para las que la aplicación ha sido diseñada, así como aquel conjunto de condicionantes que definen cuando y cómo los mencionados procesos estarán disponibles para su empleo por parte del usuario. Naturalmente, para la correcta ejecución de cada uno de estos procesos será necesaria la participación del usuario, bien para recibir la información aportada por los mismos, bien para proporcionar información requerida por ellos, o bien para realizar ambas tareas de forma simultánea en un intercambio continuo de información entre la aplicación y el usuario a través de los diálogos interactivos diseñados dinámicamente por los agentes inteligentes. Es en este punto donde el mecanismo de separación debe jugar su papel fundamental, sirviendo de negociador entre los dos entes implicados en el proceso de

comunicación (usuario y aplicación), realizando las oportunas traducciones entre los diferentes lenguajes empleados aquellos, de tal modo que dicha comunicación sea efectiva.

3.3 GADEA

De acuerdo con los objetivos recogidos en el apartado anterior se ha desarrollado GADEA, un marco conceptual general para el desarrollo de interfaces adaptables a bajo nivel (léxico y sintáctico). GADEA actúa como un sistema de gestión de interfaces de usuario capaz de crear múltiples versiones de una interfaz en tiempo real y de forma automática. Esta interfaz es generada dinámicamente en función de las características cognitivas, perceptivas y motrices del usuario destino de la aplicación a la que pertenece la interfaz. De este modo, GADEA es capaz de proporcionar un acceso universal a la interfaz de cualquier aplicación por medio de mecanismos de adaptación automáticos especialmente diseñados para satisfacer las necesidades de interacción de usuarios con diferentes grados de discapacidad física, visual y auditiva. Estos grados pueden ir desde la ausencia de discapacidad hasta un grado total de discapacidad representado por la ceguera o la sordera.

Aunque en un principio GADEA ha sido diseñado para convertirse en el sistema de gestión de interfaces de usuario del sistema operativo Oviedo-3 [Álvarez y Tajes (1997)], un sistema operativo desarrollado integralmente con tecnologías orientadas a objetos, dado que GADEA ha sido desarrollado integralmente con tecnología Java, este UIMS puede emplearse para suplantar al *toolkit* alternativo de varias plataformas operativas tales como Windows, Linux o Macintosh. Más importante aún es el hecho de que este UIMS puede servir y de hecho está sirviendo para el desarrollo de aplicaciones de portales de comercio electrónico de interfaz adaptable, así como para aplicaciones hipermedia distribuidas en Internet. El único requisito exigible para las aplicaciones cliente de este UIMS consiste en que dichas aplicaciones han de haber sido escritas en lenguaje Java, dado que tal y como se verá a partir del capítulo 5 (*Diseño Centrado en el Usuario*) uno de los mecanismos clave empleados por este UIMS para separar la funcionalidad de una aplicación de su interfaz descansa sobre la API de reflexión proporcionada por esta plataforma.

3.3.1. GADEA como Sistema Experto

GADEA está basado en el paradigma clásico de los sistemas expertos, emulando el comportamiento de un agente humano que establece un diálogo interactivo con otro humano por medio de canal de comunicación multimodal [Wahlster (1991)]. A este agente inteligente se le supone la capacidad de poder seleccionar el estilo de interacción más adecuado para establecer la comunicación de entre todos aquellos disponibles, adaptando el modo de acceso al mismo así

como la apariencia de sus contenidos en función de las características cognitivas, perceptivas y motrices únicas de aquél con quien dialoga.

La Figura 6 muestra el esquema de diseño clásico para un sistema experto [Velarde (1991a)]. Como se puede apreciar, el experto humano proporciona conocimiento acerca del dominio del problema por medio de un ingeniero de conocimiento, el cual se encarga de convertir la experiencia del experto en el conjunto de hechos y reglas que constituyen la base de conocimiento del sistema experto. Estos hechos y reglas serán empleados mas tarde por el motor de inferencia y por el módulo de adquisición de conocimiento para obtener nuevo conocimiento acerca del dominio del problema.

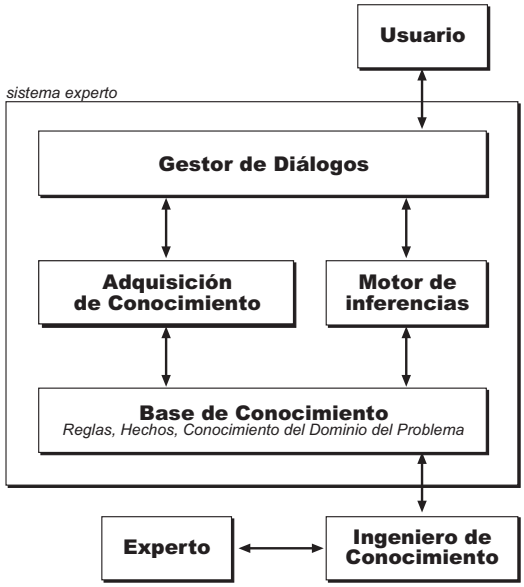


Figura 6. Esquema clásico para el diseño de un sistema experto.

En un sistema de gestión de interfaces de usuario ideal, el conocimiento proporcionado por este experto humano incluiría el estado del arte en las técnicas de interacción y comunicación humana, así como las guías de diseño específicas para la plataforma de desarrollo destino de la interfaz dinámica. En la literatura abundan las fuentes para este tipo de conocimiento, incluyendo las guías de diseño de Microsoft (2001) y Apple (1991) para el desarrollo de interfaces para las plataformas Windows y Macintosh respectivamente, o las guías de Yale [Lynch y Horton (2001)] o de Nielsen (2001) para el desarrollo de documentos web.

Sin embargo, algunas de estos principios generales de diseño tienen a realizar demasiadas suposiciones genéricas acerca de las características cognitivas, perceptivas y motrices del usuario de esa aplicación. Desafortunadamente, estas suposiciones restringen el uso de la interfaz a un pequeño grupo de los posibles usuarios de una aplicación. Las guías de diseño visual caen dentro de esta categoría. Así por ejemplo, en la guía de Nielsen (2001) encontramos que en un documento web los enlaces sirven como un mecanismo para destacar contenidos, siendo las variaciones de color y de la fuente tipográfica otro mecanismo aplicable con el mismo fin. Por su parte, en la guía de

Yale [Lynch y Horton (2001)] nos encontramos que los usuarios inician su búsqueda de información por los gráficos y solo tras un pequeño lapso de tiempo empezarán a buscarla por un medio menos digerible como es ele escrito. Aunque las dos reglas son extremadamente útiles para diseñar la jerarquía visual de un documento web, éstas resultan completamente inútiles cuando el documento diseñado va a ser empleado por un usuario con algún tipo de discapacidad visual.

Como ya se ha comentado, un sistema de gestión de interfaces de usuario inteligente requerirá que sus reglas y hechos se encuentren especialmente diseñadas para permitir un uso óptimo de sus mecanismos de interacción por parte de usuarios con características cognitivas, perceptivas y motrices especiales. Mientras que en un sistema experto clásico ,es el experto humano el que proporciona el conjunto inicial de hechos y reglas, en un sistema adaptable sin embargo

Dado que GADEA no puede fundamentar la toma de decisiones sobre generalizaciones abstractas acerca de sus posibles usuarios, este sistema experto no puede hacer uso de este tipo de reglas y hechos estáticos, necesitando en cambio equivalentes flexibles que se puedan adaptar a las necesidades concretas de un usuario dado. Como resultado, los hechos y reglas genéricos proporcionados por el experto humano han de ser complementados con hechos y reglas acerca de la propia individualidad del usuario final de la aplicación, información que será aportada por los mismísimos usuarios, puesto que estos entes son aquellos que poseen el mejor grado de conocimiento posible acerca de sus propias características. Dado que debido a razones obvias, los usuarios no puede aportan la información requerida en tiempo de diseño, lo tendrán que hacer en tiempo de real. Para ello, el papel jugado por ingeniero de conocimiento, (encargado de convertir la experiencia del experto humano en reglas y hechos) será representado ahora por agentes software inteligentes.

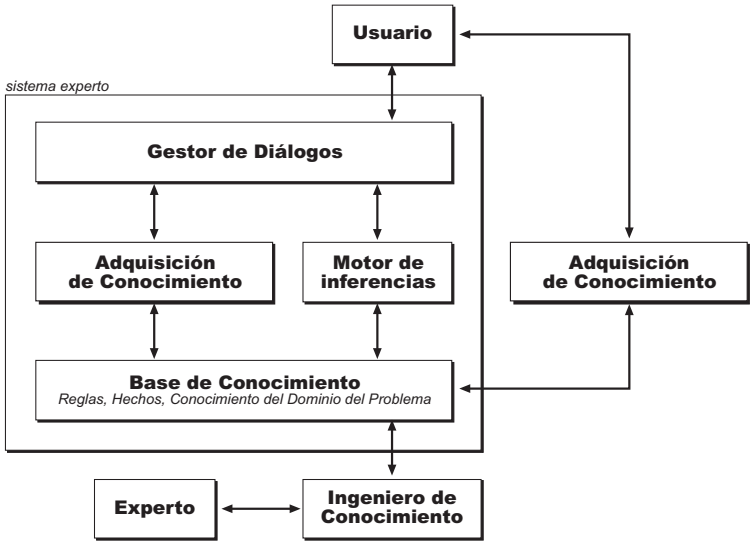


Figura 7. Esquema clásico de un sistema experto modificado para adoptar el enfoque conceptual de GADEA.

La Figura 7 representa la pequeña modificación realizada sobre el esquema general de un sistema experto para incorporar un módulo de adquisición de conocimiento experto capaz de obtener nuevos hechos acerca del usuario, incluyéndolos en la base de conocimiento de la interfaz (el modelo de usuario). Estos hechos comprenden todo el conocimiento posible acerca de las características de bajo nivel de los sistemas cognitivo, perceptivo y motriz del usuario. Estos hechos incluyen características tales con la edad y sexo del usuario, la precisión de sus sistema perceptivo (vista, oído, etc.) la precisión de su sistema motriz (en tareas de desplazamiento del ratón, pulsación de teclas, etc.) su tiempo medio de reacción, etc.

Toda esta información puede ser obtenida explícitamente por el módulo de adquisición de conocimiento externo por medio de preguntas directas (del tipo *de ¿eres zurdo o diestro?*) o de forma implícita por medio de un análisis cuidadosa del tipo de acciones realizadas por el usuario a lo largo de sus sesiones de interacción con el sistema. De todos modos, los usuarios pueden aportar información directa acerca de sus preferencias por medio de mecanismos de acceso directo al modelo de usuario.

3.3.2. Diseño General de GADEA

El diseño de GADEA está formado por tres módulos completamente independientes que trabajan en conjunto para alcanzar los objetivos previstos. Se trata de *CodeX (Code Xplorer*), DEVA (*Discourse Expert Valuator for Adaptation*) y ANTS (*Automatic Navigability Testing System*). La Figura 8 muestra estos módulos, sus componentes auxiliares y las relaciones entre los mismos.

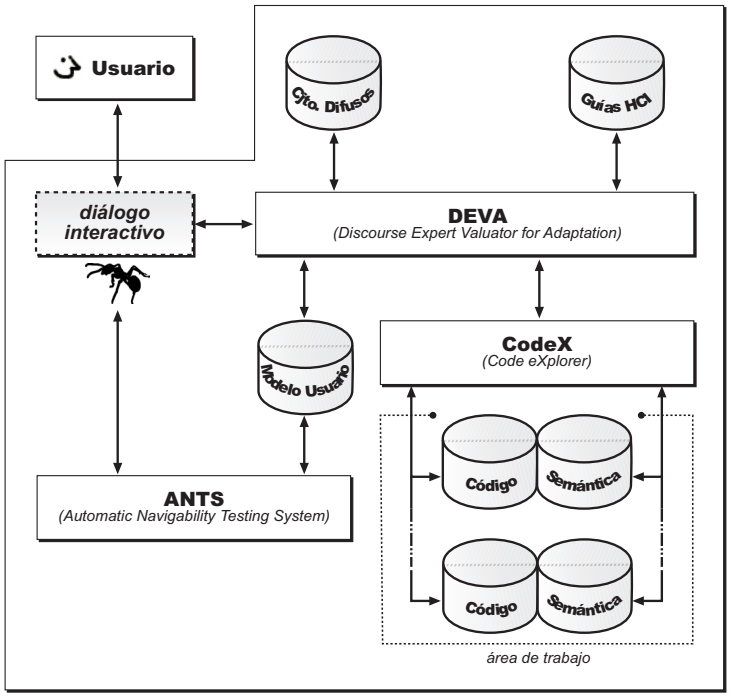


Figura 8. Diseño general de GADEA.

El módulo CodeX representa la interfaz de conexión entre GADEA y sus aplicaciones, encargándose de convertir los requerimientos de interacción del usuario en llamadas a métodos específicos de un objeto de la aplicación. Este módulo se basa en los mecanismos de reflexión de Java para examinar el código binario de una aplicación (los ficheros *class* de Java) de forma automática en busca de los procesos de usuario definidos por los programadores en tiempo de edición. La información obtenida es enviada al módulo DEVA, el cual constituye el sistema experto de GADEA. Por medio de los agentes software inteligentes incluidos en este módulo, DEVA convierte los requisitos de interacción del usuario o de la propia aplicación en un diálogo interactivo adaptado a las necesidades del usuario. Basado en el conocimiento general proporcionado por el supuesto experto en interacción y comunicación humana así como en la información específica almacenada en el modelo de usuario, DEVA emplea un motor de inferencias basado en lógica difusa para evaluar diferentes alternativas para el diseño del diálogo interactivo, seleccionado aquella que mejor se adapte a las características cognitivas, perceptivas y motrices del usuario. Por último, el módulo ANTS hace uso de diferentes familias de agentes remotos para observar el comportamiento del usuario en cualquiera de los diálogos interactivos generados dinámicamente por DEVA con el objeto de obtener información acerca de las características particulares del usuario, las cuales serán almacenada en el modelo de usuario.

Como se puede apreciar, el diseño general de GADEA se inspira en parte en el Modelo Seeheim. En este caso, las capas de presentación y de control de diálogos se encuentran integradas en el módulo DEVA, mientras que la capa de interfaz de aplicación es gestionada por el módulo CodeX. Sin embargo, a diferencia del Modelo Seeheim, el UIMS GADEA incorpora un tercer módulo (ANTS) encargado de garantizar la correcta actualización de los valores almacenados en el modelo de usuario.

4

4 La Comunicación

La selección lógica de la información suministrada a cada miembro del equipo de trabajo es la característica no solamente de un buen sistema sino de un gran sistema.

Leslie H. Matthies

4.1 Paradigma de la Comunicación

El término interacción y comunicación humana (ICH), derivación del antiguo Comunicación Hombre Máquina (CHM) y referido por algunos como Interacción Persona Ordenador (IPO) lleva implícito en su nombre el objetivo básico y fundamental de esta disciplina, es decir, el establecimiento y mantenimiento de un proceso de comunicación entre dos entes. Resulta obvio que no podrá desarrollarse una interacción efectiva entre dos objetos, si previamente dichos objetos no han establecido un canal de comunicación apropiado para el intercambio de información, así como la especificación de los protocolos y del lenguaje necesario para dicho intercambio. Evidentemente para que una aplicación pueda definirse como interactivo, dicha aplicación tendrá que situarse en el contexto de un sistema de comunicación en el que sea posible el intercambio de información y mensajes entre la funcionalidad de la misma y sus usuarios.

El marco conceptual en el que se desarrolla GADEA parte de la existencia de un sistema de comunicación más o menos complejo (en función del número de entes implicados así como de la relación ente los mismos) en el que se sitúan diferentes componentes que, haciendo las veces de moduladores, se encargan de traducir y de adaptar diferentes aspectos de los canales de comunicación empleados, así como de la información transmitida en áreas de potenciar la calidad de los mensajes que pululan por el mencionado sistema. Al abordar la problemática del diseño de interfaces desde el punto de vista de la comunicación, es posible enfocar los diferentes aspectos de diseño y construcción desde un punto de vista global y integrador que permite emplear una terminología común para los diferentes aspectos, la cual está basada en la jerga de los sistemas de transmisión de datos.

Otra importante ventaja obtenida al realizar el análisis y diseño de las interfaces desde el punto de vista de la comunicación, consiste en poder abstraer el problema en términos de la conocida *Teoría General de Sistemas* (TGS) pudiendo aplicar con ello todas sus técnicas de análisis, en especial las de modelado y predicción de sistemas, así como todos los enfoques metodológicos basados en la misma. Los dictados de esta teoría han sido empleados con éxito en la creación enfoques y modelos conceptuales para varios campos de la interacción y comunicación humana entre los que cabe señalar el desarrollo de cursos a distancia [Vos (2000)] o la substitución de espacios de trabajo por espacios de comunicación en la metáfora empleada por algunas aplicaciones [Reynolds (1988)].

De acuerdo con el enfoque de la teoría general de sistemas, todos los objetos que participan en el proceso de comunicación de un sistema interactivo no son más que componentes de un nivel de abstracción superior (el sistema) que los engloba a todos. De acuerdo con este paradigma, la comunicación entre componentes representa precisamente la relación existente entre componentes, es decir, el material aglutinante que permite que todos estos objetos puedan pertenecer a un sistema. De acuerdo con el enunciado fundamental de esta teoría, *el todo es siempre más que la suma de sus partes*, *el todo* de un sistema interactivo estará formado por la suma de todas sus *partes* (hardware, software y recursos humanos) así como de la relación existente entre estas partes, es decir, la comunicación entre los recursos empleados.

De acuerdo con este enfoque conceptual, la parte humana de un sistema de comunicación –integrada por todos y cada uno de los usuarios del sistema interactivo– también es candidata a ser considerada como un componente más de dicho sistema. De este modo es posible situar a los distintos usuarios de un sistema en un nivel de abstracción idéntico al del resto de los componentes del sistema, tales como los dispositivos hardware o las distintas jerarquías de objetos que forman parte de los componentes software interactivos. Esta nivel de abstracción permite definir clara y unívocamente los requisitos funcionales e interactivos de todos y cada uno de los usuarios requeridos para el buen funcionamiento de cualquier tipo de sistema. De este modo y al igual que ocurre con cualquier tipo de componente,

es posible especificar las características técnicas de todo usuario humano (a un nivel cognitivo, perceptivo y motriz) así como especificar las condiciones en las que éste componente debe ser empleado, de tal modo éste pueda ser ubicado exactamente en el lugar que debe corresponderle dentro del sistema (siguiendo un modelo de roles por ejemplo [Muller (1997)]).

Desde este marco conceptual se puede estudiar cualquier sistema informático –interactivo o no, construido o por construir– analizando su rendimiento por medio de técnicas de medición o mediante mecanismos de simulación con el objetivo fundamental de elevar dicho rendimiento al máximo, permitiendo así que el sistema pueda alcanzar el máximo de su capacidad teórica. En este sentido y a la hora de diseñar un sistema interactivo, en un primer nivel de abstracción no es necesario practicar ninguna diferencia importante entre componentes humanos y no humanos, puesto que en las primeras fases de análisis y desarrollo tan solo es necesario identificar todos los componentes del sistema así como establecer los requisitos de comunicación entre dichos componentes. Evidentemente, en fases de desarrollo posteriores en donde son necesarios niveles de abstracción más complejos, será el momento de destacar las peculiaridades de los componentes humanos con respecto a otro tipo de componentes para explotarlas al máximo función del objetivo global del sistema al que pertenecen.

4.1.1. Especificación de Objetivos

Como se puede observar, en el ámbito de aplicación de este modelo de comunicación, los usuarios de un sistema son concebidos como recursos a utilizar en beneficio del objetivo del sistema, cualquiera que este sea, incluyendo por su puesto alcanzar el mayor grado de satisfacción posible en la interacción llevada a cabo por sus recursos humanos. Uno de los aspectos más relevantes e importantes de este enfoque conceptual radica en la flexibilidad del modelo, ya que con él es posible modelar cualquier sistema –sea éste informático o no– y especificar diferentes jerarquías de objetivos para el mismo, situando el grado de detalle requerido bien en el sistema como entidad global, bien en alguno de los subsistemas que lo constituyen o bien sobre alguno de sus componentes. Así, retomando el ejemplo del aeropuerto empleado en el capítulo 1 (*UIMS*), éste puede ser considerado como un sistema de comunicación más o menos complejo en donde el objetivo primordial consiste en que la información proporcionada llegue al usuario adecuado en el momento adecuado. Éste objetivo general puede ser descompuesto a su vez en una serie de objetivos base, creando así una jerarquía de objetivos a cumplir para conseguir el objetivo general. De este modo, el objetivo asignado a las diferentes estaciones de atención al usuario puede ser el diseño de una interfaz intuitiva capaz de satisfacer las necesidades de información de usuarios provenientes de distintos entornos culturales, mientras que por el contrario, el objetivo marcado para las pantallas de radar de los controladores aéreos debería ser la fiabilidad y sobre todo, la reducción de la fatiga de los mismos.

Hay que destacar que el concepto de sistema manejado en este marco conceptual es tan amplio que puede ser cualquier combinación de componentes digna de tal calificativo, desde un sistema informático, en donde los recursos humanos hacen las veces de usuario, hasta por ejemplo, la sociedad de votantes de un Estado, en donde el los recursos humanos. hace las veces de meras fuentes puntuales de información. A diferencia de otros enfoques conceptuales para el diseño de interfaces de usuario en donde los esfuerzos están encaminados única y exclusivamente a facilitar la interacción del usuario con un sistema informático, en este modelo este es tan solo uno más de los objetivos que se pretenden alcanzar. Para ello es necesario analizar las características especiales del comportamiento humano (especificación del componente) para una determinada situación con el objeto de determinar aquel entorno en el cual se pueden potenciar sus cualidades innatas, beneficiando así al rendimiento global del sistema, mejorándolo por medio de una optimización en la comunicación entre sus componentes constitutivos, pudiendo ser éstos de cualquier naturaleza.

Es necesario hacer hincapié en que esta nueva propuesta engloba perfectamente a los tratamientos tradicionales de la interacción y comunicación humana puesto que se sigue estudiando toda posible mejora de la interacción entre el usuario y el resto del sistema informático. La diferencia estriba en que esa mejora ya no es un objetivo en sí mismo, sino que forma parte de objetivos más ambiciosos. Así por ejemplo, en un sistema interactivos de control de catástrofes del tipo de CHARADE o CHORIS (ver capítulo 1 y 2 respectivamente) el objetivo que a principal a alcanzar es la rápida y efectiva coordinación entre equipos de salvamento, quedando la satisfacción del usuario con respecto a la interfaz en un segundo plano. En un nivel secundario quedará esta satisfacción en un sistema interactivo de control de tráfico aéreo del tipo de CUBRICON (ver capítulo 2: *Interfaces Inteligentes*) en donde todas las posibles distracciones deben eliminarse para conseguir el objetivo de obtener el mayor grado de concentración posible por parte del usuario. No obstante, este objetivo de alcanzar la máxima satisfacción del usuario es el que posee la máxima prioridad en sistemas interactivos orientados a la diversión (juegos, portales de ocio en Internet, etc.) o a la venta de un determinado tipo de software. Como se puede apreciar, los objetivos ideales de todo software interactivo creado con tecnología cognitiva (satisfacción, diversión, productividad, facilidad de uso, seguridad, fiabilidad, etc.) no siempre han de tener el mismo grado de prioridad y éste enfoque conceptual se encarga de definir establecer y formalizar esa jerarquía de prioridad.

Dado que los recursos humanos se comportan como cualquier otro recurso del sistema, a éstos se le podrá aplicar distintas políticas de gestión con el ánimo de optimizar su empleo. Estas políticas suelen tener como objetivo reducir el tiempo que cada recurso pasa inactivo o combatir activamente las situaciones en las que se presentan cuellos de botella en el sistema, es decir cuando la capacidad limitada de un recurso incide negativamente sobre el rendimiento global de dicho sistema. La idea latente en toda política de planificación de recursos es la de distribuir de modo equitativo la carga de trabajo total del sistema

entre todos los recursos disponibles, dependiendo de la capacidad individual de cada uno de estos recursos.

Supongamos por ejemplo, el caso de un sistema para el diseño y edición de publicaciones periódicas en donde se dispone de editores, maquetistas, ilustradores y fotógrafos como recursos humanos, los cuales comparten espacio y responsabilidades con el resto de los recursos hardware y software del sistema. Supongamos además que el objetivo principal de este sistema consiste en alcanzar la edición del mayor número de páginas posible por unidad de tiempo, sin perder con ello la calidad del producto generado. Para lograr este objetivo, el ingeniero de sistemas tendrá que planificar la cantidad y tipología de los componentes hardware (impresoras, rotativas, ordenadores, etc.), software (editores de imágenes, de textos, de maquetas, etc.) y humanos (maquetistas, redactores, ilustradores, fotógrafos, etc.) a emplear al objeto de lograr un adecuado balance de la carga de trabajo para todos ellos. El diseñador del sistema deberá tomar en cuenta la capacidad individual de todas los tipos de componentes, de tal modo que no existan situaciones en las que unos se encuentren saturados de trabajo mientras que otros se encuentren ociosos. En este ejemplo, la política de administración debería calcular el número exacto del personal técnico involucrado en todos sus departamentos, así como la características de cada uno de ellos (relación tiempo/calidad) para garantizar que todos ellos trabajen el mismo número de horas con un nivel de desarrollo pleno. Obviamente, todo ello redundará en beneficio del rendimiento del sistema.

4.1.2. Las Interfaces como Adaptadores

Esta estrategia de diseño no solo permite definir las características de todos y cada uno de los recursos implicados (humanos, hardware y software) sino también la de las interfaces de comunicación entre los mismos, es decir, las características que ha de tener el punto de contacto o de conexión entre los diferentes componentes individuales del sistema; con el objeto de lograr un ensamblaje perfecto. Empleando la metáfora de una instalación eléctrica, este modelo permitiría definir las característica que han de tener los enchufes, cables y adaptadores para permitir ensamblar entre sí los distintos aparatos eléctricos que forman parte de la instalación (portalámparas, contadores, pulsadores, etc.), así como las características que éstos han de poseer (impedancia, voltaje, potencia, etc.) para que todo funcione correctamente.

En la Figura 9 ha representado la abstracción de un sistema de comunicación, del que se han eliminando los adaptadores necesarios para conectar entre sí dispositivos software y hardware (dispositivos representados por un ordenador). Los rombos representan los adaptadores necesarios para conectar entre sí los distintos componentes del sistema interactivo, los cuales pueden ser componentes humanos (usuario) o componentes máquina (software o hardware).

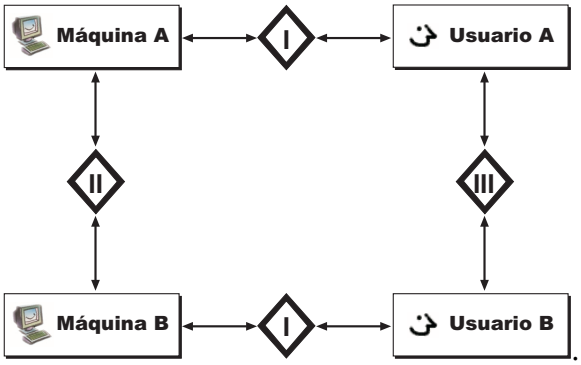


Figura 9. Diferentes tipos de adaptadores (rombos I, II y III) en un sistema de comunicación.

En la mencionada Figura 9 se recogen todas las combinaciones posibles de conexiones entre componentes, pudiendo establecerse éstas entre componentes humanos (III) , entre componentes máquina (II) o entre un humanos y máquinas (I). En todas ellas existe un adaptador (representado por un rombo) encargado de realizar la correspondiente traducción entre el modelo conceptual empleado para el diseño de cada componente, es decir, se trata de interfaces. Todas estas interfaces son –o pueden ser– diseñadas desde la óptica de la interacción y comunicación humana, dado a que a fin de cuentas, todos los componentes de un sistema son de naturaleza humana, dado que o bien son humanos o bien son dispositivos diseñados y construidos por humanos. Es por ello que el problema a resolver al diseñar estos conectores o interfaces, es un problema de comunicación entre humanos. Aunque la atención de la interacción y comunicación humana se ha centrado en el diseño de las interfaces entre humanos y máquinas, los otros dos tipos de interfaces no han pasado desapercibidos para investigadores de esta disciplina como es el caso de Maguire (1994) en el diseño de interfaces de comunicación efectivas entre componentes software o para Norman (1998a) entre otros muchos psicólogos en el caso del estudio de la comunicación entre humanos, así como del diseño de interfaces de adaptación (traducción de lenguajes) entre estos dos entes.

En la Figura 10, el logotipo de GADEA substituye al adaptador de comunicación entre componentes humanos y máquinas (I), indicando el ámbito de trabajo en el que se mueve este sistema de gestión de interfaces de usuario. A diferencia de los conectores I, a los que substituye (ver Figura 9), y que permitían adaptar la comunicación entre los usuarios A y B y sus máquinas homónimas, GADEA pretende comportarse como una suerte de adaptador universal dotado del grado de flexibilidad suficiente como para acomodarse a los requisitos de comunicación de sus dos extremos, es decir, del lado de la máquina (software de aplicación) y del lado del humano (el usuario o posibles usuarios).

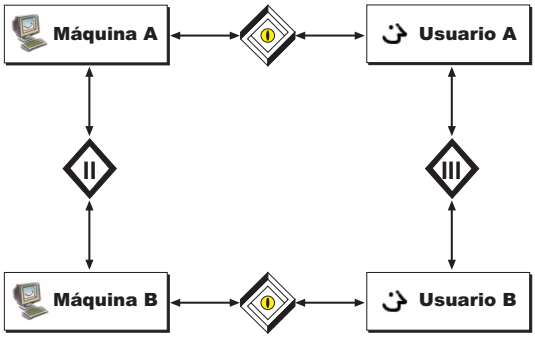


Figura 10. Ámbito de aplicación de GADEA dentro de un sistema de comunicación.

4.1.3. Canales de Comunicación

Todo proceso de comunicación implica necesariamente la transmisión de información codificada en forma de mensajes, desde un emisor hasta un receptor, por medio de un canal de comunicación. Por canal de comunicación se entiende el mecanismo físico establecido para el intercambio de las señales que forman el mensaje. Un canal de comunicación puede ser de diferentes tipos, como por ejemplo, señales visuales y auditivas, conversaciones orales o incluso escritas, como es el caso de este manojo de hojas, el cual sirve de canal de comunicación entre un emisor (el autor) y uno o varios receptores (quienquiera que lea estas líneas).

En una interfaz de usuario clásica, cada ventana, cuadro de diálogo, menú desplegable, etc. representa un canal de comunicación tendido entre un componente perteneciente a la aplicación y un usuario. Otros canal de comunicación empleados por las aplicaciones pueden ser el medio sonoro a través del cual se pueden transmitir mensajes orales y música de fondo, canal de comunicación éste, empleado con cierto grado de asiduidad por parte de aplicaciones multimedia y vídeo juegos, en donde juega un papel fundamental, dado que incide directamente sobre procesos cognitivos relacionados con la atención humana y el envío de mensajes subliminales. Ejemplos de canales de comunicación para sistemas no informáticos se pueden encontrar en las tramas visuales y auditivas de las películas, en los carteles, en la radio o incluso en los gestos empleados en la comunicación directa entre humanos.

Naturalmente, la naturaleza de cada canal de comunicación, así como la de los contenidos que se pueden transmitir a través del mismo, ofrece distintas posibilidades para la comunicación, las cuales han de tenerse en cuenta antes de optar por un determinado canal. Así por ejemplo, el uso imágenes es idóneo para transmitir grandes cantidades de información, pero tienen el defecto de poseer, grandes dosis de ambigüedad, fenómeno que no se reduce en el caso de los mensajes escritos. Por su parte el vídeo y las animaciones son idóneos para transmitir una secuencia temporal de acontecimientos o para comunicar informaciones difíciles de explicar por medio de las palabras pero requieren de gran ancho de banda, luego no son aconsejables para

transmitir grandes cantidades de información. El sonido por su parte, es ideal para enfatizar mensajes, pero tiene el defecto de ser un canal de comunicación secuencial y de percibirse a través de un sentido sensorial muy poco preciso. Como regla general, será el sistema en estudio y el estado de la tecnología quien dicte los canales de comunicación disponibles, siendo el diseñador del sistema de comunicación (sea éste un humano o un agente software inteligente) quien realice la elección final en función de las características de cada uno de estos canales de comunicación.

Cuando se analizó el caso del UIMS AUI-GUIDE (ver apartado 2.2.2) quedó patente que no deben existir restricciones en cuanto al número de canales de comunicación a emplear de forma simultánea entre dos entes. De hecho, el empleo de una comunicación multimodal y multicanal amplía la probabilidad de que un mensaje sea captado e interpretado correctamente por el receptor en situaciones en las que el conocimiento que el emisor dispone acerca de su receptor no es muy preciso. El empleo de mecanismos de transmisión multicanal posibilita la clasificación de los mensajes a transmitir en diferentes niveles de prioridad, permitiendo la asignación de diferentes canales de comunicación para cada una de las prioridades definidas. Para ello se transmite de forma simultánea distintas ideas por medio de varios canales secundarios subordinados a un canal principal, todo ello de una forma sincronizada. De hecho, tal y como se verá a partir del capítulo 12 (*El Modelo Cognitivo*), GADEA hace uso de esta técnica para transmitir información de carácter complementario empleando un canal de comunicación secundario sobre el que no se encuentra centrada la atención del usuario.

En este enfoque conceptual adoptado por GADEA, la elección y posterior configuración de los canales de comunicación a emplear, se realiza en un nivel de abstracción lo suficientemente bajo como para ocultar los detalles físicos de la transmisión, permitiendo así la realización de un eventual cambio de canal, substituyendo el canal empleado, por otro de características similares, en tiempo real. Esta característica es importante dado que el número y tipología de los canales de comunicación disponibles solo se conocerá en tiempo de ejecución. Además, este enfoque permite que la definición de los canales de comunicación, así como de la información a transmitir por ellos se encuentre en un medio libre de contexto, y por lo tanto éste se encuentre libre de las ataduras impuestas por cualquier posible interfaz. Así por ejemplo, si por algún terrible casual, el estado futuro de la tecnología cognitiva determina que el rendimiento de determinados usuarios de un sistema pueden ser estimulados mediante la transmisión de una serie cifras terminadas en el dígito cinco, se podrá aplicar este extraño descubrimiento de una forma totalmente independiente de la construcción física de los canales, pudiendo transmitir las citadas cifras visualmente mediante ventanas, acústicamente mediante mensajes hablados, de forma táctil mediante mensajes impresos en Braille, etc.

4.1.4. La Semiótica y el Proceso de Decodifiación

La Ingeniería de Semiótica está basada en el estudio de la comunicación entre humanos [(De Sousa (1993)]. De acuerdo con esta disciplina, cuando una persona desea comunicarse con otra persona, ésta expresa sus ideas a través de un canal de comunicación mediante un código basado en símbolos, el cual ha de ser comprendido tanto por el transmisor como por el receptor. El receptor obtiene el mensaje y lo codifica [Norman y Draper (1986)], teniendo en cuenta que todo mensaje estará formado por uno o más signos.

Por ejemplo, para transmitir información acerca de un perro, tanto la palabra *perro* como el dibujo de un perro son signos que pueden representar a dicho animal. Para comprender lo que el emisor ha intentado transmitir, el receptor debe crearse una imagen mental del mensaje, la cual es también un signo, de tal modo que un signo puede generar otros signos [Oliveira et. al. (1997)]. Por ejemplo, si a alguien se le enseña la fotografía de un perro y esta persona piensa en la palabra *perro*, esta palabra actuará a la vez como modelo mental y como signo de lo que es un perro. Este proceso es conocido como *semiótica indefinida* y es la base de todo proceso de comunicación [Eco (1976)].

Para poder establecer este proceso de comunicación el diseñador del sistema debe crear una representación conceptual de la aplicación, codificando esta representación en un mensaje que representará el interfaz del usuario [De Sousa (1996)]. A su vez, cada usuario de la aplicación creará su propia representación mental de la interfaz, la cual representará en última instancia su modelo conceptual de la aplicación. De este modo es posible que el significado asignado para el mensaje representado por la interfaz difiera de un usuario a otro aunque en el fondo todas las representaciones deberían ser consistentes entre sí. [Oliveira et al. (1997)]

El planteamiento adoptado por la semiótica para el diseño de interfaces de usuario concibe las interfaces como mensajes enviados por los diseñadores de la aplicación a sus usuarios. Por medio de estos mensajes, los diseñadores notifican a los usuarios acerca de los problemas que le sistema es capaz de resolver y como éstos deben comunicarse con el sistema para resolver los problemas. En sistemas multiusuario el proceso es más complejo, ya que los diseñadores deben proporcionar información adicional para indicar a los usuarios como deben establecer comunicación entre ellos mismos [Oliveira et. al. (1997)].

Una vez que se ha establecido el canal de comunicación, el emisor y el receptor podrán iniciar el envío, recepción y decodificación de sus mensajes. En el caso concreto que nos ocupa, en un extremo del canal de comunicación se situarán los recursos hardware y software y en el otro se situarán los recursos humanos. El papel jugado por estos dos entes (informático y humano) podrá ser indistintamente el del emisor y/o receptor, por lo que habrá que dotar a dichos componentes de capacidades suficientes y adecuadas para la correcta codificación del mensaje a transmitir. De nada servirá por ejemplo la existencia de un canal de comunicación auditivo si al menos uno de los dos entes

participantes en el sistema comunicativo es sordo, bien en el caso de componentes humanos con deficiencias auditivas o bien en el caso de componentes informáticos carentes de mecanismos para la decodificación de los mensajes auditivos.

4.1.5. El Ruido

Dado que los canales de comunicación suelen ser medios abiertos, las señales que por ellos transitan pueden ser interferidas por el ruido. El ruido es una idea abstracta que representa la existencia de un proceso que tiene lugar en el medio de transmisión y que introduce información irrelevante para la señal transmitida. El hecho de que el receptor llegue a captar el mensaje transmitido, dependerá de su capacidad para distinguir las señales relevantes del ruido, con el objeto decodificar correctamente el primero y de obviar el segundo. En ocasiones una estimulación ruidosa puede ser tan fuerte y penetrante que no sólo impide el reconocimiento y procesamiento de las señales del mensaje, sino que incluso puede distraer la atención del usuario hacia los estímulos interferentes. Existen más posibilidades de que este fenómeno se produzca cuando el ruido constituye en sí mismo un conjunto potencialmente significativo de señales. Para evitar los efectos negativos del ruido, una interfaz adaptable como GADEA, deberá incrementar la proporción de la señal con respecto al ruido, bien incrementando la señal, o bien reduciendo los estímulos interferentes.

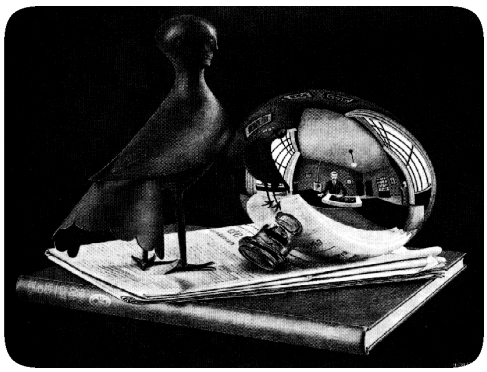
En el tipo de comunicación entre humanos y componentes informáticos, las principales fuentes de ruido suelen encontrarse en el mismo origen de la información. Si el origen de la emisión del ruido se sitúa del lado de los componentes informáticos, será posible para GADEA eliminar dicho ruido. Para ello es necesario que este sistema inteligente pueda planificar cuidadosamente la selección de los contenidos a incluir en cada uno de los distintos canales de comunicación que pueden estar activos en un momento determinado, evitando así la inclusión de información relevante junto elementos o que puedan saturar la capacidad de decodificación del usuario. Las claves para la selección correcta de esa información vienen dadas por un análisis cuidadoso del funcionamiento de procesos cognitivos superiores, como es el caso de la atención o la percepción, análisis que se incluye en los capítulos 14 (*La Atención*) y 15 (*El Procesador Perceptivo*) respectivamente.

Otro aspecto que se debe considerar al analizar el tratamiento dado el ruido, es la existencia de grandes cantidades de este elemento provenientes de canales ajenos al control del sistema de gestión de interfaces de usuario. En la mayoría de las ocasiones, no existe ningún mecanismo de defensa posible contra estas interferencias, tal es el caso por ejemplo de un simple timbre de teléfono. Obviamente ante una situación de este tipo, todos los canales de comunicación del sistema se verán invadidos por el ruido introducido por la señal emitida por el timbre. Esta señal será tan fuerte que invadirá todos los canales de comunicación, interfiriendo su transmisión y al final el usuario se verá

obligado a interrumpir momentáneamente su sesión con el fin de atender la llamada telefónica.

La presencia de situaciones de este tipo puede plantear el abandono de determinados canales de comunicación especialmente sensibles a la presencia de ruidos, como es el caso del canal de comunicación auditivo, el cual resulta especialmente vulnerable ante la presencia de ruido externo. Salvo ante usuarios con graves discapacidades visuales, GADEA evita el uso de este tipo de canal de comunicación como medio principal para la transmisión de la información, relegándolo en la mayoría de los casos a jugar un papel secundario.

La competencia entre canales de comunicación de un mismo sistema, también puede ser el origen de una comunicación ruidosa. En sistemas multitarea, la atención del componente humano es cortejada simultáneamente por toda una legión de canales de comunicación, cuya única intención es quedarse con esa atención en exclusiva, introduciendo sin quererlo grandes cantidades de ruido en los canales de la oposición. Un ejemplo claro de esta situación la tenemos en la práctica común de muchos usuarios de mantener varias ventanas abiertas y visibles, las cuales proceden de aplicaciones distintas. En este caso, la información visual adolece de un grave problema de ruido, contra el cual nada se puede hacer.



Naturaleza Muerta con Esfera Reflejante © 1934 por M. C. Escher

P A R T E I I I

CodeX: *Code Xplorer*

6 El Modelo de Interacción

El papel del modelo de interacción en el diseño centrado en el usuario. Identificación del modelo de procesos para la interacción: *precondiciones* y *postcondiciones*. Identificación del modelo de datos para la interacción: el problema de la relación semántica entre los datos de un diálogo interactivo. El paradigma de desarrollo de CodeX.

p. 135

7 Aplicaciones y Procesos de Usuario

Definición de los puntos de entrada a la funcionalidad que implementa los procesos de usuario así como al punto de inicio en la computación de una aplicación. Diseño general de Codex: el *Application Manager*, el *Application Record*, y la clase UIMS. Mecanismo automático de inspección de código binario en búsqueda de métodos que representan procesos de usuario, *precondiciones* o *postcondiciones*. Compatibilidad con otros métodos de la aplicación.

p. 143

8 El Diálogo

Importancia del modelo de datos dentro del modelo de interacción. Descripción de los canales de comunicación duales de GADEA en su dimensión visual y auditiva. El diálogo interactivo como gestor de peticiones de información contra y desde la funcionalidad de la aplicación. Diseño de primitivas básicas de datos en componentes orientados a objetos y su inclusión en los diálogos interactivos. Tratamiento de requerimientos de información en canales de comunicación variable en base a las propiedades de un objeto.

p. 163

9 La Semántica

Definición del lenguaje ACML (*Adaptive Contents Markup Language*) como puente de comunicación estándar entre los módulos de inspección de código y de presentación de información en el diseño de una arquitectura de gestión de interfaces auto-adaptables. Estructuración del conocimiento en base a la restricciones impuestas por la memoria a corto y a largo plazo. Introducción del concepto de *chunk* como unidad básica de conocimiento.

p. 177

10 El Lenguaje

Importancia del papel jugado por el lenguaje en todo sistema de comunicación. Características cognitivas del lenguaje. Ambigüedad y protocolos de comunicación. Los símbolos, signos e ideogramas. Visión del lenguaje por parte de la semiótica. Empleo de entornos culturales como conjunto envolvente del lenguaje. Estructura y diseño *del Language Manager* de CodeX y la definición jerárquica de una base de conocimiento fundamentada en entornos culturales.

p. 187

11 El Application Explorer

Diseño del *Application Explorer*, pequeña utilidad incluida en toda distribución de GADEA, responsable del registro de nuevos usuarios en el sistema, así como la verificación de la identidad de usuarios ya registrados. Esta aplicación permite además arrancar aplicaciones GADEA. Relación de las características del modelo cognitivo, perceptivo y motriz de los usuarios recogidas de forma explícita por esta aplicación.

p. 209

5

5 Diseño Centrado en el Usuario

La vida es demasiado complicada para vivirla sin orden

Martha Stewart

5.1 El Módulo CodeX

La capa de más bajo nivel del GADEA está constituida por el módulo CodeX (*Code Xplorer* o *Explorador de Código*), el cual tiene como objetivo alcanzar el mayor grado posible de separación entre la interfaz de usuario de una aplicación y su funcionalidad, es decir, entre su aspecto perceptible y la función para la que dicha aplicación está diseñada.

Mientras que la funcionalidad de la aplicación recae en el código escrito por los programadores siguiendo las especificaciones elaboradas por los diseñadores, su aspecto perceptible es gestionado de manera uniforme por el sistema experto de GADEA. Este sistema experto se encargará de adaptar el aspecto de la aplicación al modelo cognitivo,

perceptivo y motriz de cada usuario en función de la información obtenida por los agentes ANTS y mediante el motor de inferencias de lógica difusa contenido en el módulo DEVA, el cual describiremos en detalle a partir del capítulo 12 (*El Modelo Cognitivo*). Esta gestión uniforme de la interfaz de usuario permite alcanzar una consistencia total para todas las aplicaciones ejecutadas bajo el modelo GADEA, ya que el aspecto y configuración de los canales de comunicación y diálogos interactivos empleados no dependerá de las múltiples interpretaciones que los diseñadores puedan hacer de las especificaciones, normas o guías base de las plataformas de desarrollo.

Estas guías, entre las que podemos citar las *Macintosh Human Interface Guidelines* de Apple (1991) y las normativas de la *APDA (Apple Programmer's and Developer Association*) y así como la *AMTG (Apple`s Macintosh Technical Group*) para entornos Macintosh y las CUA *o Common User Access* de IBM (1989) y las normas de diseño recogidas por Dunfee et. al. (1988, p. 325-347) para entornos Windows, suelen tener el defecto de dejar descubiertos importantes aspectos del diseño de la interfaz de usuario dado el uso de especificaciones arbitrarias, ambiguas o lo que es peor, simplemente se deja importantes aspectos del diseño al criterio del diseñador, el cual sólo podrá responsabilizarse de la consistencia de dichos aspectos en las aplicaciones desarrolladas por él.

El empleo de las reglas formales de la lógica difusa albergadas en el corazón de DEVA permitirá un tratamiento homogéneo de todos los aspectos de la interfaz y la consistencia en la comunicación para todas las aplicaciones que se ejecutan bajo el modelo GADEA.

5.2 Interfaz y Funcionalidad

Como se había visto cuando se abordó el tema de sistemas de gestión de interfaces de usuario (capítulo 1: *UIMS*), los mejores resultados obtenidos en la separación de la funcionalidad de las aplicaciones de su interfaz se ha conseguido por medio de variadas técnicas en las que un equipo de especialistas en interacción diseñan por separado el aspecto de todas y cada una de las ventanas, diálogos, pantallas, botones, textos y demás componentes de la interfaz de una aplicación; para luego enlazarlas con su código por un equipo de programadores. Lamentablemente, en la mayoría de los casos el equipo de interacción y el de programación suele estar formado por las mismas personas; y eso cuando no ocurre que el grupo de desarrollo esté formado por una única persona.

En la industria, esta técnica de desarrollo ha demostrado reducir notablemente el tiempo de desarrollo de los programas de aplicación al ser notable la proporción horas de desarrollo dedicadas en exclusiva al diseño de la interfaz de usuario. La rapidez con la que se pueden diseñar pantallas, menús, ventanas y demás *widgets* en los entornos WIMP facilita el desarrollo rápido de prototipos de la interfaz, los cuales se irán refinando a lo largo de todo el proceso de desarrollo. Sin embargo, esta técnica tiende a subordinar el desarrollo de los

mecanismos de interacción de la aplicación al diseño de sus modelos de procesos y de datos, quedando relegado a un segundo plano el diseño centrado en el usuario.

Este efecto se ve reflejado de sobremanera en los proyectos de pequeña o mediana envergadura en los que la utilización de un equipo separado de expertos en interacción y comunicación humana y otro de programadores resulta prohibitivo en términos monetarios. En la mayoría de los casos el diseño de los mecanismos de interacción es superficial o incluso inexistente, llevándose a cabo a medida que avanza la fase de desarrollo del producto, creándose pantallas y ventanas allí donde se necesiten y colocando en ellas los controles necesarios para obtener los datos requeridos, siguiendo el criterio de personal no especializado en el diseño de procesos de interacción.

Y es que la propia naturaleza de algunas técnicas de la ingeniería del software facilita la producción –y reproducción– de este fenómeno. Así por ejemplo, algunas de estrategias de diseño más ampliamente utilizadas en la ingeniería del software basan su poder de abstracción en la creación de un esquema conceptual a partir del mundo real, el cual estará dirigido bien por el modelo de datos o por el modelo de procesos, pero raramente por la interacción con el usuario.

Esta definición temprana de los modelos de datos y de procesos y el protagonismo alternativo de ambos diseños a lo largo de todo el proceso de diseño obliga a relegar la interacción con el usuario a un segundo plano. Esta práctica trae como consecuencia una endeble especificación de requisitos de interacción y en definitiva la definición de unos mecanismos de comunicación con el usuario pobres y groseros, lo cual obliga a un progresivo refinamiento a lo largo de todo el proceso de desarrollo.

Estos modelos de desarrollo basados en estrategias de diseño dirigidas por el dúo conceptual dato-proceso presentan como vemos una separación muy tardía entre los procesos de interacción de la aplicación y su funcionalidad. En algunos casos extremos, la separación entre interfaz y funcionalidad se produce una vez obtenido el esquema conceptual del sistema que se está modelando, momento en el cual éste es divido a su vez en dos esquemas: el esquema interno, orientado a la máquina, y el esquema externo, orientado al usuario.

Esta es la estrategia empleada por ejemplo en la metodología *Métrica* (metodología oficial del Estado Español), cuya representación podemos ver en la Figura 11. De acuerdo con la guía de técnicas de esta metodología el proceso de separación se produce de la siguiente manera:

Cuando el usuario se plantea una serie de necesidades o cambios para su unidad, es cuando el analista encargado del análisis comienza a estudiar los datos. De la narrativa tradicional o mundo real debe llegar gráficamente al Esquema Conceptual de donde posteriormente podrá deducirse el esquema interno y el externo (orientado a la máquina y al usuario respectivamente). Así, el esquema conceptual se considera como la indirección entre los otros dos. [MAP (1994)].

Como podemos ver, el diseño centrado en el usuario queda relegado por completo a un segundo plano, ya que la toma de decisiones concernientes a la interacción y comunicación de los usuarios con la aplicación se toman una vez que las estructuras de datos y de procesos se encuentran ya consolidadas.

Estas estrategias de desarrollo, dirigidas por el diseño de componentes no humanos del sistema (en el caso que nos ocupa: datos y procesos), recogen la herencia histórica de los múltiples enfoques metodológicos para sistemas de gerencia y toma de decisiones desarrollados y refinados a lo largo de los años setenta y ochenta, como es el caso de los enfoques duros de Hall (1980) y Jenkins [recogido en Russell (1978)]. Paradójicamente, aquel esfuerzo de investigación incluía enfoques metodológicos blandos –orientados a problemas sociales en donde el humano era el protagonista– como es el caso del enfoque SOCCER (*Solutions, Owners, Culture, Customs, Expectancies, Resources*) de Checkland (1981 y 1995).

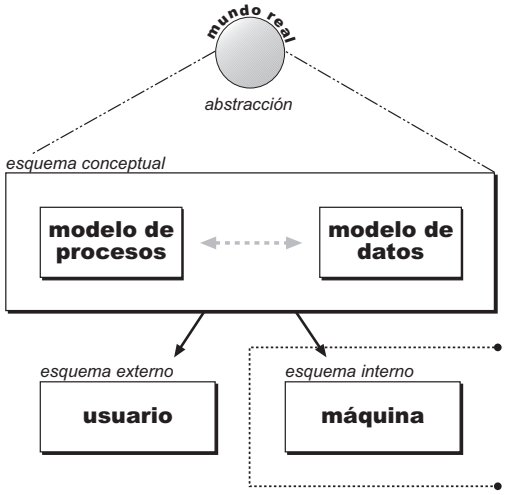


Figura 11. Esquema conceptual guiado por los modelos de datos y de procesos.

Estas estrategias de diseño alejadas del usuario no solo han pervivido en los productos que se originaron a partir de ellos, como ocurre con la propia *Métrica* o con la metodología *Merise* [Gabay (1991)] (la metodología estándar del Estado Francés) sino que también han sido incorporados al análisis y diseño orientado a objetos [Coad y Yourdon (1991)], el cual seguía hasta hace bien poco modelando los sistemas en términos de datos (objetos y propiedades) y de procesos (métodos) lo cual ha quedado reflejado en las especificaciones del enfoque metodológico de Booch (1994 y 1996) [también en Martin et al. (1996)] y en OMT (*Object Modelling Technique*) [Rumbaugh (1996)]. Afortunadamente, el modelo de interacción empieza a estar presente en los modernos enfoques metodológicos orientados a objetos, a través de los casos de uso desarrollados por Ivar Jacobson et. al. (1992) e incorporados en la notación UML (*Unified Modelling Language*) [Muller (1997)].

5.2.1. El Diseño Centrado en el Usuario

La comunidad científica internacional especializada en interacción y comunicación humana ha llamado la atención sobre los peligros de un uso inadecuado de estas técnicas en varias ocasiones [Coutaz et. al. (1993); Rouff y Horowitz (1991)] planteando en su agenda la necesidad del empleo de métodos formales de especificación de interfaces de usuario, capaces de incorporar los requisitos de uso de la aplicación desde las primeras fases de su desarrollo [Palanque y Bastide (1995); Bodart y Vanderdonckt (1996)]. Es por ello que el primer objetivo que nos planteamos al diseñar un mecanismo de separación entre funcionalidad e interfaz para GADEA ha sido mantener las ventajas de la programación visual sin que ello plantee un abuso en sus bondades que pueda conducir al desarrollo de interfaces mediocres o en el peor de los casos completamente inútiles por parte de un diseñador de aplicaciones GADEA. Es por ello que CodeX mantiene todas las ventajas de la programación visual, teniendo especial cuidado de potenciar las más importantes, entre las que destacan las siguientes:

- Ensamblaje y desarrollo sencillo de contenedores de *widgets* bajo entornos de desarrollo basados en componentes.
- Incorporación y participación del usuario en el proyecto desde las primeras fases de desarrollo.
- Desarrollo fluido de prototipos y empleo de estrategias de diseño basadas en refinamiento progresivo de los requisitos.
- Cumplimiento de requisitos de interacción especiales en el diseño de la interacción mediante el diseño, desarrollo e incorporación de componentes específicos.

Por otro lado, CodeX ha de evitar las desventajas de la programación visual cuando ésta es empleada de forma inadecuada, planteando un rígido modelo diseño centrado en el usuario. Mediante la colocación de determinados obstáculos en puntos clave en este modelo de diseño se pueden eliminar desde el principio los principales problemas que presenta la programación visual, los cuales son:

- Identificación tardía de requisitos de interacción, muchas veces en ocasiones que podrían ser evitadas con un análisis más profundo, con el ahorro en tiempo y dinero que ello conlleva.
- Creación descontrolada de canales de comunicación por personal inexperto en técnicas de interacción y comunicación humana.
- Falta de criterio en la resolución de aspectos ambiguos de una interfaz no detallados en la cuantía suficiente por las reglas de diseño.
- Generación de multitud de adaptadores que sirven de puente entre las ventanas, botones, cuadros de diálogo, etc. y el código responsable de la funcionalidad de la aplicación. Aunque los adaptadores funcionan bien para la mayoría de los casos, existen ocasiones en los que es necesario modificar dichos adaptadores con el objeto de lograr grados extremos de funcionalidad. Los

adaptadores son por lo general difíciles de modificar y su adaptación –valga la redundancia– requiere de conocimiento experto o cuando menos de un esfuerzo considerable.

- Inconsistencia en los mecanismos de interacción tanto dentro de una aplicación o serie de aplicaciones de un mismo centro de desarrollo como dentro del conjunto de las aplicaciones que funcionan bajo un mismo sistema operativo o bajo un sistema de gestión de interfaces de usuario.

Con este primer objetivo del diseño de CodeX, se pretende volver a un diseño centrado en el usuario en el que éste recobre el protagonismo. De este modo, el desarrollo de los modelos de datos y de procesos quedarán en un segundo plano, supeditados ambos a los distintos procesos que el usuario necesita para efectuar las tareas que el tiene asignadas en el sistema al cual pertenece. La identificación de estos procesos, así como la materia prima requerida por éstos (los datos) constituirán un primer paso en el diseño de las versiones preliminares de los modelos de datos y de procesos. Una vez identificadas estas necesidades del usuario, los modelos de datos y de proceso podrán refinarse ya en función del entorno y los recursos disponibles por el sistema, pudiendo obtener procesos y datos adicionales de otros subsistemas por medio de la incorporación de componentes por medio de la interacción directa a través de canales de comunicación.

Esta estrategia de diseño obliga a una identificación y definición de los requisitos de usuario desde las primeras fases del proceso de desarrollo, con lo cual es posible definir un **Modelo de Interacción**. Será este modelo de interacción quien dirija los pasos a seguir durante la fase de diseño como podemos ver en la Figura 12.

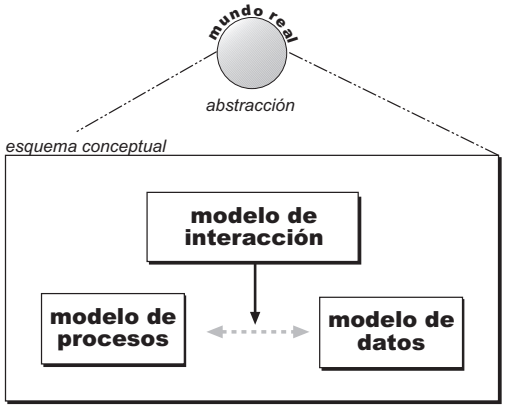


Figura 12. Esquema conceptual basado en modelos de interacción. La definición de los modelos de datos y de procesos se realiza una vez definido el modelo de interacción.

5.2.2. Objetos Adaptables

Pero el principal problema de las técnicas de programación actuales, sean éstas basadas en la programación visual o no radica en el hecho de que la selección de *widgets* y su colocación en un espacio visual se suele generar de forma estática en tiempo de compilación por lo que no es

posible una modificación posterior en tiempo de interacción con el usuario. Con ello se pierde cualquier posibilidad de adaptación de *widgets* al perfil cognitivo, perceptivo y motriz del usuario. Este razonamiento se puede extender también a la selección de la modalidad de interacción en sistemas multimodales [Wahlster (1991)] y a la creación y uso de canales de comunicación.

Este problema, que puede parecer trivial, es de magnitud considerable si tomamos en cuenta que todo el diseño estático de la interfaz de la aplicación está enfocado a satisfacer las necesidades concretas de un usuario concreto. Un usuario especial en el peor de los casos o un grupo de usuarios tipo en el mejor. Es evidente que con este planteamiento es imposible definir una interfaz adaptable.

Esta estrategia de diseño estática deja de lado las necesidades de importantes grupos humanos cuyo perfil cognitivo no coincide necesariamente con el del grupo para el cual ha sido diseñada la interfaz a pesar de que dicho grupo esté acostumbrado a realizar la misma tarea (funcionalidad) para la que está diseñada la aplicación. Por ejemplo, una aplicación para resolver ecuaciones diferenciales resulta totalmente inútil para un matemático acostumbrado a resolver este tipo de ecuaciones si la mencionada aplicación está escrita en un idioma o dialecto desconocido para ese usuario concreto.

Afortunadamente, el problema ilustrado en el ejemplo anterior ha sido solucionado hace mucho tiempo separando los textos e imágenes del código de la aplicación mediante los *Data Forks*, paquetes de internacionalización, *Bundle Resources* y soluciones similares comentadas en a partir del capítulo ¡Error!No se encuentra el origen de la referencia. (*¡Error!No se encuentra el origen de la referencia.*). Sin embargo, ese nivel de adaptación no es suficiente. En todo programa de aplicación existe un importante colectivo de usuarios con requisitos de interacción mucho más importantes y complejos que la selección de un determinado idioma. Pensemos por ejemplo en el caso de las personas con deficiencias visuales de carácter leve o moderado, que son incapaces de leer el pequeño texto diseñado para usuarios poseedores de una visión perfecta. Este ejemplo, que puede parecer algo rebuscado, es un caso de lo más común. No todos los usuarios de una aplicación tienen porque poseer una visión perfecta, de hecho, la simple variabilidad entre las edades de los usuarios implica ya una notable variación en los grados de visión. Obviamente, en condiciones normales el grado de visión de un adolescente difiere mucho del de un octogenario. Si llevamos este problema de la diferenciación al extremo veremos que los ciegos no pueden percibir determinados interfaces a los que tienen perfecto derecho de acceder.

Queda claro por tanto que el diseño estático de interfaces de usuario atenta directamente contra cualquier posibilidad que exista de adaptarlo a las necesidades de un usuario concreto. Es necesario por tanto diseñar interfaces con unas directrices estáticas del carácter lo más genérico posible, de modo que la interfaz pueda variar y adaptarse al usuario en tiempo de ejecución. Será por ello que el segundo objetivo de CodeX consistirá precisamente en identificar todos aquellos objetos del modelo de interacción que pertenecen exclusivamente al dominio de la

funcionalidad de la aplicación y por lo tanto que no sean susceptibles de una adaptación posterior.

El objetivo es pues buscar el mínimo común denominador semántico del conjunto de elementos de una aplicación con los que pueda iterar el usuario. Encontrado este, será posible adaptar la apariencia y contenido semántico de estos objetos genéricos a las necesidades del usuario a partir del contenido del modelo de usuario. Abstrayendo los objetos adaptables de los no adaptables, la tarea de los diseñadores se podrá reducir a la identificación de estos conjuntos y a la definición de los escenarios en los que éstos entrarán en juego, indicándoselo en el momento oportuno a CodeX para que inicie un canal de comunicación con el usuario y establezca una interacción a través de dicho canal con los diálogos que el sistema experto DEVA considere oportuno.

En suma, lo que se pretende con este segundo objetivo es dejar en los diseñadores de la aplicación la responsabilidad de la identificación y diseño de las diversas metáforas con las que el usuario realizará sus tareas en el sistema, dejando a GADEA las labores de selección de canales de comunicación, el diseño de los diálogos que se establecerán sobre dichos canales y el control de todo el proceso de interacción y comunicación con el humano.

Como se puede apreciar, esta estrategia de desarrollo deja en manos del experto humano (el diseñador) la parte diseño de alto nivel del modelo de interacción, mientras que el diseño de los aspectos de bajo nivel de dicho modelo queda bajo la responsabilidad del experto artificial (GADEA). Este esquema libera al diseñador de las tareas mecánicas, aburridas y por tanto propensas a errores de su trabajo, concentrando sus habilidades y destrezas en el diseño de un modelo de interacción sólido, facilitando así la participación del usuario en el desarrollo desde el inicio del proceso.

5.2.3. Facilidad de Uso

El tercer y último objetivo del esquema de separación del dúo funcionalidad-interfaz provisto por el módulo CodeX pretende que el proceso de diseño sea lo más sencillo para el experto humano, intentando reducir en la medida de lo posible la toma de decisiones de diseño por parte de éste. Para ello se pretende combatir otro de los grandes problemas de los mecanismos de programación visual, el cual consiste en la manipulación directa de las ventanas, menús, botones y demás *widgets* generados por las herramientas de diseño visual.

En la mayor parte de las herramientas, el programador ha de incluir manualmente cada una de las llamadas a procesos de usuario en su interfaz, por medio de comandos de menú, botones, órdenes introducidas mediante teclado, etc. Esta estrategia no solo presenta la desventaja de requerir grandes dosis de trabajo en la programación de estas llamadas sino que también es fuente de una gran proporción de los *bugs* o errores introducidos en el código de un programa.

Otra de las desventajas de esta estrategia de desarrollo aparece una vez que el usuario ha terminado un ciclo de interacción con la máquina

ya que tras haber introducido los datos requeridos por el agente software, siguiendo el patrón de diseño de los formularios [Grand (1999) p. 121-127], es necesario empezar un proceso de validación y verificación de los datos proporcionados, proceso que puede ser largo, tedioso, repetitivo y sobretodo, fuente de numerosos errores.

Para poder alcanzar este objetivo, CodeX debe conocer las operaciones que el usuario puede realizar con la aplicación o con los objetos interactivos que ésta posee en un momento dado. Dado que éste conjunto de operaciones se encuentra estrechamente ligado al estado de la aplicación y al flujo lógico de eventos y tareas previas realizadas por el usuario sobre el conjunto de datos y objetos de la aplicación, es totalmente imposible saber de antemano, es decir, en tiempo de edición, el rango posible de operaciones que el usuario puede realizar con una aplicación, ya que el conjunto de estas operaciones no es estático y depende directamente del estado interno de la aplicación.

Podemos ilustrar esta situación de forma clara con el ejemplo de un cliente de correo. Una de las operaciones de usuario clásicas de este tipo de aplicaciones consiste en poder reenviar un correo recibido a otra persona. Para poder hacerlo, es necesario haber recibido previamente el correo que se desea reenviar. Aunque la operación *Reenviar* se conoce ya en tiempo de diseño, el cumplimiento de la condición para su aplicación (la existencia y selección del correo a reenviar) no se puede verificar hasta que no se ejecute el código y la aplicación alcance el estado deseado (correo seleccionado). Siguiendo el patrón de diseño de un *Interfaz Explorable* para formularios [Grand (1999) p. 103-108], la operación *Reenviar* debería estar disponible entre las opciones de la aplicación, pero en modo desactivado. Solo se activaría en el momento en el que el usuario seleccionase algún correo. Este funcionamiento se complicaría aún más si se desea el empleo de los menús contextuales de un objeto, esto es, el conjunto de operaciones que se encuentran disponibles para un objeto en un momento determinado de la ejecución de la aplicación. En este caso, la operación *Reenviar* no estaría presente en la interfaz, ni activada ni desactivada hasta que no existiese un objeto correo.

5.2.4. Inspección de Código Binario

Para simplificar la comunicación entre el código y GADEA y reducir la complejidad de las condiciones mediante las cuales una operación está disponible para el usuario, CodeX plantea la inspección automática del código compilado de una aplicación en tiempo de ejecución. En esta inspección se plantea la búsqueda de métodos que definan un posible proceso de usuario, con el objeto de notificárselo al sistema experto DEVA y que éste proceda a notificar la existencia de estos procesos al usuario por medio del canal de comunicación adecuado, empleando para ello el conocimiento disponible acerca de las necesidades, cognitivas, perceptivas y motrices del usuario.

Esta estrategia de inspección automática del código compilado se puede realizar por medio de los mecanismos de reflexión e introspección proporcionados por las plataformas de desarrollo

actuales, como es el caso de Java [Weber 1997, pp. 369-382] o del sistema operativo *Oviedo-3* [Álvarez y Tajes (1997); Ortín et. al. (2000)] y sobre las que el modelo planteado en GADEA puede ejecutarse perfectamente.

Mediante el uso adecuado de estos mecanismos de reflexión, es posible analizar en tiempo de ejecución una clase compilada, por lo que con un formalismo sencillo para el etiquetado de los nombres de los métodos y propiedades de una clase que se correspondan con los procesos y datos de usuario es relativamente sencillo poder transferir información entre el código de un programa de aplicación y GADEA. El uso de formalismos de este tipo ha sido empleado con éxito en el desarrollo de protocolos de comunicación entre componentes, como ha sido en el caso de la plataforma Java y sus componentes *JavaBeans* [Sun (1996)].

Por medio de la reflexión se pretende obtener la mayor independencia posible entre GADEA y los programas de aplicación, a la vez que se reduce el trabajo de programación necesario para establecer contacto entre los dos mundos. Mediante el uso de esta poderosa herramienta y el cumplimiento de un formalismo para el nombrado de los métodos que representan operaciones de usuario, es posible lograr aplicaciones con alto nivel de independencia, logrando disminuir las comunicaciones entre el sistema experto y las aplicaciones que se ejecutan bajo su dominio, reduciendo y llegando a eliminar las desventajas de las clásicas *callbacks*.

Dado que las operaciones de usuario disponibles depende a su vez de los objetos activos, así como del estado de la aplicación, es necesario el establecimiento de *precondiciones* y *postcondiciones* para la ejecución de cada una de las operaciones disponibles. Como se vio en el capítulo 2 (*Interfaces Inteligentes*), los conceptos de *precondición* y *postcondición* fueron introducidos a escala masiva por Foley et al. (1991) en su sistema *UIDE (*mas tarde MASTERMIND), siendo potenciados posteriormente por Ivar Jacobson al ser incorporados a la notación UML [Muller (1997)].

Mientras que las *precondiciones* definen las condiciones necesarias para la ejecución de una operación, las *postcondiciones* indican el estado en que quedará la aplicación tras la ejecución satisfactoria de la misma. Así, en el ejemplo anterior del reenvío de correos en un cliente de correo electrónico, tendríamos como *precondición* la existencia y selección de un correo a reenviar y como *postcondición* su marcado como correo reenviado.

Tanto las *precondiciones* como las *postcondiciones* pueden beneficiarse del empleo de formalismos en su nomenclatura, con lo que se facilita su detección automática por agentes inspectores de código, haciendo más fácil la tarea del programador.

Gracias a esta inspección automática de código, la tarea de los diseñadores se simplifica y facilita aún más al convertirse en muchos casos en la sencilla identificación de los objetos, procesos de usuario, *precondiciones* y *postcondiciones*. Aún con la ayuda de estos poderosos mecanismos de análisis de código en tiempo real, es necesario reconocer

la ayuda del programador para poder registrar de forma explícita conjuntos bloques de interacción compuestos por un número variable de procesos de usuario.

Aunque es imposible lograr un 100% de independencia para todo el dominio de las aplicaciones mediante mecanismos de análisis automático, si es posible alcanzar cotas de excelencia en torno al 90% para la mayoría de ellas, sobre todo en aquellas que no requieran modelos de interacción especiales.

Una vez definidas las operaciones de usuario y los objetos sobre los que éstos actúan de forma automática, por medio de la reflexión, queda pendiente la tarea de la definición del contenido de los diálogos necesarios para establecer comunicación con el usuario. Obviamente, GADEA desconoce en un principio la naturaleza de estos diálogos y tendrá que buscar su definición en la información provista por los diseñadores y programadores del producto a través del código escrito.

Basándonos en la estructura básica no adaptable del segundo objetivo de CodeX, es posible diseñar los diálogos establecidos sobre los canales de comunicación en función de una colección de objetos base. De este modo, el programador no definirá los *widgets* que necesite en cada diálogo (botones, *checkboxes*, etiquetas, etc.) sino estrictamente los datos que necesite (una cadena, un entero, un real, etc.). De este modo el proceso desarrollo se simplifica aún más al poder trabajar con un nivel de abstracción superior. Si los diálogos se establecen en términos de datos en lugar de en función de los *widgets* necesarios para obtener dichos datos, es posible establecer relaciones directas entre los datos requeridos por el modelo de interacción y por el modelo de datos. Con ello se logra que el flujo de información sea más natural.

Al poder realizarse una analogía directa entre los datos necesarios para construir el modelo de interacción y el modelo de datos, la calidad del producto obtenido es superior, ya que se pueden rastrear los requisitos del sistema en ambos modelos y verificar que el conjunto de datos es el mismo (especificación correcta) o no (especificación incorrecta). Como se puede apreciar, todo ello se puede llevar a cabo desde las primeras fases del desarrollo del producto.

Como consecuencia de la definición de este objetivo, será necesario la creación de una sólida jerarquía de clases de datos ambivalentes, que puedan servir tanto para cubrir las necesidades del modelo de datos como las del modelo de iteración. Estas clases serán empleadas mas tarde por DEVA para seleccionar el *widget* más adecuado en función del tipo de dato requerido por la aplicación y del modelo de usuario concreto con el cual se relacione la interfaz GADEA en un momento determinado de la ejecución de la aplicación.

6

6 El Modelo de Interacción

Mientras no sepamos cómo construir interfaces efectivos, poco sentido tendrá invertir esfuerzos en lograr un software estructurado y perfectamente documentado

Nathaniel S. Borenstein

6.1 Paradigma de Diseño Centrado en el Usuario

El esquema de separación propuesto por el módulo CodeX se basa en la aplicación de las técnicas clásicas del diseño orientado a componentes y de la ingeniería del software en dos capas o niveles de abstracción: una de procesos y otra de datos. Estas capas se corresponden con los procesos de usuario y con los datos empleados por dichas técnicas y podrán ser definidas dando preferencia a los procesos sobre los datos, bien dando prioridad a los datos o en el mejor de los casos a la vez, desarrollando ambos procesos de forma paralela.

La unión de estas dos capas en un marco conceptual único permite configurar el modelo de interacción referido en el apartado 5.2. Se pretende con ello definir y configurar dicho modelo desde las primeras fases del diseño de la aplicación. Este modelo de interacción será refinado y detallado mediante las técnicas de desarrollo evolutivo

basadas en la construcción de prototipos, siendo finalmente descompuesto en el modelo de datos y en el modelo de procesos requerido por la metodología de desarrollo empleada por la organización o por las técnicas de análisis y diseño de la ingeniería del software tradicional.

6.1.1. Los Procesos de Usuario

El empleo del módulo CodeX requiere como único requisito la identificación previa de los principales procesos de usuario durante la fase de análisis del modelo de interacción de la aplicación. Estos procesos de usuario comprenden todos aquellos procesos del modelo de procesos tradicional que puedan ser invocados de una forma directa o indirecta por el usuario, es decir, comprenden todos aquellos procesos mediante los cuales el usuario puede actuar sobre la funcionalidad del software.

Un buen diseño del modelo de interacción prevé la identificación de todas las tareas que el usuario realiza en su trabajo cotidiano y que deberán tener un equivalente software en la aplicación a desarrollar. Esta identificación de tareas se realiza con independencia de la metáfora de interacción que se implementará sobre el producto software, teniendo en cuenta que la funcionalidad ha de estar presente en la aplicación de un modo u otro.

Así por ejemplo, si la aplicación a desarrollar se trata de un IDE (*Integrated Development Environment*) para programadores avanzados, los procesos de compilar, depurar y ejecutar han de estar presentes independientemente de la metáfora de desarrollo empleada, sea esta la basada en proyectos, del tipo de la empleada en el *Kawa* de *Tek-Tools* (1995) o en el *THINK C++* de *Symantec* (1989) o sea ésta la metáfora basada en las áreas de trabajo (*workspaces*) del tipo de la adoptada por IBM en su gama de productos *Visual Age* [IBM (1994)].

Del mismo modo, la presencia de estos procesos de usuario, al estar basada en la tarea a realizar y no en la funcionalidad (el *qué* en lugar del *cómo*) ha de ser independiente de la carga semántica de los mismos. Siguiendo el ejemplo del IDE, las operaciones compilar, depurar y ejecutar son independientes de la naturaleza de los lenguajes de programación empleados por el usuario y de la modalidad de desarrollo, pudiendo ser esta funcional, estructurada u orientada a objetos, tal y como señala Whitelaw y Weckert (1997) en su estudio sobre la naturaleza cognitiva de las distintas modalidades de programación.

Se pueden encontrar otros ejemplos de esta independencia de los procesos de usuario con respecto a su metáfora de desarrollo o con respecto a su carga semántica en diversas fuentes. En el primer caso, es recomendable observar como evolucionaron las interfaces CLI (*Command Line Interface*) de algunos sistemas operativos mediante el desarrollo de metáforas basadas en el escritorio hasta llegar a interfaces GUI *(Graphic User Interface*) manteniendo exactamente la tipología y cantidad de sus procesos de usuario [Apple (1992)]. En el segundo caso,

es recomendable el análisis de la semántica de los procesos de usuario llevada a cabo por Shulz et al. (1997) con herramientas para la medición de ondas electromagnéticas, observando como diferentes diseños dan respuesta a los mismos procesos de usuario.

También es importante destacar como ejemplo integrador el planteamiento de algunos enfoques metodológicos, que como es el caso de la metodología OOHDM [Schwabe et al. (1998) y Rossi et al. (1997)] para el desarrollo de productos multimedia, salvaguardan los procesos de usuario (enfocados en la obtención de la información) de las metáforas de navegación y la semántica de los datos.

La identificación de estos procesos de usuario, así como el momento en el que éstos estarán disponibles (*precondiciones*) y los efectos que una ejecución satisfactoria de los mismos puede producir sobre el estado de la aplicación o los objetos sobre el que se aplican (*postcondiciones*), es el único requisito necesario para la configuración del modelo de interacción. Esta identificación se puede realizar mediante cualquiera de las técnicas que la ingeniería del software propone para ello, tales como los *casos de uso* de Ivar Jacobson (1992) o las técnicas clásicas de integración del usuario en el proceso productivo [Apple (1992) p. 41].

Obviamente, la definición de las *precondiciones* y *postcondiciones* de los múltiples procesos de usuario que integran una aplicación equivale a definir todo el complejo entramado de mecanismos de interacción de la aplicación. Es aquí cuando la habilidad del diseñador es requerida para abstraer el mundo real y convertirlo en un modelo software compatible con el modelo mental empleado por los usuarios de la aplicación en la realización de sus tareas cotidianas. Es aquí por tanto donde es requerido el diseño de las metáforas necesarias para la correcta traducción de la ejecución de procesos de un medio real a un medio software.

6.1.2. Los Datos de Usuario

Una vez que los procesos de usuario han sido definidos, es preciso identificar los datos que dichos procesos van a requerir para su ejecución o que han de ser mostrados al usuario durante la misma, con el objeto de definir los diversos canales de comunicación necesarios para su transmisión.

Si el sistema experto DEVA está presente en el desarrollo de la aplicación, sólo será necesaria en esta fase la identificación de los datos requeridos por el proceso de usuario y su tipo (entero, cadena de caracteres, fecha, etc.), ya que la asociación entre el tipo de dato y el *widget* o control necesario para su obtención dependerá del usuario en concreto que emplee la aplicación. En el caso de no disponer de un sistema experto de este tipo, en el diseño dirigido por interacción será imprescindible seleccionar el tipo de *widget* necesario para la obtención del de una información en concreto, así como su configuración y apariencia externa en función del usuario tipo del programa.

Esta relación de datos agrupados por proceso de usuario será empleada para diseñar y definir los distintos canales de comunicación

necesarios para obtener dichos datos (visuales, auditivos, ventanas, cuadros de diálogo, cuadros de alerta, etc.), así como los múltiples diálogos que se pueden establecer sobre dichos canales.

Es necesario destacar que los canales de comunicación necesarios para esta interacción son de duración variable. Dichos canales pueden ser de existencia permanente o de existencia efímera, existiendo como es natural una variedad de canales cuya duración se sitúa entre ambos extremos, estableciéndose y destruyéndose en momentos muy concretos de la vida activa de una aplicación o de un proceso de usuario, satisfaciendo las necesidades puntuales de datos de la aplicación.

Cómo ejemplo de canales de comunicación de existencia permanente podemos citar el caso de la ventana principal de una aplicación de control de procesos en tiempo real, en la cuál el usuario puede observar y modificar un conjunto de variables del sistema (datos) a lo largo de toda su sesión de trabajo. Como ejemplo de canales de comunicación efímeros podemos citar los sonidos y/o cuadros alerta que avisan al usuario de la finalización de un proceso o de la existencia de una situación anómala en el sistema. Por último, como ejemplos de canales de comunicación de duración variable, se pueden citar las opciones de configuración de una impresora (tipo de papel, bandeja de salida, calidad e impresión, etc.), las cuales son accedidas por el usuario cuando necesita realizar un trabajo especial y las cuales permanecen dentro de su campo de acción por un tiempo prolongado.

Es por ello que los grupos de datos identificados deben estar asociados a un canal de comunicación y diálogo concretos, especificando además el momento exacto de su aparición y el de su posterior extinción. Este momento exacto se refiere obviamente al estado de la aplicación, proceso u objeto en el que es factible la creación o destrucción del canal de comunicación y diálogo asociado, así como el evento que genera su construcción y/o destrucción.

Tabla 2. Componentes básicos para un posible diálogo interactivo.

NOMBRE	TIPO	RANGO
DNI	STRING con plantilla	libre
Nombre	STRING	libre
Apellidos	STRING	libre
Calle	STRING	libre
Código Postal	INTEGER con plantilla	Lista predeterminada
Ciudad	STRING	Lista predeterminada
País	STRING	Lista predeterminada

Aparte de la colección de datos requerida por un proceso de interacción, será necesario también establecer la estructura lógica de los datos que forman parte de un mismo diálogo, con el objeto de facilitar la percepción de los mismos por parte del usuario y agilizar su

correspondencia con el modelo mental empleado por el usuario para su comprensión.

Figura 13. Posible distribución de los componentes de la Tabla 2 en un diálogo interactivo sin tomar en cuenta las relaciones semánticas entre los mismos. Resulta obvio que este esquema es desastroso en términos de usabilidad (consultar versión en color en el capítulo 24: *Imágenes en Color*, p. 445).

Así por ejemplo, supongamos que en un proceso requiere los datos incluidos en la Tabla 2 correspondientes a la identificación completa del usuario (DNI, nombre, apellidos, calle, código postal, ciudad y país). Suponiendo que para la obtención de estos datos se va a emplear un único diálogo interactivo y una distribución espacial de los *widgets* de tipo lineal en dirección vertical (un *widget* tras otro), el número de combinaciones posibles en las que se pueden distribuir los elementos en un campo visual o auditivo asciende a 7! = 10.080, de las que podemos ver dos en el ejemplo de la Figura 13 y en el de la Figura 14.

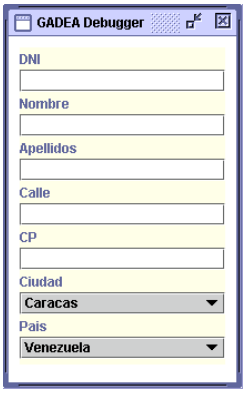


Figura 14. Distribución de los componentes de la Tabla 2 en función de las relaciones semánticas de precedencia y de cohesión que han de existir los mismos (consultar versión en color en el capítulo 24: *Imágenes en Color*, p. 445).

Como es lógico, el cuadro de diálogo de la Figura 14 presenta una estructura mucho más adecuada para un usuario que el cuadro representado en la Figura 13. La separación de los campos destinados a la identificación personal (DNI, nombre y apellidos) y de la dirección (calle, código postal, ciudad y país), así como el orden lógico de los mismos (identificación personal antes que la dirección) hacen más

perceptible la información que se desea transmitir al usuario en la Figura 14.

Como se puede apreciar en este grosero ejemplo, es imprescindible completar la información que el modelo de interacción posee acerca de los datos de usuario que formarán un diálogo con la estructura lógica necesaria para realizar su agrupamiento, así como el orden en que irán colocados los distintos grupos.

6.1.3. Diseño del Modelo de Interacción

Es necesario destacar que la tarea de identificación de datos puede ser realizada en paralelo con la tarea de identificación de los procesos de usuario, siendo del todo recomendable que ambas tareas se desarrollen en una estrecha colaboración mutua. Es necesario tener en cuenta que la identificación de procesos suele llevar a la identificación de nuevos datos y que estos a su vez pueden conducir a la identificación de los procesos necesarios para su captura, enriqueciendo con ello el proceso de definición del modelo de interacción y la identificación de nuevos requisitos de interacción.

En función del sistema de comunicación a modelar, en base a la naturaleza del problema a resolver o dependiendo de la aptitud y orientación psicológica del diseñador, la definición del modelo de interacción puede empezarse bien desde la óptica de los datos o bien desde el punto de vista de los procesos. De este modo, el proceso de diseño se puede dirigir indistintamente en función de los datos o de los procesos. Esta libertad de acción se encuentra presente en enfoques metodológicos como *Métrica* [MAP (1994a) y (1991)] o *Merise* [Gabay (1991)] en donde han demostrado alcanzar excelentes resultados en cuanto a la calidad de los productos obtenidos [D'Amore (2000); García Alcázar y Monzón (2000)].

No podemos olvidar que la definición de los datos y procesos requeridos por el modelo de interacción de la aplicación no son más que un subconjunto de los modelos de datos y procesos de los paradigmas de diseño comentados en el apartado *5.2 Interfaz y Funcionalidad* del capítulo 5 (*Diseño Centrado en el Usuario*)e ilustrados en la Figura 12. Es decir, los datos y procesos del modelo de interacción se corresponden directamente con aquellos datos y procesos relacionados con el proceso de interacción con el usuario, e identificados en el modelo de datos y en el modelo de procesos respectivamente.

Como consecuencia de esta separación entre funcionalidad e interfaz podremos hablar de dos niveles distintos para los modelos de datos y para los modelos de procesos (ver Figura 15). En un primer nivel se encuentran agrupados en un mismo modelo de interacción un modelo de datos y un modelo de procesos interactivo, es decir, datos y procesos implicados en la interacción con el usuario.

Centrando el proceso de diseño en las fases iniciales del proyecto en la definición y configuración de este modelo de interacción, se irá extrayendo el antiguo modelo conceptual que integre a los modelos de datos y de procesos funcionales, es decir, aquellos datos y procesos

internos o auxiliares necesarios para el buen funcionamiento de la aplicación, pero que no presentan una interacción directa con el usuario, sino que lo hacen de forma indirecta a través de los datos y procesos pertenecientes al modelo de interacción.

Este será el modelo empleado por CodeX para separar la interfaz (interacción) de la funcionalidad de la aplicación. Para ello, CodeX solo necesita conocer el modelo de interacción (como es evidente), dejando el modelo de datos y de procesos del esquema interno al buen criterio de las técnicas y mecanismos de diseño y desarrollo de la ingeniería del software.

Nótese que a pesar de que CodeX requiere un paradigma de diseño centrado y dirigido por el modelo de interacción, se sigue manteniendo la libertad para la selección del submodelo que dirigirá el resto del proceso de diseño (datos y procesos), libertad que tan buenos resultados ha proporcionado a la ingeniería del software.

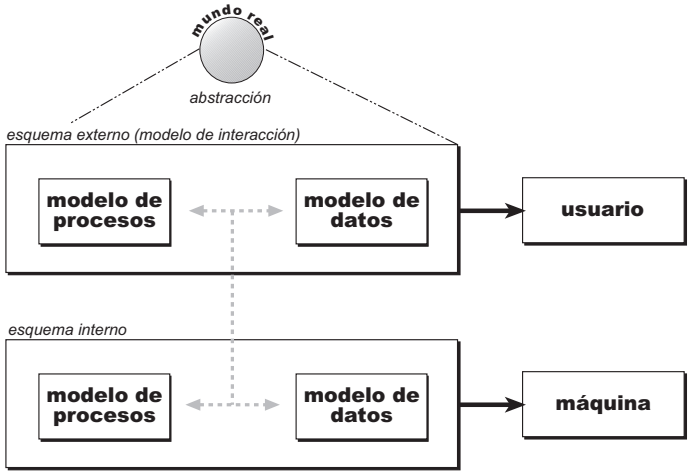


Figura 15. Paradigma de desarrollo empleado por Codex: se descompone el sistema a modelar en un modelo de interacción, un modelo de datos y un modelo de procesos, dando prioridad al primero de ellos. A su vez, el modelo de interacción se descompone en un modelo de datos y otro de procesos, los cuales representan los datos y procesos de usuario.

Si aplicamos las reglas básicas de la ingeniería de sistemas [Klir (1972)] a este modelo de desarrollo, podemos predecir que la descomposición de datos y procesos en dos niveles no disminuirá la calidad del producto obtenido sino todo lo contrario, pues al enriquecerse el sistema con las relaciones existentes entre los dos bloques y al poder definir y acotar con mayor precisión la funcionalidad de cada uno de ellos, se aumenta el grado de equivalencia del diseño al sistema que se pretende modelar.

Si en un diseño monolítico, la libertad de elección entre un hilo conductor basado en datos o en procesos obtenía buenos resultados, es de esperar por tanto que estos mismos resultados positivos se mantengan ahora tanto en las partes (modelo de interacción y esquema conceptual) como en la totalidad del sistema [Bertalanffy (1968)].

7

7 Aplicaciones y Procesos de Usuario

Lo cierto es que muchas personas fijan reglas para no tener que tomar decisiones

Mike Krzyzewsk

7.1 Registro de Aplicaciones GADEA

A diferencia de lo que ocurre en los sistemas de gestión de interfaces de usuario (SGIU o UIMS) de los sistema operativos tradicionales, la comunicación entre una aplicación y el conjunto de herramientas destinadas a la interacción no se realiza por medio de llamadas estáticas al sistema. Para garantizar el mayor grado de flexibilidad posible en esta comunicación, GADEA no proporciona de antemano un conjunto de primitivas básicas que se pueden invocar en tiempo de edición/compilación sino que éstas se emplearán directamente en tiempo de ejecución. Como se ha comentado en su momento, estas primitivas tienen un marcado carácter abstracto, que ha diferencia de las primitivas contenidas en juegos de instrucciones más liberales como la AWT (*Abstact Window Toolkit*) de la plataforma Java [Lemay (1996)] o

el *ToolBox* de la plataforma Macintosh [Apple (1992a); Apple (1992b) y Chernicoff (1985 y 1985a)] no permiten trabajar directamente con los mecanismos de interacción de la plataforma (*widgets*).

El carácter conservador de las primitivas de interacción de CodeX, así como su modo de empleo dinámico, proporcionan a GADEA la capacidad de adaptar los diálogos interactivos de la aplicación en función del estado cognitivo, perceptivo y motriz del usuario y del propio estado interno de la aplicación.

Dado el carácter dinámico de la comunicación entre aplicación y el sistema de gestión de interfaces de usuario (SGIU), es necesario que la aplicación esté registrada dentro del marco conceptual de GADEA y que ésta a su vez le proporcione un punto de apoyo que sirva de emisor de los requerimientos de interacción de la aplicación y a la vez de receptor de las peticiones de información por parte del usuario.

El establecimiento de ese canal de comunicación entre CodeX y una aplicación se realiza en el preciso momento de instalar la misma en el marco de trabajo de la plataforma en la que se ejecuta GADEA. Para ello, el proceso de instalación proporciona a CodeX la ubicación física en donde se encuentran las librerías, componentes y demás utilidades que integran el código de la aplicación (dentro del sistema de ficheros de la plataforma), así la clase concreta en la que se encuentra el punto de entrada de la computación de la aplicación. En el caso de aplicaciones escritas en lenguaje Java, dicha clase será aquella que alberga el método *main*. Esta información, así como el entorno cultural empleado por el usuario destino (que se analizará en detalle en el capítulo 10: *El Lenguaje*) se recoge en la Tabla 3.

Tabla 3. Valores Almacenados en *un Application Record.*

CAMPO	DESCRIPCION
PATH	Descripción completa de la ubicación del paquete de clases que forman la aplicación, indicando el volumen y directorio
MAIN OBJECT	Nombre completo del binario (.class) que actúa como punto de entrada de la computación de la aplicación.

Esta información es gestionada cuidadosamente por el *Application Manager* de CodeX, quien será el componente encargado de administrar la interacción concurrente por parte del usuario (a través de DEVA) sobre las distintas aplicaciones activas en la plataforma de ejecución. Dicha información puede ser almacenada directamente en un documento al que denominamos *Application Record.*

Código 1. Ejemplo de posible *Application Record.*

```
C:/Archivos de Programa/  DOBRA.productos.UE      es_sp
C:/Archivos de Programa/  uniovi.gadea.Explorer   es_sp
D:/Personal/Uniovi/EDI    uniovi.edi.EDIReg       es_sp_as
```

En el código Código 1 podemos ver un ejemplo del contenido de un *Application Record.* Como podemos observar, para cada aplicación existe un registro en el que se especifica el directorio en el que se encuentran

depositadas las clases que forman dicha aplicación y el nombre de la clase principal (aquella que contiene la función *main*). El nombre de esta última se especifica siguiendo la notación completa del paquete al que pertenece. Así, en el mencionado *Application Record* se indica que en el directorio *c:/Archivos de Programa* debe existir un directorio llamado *dobra* y otro llamado *uniovi*. Dentro del primero existirá un directorio conocido por el nombre de productos, dentro del cual existe un fichero llamado *UE.class*, el cual contiene la función *main* de la aplicación especificada en la primera línea. A su vez, en el segundo directorio (*uniovi*) existirá un directorio llamado *gadea*, el cual contendrá una clase denominada *Explorer.class* con el *main* de la segunda aplicación. La tercera línea del fichero especifica que una clase denominada *uniovi.edi.EDIReg.class* se encuentra dentro del directorio *d:/Personal/Uniovi/EDI*. El tercer atributo de cada registro representa el entorno cultural del usuario y será analizado en el capítulo 10 (*El Lenguaje*).

Cuando se inicia la plataforma GADEA, su *Application Manager* inspecciona el *Application Record* y carga en memoria una referencia de todas las posibles aplicaciones que se pueden emplear dentro de la sesión. Esta referencia es mostrada al usuario por una pequeña aplicación denominada *Application Explorer*, mediante la cual se puede seleccionar y arrancar cualquiera de las aplicaciones incluidas en el *Application Record*.

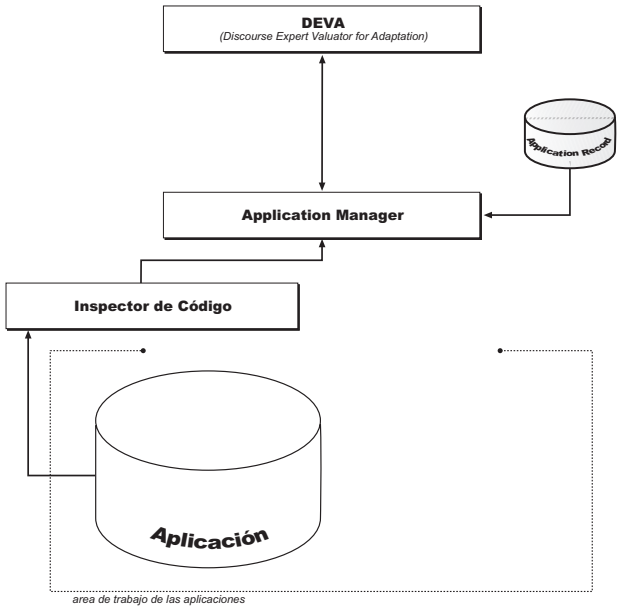


Figura 16. Diseño general de CodeX. La información almacenada en el *Application Record* permite encontrar los ficheros binarios de una aplicación, de tal modo que éstos puedan ser explorados por el inspector de código. La información obtenida es enviada al *Application Manager*, quien será el ente responsable de arrancar la aplicación.

Esta aplicación forma parte de la distribución estándar de GADEA y será descrita en el capítulo 11 (*El Application Explorer*). Por el momento solo necesitamos saber que esta aplicación sirve de selector de aplicaciones, facilitando al usuario la sincronización y gestión de la

interfaz de múltiples aplicaciones GADEA que se ejecutan en paralelo. De hecho, todas las aplicaciones registradas en el *Application Record* aparecen disponibles en el canal de comunicación principal de esta aplicación (auditivo y visual) vía menú, vía hipertexto, vía botones, etc. De este modo, el usuario podrá arrancar cualquiera de las aplicaciones GADEA disponibles, las cuales se ejecutarán bajo la supervisión del *Application Explorer*.

7.1.1. Comunicación con el Espacio de Objetos de la Aplicación

Cuando el usuario selecciona una aplicación a través del *Application Explorer*, CodeX asigna un espacio de ejecución para la aplicación seleccionada en el cual se mantendrá una referencia activa a todos aquellos objetos provistos de procesos de usuario, los cuales serán invocados según las necesidades del usuario y en el momento y condiciones en el que éste lo considere oportuno.

Por medio del objeto principal de la aplicación incluido en el *Application Record*, GADEA es capaz de establecer un canal de comunicación directo con la aplicación y registrar de forma automática y por medio de reflexión todos aquellos métodos pertenecientes a dicho objeto que representen procesos de usuario. Sin embargo, éste canal de comunicación no trabaja en dos direcciones ya que el objeto principal de la aplicación no puede contactar directamente con CodeX.

Para resolver este problema, una vez que CodeX ha preparado el espacio de objetos de la aplicación cliente, iniciando las estructuras necesarias para su mantenimiento y gestión en tiempo de ejecución, éste módulo de GADEA proporciona al objeto principal de la aplicación una referencia al propio CodeX, a través de una interfaz de comunicación que hace las veces de la caja de herramientas de interacción (*ToolBox*) de los sistemas tradicionales. Esta interfaz de comunicación se encuentra encapsulada en una instancia de la clase UIMS (*User Interface Management System*).

En el objeto UIMS proporcionado a la aplicación cliente se encuentra declarados todos aquellos métodos con los que dicha aplicación cuenta para establecer un diálogo interactivo con el usuario o para registrar y dar de baja procesos de usuario y objetos perceptibles por dicho usuario. Este objeto funcionará como canal de comunicación directo entre cualquiera de las aplicaciones clientes de GADEA y el módulo CodeX.

El mecanismo empleado para proporcionar esta referencia a la aplicación cliente se basa en la invocación dinámica de métodos disponible en toda arquitectura basada en mecanismos de reflexión. Por medio de la información almacenada en el *Application Record*, en el momento en el que usuario decide arrancar una aplicación, CodeX crea dinámicamente una instancia del objeto principal de dicha aplicación e invoca un método constructor especial presente en dicho objeto, el cual ha de contener como único parámetro una instancia de la clase UIMS, tal y como puede verse en la función *launchApplication* del Código 2.

En este ejemplo, es el propio objeto UIMS el que arranca la aplicación comprobando que la aplicación cliente dispone de un constructor en el que se recibe como parámetro una instancia de la clase UIMS.

Código 2. Ejemplo de arranque automático de una aplicación cliente en GADEA.

```
public void launchApplication (String mainObject)
 {
 try
 {
  // Inspects the <mainObject>'s constructor list
  Class cl = Class.forName (mainObject);
  Constructor constructors [] = cl.getDeclaredConstructors();

  // Search for a constructor with a unique parameter of the
     // UIMS Class
  int theConstructor = 0;

     for (int x=0; x<constructors.length; x++)
  {
  Class parameters[] = constructors [x].getParameterTypes ();

     // Checks constructors with one parameter only.
  if (parameters.length == 1)
  for (int y = 0; y < parameters.length; y++)
  {
        if (parameters[y].getName() ==
           (this.getClass()).getName())
   theConstructor = x;
  }
  }

  // Invokes the right constructor and provides the
     // UIMS instance
  Object paramValues[] = new Object[1];
  paramValues[0] = this;
  Object app =
     constructors[theConstructor].newInstance(paramValues);
 }
 catch (InstantiationException e)
 {
  System.err.println ("We are not allowed instantiate
     the class " + e.toString());
 }
 catch (IllegalAccessException e)
 {
  System.err.println ("We are not allowed to get the
     class info " + e.toString());
 }
 catch (InvocationTargetException e)
 {
  System.err.println ("We are not allowed call the
     object's constructor " + e.toString());
 }
 catch (ClassNotFoundException e)
 {
  System. err.println ("Not allowed to get the class
     info" + e.toString());
 }
 }
```

De este modo se garantiza que la clase cliente dispone de un objeto UIMS para su comunicación con el módulo CodeX y ello se logra sin tener que imponer ninguna restricción especial a la clase principal de la aplicación ya que como puede apreciarse, no se insta a que dicha clase implemente ningún tipo de interfaz especial en la que se obligue a implementar un método que reciba como parámetro una instancia de la clase UIMS. Si no existe ningún constructor que cumpla esta característica, simplemente se ignora la petición del usuario ya que la

aplicación destino no es una aplicación GADEA. Nótese además que éste mecanismo de registro tampoco impone una nomenclatura especial para el método de arranque (se trata del constructor y dependerá del nombre que el diseñador de la aplicación le ha otorgado a la clase). Tampoco se impone heredar de ninguna clase especial de GADEA para beneficiarse de este sistema, ya que la selección de la clase de la cual hereda la clase principal registrada en este sistema, dependerá única y exclusivamente de condicionantes propios de la ingeniería del software y no del diseño del modelo de interacción.

El objeto UIMS recibido por el constructor del objeto cliente invocado será empleado como veremos más adelante para la comunicación entre la aplicación cliente y el mecanismo de separación de funcionalidad de GADEA. En el diseño de este mecanismo de arranque automático se prevé que sea responsabilidad de la propia aplicación cliente el almacenado y gestión de la instancia de la clase UIMS, teniendo en cuenta especialmente que cualquier objeto de la aplicación que requiera registrar procesos de usuario, darlos de baja o realizar cualquier secuencia de tareas de interacción con el usuario necesitará este objeto para ordenar dichas tareas a GADEA. Es por ello que se sugiere implementar el constructor de modo similar al objeto incluido como ejemplo en el Código 3.

Código 3. Ejemplo del modo de iniciación recomendado para una aplicación cliente del sistema GADEA.

```
import // GADEA's packages
import // other packages

public class MyMainObject extends myCode
{
 UIMS aInterface = null;

   // Object's constructors
   MyMainObject (UIMS myAInterface)
 {
 super ();
      aInterface = myAInterface;

      // Provides a reference to a <aInterface> object
      // to any other interactive object.
      OtherIObject otherIObject = new OtherIObject (aInterface);
      …
      // more objects.
   }

   // Other constructors
   myMainObject ()
   {
      …
   }
}
```

Como se puede apreciar, la clase *myCode* no hereda de ninguna clase especial proporcionada por GADEA ni tampoco implementa ningún interfaz de este sistema. La única condición necesaria para garantizar una compatibilidad plena con GADEA está en que esta clase disponga de un constructor que reciba como único parámetro una instancia de la clase UIMS, pudiendo tener otros constructores alternativos, pero teniendo en cuenta que el primero de ellos en ser invocado será el constructor descrito.

La tarea fundamental de este constructor será almacenar la instancia de la clase UIMS proporcionada por CodeX y proporcionar una referencia a la misma para todo objeto interactivo que la necesite a la hora de realizar tareas de interacción con el usuario.

7.2 Registro Automático de Procesos de Usuario

En la literatura hay constancia abundante del esfuerzo que la comunidad de interacción y comunicación humana se ha tomado en los últimos años para desarrollar especificaciones formales en la definición detallada de los requisitos de interacción de una interfaz de usuario [Paterno y Palanque (1996) y Rouff (1995)].

Este línea de investigación ha permitido definir el comportamiento perceptible de una aplicación en función de una secuencia temporal de acciones así como del ciclo de vida de sus objetos. En este sentido hay que destacar el trabajo de Palenque y Bastide (1990) en el que ésta secuencia de acciones es modelada por medio de redes de Petri.

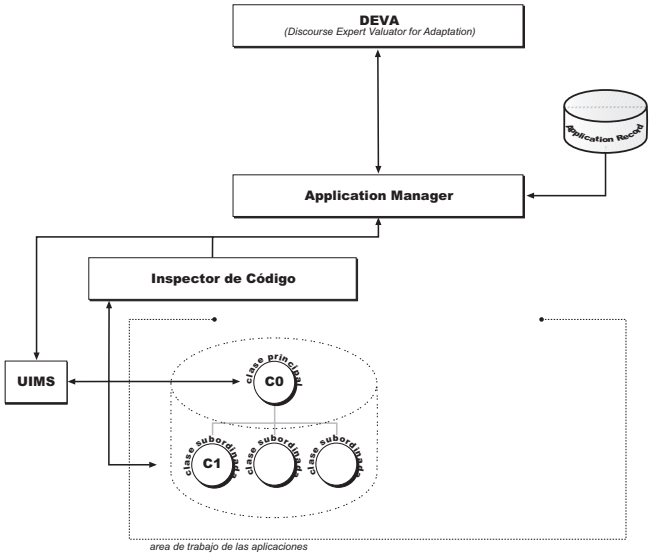


Figura 17. El componente UIMS sirve de puente entre cualquier fichero binario de la aplicación y el *Application Manager* de CodeX.

Un planteamiento similar ha sido empleado por DeCarolis y DeRosis (1994) en el que se amplía el poder de representación de las redes de Petri, tomando en cuenta no solo el papel de las acciones (procesos) sino también el del flujo de datos mediante redes coloreadas de Petri, empleadas en este caso para definir y evaluar interfaces sensibles al contexto. Estos autores [DeCarolis y DeRosis (1993)] introdujeron relaciones adaptables entre los nodos de las redes coloreadas de Petri y los objetos perceptibles de la interfaz. Sobre este modelo se pueden añadir restricciones temporales en las acciones, así como puntos de referencia y transiciones entre acciones dependiendo del estilo de interacción adoptado.

Otra estrategia similar a las comentadas, pasa por el empleo de grafos dirigidos, cuyos nodos representan estados de una aplicación y

en donde los arcos representan el flujo de datos entre los mismos [Bumbulis et. al. (1995)]. Existen variantes de este método en el que se emplean diagramas de transición de estados [Carneiro-Coffin (1995)] modelando los estados de una aplicación desde el punto de vista del usuario y separando el comportamiento de la aplicación de su interfaz, simplificando por ello el modelo desde las primeras fases de su construcción. En estos diagramas las transiciones de un estado a otro son causadas por eventos generados externamente (acciones invocadas por el usuario) y funciones de manipulación de datos que son realizadas durante las transiciones, como ocurre en el modelo propuesto por Timo Jokela [c. Rouff (1995)]. También se han diseñado diagramas de transición de estados ampliados en los que se especifican las restricciones temporales y los flujos de datos, tales como el modelo propuesto por Christopher Rouff (1995).

Como podemos observar, las diferentes técnicas empleadas para la descripción formal de interfaces tienden a converger en único modelo en el que se representa a la aplicación como una serie limitada de estados perceptibles por el usuario a la que se accede por medio de la invocación de determinados procesos o por la modificación de determinadas propiedades de un objeto o colección de objetos.

De forma simplificada, es posible representar el modelo de procesos de una aplicación como un gran autómata que posee una colección de estados a los que se puede acceder (partiendo de un estado inicial) por medio de una serie de acciones externas invocadas por los usuarios o por medio de acciones internas invocadas por la propia aplicación en función de determinados procesos de tratamiento de la información.

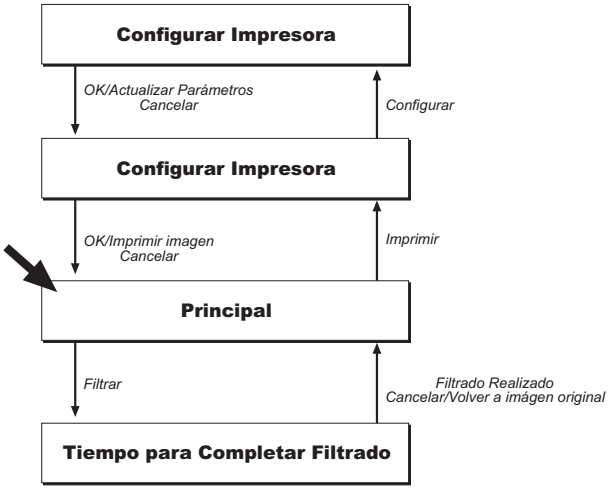


Figura 18. Representación abstracta del estado interno de una supuesta aplicación de tratamiento de imágenes, la cual es guiada por los procesos de usuario *Imprimir* y *Filtrar* .

En la Figura 18 se muestra un ejemplo de este tipo de modelos para una aplicación de edición de imágenes en las que, para simplificar, sólo existen dos procesos de usuario: *imprimir* una imagen y *filtrar* una imagen. La invocación directa por parte del usuario de cada uno de estos procesos permite acceder a los estados que permiten configurar la impresora para *imprimir* o *filtrar* la imagen. En el caso del estado

imprimir, es el usuario el que toma la decisión de regresar al estado inicial, bien proporcionando al sistema los parámetros necesarios para realizar la impresión (número de copias, calidad de impresión, etc.) o bien cancelando el proceso. En el caso del estado filtrar, es la propia aplicación la que toma la decisión de retornar al estado inicial tras filtrar la imagen.

La capacidad de simplificar el modelo de procesos (y en cierta medida el modelo de interacción) de una aplicación mediante el empleo autómatas, nos permite dar un paso más en la formalización de requisitos de interacción, pudiendo convertir directamente cualquiera de los modelos descritos en código ejecutable, simulando los estados mediante métodos que representan procesos de usuario y la transición entre los mismos mediante invocaciones directas a tales métodos.

Esta es precisamente la estrategia planteada por CodeX para separar la funcionalidad de las aplicaciones (la representación formal de su estado interno) de la interfaz. Para ello, éste módulo considera una notación simple para el nombrado de cada uno de las métodos que implementan los procesos de usuario de la aplicación. Así, en función del nombre de un método, CodeX sabrá si se trata de un proceso de usuario o de un método interno de la aplicación. Este sencillo mecanismo permite convertir en código cualquier especificación formal basada en transiciones de estados.

Aún así no será suficiente con conocer que métodos son internos y cuales son externos ya que también es necesario saber que procesos han de estar disponibles para el usuario en función del estado interno de la aplicación. Es por ello que también es preciso detectar de forma automática las *precondiciones* y *postcondiciones* comentadas en el apartado *Facilidad de Uso* del capítulo 5 (*Diseño Centrado en el Usuario*) y asociarlas de algún modo a cada uno de los procesos de usuario disponible.

En CodeX la definición de las *precondiciones* y *postcondiciones* se realizará también por medio métodos internos de la aplicación, los cuales devolverán verdadero si la condición se cumple y falso en caso contrario. El hecho de codificar las condiciones por medio de métodos y no de propiedades *booleanas* permite alcanzar el mayor grado de flexibilidad posible, ya que dentro de un método es posible especificar complejos predicados en el que participen múltiples variables, caso que no se daría en caso de emplear variables *booleanas* de manera exclusiva. Nótese que el estado de una condición no dependerá solamente del valor de las variables que forman parte de un predicado, sino que en ocasiones será necesario llegar a ejecutar procesos internos de control para determinar si un proceso de usuario puede ejecutarse o no. Así, en el ejemplo de la figura Figura 18, el proceso de filtrado de la imagen no sólo depende de tener una imagen activa (condición *booleana*) sino también de poseer memoria auxiliar suficiente para ejecutar la operación, condición que sólo se puede detectar mediante la invocación de procesos internos que calculen la cantidad de memoria necesaria y su disponibilidad en el sistema.

La asociación entre los métodos que representan procesos de usuario y sus *precondiciones* y *postcondiciones* se realiza también por medio de una sencilla notación.

En definitiva, podemos decir que CodeX clasificará los métodos de cualquier aplicación cliente en función de las siguientes categorías:

- Procesos de Usuario.
- *Precondiciones.*
- *Postcondiciones.*
- Métodos internos de la aplicación.

El diseñador de la aplicación dispone de libertad total para clasificar los métodos de sus aplicaciones dentro de cualquiera de las categorías precedentes valiéndose para ello de un sencillo mecanismo de notación para el nombre de dichos métodos, el cual pasamos a describir.

7.2.1. Procesos de Usuario

Los procesos de usuario son aquellos que se pueden invocar directamente desde la interfaz de una aplicación por orden expresa del usuario, a través de los distintos mecanismos de interacción disponibles en un momento determinado en dicha interfaz. Estos métodos son del tipo de comandos escritos, órdenes orales, pulsaciones del ratón etc.

Nótese que la ejecución de un proceso de usuario puede dar pie a la ejecución de otros procesos de usuario. Así por ejemplo, el proceso de usuario Imprimir del ejemplo de la Figura 18 puede dar lugar al proceso de usuario *Configurar Impresora.*

En tiempo de ejecución, GADEA considerará que todos aquellos métodos de un objeto que empiecen por el acrónimo UP_ (*User Process*) serán puntos de entrada para la computación de los procesos de usuario respectivos. Al referirnos a estos métodos como puntos de entrada estamos definiendo que la definición de un proceso de usuario puede estar formada por uno o más métodos, proporcionando así la mayor flexibilidad a los diseñadores y programadores para la estructuración de sus artefactos software.

En tiempo de ejecución, en función de la lógica interna de la aplicación, CodeX podrá iniciar el registro automático de los métodos de un objeto, registrando todos aquellos métodos cuyo nombre empiece por *UP_* como procesos de usuario.

Supongamos por ejemplo que se dispone de una aplicación de procesamiento de textos en donde existe un objeto denominado *Document* que representa los documentos con los que trabaja el procesador. Si el programador quiere definir un proceso de usuario para cerrar el documento, debería definir un método en el objeto *Document* cuyo nombre empiece por *UP_* seguido del nombre interno con el que el equipo de diseño se refiere el proceso de *usuario Cerrar Documento.* Obviamente, dicho nombre interno es un parámetro que debería quedar reflejado en la documentación del análisis de la

aplicación y por lo tanto debería ser conocido por todos los efectivos implicados en el desarrollo de la aplicación. Supongamos que el nombre asignado es *closeDocument*. En este caso, el nombre con el que el programador debería nombrar al método que hace las veces de punto de entrada del proceso de usuario debería ser *UP_closeDocument*, tal y como se puede ver en el código Código 4.

Código 4. Punto de entrada para el proceso de usuario *closeDocument*.

```
public class Document
{
   public int openedDocs = 0;

   // constructor
   Document (…)
   {
…
   }

   // User processes
   public boolean UP_closeDocument ()
   {
      // Code needed to close the document.
   }
}
```

Cuando el programador le dé la orden de registrar los puntos de entrada a procesos de usuario del objeto *Document*, CodeX inspeccionará todos los métodos de la clase *Document*, en busca de métodos de tipo *UP_*, detectando el método *UP_closeDocument* y registrando este método en sus estructuras de datos internas, tomando buena nota de la clase que contiene el método. En el Código 5 se puede ver un sencillo ejemplo escrito en lenguaje Java en el que dado un objeto, se inspeccionan todos sus métodos mediante mecanismos de reflexión, tomando nota de todos aquellos cuyo nombre empiece por *UP_*.

Código 5. Inspección de todos los métodos de una clase en búsqueda de puntos de entrada a procesos de usuario.

```
public void registerUPEntryPoints (Object object)
{
   try
   {
      Method methods [] = (object.getClass()).getDeclaredMethods();

  for (int x=0; x<methods.length; x++)
  {
        Class parameters[] = methods [x].getParameterTypes ();

        // We inspect methods without parameters only
        if ((methods [x].getParameterTypes ()).length == 0)
   if ((methods[x].getName()).startsWith ("UP_"))
   {
   // UP registration here
         registerUP (object, methods[x].getName());
   }
  }
   }
   catch (IllegalAccessException e)
 {
  System.err.println ("We are not allowed to get the
     class info " + e.toString());
 }
 catch (ClassNotFoundException e)
 {
     ;
```

```
  }
}
```

Una vez que el método de tipo *UP_* ha sido detectado mediante el mecanismo de reflexión descrito, CodeX notificará a DEVA la existencia del nuevo método disponible en el objeto inspeccionado mediante la generación de un evento del cual DEVA es receptor. A partir de este momento y si el proceso de usuario se encuentra activo, DEVA podrá proporcionar el proceso *closeDocument* al usuario mediante una modalidad y mecanismo de interacción concretos, incluyéndolo por ejemplo en uno de los menús de la aplicación.

Cuando el usuario seleccione este proceso de entre los disponibles en la aplicación (dependiendo del estado de la misma), DEVA se lo notificará a CodeX mediante otro evento y éste a su vez localizará la instancia correspondiente de la clase *Document* e invocará a su método *UP_closeDocument* tal y como se puede ver en el Código 6.

Código 6. Mecanismo de invocación de puntos de entrada de procesos de usuario.

```
public void invokeUPEntryPoint (Object object, String methodName)
{
   try
   {
      Method method = (object.getClass()).getDeclaredMethod ("UP_" +
      methodName);
  method.invoke (app, null);
   }
 catch (InvocationTargetException e)
 {
 System.err.println ("We are not allowed call the method
      " + e.toString());
 }
}
```

Nótese que el método que representa el punto de entrada para el proceso de usuario es una función *booleana* sin parámetros (ver Código 4). Esta función *booleana* deberá devolver un valor verdadero en el caso en el que la totalidad del proceso de usuario se haya realizado sin novedad, es decir, sin haber sido cancelado por el usuario en cualquier momento de su ejecución u abortado debido a causas de índole técnica (problemas al abrir un fichero, falta de memoria, etc.). En caso contrario, si el proceso ha sido cancelado u abortado, el programador tendrá la responsabilidad de hacer que éste método devuelva un valor igual a falso.

En el caso de que el método se ejecute correctamente, devolviendo un valor verdadero, CodeX asume la posibilidad de que ésta ejecución exitosa del proceso haya afectado al estado interno de la aplicación, modificando los valores de propiedades asociadas de un modo u otro con la interfaz de usuario. En este caso, CodeX se verá obligado a comprobar la disponibilidad de todos y cada uno de los procesos de usuario registrados, notificando a DEVA cualquier cambio detectado.

Siempre que tras la ejecución del punto de entrada de un proceso de usuario, el método respectivo devuelva un valor igual a falso, todo el proceso de comprobación del estado de la aplicación se omite, con lo que se gana en eficiencia. Aunque la política de funcionamiento de todo

gestor de interfaces de usuario basado en la separación de funcionalidad de la interfaz debe estar regida por el conservadurismo, ésta no tiene por qué estar reñida con la eficiencia. En todo caso y ante políticas de trabajo más restrictivas, un sistema del tipo de CodeX, podría evaluar siempre el estado interno de la aplicación cada vez que se ejecute un método de este tipo, eliminando la condición *booleana* del método *UP_*.

7.2.2. Precondiciones

Las *precondiciones* o condiciones de ejecución se refirieren a las precondiciones comentadas en el apartado 5.2.3 y por lo tanto indican el conjunto de condiciones que han de ser satisfechas en tiempo de ejecución para que el proceso de usuario esté disponible. Estas condiciones dependerán del estado interno de la aplicación o del objeto sobre el que éstas se puedan aplicar.

Las *precondiciones* de un proceso de usuario representan todas aquellas condiciones que son necesarias para que dicho proceso pueda ejecutarse, siempre desde el punto de vista del usuario del sistema y por lo tanto no deben interpretarse desde la óptica de la ingeniería del software ni desde la del modelado del procesos.

Las *precondiciones*, adoptadas de sistemas inteligentes para el desarrollo de interfaces de usuario como el MASTERMIND [Foley et. al. (1991)] (mas tarde Humanoid y Mastermind), se emplean por vez primera en tiempo de ejecución en GADEA, ya que con anterioridad sólo se usaban en tiempo de edición y diseño de las interfaces. Hasta el momento, el concepto de *precondición* su *postcondición* asociada (la cual analizaremos en el apartado 7.2.3) eran parte del análisis y diseño de un modelo de interacción, definidos en el momento de establecer los requisitos de todo proceso de usuario. Sin embargo, tanto las *precondiciones* como las *postcondiciones* de un proceso de usuario no tenían continuidad clara en la construcción del mismo, siendo programadas sin ningún tipo de formalismo ni metodología en los puntos del código en donde intervenían.

Gracias al mecanismo de reflexión e inspección automática de código de GADEA, estas condiciones especiales de ejecución de los procesos de usuario pueden incorporarse de manera formal al código representativo de un modelo de usuario, siendo pieza clave de la concepción espacial y temporal del estado de una aplicación compatible con el modelo GADEA.

Todo punto de entrada a un proceso de usuario, puede tener asociada una precondición, la cual estará representada por una serie de predicados que han de cumplirse para que la acción representada por dicho proceso esté activa en la interfaz de la aplicación y por lo tanto pueda ser seleccionada por el usuario.

Todos los predicados que forman parte de una *precondición* han de incorporarse en un método especial que hará las veces de *precondición*. Dicho método será una función *booleana* que devolverá cierto si la *precondición* se cumple y falso en caso contrario. La notación adoptada

por GADEA implica que los métodos representativos de *precondiciones* han de denominarse de manera idéntica a los procesos de usuario asociados, pero incluyendo el prefijo *PRE_*. Así, la precondición asociada para el punto de entrada al proceso de usuario *UP_closeDocument* empleado en los ejemplos anteriores, habrá de denominarse *PRE_UP_closeDocument*.

Código 7. proceso de usuario *closeDocument* y su *precondición* asociada.

```
public class Document
{
   public int openedDocs = 0;

   // constructor
   Document (…)
   {
 …
   }

   // User processes
   public boolean UP_closeDocument ()
   {
      // Code needed to close the document.
      …
   }

   // Preconditions
   public boolean PRE_UP_closeDocument ()
   {
      // Is there any opened document?
      Return (openedDocs > 0);
   }
}
```

En el ejemplo incluido en el **Código 7** puede apreciarse que una posible *precondición* para el cierre de un documento consiste en comprobar que existen documentos para cerrar (número de documentos abiertos mayor que cero). En el caso en el que el número de documentos abiertos sea igual a cero, lógicamente no será posible cerrar ningún documento por lo que la función *PRE_UP_closeDocument* devolverá un valor igual a falso y el método *UP_closeDocument* no será activado en la interfaz de usuario.

Cuando se registran los puntos de entrada para los procesos de usuario de una clase, GADEA registra también las *precondiciones* empleando un mecanismo similar al usado en el Código 5, almacenando una referencia a dichas *precondiciones* en las estructuras de datos internas de CodeX. De este modo, se puede invocar dichas *precondiciones* en tiempo de ejecución cuando el sistema experto así lo considere oportuno, siendo empleadas por los prototipos implementados de CodeX para activar o desactivar comandos, aunque la riqueza semántica de las mismas hacen factible su empleo futuro en el desarrollo de sistemas de ayuda (que expliquen la razón por la cuál un determinado comando esté desactivado) o tutores inteligentes que, aplicando un mecanismo de backward-chain permitan indicar al usuario cómo satisfacer las *precondiciones* necesarias para la ejecución de determinados comandos.

Como se puede apreciar, éste modo de obrar permite a los diseñadores y programadores de las aplicaciones GADEA una gran flexibilidad en el diseño, ya que la confección de las precondiciones se

reduce al empleo de variables contextuales que se pueden enfocar directamente sobre las dependencias semánticas de la aplicación en función de su estado y siempre a partir de cualquier representación del mismo basada en los métodos formales descritos (diagramas de transición de estados, redes de Petri, etc.).

Aunque todo punto de entrada para un proceso de usuario debería tener una *precondición* asociada, GADEA no considera un error la ausencia de dicha precondición, asumiendo un valor verdadero para la misma, cosa que ocurre en la mayoría de los procesos de usuario. Esta estrategia de diseño proporciona una mayor facilidad de uso que la presente en los sistemas tradicionales, ya que la expresión final de una *precondición* no solamente puede ser representada fácilmente en una forma que pueda ser evaluada convenientemente en tiempo de ejecución, sino que ésta se evalúa de forma automática sin la necesidad de incorporar código extra.

7.2.3. Postcondiciones

Las *postcondiciones* o efectos de ejecución se refieren a las *postcondiciones* del apartado 5.2.3, indicando por tanto al conjunto de operaciones a realizar una vez que se ha ejecutado satisfactoriamente un proceso de usuario. Estas operaciones pueden cambiar el estado de la aplicación, de un objeto o de una propiedad.

Las *postcondiciones* representan bloques de código cuya funcionalidad consiste en asignar nuevos valores para las variables contextuales empleadas en las *precondiciones* del punto de entrada de un proceso de usuario. Dichas *postcondiciones* se ejecutan justo a continuación de los procesos de usuario asignados, siempre y cuando éstos terminen de forma correcta, es decir, siempre que estos no hayan sido abortados por el usuario o siempre que no se haya producido ningún falo técnico en su ejecución.

Las *postcondiciones* representan pues el mecanismo natural empleado por los programadores y diseñadores de una aplicación para modificar los valores de la *precondición* de un proceso y por ende son empleadas para activarlo o desactivarlo en un punto concreto de la aplicación, siempre dependiendo de su estado interno.

Al igual que ocurre con las *precondiciones*, GADEA permite emplear por primera vez este concepto en tiempo de ejecución, siendo compatible con las técnicas estáticas precedentes en cuanto a la definición de *precondiciones* en tiempo de análisis y diseño de los modelos de interacción, ya que el mecanismo de reflexión empleado por GADEA permite definir las *postcondiciones* en una forma en la cual pueden ser evaluadas convenientemente en tiempo de ejecución.

En la notación empleada por GADEA, la *postcondición* asociada a un proceso de usuario define que su nombre ha de ser idéntico al punto de entrada de dicho proceso de usuario, pero empezando por el prefijo *POST_*. Siguiendo con el ejemplo del *Document* y su proceso *closeDocument*, el método representativo de la *postcondición* asociada a dicho proceso *closeDocument* debería denominarse

POST_UP_closeDocument, tal y como puede apreciarse en el ejemplo incluido en el Código 8.

Código 8. proceso de usuario *closeDocument* con su *precondición* y *postcondición* asociadas.

```
public class Document
{
   public int openedDocs = 0;

   // constructor
   Document (…)
   {
 …
   }

   // User processes
   public boolean UP_closeDocument ()
   {
      // Code needed to close the document.
      …
   }

   // Preconditions
   public boolean PRE_UP_closeDocument ()
   {
      // Is there any opened document?
      Return (openedDocs > 0);
   }

   // Postconditions
   public void POST_UP_closeDocument ()
   {
      // The number of opened docs decreases.
      openedDocs--;
   }
}
```

En el ejemplo del Código 8 se incluye como posible *postcondición* para la acción de cerrar un documento, la disminución en un grado del valor de la propiedad de la clase *Document* que lleva la cuenta del número de documentos abiertos (*openedDocs*).

Siguiendo su parecido con las *precondiciones*, la especificación de *postcondiciones* no es de uso obligado en un código GADEA, de tal modo que pueden existir puntos de entrada para un proceso de usuario sin su correspondiente *postcondición*. En estos casos, al no existir *postcondición* asociada, obviamente ésta no se invoca.

7.2.4. Métodos Internos de la Aplicación

Aquellos métodos que por su nombre no cumplen con la notación prevista para los puntos de entrada a procesos de usuario, *precondiciones* y *postcondiciones* son considerados por CodeX como métodos internos de la aplicación y por lo tanto pertenecen a la definición a la especificación del esquema interno de la misma (descrito en 6.1.3), bien formando parte de su modelo de procesos o bien perteneciendo a su modelo de datos.

Sin embargo, el sistema de notación previsto por GADEA puede ser ignorado por los programadores y/o diseñadores de una aplicación, especificando manualmente el nombre de aquellos métodos que actuarán como procesos de usuario, como *precondiciones* o como

postcondiciones por medio de los métodos *registerUP*, *registerPRE_UP* y *registerPOST_UP*, todos ellos pertenecientes a la clase UIMS comentada en el apartado 7.1.1 (*Comunicación con el Espacio de Objetos de la Aplicación*).

Por medio de estos métodos, el programador puede indicar a CodeX el nombre y el objeto que implementa cualquiera de los métodos descritos en los apartados precedentes, aún cuando no cumplan con el criterio de notación descrito, proporcionando el mayor nivel de flexibilidad posible a la estructuración del código de las aplicaciones GADEA, permitiendo con ello una adaptabilidad total al enfoque metodológico empleado por la organización que desarrolla el software.

Los tres métodos descritos reciben como parámetro una referencia al objeto que posee el método a invocar y el nombre de dicho método. La única condición que han de cumplir estos métodos es la propia de la funcionalidad para la que están diseñados. Para los métodos que representan procesos de usuario y precondiciones, el único requisito exigible es el de que se trate de métodos booleanos sin parámetros. En el caso de métodos que implementen postcondiciones, el requisito exigido es que se trate de métodos sin valor de retorno (*void*) y que carezcan de parámetros. En el caso en el que los métodos registrados no cumplan con los requisitos mínimos exigidos, GADEA lo notificará mediante una excepción en el propio método de registro y detendrá la ejecución de la aplicación, ya que no puede garantizar su correcta ejecución.

Código 9. Registro manual de puntos de entrada a procesos de usuario, *precondiciones* y *postcondiciones* en el ejemplo *Document*.

```
public class Document
{
   public int openedDocs = 0;

   // constructor
   Document (UIMS interface)
   {
      try
      {
      interface.registerUP (this, "myClose");
         interface.registerPRE_UP (this, "myClose",
            "myClosePrecondition");
         interface.registerPOST_UP (this, "myClose",
            "myClosePostcondition");
      }
      catch (Exception e)
      {
         System.err.println ("Methods don't match basic
         requirements");
         System.exit (1);
      }
   }

   // User processes
   public boolean myClose ()
   {
      // Code needed to close the document.
      …
   }

   // Preconditions
   public boolean myClosePrecondition ()
   {
      // Is there any opened document?
      Return (openedDocs > 0);
   }
```

```
   // Postconditions
   public void myClosePostcondition ()
   {
      // The number of opened docs decreases.
      openedDocs--;
   }
}
```

En el ejemplo del Código 9 se puede apreciar una versión especial de la clase *Document*, en la que el programador emplea nombres de puntos de entrada a procesos de usuario, *precondiciones* y *postcondiciones* distintos de los esperados por GADEA (*myClose*, *myClosePrecondition* y *myClosePostcondition* en lugar de los respectivos *UP_closeDocument*, *PRE_UP_closeDocument* y *POST_UP_closeDocument*). Nótese como el nombre del proceso de usuario (myClose) actúa como clave con la que se relaciona su *precondición* (*myClosePrecondition*) y *postcondición* (*myClosePostcondition*) siguiendo una relación uno a uno.

En el ejemplo, el registro de éstos métodos se realiza en el propio constructor del objeto, el cual recibe una instancia de la clase UIMS como parámetro. Esta clase, proporcionada a su vez por la clase que crea el objeto *Document*, es empleada para registrar los métodos. Naturalmente, el registro de estos métodos se puede realizar en cualquier parte del código y en el momento en el que la especificación del modelo de usuario así lo crea conveniente.

Como es lógico, si se pueden registrar métodos de forma manual, también se pueden dar de baja de forma manual. Esto se logra mediante los métodos *unregisterUP*, *unregisterPRE_UP* y *unregisterPOST_UP* para las entradas a procesos de usuario, *precondiciones* y *postcondiciones* respectivamente.

7.3 Invocación Automática de Procesos de Usuario

El proceso automático de registro de métodos en GADEA no se ejecuta de manera directa, sino que necesita de orden directa del programador, quien siguiendo los dictados del modelo de interacción, será el responsable de especificar el momento en el que un determinado conjunto de métodos pasan a estar activos (registrados) o desactivados.

El punto idóneo para el registro automático de los métodos de una clase es su propio constructor de la misma, aunque como ya se ha comentado en el apartado 7.2.4 (*Métodos Internos de la Aplicación*), la arquitectura de GADEA permite registrar procesos y darlos de baja en cualquier parte del código.

En el ejemplo del Código 10, se puede ver el proceso clásico de registro automático de procesos de usuario para una aplicación GADEA, siguiendo el ejemplo de la clase *Document* desarrollado en este capítulo.

Código 10. Registro automático de procesos de usuario en la clase *Document*.

```
public class Document
{
   public int openedDocs = 0;

   // constructor
   Document (UIMS interface)
   {
 interface.registerEveryUP (this);
   }

   // User processes
   public boolean UP_openDocument ()
   {
      // Code needed to open the document.
      …
      openedDocs++;
   }

   public boolean UP_closeDocument ()
   {
      // Code needed to close the document.
      …
   }

   // Preconditions
   public boolean PRE_UP_closeDocument ()
   {
      // Is there any opened document?
      Return (openedDocs > 0);
   }

   // Postconditions
   public void POST_UP_closeDocument ()
   {
      // The number of opened docs decreases.
      openedDocs--;
   }
}
```

En el constructor se recibe como parámetro una instancia de la clase UIMS, la cual es empleada para invocar al método *registerEveryUP*, el cual es una variante del *registerUPEntryPoints* (esbozado en el Código 5) en la cual se registran en una sola operación todos los puntos de entrada a procesos de usuario, *precondiciones* y *postcondiciones* existentes en la clase. Este método empieza su trabajo registrando todos los puntos de entrada a procesos de usuario de la clase, es decir, buscando todos los métodos de tipo *UP_*. Una vez hecho esto, procederá a buscar todas las *precondiciones* y *postcondiciones* asociadas a los métodos *UP_* registrados previamente. Este mecanismo elimina la posibilidad de registrar una *precondición* (método *PRE_UP_*) o una *postcondición* (método *POST_UP_*) sin su correspondiente proceso de usuario asociado. En el caso de que para un determinado método UP exista más de una *precondición* o *postcondición*, el proceso de registro abortará, notificando el error mediante una excepción.

Si la operación concluye satisfactoriamente y todos los procesos de usuario de la clase han sido registrados (*openDocument* y *closeDocument*), se procederá a invocar a todos y cada uno de los métodos de tipo *PRE_UP_* con el objeto de conocer si los métodos *UP_* asociados pueden estar disponibles o no en DEVA en función del estado actual de la aplicación.

En el ejemplo del Código 10 se invocará al método *PRE_UP_closeDocument* obteniendo un valor falso, ya que el número de documentos abiertos al crear el objeto *Document* es cero. Nótese que dado que no existe un método llamado *PRE_UP_openDocument*, se asume que la precondición para el proceso de usuario *openDocument* es siempre verdadera (valor por defecto) por lo que el método *UP_openDocument* siempre estará activado en la interfaz para que el actual usuario pueda seleccionarlo.

Cuando un usuario ordena ejecutar el punto de entrada de un proceso de usuario, seleccionándolo en la interfaz proporcionada por GADEA, CodeX invoca al correspondiente método *UP_*. Si la ejecución de éste método devuelve un valor verdadero (el usuario no ha cancelado su ejecución), entonces se ejecuta automáticamente la *postcondición* asociada a dicho proceso de usuario. Dado que la ejecución de la *postcondición* puede haber alterado el valor de las variables contextuales de la aplicación, la ejecución de todo proceso de usuario implica un posible cambio en el estado interno de la aplicación lo que se puede traducir en un cambio del estado de disponibilidad de cualquier proceso de usuario, no coincidiendo necesariamente con el proceso invocado por el usuario. Por ello, CodeX evalúa de nuevo las *precondiciones* de todos los procesos de usuario registrados, activando aquellos cuyas *precondiciones* asociadas haya devuelto un valor igual a verdadero y desactivando aquellos cuyas *precondiciones* hayan devuelto valores igual a falso.

En el ejemplo seguido, si el usuario elige la opción *openDocument*, CodeX procederá automáticamente a la invocación del método *UP_openDocument* evaluando el valor de respuesta. Si este es verdadero (el usuario ejecutó el proceso hasta el final), se procederá a la evaluación automática de su *postcondición* (*POST_UP_openDocument*), la cual, como se puede apreciar, puede modificar el valor de la variable *openedDocs*, incrementando en uno su valor (caso de poder abrir un documento). Siguiendo una política evidentemente conservadora, CodeX repasará a continuación el estado de todas las precondiciones registradas invocando al método *PRE_UP_* asociado a todo método *UP_*. En el ejemplo se invocará al método *PRE_UP_closeDocument* obteniendo ahora un valor verdadero, por lo que el método asociado (*UP_closeDocument*) pasará a estar activo en la interfaz de la aplicación.

8

8 El Diálogo

Diálogo es colaboración de dos personas en un tema

Bontempelli

8.1 Canales de Comunicación

En base al paradigma de los sistemas de comunicación y la Teoría General de Sistemas (TGS) descrito en el capítulo 4 (*La Comunicación*), los mecanismos de interacción de GADEA están fundamentados en una serie de canales de comunicación sobre los cuales se establecen los correspondientes diálogos interactivos para el intercambio de información con el usuario.

Todo canal de comunicación en GADEA está basado en un modelo de interacción dual en el que se transmite y se recibe información por medio de mecanismos de interacción visuales y auditivos.

Los mecanismos de interacción visuales se traducen en ventanas y cuadros de diálogo sobre los que se pueden situar los diversos *widgets* que forman parte de un proceso de intercambio de información entre el usuario y la aplicación. Hay que destacar que estos mecanismos de interacción visual sobre los que es posible establecer canales de

comunicación son simples contenedores de la información, la cual se transmite por medio de diálogos interactivos individuales.

Los mecanismos de interacción auditivos por otro lado, se traducen en narraciones sonoras de la información que se pretende transmitir por medio de los diálogos interactivos asociados al canal de comunicación. Para ello se emplean los sistemas de voz de la plataforma destino, describiendo los mensajes a ser transmitidos, disponiendo de la capacidad de recibir la información proveniente del usuario haciendo uso también del medio auditivo.

Como se puede apreciar, con la notable excepción de los ideogramas y otras representaciones gráficas, toda información contenida en un canal de comunicación es transmitida dos veces: una haciendo uso de mecanismos visuales y otra empleando mecanismos auditivos. De este modo, GADEA proporciona un canal de comunicación dual, el cual puede orientarse a dos tipos de usuarios diferentes: usuarios ciegos y usuarios con visión perfecta. Naturalmente existen diferentes grados entre estos dos valores extremos para el grado de capacidad visual de un individuo. El mecanismo de transmisión dual de información planteado por GADEA permite dar un soporte global a todo el abanico de tipos posibles de usuario situado entre los dos valores, complementando la información recibida por medio de dos tipos diferentes de órganos sensoriales partícipes de la experiencia interactiva: la vista y el oído.

Como se verá a partir del capítulo 12(*El Modelo Cognitivo*), las dos percepciones del canal de comunicación (visual y sonoro) podrán ajustarse de forma automática dependiendo de las capacidades perceptivas e incluso motrices de los usuarios. Así, si el usuario demuestra una predilección especial por un mecanismo de transmisión concreto o si uno de los órganos sensoriales implicado en el proceso se encuentra dañado (total o parcialmente), el canal de comunicación potenciará el mecanismo de interacción que favorezca la transmisión de información. Por ejemplo, el uso de la transmisión de mensajes sonoros puede ser intensivo en el caso de usuarios con graves discapacidades visuales, mientras que su empleo puede reducirse para usuarios con un grado óptimo de percepción visual.

La naturaleza de todo canal de comunicación en GADEA puede ser temporal o perenne. En el primero de los casos, se trata de canales de comunicación empleados puntualmente por la aplicación o solicitados por el usuario para intercambiar la información requerida por procesos de usuario de carácter efímero, existiendo tan solo en algunos estados aislados de la aplicación. En el segundo de los casos, la duración del canal de comunicación se puede asociar con el ciclo de vida completo de una aplicación, estando presente en todos y cada uno de los estados internos por los que puede pasar la aplicación. En los interfaces de usuario tradicionales, los canales de comunicación visual perennes pueden asimilarse con las ventanas de una aplicación, mientras que los canales de comunicación efímeros pueden identificarse con los cuadros de diálogo o cuadros de alerta, los cuales son destruidos una vez que el usuario ha proporcionado u obtenido la información requerida.

Naturalmente, los programadores y diseñadores de aplicaciones GADEA disponen de las herramientas necesarias para crear canales de comunicación de las dos tipologías mencionadas, o cambiar en tiempo de ejecución la duración de todo canal de comunicación. La duración prevista para todo canal de comunicación dependerá de los requisitos identificados durante la fase de análisis del modelo de interacción de la aplicación en desarrollo y será asimilable con la metáfora de alto nivel empleada por los diseñadores.

Toda aplicación GADEA dispone de un canal de comunicación permanente, creado por defecto en el momento de su inicio. Si el programador no indica lo contrario, todos los diálogos interactivos creados en el espacio de interacción de la aplicación se montarán sobre esta canal de comunicación. Obviamente, el programador no solo podrá crear todos los canales de comunicación alternativos que considere necesario, sino que podrá habilitar y desactivar cualquiera de ellos en cualquier momento a lo largo de la ejecución de la aplicación (incluido el canal por defecto).

El establecimiento de nuevos canales de comunicación, así como la configuración de sus características y el empleo sobre ellos de los diálogos interactivos se realiza en aquellos puntos de la aplicación en los que ésta recibe el control por parte de GADEA. Estamos hablando por tanto de los procesos de usuario, en la terna referida en el capitulo 7 (*Aplicaciones y Procesos de Usuario*), formada por puntos de entrada (*UP_*), *precondiciones* (*PRE_UP_*) y *postcondiciones* (*POST_UP_*).

8.2 Diálogos Interactivos

Cuando CodeX invoca un método *UP_* de forma automática ante un solicitud del usuario, el programador es responsable de implementar el proceso de usuario correspondiente dentro de dicho método o por medio de una serie de métodos encadenados que tienen su raíz en el método *UP_* mencionado.

Si el proceso de usuario invocado requiere establecer comunicación con el usuario para proporcionarle datos u obtenerlos de él, lo cual ocurre en prácticamente la totalidad de los casos, el programador podrá hacer uso de los servicios proporcionados por la clase UIMS que le sirve de interfaz de comunicación con GADEA.

El primer paso aconsejado será la selección del canal de comunicación a través del cual podrá establecer el diálogo con el usuario, creando uno nuevo si el canal activo no es aconsejado y pasando a configurar los parámetros del mismo. El segundo paso consistirá en la creación del diálogo con el usuario.

Para GADEA, un diálogo interactivo consiste en el conjunto de datos que el proceso de usuario debe solicitar del usuario para su ejecución y la información que ha de transmitirse al usuario como resultado de dicho proceso. Como se puede apreciar, técnicamente hablando, un diálogo interactivo estará formado por una serie de primitivas de datos necesarios para ejecutar un proceso de usuario con éxito.

En la versión actual del prototipo GADEA se proporciona soporte para cuatro tipos básicos de datos. En concreto para los tipos entero (clase *Integer*), cadena de caracteres (clase *String*), fecha (clase *Date*) y proceso de usuario (clase *UP*). Todos ellos son componentes *JavaBeans* [Sun Microsystems (1996)] que pueden ser configurados por medio de su conjunto de propiedades públicas. Para comodidad del programador, estas primitivas básicas de datos se encuentran agrupadas en el paquete *gadea.datatypes*, diferenciándolas así de las clases de Java homónimas (*Integer*, *String* y *Date*) incluidas en el paquete estándar *java.lang*.

Tabla 4. Componente: *gadea.datatypes.Integer*: descripción de sus principales propiedades.

PROPIEDAD	DESCRIPCIÓN
value	Vector de contenedores de la clase *Integer*, representando los valores actuales para esta primitiva.
isFixed	Si esta propiedad tiene un valor *verdadero*, el valor de la propiedad *value* no podrá ser modificado y se considerará que el componente sólo tiene funcionalidad descriptiva, por lo que no puede emplearse para proporcionar datos a la aplicación.
Mask	Máscara de entrada para este valor. Los caracteres de la máscara indican el formato con el que DEVA mostrará la información al usuario o esperará que éste la introduzca en el sistema.
IsRequired	Si su valor es *verdadero*, el usuario deberá rellenar obligatoriamente el valor de este componente en el *widget* correspondiente para el diálogo interactivo en el que se encuentre, una vez desplegado por DEVA. Si el usuario cierra el diálogo interactivo sin proporcionar ningún valor para este componente, DEVA se lo notificará al usuario, impidiéndoselo (a menos que el usuario decida cancelar el proceso de recolección de datos).
range	Rango de valores continuos que puede asumir esta primitiva de datos. Este rango está determinado por un valor mínimo y un valor máximo. El programador puede especificar cualquiera de ellos, o ambos. El usuario solo podrá asignar a este componente aquellos valores que se encuentren dentro del rango previsto. Si el valor proporcionado no se encuentra dentro de este rango, DEVA lo notificará al usuario invalidando la introducción de datos.
possibleValues	Lista de valores discretos que puede tener este componente. El usuario sólo está autorizado a introducir valores incluidos en esta lista.
multipleValues	Si su valor es *verdadero*, este componente puede contener más de un valor.

Como se puede apreciar, el paradigma de desarrollo planteado por GADEA elimina totalmente el uso directo de *widgets* por parte del programador, lo cual, como se explicó en el capítulo 5 (*Diseño Centrado en el Usuario*) representa uno de los grandes problemas planteados por la programación visual desde un punto de vista de interacción y comunicación humana.

Dado que desde el código de un programa GADEA no se hace mención alguna a los *widgets*, será responsabilidad de este sistema elegir y situar los mismos dentro del canal de comunicación seleccionado por el programador. Nótese que en un sistema de acceso universal, la dimensión puramente visual de los *widgets* se pierde en beneficio de una naturaleza dual (visual y auditiva).

Como se verá en a partir del capítulo 12(*El Modelo Cognitivo*), será el módulo DEVA quien decida que *widgets* son los más adecuados para su empleo en la interfaz de la aplicación, dependiendo siempre de las características cognitivas, perceptivas y motrices del usuario actual de la aplicación.

Tabla 5. Componente: *gadea.datatypes.String*: descripción de sus principales propiedades.

PROPIEDAD	DESCRIPCIÓN
value	Vector de contenedores de la clase *String*, representando los valores actuales para esta primitiva.
isFixed	Si esta propiedad tiene un valor *verdadero*, el valor de la propiedad *value* no podrá ser modificado y se considerará que el componente sólo tiene funcionalidad descriptiva, por lo que no puede emplearse para proporcionar datos a la aplicación.
Mask	Máscara de entrada para este valor. Los caracteres de la máscara indican el formato con el que DEVA mostrará la información al usuario o esperará que éste la introduzca en el sistema.
isPerceptible	Si esta propiedad tiene un valor *verdadero*, DEVA podrá mostrar el contenido de la propiedad *value* en el *widget* correspondiente y a través del canal de comunicación auditivo. Si tiene un valor *falso*, se entiende que este componente posee un contenido confidencial que no puede ser mostrado al usuario, como una contraseña, por ejemplo..
IsRequired	Si su valor es *verdadero*, el usuario deberá rellenar obligatoriamente el valor de este componente en el *widget* correspondiente para el diálogo interactivo en el que se encuentre, una vez desplegado por DEVA. Si el usuario cierra el diálogo interactivo sin proporcionar ningún valor para este componente, DEVA se lo notificará al usuario, impidiéndoselo (a menos que el usuario decida cancelar el proceso de recolección de datos).
possibleValues	Lista de valores discretos que puede tener este componente. El usuario sólo está autorizado a introducir valores incluidos en esta lista.
multipleValues	Si su valor es *verdadero*, este componente puede contener más de un valor.

Por medio de la información almacenada en el modelo individual del usuario actual y en función de la Teoría Unificada de la Cognición [Newel (1991)], DEVA seleccionará y configurará *widgets* adecuados para el tipo de información que se pretende obtener y/o mostrar al usuario, adaptando también el aspecto perceptible de los mismos, sea este visual o auditivo.

Mientras mayor sea la riqueza de conocimiento del sistema experto, tanto en la precisión y adecuación de sus reglas como en la veracidad de los hechos de partida (el conocimiento que el propio sistema tiene acerca del usuario) mayor será la efectividad en la adaptación de la interfaz.

El poder configurar un diálogo iterativo en función del submodelo de datos del modelo de interacción en lugar de configurarlo en función de *widgets*, como viene siendo costumbre es una de las claves de GADEA , ya que le aporta flexibilidad y facilidad de uso. Flexibilidad porque a partir de una única lista de requisitos de información GADEA es capaz de generar de forma automática múltiples versiones de una interfaz, combinando diferentes *widgets* en diferentes ubicaciones

(visuales y auditivas) y con diferentes configuraciones (también visual y auditiva) sin la limitación de las interfaces generadas de manera estática. Facilidad porque la responsabilidad de seleccionar el *widget* adecuado, así como su correcta configuración no depende del programador o del equipo de interacción y comunicación humana en el mejor de lo casos, sino del propio sistema de gestión de interfaces de usuario. De este modo GADEA libera al equipo de desarrollo de una gran responsabilidad y de una gran carga de trabajo, reduciendo en todo caso los tiempos de desarrollo de forma drástica.

Tabla 6. Componente: *gadea.datatypes.Date*: descripción de sus principales propiedades.

PROPIEDAD	DESCRIPCIÓN
value	Vector de contenedores de la clase *Date*, representando los valores actuales para esta primitiva.
isFixed	Si esta propiedad tiene un valor *verdadero*, el valor de la propiedad *value* no podrá ser modificado y se considerará que el componente sólo tiene funcionalidad descriptiva, por lo que no puede emplearse para proporcionar datos a la aplicación.
Mask	Máscara de entrada para este valor. Los caracteres de la máscara indican el formato con el que DEVA mostrará la información al usuario o esperará que éste la introduzca en el sistema.
IsRequired	Si su valor es *verdadero*, el usuario deberá rellenar obligatoriamente el valor de este componente en el *widget* correspondiente para el diálogo interactivo en el que se encuentre, una vez desplegado por DEVA. Si el usuario cierra el diálogo interactivo sin proporcionar ningún valor para este componente, DEVA se lo notificará al usuario, impidiéndoselo (a menos que el usuario decida cancelar el proceso de recolección de datos).
range	Rango de valores continuos que puede asumir esta primitiva de datos. Este rango está determinado por un valor mínimo y un valor máximo. El programador puede especificar cualquiera de ellos, o ambos. El usuario solo podrá asignar a este componente aquellos valores que se encuentren dentro del rango previsto. Si el valor proporcionado no se encuentra dentro de este rango, DEVA lo notificará al usuario invalidando la introducción de datos.
possibleValues	Lista de valores discretos que puede tener este componente. El usuario sólo está autorizado a introducir valores incluidos en esta lista.
multipleValues	Si su valor es *verdadero*, este componente puede contener más de un valor.

La selección de los tipos de datos a emplear depende directamente del dominio del problema y en todo caso del submodelo de datos del modelo de interacción. En todo caso, este trabajo de selección puede ser acometido por las técnicas de la ingeniería del software tradicional, siendo un trabajo presente en todo producto software. Es la selección de *widgets* en función de estas necesidades de datos y su posterior configuración la carga de trabajo que se reduce al emplear el paradigma de desarrollo propuesto por GADEA.

Nótese que dado que el proceso de selección de primitivas de datos es independiente de la interfaz, se logra una separación casi total entre la funcionalidad de una aplicación (datos y procesos) de su interfaz, alcanzando con ello uno de los objetivos básicos propuestos para GADEA y aunque éste no su objetivo fundamental, es paso obligado para alcanzar el objetivo de la adaptabilidad.

Dependiendo del tipo de datos requeridos por la aplicación para la ejecución del proceso de usuario o por el usuario tras la ejecución de dicho proceso, el programador seleccionara las primitivas de datos necesarias y configurará adecuadamente sus propiedades. Una vez que ha creado todas y cada una de las primitivas de datos necesarias, las incluirá dentro de un objeto de tipo *InteractiveDiscourse* (diálogo o conversación interactiva) el cual servirá de contenedor datos para su transmisión al usuario. Una vez creado el diálogo interactivo, éste puede ser transmitido por cualquiera de los canales de comunicación disponibles.

Tabla 7. Componente: *gadea.datatypes.UP*: descripción de sus principales propiedades.

PROPIEDAD	DESCRIPCIÓN
value	Nombre del método que actúa como punto de entrada al proceso de usuario representado por este método.
precondition	Nombre del método que actúa como *precondición* del proceso de usuario representado por este método.
postcondition	Nombre del método que actúa como *postcondición* del proceso de usuario representado por este método.

En el ejemplo del Código 11 podemos apreciar los pasos necesarios para obtener el nombre y clave de acceso de un usuario en un supuesto proceso de usuario para la verificación de acceso a una aplicación. La clase *Access* dispone de un solo proceso de usuario, denominado *UP_checkAccess* y por lo tanto registrado de forma automática por CodeX cuando el constructor de ésta clase invoca al método *registerEveryUP* de la clase UIMS.

Código 11. Creación de un diálogo interactivo para la obtención del nombre y clave de acceso de un usuario.

```
public class Access
{
   UIMS interface = null;

   // constructor
   Access (UIMS myInterface)
   {
   interface = myInterface;
      interface.registerEveryUP (this);
   }

   // User processes
   public boolean UP_checkAccess ()
   {
      gadea.datatypes.String userLabel, passwordLabel, user,
      password;
      boolean value = false;

      // Data primitives instantiation
      userLabel = new gadea.datatypes.String ("Name: ", true);
      passwordLabel = new gadea.datatypes.String ("Password: ",
         true);
      user = new gadea.datatypes.String ("", false);
      password = new gadea.datatypes.String ("", false);

      // Components configuration
      user.setRequired (true);
      password.setRequired (true);
      password.setPerceptible (false);

      Dialogue dialogue = new Dialogue ();
```

```
      dialogue.add (userLabel);
      dialogue.add (user);
      dialogue.add (passwordLabel);
      dialogue.add (password);

      CommunicationChannel comChannel = new CommunicationChannel
         ("GADEA Registration" , CommunicationChannel.MAX_PRIORITY);

      interface.load (comChannel);
      if (comChannel.display (dialogue))
      {
         // User selected OK
         // Code needed to check access here. Data is contained in
         // the <user> and <password> properties.
         System.out.println ("Name: " + user.getFirstValue ());
         System.out.println ("Password: " +
            password.getFirstValue ());
         value = true;
      }
      else
      {
         // User selected CANCEL
      }

      interface.unload (comChannel);
      return (value);
}
```

Cuando el usuario selecciona el proceso de usuario *checkAccess*, CodeX invoca automática al método *UP_checkAccess*, el cual está siempre activo pues, como se puede apreciar, carece de *precondiciones* (y también de *postcondiciones*). La primera tarea realizada en este método consiste en crear cuatro instancias de la clase *gadea.datatypes.String*, las cuales servirán para indicar al usuario el tipo de información requerida (su nombre y clave de acceso). En el modelo de constructor empleado se especifica el valor de las propiedades *isFixed y value*. La primera tendrá un valor verdadero para las etiquetas que indican el valor solicitado y falso para las cadenas en donde el usuario depositará el valor.

A continuación se configuran las primitivas que lo necesitan, en concreto los datos *user* y *password*, indicando que ambos son de cumplimiento obligatorio y especificando en particular para el dato *password* que éste no debe ser mostrado directamente al usuario (ya que se trata de una clave).

Terminado este paso, se crea el correspondiente diálogo interactivo y se añaden a éste los objetos recién creados. El orden en el se añaden éstos objetos resulta sumamente importante ya que indica la prioridad que tendrán estos en canal de comunicación en el que serán transmitidos. Aquellos objetos con la mayor prioridad consumirán mayor ancho de banda u ocuparán los puestos mas prominentes del canal de comunicación según lo estime oportuno el módulo encargado de su distribución espacial (DEVA). De todos modos, la prioridad asignada a cada objeto puede variarse en cualquier momento gracias a los métodos creados al efecto en la clase *InteractiveDiscourse*.

Una vez creado el diálogo interactivo, éste puede asignarse al canal de comunicación activo. En el ejemplo del, se crea un canal de comunicación auxiliar y se registra en la clase UIMS. A continuación se registra el diálogo creado en el canal de comunicación y se proporcionan instrucciones a GADEA para que muestre el canal de

comunicación y su diálogo interactivo asociado de modo puntual, es decir, dándole la mayor prioridad para que se muestre al usuario de inmediato, paralizando cualquier otra posible actividad.

En otras palabras, con la tecnología disponible, este diálogo interactivo se mostrará en un cuadro de diálogo similar al de la Figura 19, del que sólo se saldrá seleccionado el botón *OK* (siempre y cuando los datos *user* y *password* hayan sido cubiertos) o el botón *CANCEL*. La misma información es transmitida por medios auditivos. Para usuarios con serias discapacidades visuales, la recogida de información se realiza por medio de una pequeña modificación de los estilos de interacción basados en sistemas de barrido. En estos sistemas, las opciones son resaltadas de forma secuencial. Cuando un determinado ítem es resaltado, el usuario puede seleccionarlo mediante la pulsación de una tecla [Abascal et. al. (2000)]. En nuestro caso, el resaltado de las opciones se realiza de forma oral a través del altavoz del ordenador.

Una vez que el usuario ha seleccionado una de las dos opciones de salida, el método *comChannel.display (dialogue)* finaliza, devolviendo un valor verdadero si el usuario ha seleccionado la opción *OK* o falso si ha seleccionado la opción *CANCEL*. En este momento, el programador tiene a su disposición toda la información necesaria para obrar en consecuencia. En caso de obtener una interacción positiva (selección *OK*) en los componentes *user* y *password* tendrá los valores proporcionados por el usuario. En caso negativo (selección *CANCEL*) en dichos componentes tendrá los valores previos a la invocación del método *display* de la clase *CommunicationChannel*. En todo caso, en el ejemplo se elimina el canal de comunicación activo mediante una llamada al método *unload* de la clase *UIMS* (*interface*) con lo que tanto el cuadro de diálogo como la transmisión continua de voz se da por finalizada.

Figura 19. Posible construcción del canal de comunicación y diálogo interactivo asociado del Código 11 (consultar versión en color en el capítulo 24: *Imágenes en Color*, p. 445).

El motivo de empaquetar las primitivas de datos sobre un objeto de tipo *InteractiveDiscourse* en lugar de enviarlos directamente a través de un objeto de tipo *CommunicationChannel* radica en la flexibilidad del primer planteamiento. De este modo, sobre un único canal de comunicación es posible establecer una secuencia de diálogos interactivos limitando la cantidad de información que el usuario recibe de cada vez, favoreciendo así la comunicación, la interacción y el aprendizaje de la interfaz por parte del propio usuario. Este tema se

tratará en profundidad a partir del capítulo 12 (*El Modelo Cognitivo*) en el que se verá como los propios diálogos interactivos serán fragmentados para adaptarse a la propia experiencia del usuario con la aplicación y a la capacidad de su memoria a corto plazo.

De este modo, es posible establecer modelos de interacción en secuencia, del tipo de los empleados por los *wizards*, simplemente realizando un despliegue selectivo de diálogos interactivos. Como se puede apreciar, en el código que nos sirve de ejemplo, todas las decisiones concernientes a los mecanismos de interacción empleados son de altísimo nivel (metáfora, modelo de interacción), dejando las consideraciones de bajo nivel (selección de *widgets*, ubicación, tamaño, color, etc.) al sistema experto.

8.3 Un Camino de Ida y Vuelta

Como se ha comentado al inicio de este capítulo, el tiempo de vida de cualquier canal de comunicación establecido por GADEA puede ser efímero o tener un carácter permanente. En el primero de los casos, asociado generalmente a cuadros de diálogo o a cuadros de alerta, la información proporcionada por el usuario se almacena directamente en las primitivas básicas de datos incorporadas en el diálogo interactivo activo, los cuales podrán ser recuperados por la aplicación y dentro del proceso de usuario en curso al finalizar dicho diálogo; tal y como se ha visto en los ejemplos que ilustran este capítulo. Sin embargo, en el segundo de los casos, es necesario establecer mecanismos para que la información que se está mostrando en un canal de comunicación no dependa directamente de la ejecución de un determinado proceso de usuario y por lo tanto pueda permanecer en dicho canal de comunicación entre dos o más llamadas consecutivas a distintos procesos de usuario.

Es decir, es necesario garantizar que cierta cantidad de información puede transmitirse de forma permanente en determinados canales de comunicación, la cual debe ser actualizada de forma continúa con los datos proporcionados tanto por el usuario como por la propia aplicación.

En estos casos, la información de carácter permanente no debe incorporarse al canal de comunicación respectivo por medio de variables locales pertenecientes a determinados procesos de usuario, sino por variables de mayor duración, como son los propiedades de los objetos, asociadas a la duración del objeto al que pertenecen. Es por ello que el diseño general de GADEA prevé la incorporación directa de los valores de las propiedades de los objetos al canal de comunicación activo, permitiendo el registro en tiempo de ejecución de las mismas por medio de la instancia respectiva de la clase UIMS que sirve de interfaz con CodeX.

Para garantizar el registro de propiedades en los canales de comunicación, GADEA se apoya una vez más en un sistema de notación para los métodos de acceso a las propiedades de un objeto.

Esta notación no es otra que la definida para los *JavaBeans* del lenguaje Java [Sun Microsystems (1996)] así como para su sistema eventos. En pocas palabras, se espera que todo método de acceso a una propiedad se nombre con el prefijo *set*, seguido del nombre de la propiedad.

Esta notación permite a GADEA acceder al valor de una propiedad dentro de un objeto cada vez que el usuario modifique el valor del *widget* asociado a dicha propiedad en la interfaz proporcionada por DEVA. Para ello, se empleará de nuevo el mecanismo de reflexión en el que se fundamenta CodeX. En el ejemplo del Código 12 se muestra de manera simplificada el contenido del método *updateProperty* de la clase UIMS, invocado por DEVA cuando el usuario modifica una propiedad. Como se puede apreciar, CodeX, tras localizar el nombre de la propiedad y el objeto al que pertenece en su registro interno, procede a invocar al método de acceso de la propiedad, el cual se tendrá que nombrar de manera obligatoria con el prefijo *set* y el nombre de la propiedad en cuestión. A dicho método se le pasa como parámetro el valor actualizado de la propiedad.

Código 12. Método *setProperty* para la actualización de una propiedad desde la interfaz de usuario al código de la aplicación.

```
public void setProperty (name String, type String, class String,
object Object, value Object)
{
   try
 {
   // search for a 'set' <name> (<type>) method in <object>.
  Class parameters[] = new Class[1];
  parameters[0] = Class.forName (type);
  Class cl = Class.forName (class);
  Method method = cl.getDeclaredMethod ("set" + name,
        parameters);

     // invokes the 'set' method.
  if (method != null)
  {
  Object finalParameters [] = new Object [1];
  finalParameters[0] = value;
  method.invoke (object, finalParameters);
  }
 }
 catch (Exception e)
 {
  System.err.println ("[UIMS] : unable to update property..." +
        e.toString());
 }
}
```

Naturalmente, para que CodeX pueda actualizar una propiedad es necesario que dicha propiedad haya sido registrada en este módulo, así como haber sido asignada a un canal de comunicación. Para ello, todo canal de comunicación dispone de un método diseñado a tal efecto, denominado *registerProperty*, el cual recibe como parámetro la propiedad, el objeto al que pertenece y su nombre.

En el constructor de la clase *EasyInteger*, empleado como ejemplo en el Código 13, se muestra el registro de una propiedad denominada *value*, la cual se pretende que esté presente en el canal de comunicación por defecto de la aplicación. Dicha primitiva es de tipo *gadea.datatypes.Integer* y al ser modificada por el usuario en la interfaz

imprimirá su valor por la salida estándar. En el ejemplo, el registro es realizado en el propio constructor de la clase, aunque en realidad, dicho registro puede efectuarse en cualquier punto del código de la mencionada clase. Como se puede apreciar, existe un método denominado *setValue* el cual será invocado por CodeX cuando el usuario modifique el valor de la propiedad *value* a través de la interfaz de usuario.

Código 13. Clase *EasyInteger*. Se registra la propiedad *value* en CodeX para que el usuario pueda modificarla..

```
import gadea.datatypes.*;

public class EasyInteger
{
   gadea.datatypes.Integer value = null;

   // constructor
   EasyInteger (UIMS myInterface)
   {
      value = new gadea.datatypes.Integer (0, false);
   (interface.getDefaultCommunicationChannel()).registerProperty
         (value, "value", this);
   }

   public void setValue (myValue Object)
   {
      value = (gadea.datatypes.Integer) myValue;
      System.out.println ("VALUE: " + value.getValue ());
   }
}
```

Así como el usuario debe poseer la capacidad de modificar el estado de una propiedad directamente desde la interfaz, la aplicación también debe poseer la capacidad de que los cambios efectuados por esta sobre una propiedad queden reflejados en la interfaz de usuario. Este mecanismo básico de transmisión de información desde el lado de la aplicación al del usuario se puede efectuar de forma muy sencilla, siguiendo tan solo los preceptos del sistema de eventos de Java establecido a partir de la versión 1.1 de esta plataforma de desarrollo [Weber (1997) p. 333], notificando cualquier cambio en el estado interno de una propiedad mediante la emisión de un evento.

Dicho evento será capturado por todos los objetos subscriptos al mismo (*listeners*) entre los que se encuentra CodeX. De este modo, el mecanismo de transmisión y captura de eventos estándar de Java servirá para que CodeX detecte la modificación del valor de una propiedad y se lo notifique de forma adecuada al usuario a través de DEVA. En el ejemplo incluido en el Código 14 se ilustra este proceso empleando una clase denominada *EasyCounter*, la cual contiene una propiedad de tipo *gadea.datatypes.Integer*, que es actualizada cada segundo en un hilo de ejecución paralelo. Cada vez que se produce una actualización se genera un evento, el cual es capturado por CodeX quien se subscribe a dicho evento en el mismo momento en que la propiedad es registrada mediante el método *registerProperty* incluido en el constructor. De este modo, el contador es actualizado en la interfaz cada segundo. Nótese como el Código 14 referido incluye métodos para el registro y la baja de subscriptores de eventos de tipo *PropertyChange*.

Código 14. Clase *EasyCounter*. Un contador actualizado en el hilo representado por el método *run*, es mostrado en el canal de comunicación por defecto.

```
import java.beans.*;
import gadea.datatypes.*;

public class EasyInteger
{
   gadea.datatypes.Integer counter = null;
   protected PropertyChangeSupport changeAgent = null;

   // constructor
   EasyInteger (UIMS myInterface)
   {
      counter = new gadea.datatypes.Integer (0, true);
   (interface.getDefaultCommunicationChannel()).registerProperty
         (counter, "counter", this);

      // agent for events broadcasting.
      changeAgent = new PropertyChangeSupport (this);

      // creates the counter in its own thread
      (new Thread (this)).start();
   }

   public void addPropertyChangeListener (PropertyChangeListener l)
   {
   changeAgent.addPropertyChangeListener (l);
   }

   public void removePropertyChangeListener (PropertyChangeListener
      l)
   {
   changeAgent.removePropertyChangeListener (l);
   }

   public void run ()
 {

 for (;;)
 {
  try
  {
  Thread.sleep (1000);
  }
  catch (Exception e)
  {;}

  counter.setValue (counter.getValue() + 1);
  changeAgent.firePropertyChange ("counter", counter, counter);
 }
 }

}
```

9

9 La Semántica

Nada perece en el Universo; cuanto acontece en él no pasa de meras transformaciones

Pitágoras

9.1 ACML

Como se podido apreciar en los capítulos precedentes, la definición de los diálogos interactivos planteada en términos de datos en lugar de en base a *widgets* permite alcanzar un elevado grado de separación entre la funcionalidad de una aplicación y su interfaz. Dado que los datos se mantienen en su estado puro (un estado tratable única y exclusivamente por el modelo de datos de la aplicación y separado del modelo de interacción), el componente DEVA es capaz de establecer la vista que mejor se adapte a las necesidades cognitivas, perceptivas y motrices del usuario puntual de la aplicación, extendiéndose aquí el concepto de *vista* a uno más amplio al que podríamos denominar *percepción*, ya que las técnicas de comunicación duales descritas permiten satisfacer también a usuarios con deficiencias visuales parciales o incluso totales.

Aprovechando el nivel de separación entre funcionalidad e interfaz alcanzado con el paradigma de desarrollo de CodeX, se ha pretendido aislar los datos pertenecientes al submodelo de datos del modelo de interacción de cualquier manipulación posterior, almacenándolos en una estructura libre de sintaxis, es decir, totalmente neutral ante diseños concretos de una interfaz basada en *widgets*.

Naturalmente, aunque esta estructura de datos es libre de sintaxis, no lo es de semántica ya que los datos almacenados en ella han de poder relacionarse entre sí de acuerdo con la semántica o metáfora definida por los diseñadores para la aplicación. De este modo es posible aprovechar al máximo las ventajas que aporta el modelo vista controlador [Gamma (1996)] para llevarlo a un modelo percepción controlador.

Dado que la información recopilada por CodeX con respecto al submodelo de datos del modelo de interacción es independiente del tratamiento posterior que le pueda proporcionar DEVA (o cualquier otro sistema experto), la estructura de datos para el almacenamiento de las primitivas de datos se ha diseñado en base a las premisas de sencillez y compatibilidad entre los módulos del sistema de gestión de interfaces de usuario, sean éstos los propuestos en este documento u otros que la tecnología futura permita construir. Por ello, la estructura de datos que representa a un diálogo interactivo puede convertirse fácilmente en código legible y transferible de una aplicación a otra.

Para lograr esta sencillez y compatibilidad se decidió crear un lenguaje de intercambio de datos propio, basado en una extensión del lenguaje XML (*Xtensible Mark-Up Language*) [Johnson (1999)] y por lo tanto fácil de editar y verificar con aplicaciones diseñadas para el tratamiento de este lenguaje como el XML Writer [Howard y Cronje (2001)] o el XML Spy [Altova (2001)]. Este lenguaje recibe el nombre de ACML y atiende a las siglas de *Adaptive Contents Mark-Up Language*, es decir, un *Lenguaje de Marcas para Contenidos Adaptables*.

La sintaxis de este lenguaje es muy sencilla, representando una transcripción directa del contenido de las primitivas de datos básicas empleadas por CodeX, así como del diálogo interactivo en el que están inmersos. En el Código 15 se muestra un ejemplo en el que se representa el diálogo interactivo incluido en el Código 11 escrito en formato ACML. Como se puede apreciar, el lenguaje es bastante fácil de generar a partir de un objeto de tipo *InteractiveDiscourse* e incluso, dado el caso, puede ser codificado manualmente por un operario humano.

Código 15. Dialogo interactivo incluido en el Código 11 convertido en un documento ACML versión 1.

```
<? ACML version = 1.0' ?>
<DIALOGUE NAME=''>
   <STRING ISFIXED='TRUE' ISPERCEPTIBLE='TRUE' ISREQUIRED='FALSE'
      MULTIPLEVALUES='FALSE'>
      <VALUES>
         <VALUE ID='Name: '/>
      </VALUES>
   </STRING>
   <STRING ISFIXED='FALSE' ISPERCEPTIBLE='TRUE' ISREQUIRED='TRUE'
      MULTIPLEVALUES='FALSE'>
      VALUES/>
   </STRING>
```

```
   <STRING ISFIXED='TRUE' ISPERCEPTIBLE='TRUE' ISREQUIRED='FALSE'
      MULTIPLEVALUES='FALSE'>
      <VALUES>
         <VALUE ID ='Password: '/>
      </VALUES>
   </STRING>
   <STRING ISFIXED='FALSE' ISPERCEPTIBLE='FALSE' ISREQUIRED='TRUE'
      MULTIPLEVALUES='FALSE'>
      <VALUES/>
   </STRING>
</DIALOGUE>
```

La gran ventaja del empleo de un lenguaje del tipo del ACML estriba en que disminuye el grado de acoplamiento entre los módulos implicados en el procesamiento del submodelo de datos del modelo de interacción de una aplicación, los cuales de otro modo estarían ligados unos a otros por fuertes relaciones de dependencia. De este modo, ACML sirve como lenguaje de comunicación común y puente entre los distintos módulos de GADEA implicados en la transferencia y gestión de primitivas de datos (en nuestro caso CodeX y DEVA).

El empleo estándar de este lenguaje facilita el uso y aprovechamiento de componentes fabricados por terceras partes dentro de la misma arquitectura básica del gestor de interfaces de usuario, sirviendo incluso como interfaz común para diferentes versiones de los mismos módulos propuestos en este documento, tal y como se puede apreciar en la Figura 20. En este sentido, el lenguaje ACML actúa como un lenguaje intermedio, con las ventajas que ello aporta a la flexibilidad del sistema [Cueva (1994) p.34].

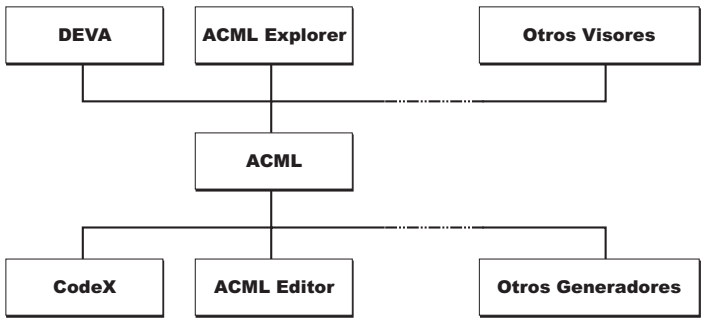


Figura 20. Empleo del lenguaje ACML como nexo común entre los distintos módulos que forman un gestor de interfaces de usuario auto-adaptables, tanto del presente como del futuro.

Sin embargo, la característica de ACML que mayor ventaja aporta es la sencillez de su sintaxis, ya que como se puede ver, se limita a definir un objeto XML por cada primitiva de datos, en cuya declaración de atributos se especificarán los valores proporcionados a sus propiedades. La sencillez del diseño de ACML permite su uso directo por parte de operarios humanos o cuando menos, bajo la ayuda de herramientas diseñadas para una fácil edición de este tipo de documentos. Dado que el lenguaje contiene un submodelo de datos en estado puro, es decir, libre de las ataduras de una interfaz concreta, es perfectamente factible el hecho de que un operario humano pueda diseñar diálogos interactivos destinados a otros operarios humanos, los cuales serán adaptados por DEVA a las características cognitivas, perceptivas y

motrices del usuario destino. Volvemos entonces al marco conceptual basado en la teoría general de sistemas en la que se fundamenta todo nuestro desarrollo teórico y que fue planteado en el capítulo 4 (*La Comunicación*).

Hasta el momento, en el marco conceptual general en el que se desenvuelve nuestro sistema de comunicación, GADEA actuaba única y exclusivamente en la comunicación entre un usuario humano y una máquina y viceversa, adaptando la interfaz de comunicación de uno y otro emisor a su correspondiente receptor. Sin embargo, gracias al concepto del lenguaje de adaptación de contenidos es posible emplear GADEA para la comunicación entre dos humanos, aislando el mensaje a transmitir en el punto emisor de cualquier posible encarnación física, adaptándolo a las necesidades del receptor una vez que ha llegado a éste, eliminando en parte el ruido que pueda existir en el canal de comunicación.

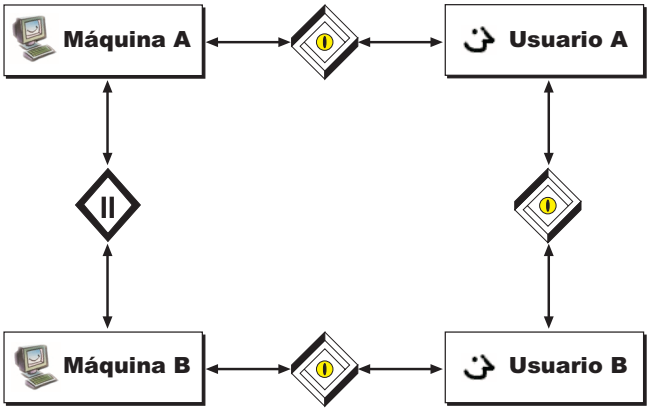


Figura 21. Con ACML es posible adaptar la comunicación directa entre dos componentes humanos.

Una aplicación práctica directa del lenguaje ACML puede ser la de su uso para generar contenidos libres de sintaxis para Internet, distribuyéndolos en servidores web compatibles con el lenguaje ACML. De este modo, cualquier usuario que disponga de un explorador de Internet basado en tecnología GADEA podrá acceder a una versión personalizada a bajo nivel del documento depositado en el servidor, teniendo en cuenta la propia individualidad del receptor.

Si XML pretende facilitar el diseño de contenidos independientes de una vista u aplicación concreta, el objetivo de ACML se concreta en facilitar el diseño de contenidos independientes de un receptor concreto. Así, en el ejemplo anterior, un documento ACML podrá ser consultado con el mismo beneficio tanto por usuarios en perfecto estado físico, como por aquellos que posean discapacidades físicas parciales o totales.

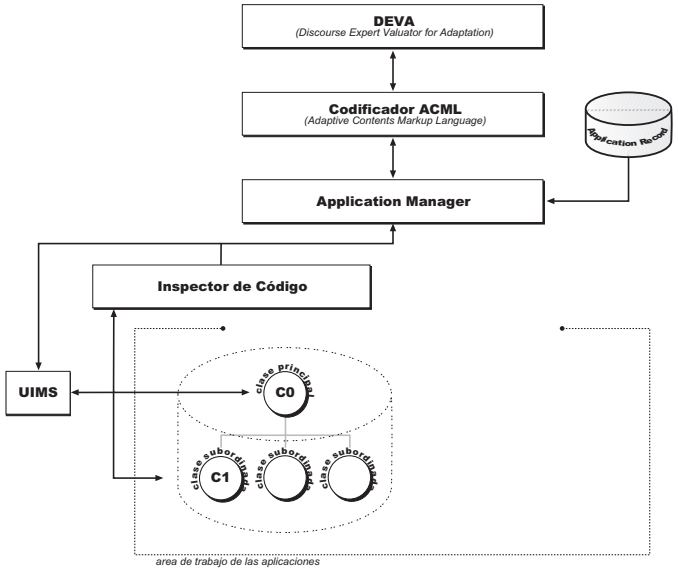


Figura 22. El decodificador de lenguaje ACML actúa de puente entre DEVA y el *Application Manager* de DEVA sirviendo como formato común para el intercambio de información.

9.2 Los Chunks

El paradigma de desarrollo planteado por CodeX implica que, dependiendo del tipo de información requerida tanto por el usuario como por la propia aplicación, el programador seleccionará y configurará los componentes apropiados, registrándolos en un diálogo interactivo, siendo la prioridad asignada a los mismos, la encargada de indicar a DEVA el orden a adecuado en el que los *wigdets* seleccionados serán mostrados, resolviendo así el problema de ambigüedad en la ordenación espacial de los *widgets* planteado en la sección 6.1.2 (*Los Datos de Usuario*) del capítulo 6.

Sin embargo, una especificación de prioridades puede no ser suficiente en casos en el que es necesario agrupar una serie de primitivas básicas de acuerdo con la carga semántica que puedan tener para el usuario. Por ejemplo, en el ejemplo planteado en la sección 6.1.2 (*Los Datos de Usuario*) del capítulo 6, la información correspondiente a la información básica del individuo (DNI, Nombre y Apellidos) podría ser separada de la información correspondiente a su domicilio (Calle, Código Postal, Ciudad y País) para formar dos bloques cognitivos distintos, facilitando así la asimilación de la información por parte del usuario.

Como se verá en detalle cuando se analice el módulo DEVA, esta agrupación de elementos en bloques tiene mucho que ver con la organización interna de los objetos percibidos en la memoria a corto y a largo plazo de los seres humanos. Para adelantar terreno, podemos decir que la información percibida por un sujeto cualquiera, rara vez se concibe en términos de una secuencia de elementos, sino que éstos son clasificados y agrupados en bloques o trozos con un significado determinado, los cuales son denominados *chunks* en la literatura

[Newell (1991)]. De ahí que en este documento se conserve la nomenclatura original.

Así por ejemplo, la información contenida en este documento no es percibida como una secuencia de letras, como sería lo lógico, sino que un lector bien entrenado percibe secuencias de palabras que son agrupadas en sentencias y oraciones, las cuales son almacenadas en la memoria a corto plazo (MCP) para su procesamiento inmediato. Si la atención del lector considera importante el contenido de estas frases y oraciones, dichos bloques de información serán estructurados y reagrupados en la memoria a largo plazo por medio de bloques de tamaño mayor pero más fáciles de almacenar y de recobrar.

Siguiendo este ejemplo, las unidades básicas de información de este documento (las letras) son agrupadas en *chunks* formados por las palabras, las cuales a su vez formaran parte de *chunks* mayores a los que denominamos oraciones. Estas oraciones podrán ser reagrupadas en *chunks* de tamaño mayor denominados párrafos y así sucesivamente hasta llegar al documento.

Este sistema de clasificación permite realizar una referencia directa a cualquier *chunk* empleando para ello muy poca información, lo cual es una ventaja muy importante de cara a la recuperación del material almacenado tanto en la memoria a corto plazo como en la memoria a largo plazo. Nótese lo sencillo que es referir una página o capítulo de este documento, práctica habitual en toda obra, con solo mencionar un capítulo, número de página o párafo. En la Figura 23 podemos ver una representación esquemática de esta organización en la que se resalta el carácter jerárquico de la misma.

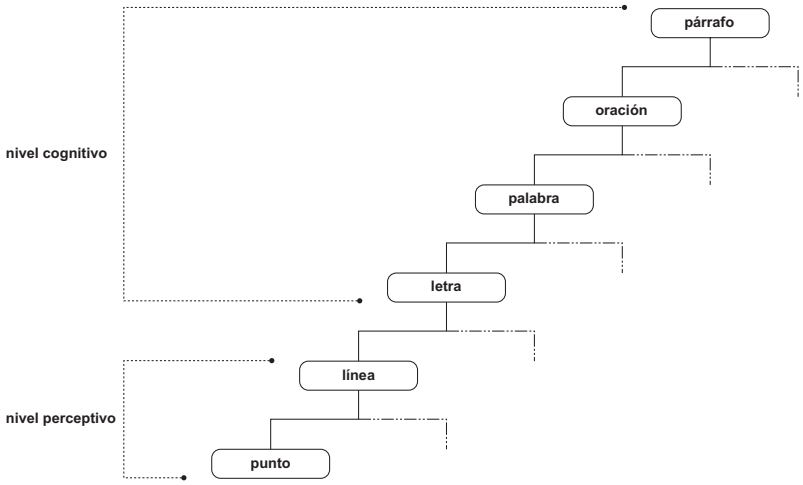


Figura 23. Esquema de memorización y percepción de un libro basado en *chunks*.

Este sistema de clasificación jerárquica basado en *chunks* es común a todo ser humano, de ahí el interés en su empleo en un sistema de gestión de interfaces de usuario. Por ello, GADEA incorpora este concepto a través de su clase *Chunk*, la cual actúa como contenedor, tanto de primitivas de datos como de otros objetos de la propia clase *Chunk*. Metafóricamente hablando, podemos equiparar un chunk en

GADEA con un panel en la jerarquía de clases habitual de todo paquete basado en el modelo vista controlador [Gamma (1996)], con la notable diferencia de que un *chunk* de GADEA trabaja con primitivas de datos, mientras que un panel trabaja directamente con *widgets*. Es decir, un *chunk* trabaja a un nivel cognitivo superior (nivel semántico) del empleado por un panel (nivel sintáctico).

En la Teoría Unificada de la Cognición (TUC) [Newell (1991)], todo *chunk* tiene una capacidad limitada en la memoria a corto plazo, la cual, aunque dependiendo ligeramente del tipo de información percibida, ronda en torno a los 2,5 bits (unos siete elementos). Por lo tanto, el tamaño de todo *chunk* debería situarse entre los 7 (±2) elementos. Sin embargo, cuando el *chunk* se emplea en procesos de memoria a largo plazo (MLP), en los que las relaciones entre elementos son más ricas, pero a la vez más complicadas, no hay límite para el número de elementos que posee un *chunk*.

Por las razones comentadas y poniéndonos en el mejor de los casos (memoria a largo plazo), la clase *Chunk* empleada por CodeX no impone ninguna limitación al número de elementos que pueda contener.

Sin embargo, cuando dichos *chunks* va a ser mostrados al usuario en un proceso de diálogo interactivo bajo la responsabilidad de DEVA, la cantidad de información real a transmitir estará condicionada por el tipo de memoria que el usuario debe emplear para percibir la información. Si la memoria a emplear es del tipo de memoria a corto plazo, DEVA limitará la cantidad de información a transmitir a 2,5 bits. Si por el contrario la memoria empleada es memoria a largo plazo, se mostrará toda la información contenida en el *chunk*. Como se analizará cuando se trate el diseño de DEVA, la primera situación ocurrirá cuando el usuario es novato en el empleo de un determinado proceso de usuario o cuando la información a transmitir es mostrada durante un período de tiempo muy corto (el despliegue de un menú, por ejemplo). Por el contrario, la segunda situación se dará cuando el proceso de usuario en el que se está empleando el *chunk* es bien conocido por el usuario y por lo tanto se encuentra asimilado en su memoria a corto plazo.

Técnicamente, la clase *Chunk* se comporta como cualquier otra primitiva de datos y por lo tanto se incluye también dentro del paquete *gadea.datatypes*. Si el diseñador del modelo de interacción de la aplicación desea agrupar semánticamente una serie de primitivas de datos dentro de un *chunk*, creará una instancia de la clase *Chunk* y le añadirá los componentes que forman parte de este bloque semántico como si se tratase de un diálogo interactivo. Naturalmente, el programador podrá asignar valores de prioridad a todos los componentes del *chunk*, incluso a otros *chunks* que formen parte del *chunk* contenedor. Como ya se comentó en su momento, DEVA empleará estos valores de prioridad para asignar el ancho de banda disponible en el canal de comunicación en el que se transmita el diálogo interactivo del que el *chunk* forme parte. A un mismo nivel de prioridad, los *chunks* se comportarán como cualquier otro componente a la hora del reparto del ancho de banda. A su vez, la fracción del ancho

de banda asignado a un *chunk* será repartida entre los componentes que contiene en función de la prioridad asignada a cada uno de ellos. Si entre esos componentes existe algún *chunk*, la estrategia de reparto se aplicará de nuevo de forma recurrente.

Para mostrar el uso previsto de los *chunks* será preciso volver al ejemplo planteado en el proceso de usuario del Código 11 en el que se pretendía obtener información acerca del nombre de un usuario y su clave en un determinado sistema informático. Dependiendo del carácter con el que los diseñadores han definido el modelo de interacción de la aplicación, cabe la posibilidad de emplear el nivel de agrupación proporcionado por la prioridad de las cuatro primitivas de datos involucradas en el proceso (Código 11), o identificar un bloque semántico para la información concerniente al nombre del usuario (con su correspondiente etiqueta) y otro para la información relativa a su clave de acceso (también con su correspondiente etiqueta). Si se opta por implementar la segunda opción, se llegará a una definición similar a la incluida en el Código 16.

Código 16. Empleo de dos chunks para la obtención del nombre (*getUser*) y clave de acceso (*getPassword*) de un usuario.

```
public class Access
{
   UIMS interface = null;

   // constructor
   Access (UIMS myInterface)
   {
   interface = myInterface;
      interface.registerEveryUP (this);
   }

   // User processes
   public boolean UP_checkAccess ()
   {
      gadea.datatypes.String userLabel, passwordLabel,
      user, password;
      boolean value = false;

      // Data primitives
      userLabel = new gadea.datatypes.String ("Name: ", true);
      passwordLabel = new gadea.datatypes.String ("Password: ",
         true);
      user = new gadea.datatypes.String ("", false);
      password = new gadea.datatypes.String ("", false);

      // Components configuration
      user.setRequired (true);
      password.setRequired (true);
      password.setPerceptible (false);

      Dialogue dialogue = new Dialogue ();
      Chunk getUser = new Chunk ("");
      getUser.add (userLabel);
      getUser.add (user);
      dialogue.add(getUser);

      Chunk getPassword = new Chunk ("");
      getPassword.add (passwordLabel);
      getPassword.add (password);
      dialogue.add(getPassword);

      CommunicationChannel comChannel = new CommunicationChannel
         ("GADEA Registration", CommunicationChannel.MAX_PRIORITY);

      interface.load (comChannel);
      if (comChannel.display (dialogue))
      {
```

```
         // User selected OK
         // Code needed to check access here. Data is contained in
         // the <user> and <password> properties.
         System.out.println ("Name: " + user.getFirstValue ());
         System.out.println ("Password: " +
            password.getFirstValue ());
         value = true;
      }
      else
      {
         // User selected CANCEL
      }

      interface.unload (comChannel);
      return (value);
}
```

Como se puede apreciar, la única diferencia entre el Código 11 y el Código 11 estriba en que en la segunda versión las primitivas de datos se añaden directamente a los *chunks* creados, los cuales se insertan a su vez en el diálogo interactivo. La prioridad 1 se asigna al *chunk getName* mientras que la prioridad 2 recae en el *chunk getPassword*. Podremos concluir que las primitivas *nameLabel* y *name* recibirán las prioridades 1.1 y 1.2 respectivamente, mientras que las primitivas *passwordLabel* y *password* obtendrán las prioridades 2.1 y 2.2.

Figura 24. Diálogo interactivo del Código 11. Nótese la numeración y color de fondo distinto para cada *chunk* (consultar versión en color en el capítulo 24: *Imágenes en Color*, p. 445).

A la hora de mostrar el diálogo creado en el Código 11 al usuario, DEVA empleará criterios de diferenciación para cada *chunk*, como se verá al tratar éste componente. Basta mencionar por el momento que el color de fondo de cada *chunk* será distinto en el canal de comunicación oral y que los *chunks* recibirán numeraciones diferentes para una acceso individual a la hora de emplear técnicas de selección por barrido para usuarios con discapacidades físicas y visuales. Una posible versión visual del ejemplo incluido en el Código 11 se puede contemplar en la Figura 24 (nótese que ésta construcción variará de un usuario a otro dependiendo de sus características cognitivas, perceptivas y motrices).

Como es lógico, el lenguaje ACML introducido en este capítulo también proporciona soporte al concepto de *chunk*. En el Código 17 se incluye la versión ACML del diálogo interactivo incluido en el Código 11. En este ejemplo se puede apreciar como la naturaleza jerárquica de ACML, heredada precisamente del lenguaje XML empleado para su definición, se adapta perfectamente al empleo de *chunks*. Nótese como la definición de un *chunk* comprende su nombre (ninguno en el

ejemplo) y la lista de primitivas de datos básicas que forman parte del *chunk*. El orden en que vienen definidas estas primitivas constituye el valor de la prioridad de las mismas.

Código 17. Versión ACML del diálogo interactivo incluido en el Código 11.

```
<? ACML version = 1.0' ?>
<DIALOGUE NAME=''>
   <CHUNK NAME=''>
      <STRING ISFIXED='TRUE' ISPERCEPTIBLE='TRUE' ISREQUIRED='FALSE'
         MULTIPLEVALUES='FALSE'>
         <VALUES>
            <VALUE ID ='Name: '/>
         </VALUES>
      </STRING>
      <STRING ISFIXED='FALSE' ISPERCEPTIBLE='TRUE' ISREQUIRED='TRUE'
         MULTIPLEVALUES='FALSE'>
         <VALUES/>
      </STRING>
   </CHUNK>
   <CHUNK NAME=''>
      <STRING ISFIXED='TRUE' ISPERCEPTIBLE='TRUE' ISREQUIRED='FALSE'
         MULTIPLEVALUES='FALSE'>
         <VALUES>
            <VALUE ID ='Password: '/>
         </VALUES>
      </STRING>
      <STRING ISFIXED='FALSE' ISPERCEPTIBLE='FALSE'
         ISREQUIRED='TRUE'
         MULTIPLEVALUES='FALSE'>
         <VALUES/>
      </STRING>
   </CHUNK>
</DIALOGUE>
```

10

10 El Lenguaje

La palabra es mitad de quien la pronuncia, mitad de quien la escucha

Michel E. de Montaigne

10.1 Características Cognitivas del Lenguaje

Una premisa básica y fundamental para que el concepto de *comunicación* planteado en el capítulo 4 (*La Comunicación*) exista, es la utilización del mismo lenguaje por parte del emisor y del receptor, de modo que ambos manipulen el mismo concepto a través de las señales que utilizan.

La comunicación alcanzará el éxito esperado en la medida en que el receptor termine reaccionando de la forma pretendida por el emisor. En este sentido, existirá un grave problema de comunicación cuando se presenta un falso emparejamiento entre los esquemas empleados por el emisor y el receptor. En otras palabras, puede ocurrir que el emisor y el receptor se estén refiriendo a conceptos distintos, o que el receptor se vea obligado a ignorar la información que recibe al no poseer los mecanismos adecuados para su decodificación, todo lo cual conducirá a un fracaso en la comunicación. En el caso particular que nos ocupa es

de especial importancia que ambos extremos del canal de comunicación (la interfaz y el usuario) dominen el mismo lenguaje.

El lenguaje es uno de las herramientas más importantes entre las empleadas por la cognición humana y representa una de las características clave en la diferenciación de los seres humanos de los animales. Como ocurre con otros aspectos cognitivos, la comprensión de un determinado lenguaje se ve influenciado en gran medida por la experiencia previa de quien lo utiliza, dado que los humanos decodificamos los mensajes recibidos en función de esquemas mentales previos [Álvarez (1994)].

Las experiencias de Loftus (1979) con testigos simulados de accidentes de tráfico vienen a ilustrar como el lenguaje influye notablemente en el modelo cognitivo, en especial en la percepción. A los participantes de estas experiencias se les sometió a la visión de las imágenes de un accidente, del que más adelante serían interrogados. Tras recoger varios testimonios de un mismo accidente se descubrió cómo en función de las palabras empleadas para interrogar a los observadores existía una variación significativa en cuanto a lo que los participantes decían haber percibido acerca de los detalles siniestro. Así, cuando en las preguntas se empleaban verbos *duros*, del tipo de *estrellar*, *reventar*, etc. (en lugar de *colisionar* o *tomar contacto),* los testigos eran más propensos a exagerar sobre las consecuencias del accidente.

Otro ejemplo de cómo el lenguaje influye en la percepción lo tenemos en las experiencias de Gottschaldt (1938). En los experimentos realizados por este investigador se presentaba a los observadores una figura sencilla para luego pedirles que la localizaran dentro de una imagen más compleja de la cual forma parte. Si bien el hecho de dejar que los participantes en el experimento se familiarizaran con la figura no parece facilitar la búsqueda, dejar que los observadores se entrenaran con la figura, dibujándola un determinado de veces, si parece producir mejoras en la localización posterior.

Sin embargo, cuando se introduce el lenguaje en la búsqueda, haciendo que los observadores asocien una palabra para cada una de las figuras simples, se produce automáticamente una notable mejora en el proceso. Si se divide a los observadores en grupos que empleen distintos niveles de lenguaje, los resultados son aún más sorprendentes. En el experimento original se dividió a los observadores en tres grupos: en el primero se asociaba una palabra a cada figura, en el segundo se asociaba una palabra para cada cuatro figuras, mientras que en el último no se realizaba ninguna asociación. El resultado fue que mientras los primeros mejoraron su capacidad de reconocer las figuras, en los segundos se vio empeorada incluso por debajo de la de los que no realizaban asociaciones lingüísticas. Esto se debe a que una asociación amplia tiende a enfatizar las similitudes existentes entre los objetos haciendo que empeore la capacidad existente para distinguirlos individualmente.

Como se puede apreciar, el lenguaje es uno de los parámetros más importantes a tener en cuenta a la hora de confeccionar el protocolo de comunicación de un sistema, ya que condiciona la actitud del sistema

perceptivo de los humanos que participan en él. En determinados casos puede ocurrir que el usuario destinatario de un mensaje no disponga de experiencias previas que le puedan dotar de los esquemas mentales necesarios para decodificarlo. Nos encontramos entonces ante una situación crítica para todo sistema de comunicación, la cual hay que evitar a toda costa, ya que puede conducir a un absurdo absoluto y provocar como consecuencia la inutilidad del sistema. Así por ejemplo, poco sentido tendrá una aplicación para la resolución de ecuaciones diferenciales destinada a un usuario desconocedor de los más elementales principios del cálculo.

Para facilitar la comunicación será pues necesario emplear principios establecidos al formular los mensajes; principios que deben conocer tanto los emisores como los receptores y que deben formar parte del entorno cultural en el que se ambos mueven. La existencia de un marco cultural de referencia común es requisito imprescindible para todo proceso de comunicación, ya que en caso contrario establecer dicho marco sería del todo imposible. Si cada vez que un concepto fuese mencionado a lo largo de una comunicación cualquiera se tuviese que proporcionar una definición explícita y completa de dicho concepto, la comunicación sería inviable, al menos en un lapso de tiempo finito [Cadrecha et. al. (1999)]. Esta es la base de cualquier proceso de aprendizaje.

10.1.1. Protocolos de Comunicación

Como se puede apreciar, la comunicación entre humanos está limitada por las capacidades cognitivas de éstos y por los esquemas del receptor; límite en el que se deberá restringir todo el sistema de comunicación, GADEA incluido. Es por ello que antes de poder planificar en el diseño del lenguaje de comunicación para cualquier sistema de información, se tendrá que establecer claramente la clase de usuarios que emplearán dicho sistema, adaptando el lenguaje empleado a la capacidad cognitiva y experiencias personales de dichos usuarios. A partir de este momento, la comunicación deberá establecerse hacia las clases de usuarios definidas, tal y como se planteó en el capítulo 4 (*La Comunicación*).

Dado que los humanos representamos y almacenamos la información por referencia a esquemas mentales previos, una manera muy conveniente de establecer una comunicación eficaz será la representación y adecuación del lenguaje a emplear, así como de la información a transmitir por medio de múltiples modelos, estando cada uno de ellos diseñado en función de una clase de usuario concreta. De este modo se puede resolver el problema de comunicación en los casos en los que el sistema disponga de múltiples clases de usuarios.

Supongamos por ejemplo el caso del quiosco multimedia que actúa como punto de información de un museo. En este caso, el quiosco se enfrenta con un extenso abanico de usuarios dotados todos ellos de modelos mentales completamente distintos entre sí, los cuales irán desde el experto interesado en información muy concreta y detallada,

hasta el escolar que hace una visita colectiva al museo con su escuela y que poco o nada sabe de lo que allí se exhibe.

Esta claro que para que el sistema alcance su máxima eficacia, todos sus usuarios deben obtener de él las respuestas que plantean. Para ello, el sistema debe disponer de un tipo de lenguaje para cada una de las clases de usuarios, es decir, para cada entorno cultural. Por ejemplo, los canales de comunicación empleados para establecer el contacto con el experto del ejemplo anterior deberán mostrar información detallada, empleando para ello un lenguaje muy técnico y preciso, mientras que la comunicación establecida con el escolar deberá realizarse por medio de información general y mediante el empleo del lenguaje propio del público infantil.

La planificación de la comunicación no debe limitarse al lenguaje a emplear, sino que debe incluir también el tipo y uso de los canales de comunicación empleados para establecer una comunicación efectiva con dicho lenguaje. Se debe tomar en cuenta que la velocidad y cantidad de información que un receptor puede asimilar es también limitada. Para establecer un sistema de comunicación efectivo que reduzca o elimine los problemas descritos, la información deberá transmitirse empleando el mayor número de canales de comunicación que la tecnología del momento proporcione. Si la información es presentada mediante distintas formas alternativas, de modo que el usuario perciba múltiples representaciones de la misma información, la comprensión de la información será más efectiva.

Como recomendaciones generales para el diseño de la información y del lenguaje a emplear por una aplicación, en los mensajes en los que se emplee el lenguaje deberían utilizarse frases con un tamaño máximo de veinticinco a treinta palabras. Dado que una oración no necesita contener más de un pensamiento, los párrafos han de ser cortos. También es oportuno estructurar las oraciones de modo que se evite el empleo de la voz pasiva ya que este mecanismo representa toda una inversión en el modelo mental y por lo tanto es causa de una interrupción momentánea en la comunicación.

En todo momento, se deben evitar los clichés ya que tienden a clasificar al espectador de acuerdo con determinados estereotipos, limitando su propia individualidad, lo cual nadie desea, puesto que el objetivo es tratar a cada persona como a un ente individual distinto del conjunto. En este apartado se debería tener especial cuidado con los chistes y las ironías. El hecho de que un sector del público –al que van dirigidos– los entienda o les parezcan divertidos, no significa necesariamente que ocurra lo mismo con toda audiencia. Si el sistema de comunicación pretende ser internacional, el efecto producido es peor aún, ya que este tipo de estructuras del lenguaje son difícilmente traducibles. Lo mismo ocurre con las expresiones populares.

También resulta de especial relevancia vigilar las palabras empleadas en el lenguaje y cuidar su significado o se corre el peligro de caer en los bochornosos errores del tipo de afirmar que algo es *casi* único, lo cual es equivalente, desde un punto estrictamente semántico, a afirmar que una mujer está *medio* embarazada o que una persona es *casi*

virgen. En todos estos casos, la comunicación se rompe al emplear términos ambiguos y construcciones lingüísticas antinaturales.

10.1.2. El Papel de la Semiótica

La extensión y potencia de un lenguaje no se limita a su expresión oral o escrita sino que comprende además todos aquellos elementos gráficos que, sin llegar a la complejidad y estructuración de un sistema de escritura, son capaces de transmitir información de algún modo. En un canal de comunicación visual la materia prima con la que se trabaja a la hora de transmitir ideas, es precisamente la compuesta por dichos elementos gráficos o signos. De hecho, todos los elementos constituyentes de dicho canal, desde los iconos hasta las letras empleadas para crear palabras, son signos dotados de un significado muy concreto y por lo tanto jugarán parte muy importante del lenguaje empleado en el canal de comunicación. Pierce (1992) define irónicamente un signo *como algo que representa algo para alguien*.

Los elementos gráficos presentes en un canal de comunicación visual suelen ser de muy variada tipología, aunque autores como Liungman (1992) los clasifican de acuerdo con la carga semántica de los mismos en iconos e ideogramas. Mientras que los iconos (del griego *Eikon* o *imagen*) representan de manera aproximada la imagen de un concepto presente en la realidad del entorno cultural que ha creado el icono, los ideogramas son símbolos que representan una idea o concepto, cuyo origen probablemente se remonte a la degeneración progresiva de un icono [Elíade (1983)]. Un ejemplo clásico de lenguaje basado en ideogramas lo tenemos en el sistema de escritura del antiguo Egipto en donde cada símbolo representa una idea en lugar de un fonema.

Para Greene (1986) los ideogramas se pueden clasificar a su vez en símbolos y señales, dependiendo del significado de los mismos. Para este autor, los símbolos pueden sugerir un significado, mientras que las señales transmiten solo uno. Esta será la nomenclatura que se emplee de ahora en adelante en este documento.

El significado de un signo se fundamenta en las ideas compartidas por la mayoría de la gente a la que el símbolo le es familiar. Al igual que ocurre con el resto de los elementos de un lenguaje, los signos han de tener un significado concreto que por lo general sólo tiene sentido dentro del entorno cultural que los ha engendrado y rara vez tienen algún significado para personas ajenas al mismo. En algunas ocasiones, los símbolos se crean ex profeso para un entorno cultural concreto, de tal modo que no puedan ser descifrados por otros colectivos. Este es el caso por ejemplo de los códigos militares o la jerga iconográfica empleada por el hampa internacional. Como ejemplo anecdótico, en la Tabla 8 se reproducen los símbolos empleados por un grupo de delincuentes de la Comunidad de Madrid para transmitir información confidencial a otros colegas del gremio acerca del valor de los posibles objetivos de sus fechorías.

Es necesario destacar que el significado de un símbolo puede variar notablemente aún dentro del entorno cultural dentro del que ha sido

creado, existiendo al menos cuatro factores que pueden afectar notablemente a su significado [Liugman (1972)].

Tabla 8. Signos pintados por delincuentes con tiza, lápiz o marcados con algún objeto punzante en timbres de entrada, suelo e incluso debajo del felpudo de entrada a domicilios de Madrid. [Fuente: Comisaría de policía del distrito de Salamanca, Madrid, verano de 2000].

No.	SIGNIFICADO	SIGNO
01	Casa deshabitada	
02	Casa ya robada	
03	Mujer sola	
04	Inútil insistir	
05	De vacaciones	
06	Robar, inválido	
07	Rápido, vuelven pronto	
08	Usar palanca	
09	Dispuesta para robo	
10	No robar subnormal	
11	Abren con cadena	
12	Aquí, nada	
13	Cuidado, policía	
14	Nada de interés	
15	Casa caritativa	
16	Muy buena	
17	Cuidado, hay perro	
18	Aquí se puede robar	
19	Sólo viven mujeres	

20	Buena acogida, si se habla de Dios	

En primer lugar se encuentra la propia estructura del símbolo, es decir, el conjunto de primitivas con las que está construido (líneas rectas, segmentos circulares, espirales o puntos), ya que dicha estructura puede ser analizada a la luz de connotaciones genéticas intrínsecas en el receptor (discapacidades físicas, por ejemplo) o de connotaciones espaciales propias los usuarios del símbolo. En este último espacio cabe destacar la orientación que asumen muchos símbolos, cuya estructura está diseñada en función de aspectos tales como la gravedad. De este modo, representaciones iconográficas como la yegua invertida de la **Figura 25** (izquierda), pierde por completo su significado al adoptar una orientación errónea.

Figura 25. Yegua invertida difícilmente reconocible como tal al haber perdido su significado iconográfico debido a una orientación contraria a la habitual.

En segundo lugar hay que situar el entorno espacial en el que se ubica un símbolo, en especial si existen otros signos en la vecindad. Por ejemplo, una simple cruz puede adoptar múltiples significados al situarla en entornos diferentes. La cruz de la Figura 26 puede convertirse en el operador suma (Figura 27) si se la rodea de dos números; o, si se la sitúa sobre una batería, dicha cruz pasará a representar el concepto de polaridad (Figura 28).

Figura 26. Cruz. Símbolo presente en muchas culturas, producto de la intersección de un línea horizontal y de otra vertical.

En tercer lugar, el significado de un signo puede variar en función del fondo histórico y social del propio signo. De este modo, la cruz del ejemplo anterior representa la religión cristiana en un entorno cultural muy amplio y concreto y su significado variará enormemente en función de que el observador pertenezca al mismo o no.

El cuarto factor que afecta al significado de un símbolo será la propia experiencia personal previa o fondo cultural del observador. Siguiendo con el ejemplo de la cruz, ésta no tendrá el mismo significado para un observador ateo que para uno practicante; aunque ambos se encuentren inmersos en el mismo entorno cultural occidental.

2+2

Figura 27. La cruz de la Figura 26 rodeada de dos dígitos se convierte inmediatamente en el operador suma.

Hay que destacar que el significado de un signo no es estático, sino que puede variar con el tiempo en función de la evolución histórica y social del entorno al cual pertenece, el cual, obviamente, evoluciona también de generación a generación. Esta evolución puede seguir un curso natural o ser forzada en un determinado momento. El paradigma clásico de cambio radical en el significado de un símbolo lo tenemos en la svástica, primitiva espiral que a lo largo de los siglos y en diversas culturas fue adoptando diversos significados. Para las primitivas tribus indoeuropeas este era un símbolo solar. Para los romanos del bajo imperio este símbolo representaba el concepto de *paz* o el de *guerra*, dependiendo de si su orientación coincidía con el de las manecillas de un reloj (guerra) o no (paz). Tras la primera guerra mundial, este signo cambió intencionalmente de significado de manera radical al ser adoptado como símbolo del nazismo, según reconoció el propio Hitler (1925) en sus memorias.

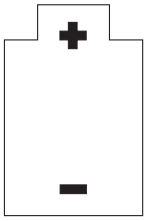
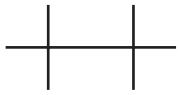


Figura 28. La misma cruz de la Figura 26 y de la Figura 27 situada ahora sobre el contorno de una batería, cambia su significado para representar el concepto de polaridad positiva.

También es necesario destacar que el significado de un símbolo se ve potenciado la asociación de símbolos de la misma clase. Así, en el ejército, suele tener más rango quien más estrellas tenga en su pechera, sirviendo las estrellas como símbolo de poder. En la Tabla 9, se ilustra el empleo de otros símbolos en los que se refuerza o dobla su significado individual por medio de la duplicación. Naturalmente, existen excepciones a esta regla. Por ejemplo, en la gramática castellana el signo punto (.) se emplea para finalizar una frase. El doble punto (:) sin embargo, no se emplea para finalizar un documento como podría parecer, sino todo lo contrario, ya que se usa para empezar una frase.

Tabla 9. Refuerzo del significado de algunos símbolos gracias a su duplicación.

No.	SIGNIFICADO	SIGNO
01	Peligro	

02	SS, doble peligro	
03	Jaque	
04	Jaque Mate	
05	Hongo venenoso	
06	Hongo mortal	

Otro aspecto destacable de los símbolos es su capacidad para mantener significados contrapuestos incluso para un mismo entorno cultural, tal y como se puede apreciar en la Tabla 10. Hay que destacar que esta característica parece estar presente solamente en aquellos símbolos rudimentarios. A medida que los símbolos adquieren mayor complejidad, su carga semántica suele concentrarse en un solo significado, tal y como ocurre en el corazón representado en la mencionada Tabla 10.

Tabla 10. Los símbolos rudimentarios suelen sugerir al menos dos significados opuestos (símbolos 1 y 2), mientras que los símbolos más elaborados sugieren un significados no contrapuestos (símbolo 3).

No.	SIGNIFICADOS	SIGNO
01	Muerte o vida eterna	
02	Cero o infinito	
03	Corazón, amor...	

10.1.3. El Lenguaje de la Imagen

La mayoría de nuestras percepciones se producen a partir de estímulos incompletos cuyas lagunas son cubiertas por información almacenada en nuestra memoria a largo plazo completando en parte la información contenida en el mensaje percibido. Aun en el caso en el que el objeto representado no sea fácil de identificar, en la mayoría de los casos basta con que se proporcione una pista del objeto representado, para que éste sea identificado inmediatamente. En experimentos realizados por Hershenson (1998), se sometía a los observadores a

escuchar el sonido de una palabra durante un período de tiempo tan corto que ésta era irreconocible. Sin embargo, si ésta se repetía una y otra vez, la palabra era finalmente reconocida. Mediante este mecanismo de percepción, aún ante una figura construida con manchas inconexas, el ser humano es capaz de reconocer representaciones de objetos reales, discerniendo claramente entre lo que es una representación y lo que es el objeto real.

En el caso de las imágenes, para su correcta interpretación es necesario cierto grado de selección de los indicadores perceptivos que se encuentran presentes en ella. Para poder reconocer las imágenes se debe partir de la esencia de que se trata de una representación plana. Por otro lado, es necesario atender a los indicadores de profundidad, como por ejemplo la perspectiva lineal o la interposición de los objetos. La estrategia perceptiva de un individuo dependerá de la frecuencia con que encuentre determinados indicadores en su entorno. La exposición continua a determinados tipos de estímulo, así como el aislamiento con respecto de otros, puede condicionar las experiencias visuales del individuo, quien solo será capaz de reconocer sin ayuda representaciones gráficas con cierto grado de similitud con otras percibidas en tiempos pasados.

Una de las facetas más importantes de esta modo de operar es sin duda la posibilidad de poder captar formas incompletas como si se fuesen formas completas. Esta es la forma más sutil que existe de disfrazar un estímulo; pues permite enviar mensajes incompletos al usuario, quien se encargará de completarlos de acuerdo con su entorno cultural. Si ese mensaje incompleto carece de ambigüedad y su significado es de conocimiento general, podrá acentuar su fuerza dentro del marco conceptual particular del usuario, al particularizarlo en función de los sus esquemas culturales. De este modo es posible amplificar la señal del canal de comunicación, disminuyendo el ruido y potenciando el mensaje. En la Figura 29 se reproduce la representación esquemática de una yegua. Como se puede apreciar, aunque parte los trazos están ausentes (sobre todo en las patas del animal), esto no es óbice para identificar al animal como un equino realizando una reconstrucción mental de las zonas poco detalladas.

Figura 29. Representación paleolítica de una yegua en estado de gravidez (Cueva de la Peña de Candamo, Asturias). Aunque parte del contorno del animal no está definido, la experiencia previa del observador es empleada para rellenar el trazo ausente.

La propia naturaleza del proceso perceptivo que nos permite reconstruir imágenes completas a partir de estímulos incompletos, es causa de en un buen número de situaciones se produzcan percepciones erróneas. En un experimento llevado a cabo por Ross (1997), se

mostraban imágenes a los observadores voluntarios en una fracción de tiempo tan minúscula que a éstos no les era posible apreciar todos los detalles. Aunque las formas empleadas eran montajes irreales en los que aparecían por ejemplo perros con dos patas, personas con tres brazos o caras con tres ojos, los observadores corregían inconscientemente dichas irregularidades describiendo las imágenes con todo detalle sin reportar ninguna irregularidad. Lo más destacado de este experimento es que este fenómeno se presentaba aunque la imagen se mostrara repetidamente o incluso cuando se dirigía la atención de observador de forma intencionada hacia el aspecto anómalo de la imagen. En la Figura 30 se muestra un ejemplo de ilusión visual en la que el conocimiento previo del mundo real no coincide con lo representado en la imagen.

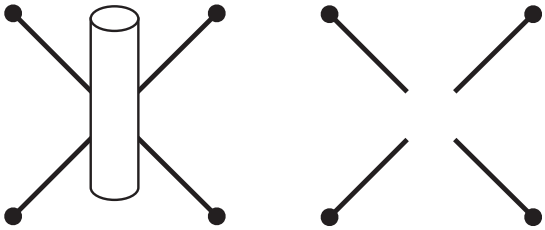


Figura 30. En la imagen de la izquierda, el observador imagina la existencia de dos líneas cruzadas sobre las que reposa el cilindro, cuando en realidad se trata de cuatro segmentos (derecha).

10.2 Adaptación de Contenidos

De este análisis previo de las características cognitivas y perceptivas del lenguaje podemos extraer como conclusión que la carga semántica de los contenidos de una aplicación depende directamente de las experiencias previas del usuario receptor de la información. Aún cuando en líneas generales es posible clasificar y agrupar usuarios en entornos culturales caracterizados por un el empleo de un significado similar para una misma colección de símbolos, las diferencias existentes entre el fondo cultural de cada individuo hace imposible alcanzar el objetivo de una individualización perfecta del mensaje transmitido por la interfaz, al menos desde el punto de vista del emisor.

Dada esta premisa negativa, todo sistema de gestión de interfaces de usuario tendrá que limitar el alcance de su adaptación a objetivos muy genéricos ya que es del todo imposible conseguir que la imagen mental de todos y cada uno de los usuarios de una aplicación coincida con el esquema mental propuesto por su diseñador, según se ha comentado en el apartado 4.1.4 (*La Semiótic*) del capítulo 4 (*La Comunicación*). Nos movemos entonces en las turbias aguas de la generalización, intentando transmitir un mensaje lo más adaptado posible a las supuestas características del usuario destino, pero sabiendo que el propio ruido del canal de comunicación, así como una posible alineación incorrecta entre los modelos mentales de quien transmite el mensaje y de quien lo recibe, pueden hacer que parte de la señal se pierda.

Por ello, el objetivo del proceso de adaptación de contenidos será intentar realizar una traducción aproximada (que no exacta) del modelo mental del emisor, traduciendo el mensaje, en la medida de lo posible al lenguaje empleado por el destinatario.

Siguiendo con el objetivo propuesto de separar la funcionalidad de una aplicación de su interfaz (mensajes incluidos), el esquema mental propuesto por los diseñadores de la aplicación durante las fases de análisis y diseño, es decir, el conjunto formado por el modelo de procesos, el modelo de datos y el modelo de interacción, será traducido a un lenguaje capaz de ser comprendido por el usuario, proporcionando tantas versiones de los mensajes incluidos en el esquema original como posibles destinatarios de la información.

El mecanismo de traducción de un esquema a otro es responsabilidad del *Language Manager*, un submódulo de CodeX. Aprovechando el mecanismo de separación de funcionalidad e interfaz adoptado por CodeX, el *Language Manager* inspeccionará el lenguaje empleado en la aplicación, realizando una traducción automática de bajo nivel con la que será posible proporcionar a DEVA los mensajes convertidos ya al lenguaje del posible receptor. De este modo, la capa de más bajo nivel de GADEA es capaz de realizar el primer paso en el proceso de adaptación, dejando que las capas superiores concentren sus esfuerzos en la adaptación de la sintaxis de la información.

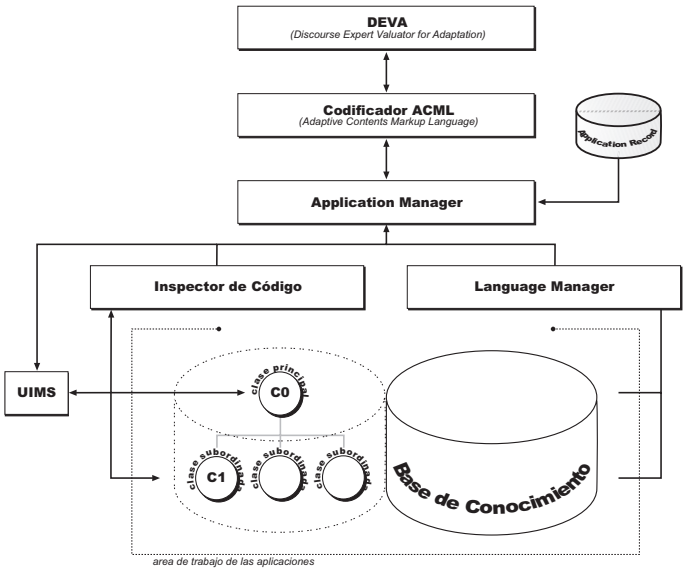


Figura 31. El *Language Manager* permite separar la funcionalidad de su base de conocimiento, permitiendo un acceso separado a la misma, lo cual posibilita realizar una adaptación de contenidos.

10.2.1. El *Language Manager*

La estrategia adoptada por el *Language Manager* para la traducción de la semántica interna de una aplicación está basada en la empleada por sistema de internacionalización de Java 1.1 [Weber (1997) p. 267] pero aumentando su flexibilidad, integrándola con los mecanismos de reconocimiento automático de procesos de usuario y diálogos

interactivos, así como aumentando su poder al lograr que la conversión se realice a nivel de valor cultural en lugar de a nivel de idioma o dialecto como ocurre en dicho sistema. Al fin y al cabo, el lenguaje es uno más de los valores que forman parte de un entorno cultural [Yeo (1996)].

En el sistema de internacionalización de Java, un objeto de la clase *Locale* representa información acerca de una ubicación espacial concreta a nivel de idioma, identificando la región del planeta en que se utiliza dicho idioma o dialecto. Todo objeto *Locale* es construido a partir de una terna formada por un código que representa el idioma, un código para indicar el país y cn código opcional para indicar la variante del idioma (dialecto). Dichos códigos son cadenas de caracteres y se pueden emplear a discreción. En el Código 18 se incluye como ejemplo la definición de un *Locale* para el inglés (*en*) de Estados Unidos (*US*), otro para el inglés del Reino Unido (*UK*) y otro para el castellano (*es*) de España (*sp*) hablado en Asturias (*as*).

Código 18. Definición de un *Locale* para varios idiomas y sus variantes.

```
// US Locale.
Locale usLocale = new Locale ("en", "US", "en");

// UK Locale.
Locale ukLocale = new Locale ("en", "UK");

// Spanish Locale.
Locale spanishLocale = new Locale ("es", "sp", "as");
```

Cuando a un método sensible a un ubicación concreta se le pasa como parámetro un objeto *Locale*, éste intentará modificar su comportamiento para el idioma o dialecto representado por el *Locale*. Por ejemplo, si se intenta mostrar la fecha actual con el objeto *usLocale* del Código 18 en su representación se mostrará el mes antes que el día, como ocurre en el inglés empleado en los Estados Unidos. Justo lo contrario ocurrirá al emplear la clase *spanishLocale*.

Aunque potente, esta estrategia de conversión de mensajes deja algunas importantes lagunas ya que el idioma en el que se expresan las ideas es sólo uno de los posibles parámetros que se pueden emplear a la hora de clasificar al usuario de una aplicación. Como hemos visto, existen otros condicionantes culturales que afectan a la percepción que tienen un determinado usuario acerca del significado de un mensaje. Por ejemplo, la percepción que un botánico pueda tener de el concepto *árbol* no es la misma que pueda tener un informático, aunque ambos puedan compartir a un mismo *Locale*. Estas notables diferencias son detectadas no solo en la jerga empleada por la aplicación sino también en las representaciones gráficas de los objetos de la misma, especialmente en los iconos e ideogramas.

Es por ello que el planteamiento del *Language Manager* debe ir más allá del soporte proporcionado por el mecanismo de internacionalización descrito, permitiendo que el diseñador de una aplicación pueda definir tantas categorías como sean necesarias y sobretodo, sin restringirlas al idioma empleado por los posibles usuarios de la aplicación. Así, siguiendo con el ejemplo anterior, en una

aplicación de propósito general, su diseñador podría definir contenidos para aquellos usuarios que hablen el castellano (*es*) de España (*sp*) en Asturias (*as*) y que bien sean botánicos (*bo*) o bien sean informáticos (*cs*). Pasamos entonces a hablar de la definición de un *Culture* (cultura) en lugar de un *Locale*, lo que constituirá la materia prima del *Language Manager*. En el Código 19 se incluye la definición de los entornos culturales empleados en el ejemplo por medio de sendas instancias de la clase *Culture*. Nótese que dado que no existe ningún límite en el número de palabras clave a emplear en la denominación de una cultura, éstas son añadidas en secuencia empleando el caracter subrayado (_) para agruparlas.

Código 19. Definición de un *Culture* para dos entornos culturales que comparten un mismo idioma.

```
/ Botanic Culture.
Culture botanicCulture = new Culture ("es_sp_as_bo");

// Computer Science Culture.
Culture csCulture = new Culture ("es_sp_as_cs");
```

10.2.2. La Base de Conocimiento

Una vez que se ha definido un objeto para representar el entorno cultural al que pertenece un determinado grupo de usuarios, los métodos empleados para mostrar información al usuario (definición de diálogos interactivos, creación de primitivas de datos, etc.) emplearán dicho objeto para realizar la traducción entre los conceptos definidos durante la fase de análisis y diseño de la aplicación en bloques de conocimiento comprensibles por el usuario.

Para ello, los recursos que serán mostrados al usuario tras la aplicación del proceso de traducción mencionados, habrán de almacenarse en la base de conocimiento de la aplicación, la cual está compuesta por un conjunto de documentos almacenados a partir de un directorio destinado a tal fin y que debe estar presente en toda aplicación GADEA. Dentro de estos documentos se ubicará la información traducida al lenguaje empleado por los usuarios de la aplicación o, dado el caso, una referencia al lugar en el que se encuentre almacenada esta información.

Para el nombrado de estos documentos se empleará una vez más unas estrictas normas de notación, que de cumplirse, garantizarán a CodeX una acceso fiable a la información necesaria para realizar la traducción automática de gran parte del lenguaje empleado por una aplicación. La notación propuesta por Codex es bastante sencilla y consiste simplemente en nombrar los documentos empleando las claves usadas en el nombrado del entorno cultural que representan, terminando el nombre con la extensión *sem* (semántica). Así, la información representada en los objetos *Culture* del ejemplo del Código 19 debería ubicarse en un par de documentos denominados respectivamente *es_sp_as_bo.sem* y *es_sp_as_cs.sem*.

La información almacenada en los dos documentos del ejemplo se corresponde única y exclusivamente con información específica que haga énfasis en las diferencias culturales existentes entre botánicos (*bo*) e informáticos (*cs*). Así, en el documento específico de los botánicos (*es_sp_as_bo.sem*) se debería especificar la imagen con la que debería representarse el concepto de *árbol* para los botánicos, mientras que en el documento representativo del entorno cultural de los informáticos (*es_sp_as_cs.sem*) se deberá especificar la imagen con la que se ha de representar el mismo concepto para este colectivo concreto.

Por su parte, toda la información compartida por los dos colectivos empleados en el ejemplo (botánicos e informáticos) debería agruparse en otro documento denominado *es_sp_as.sem*. En este documento se especificaría la carga semántica compartida por todos aquellos mensajes de traducción común para usuarios que hablen el castellano (*es*) de España (*sp*) en Asturias (*as*), independientemente de si son botánicos o informáticos. Es necesario destacar que el concepto árbol que nos ha servido como hilo conductor de este ejemplo y como excusa para diferenciar a botánicos de informáticos también puede estar presente en este documento, tratándose en este caso de representar el concepto de árbol sin tomar en cuenta la profesión del usuario. En este caso se intentaría representar un árbol cercano a la cultura asturiana (un roble, por ejemplo) ya que ésta es la última clave del nombrado del documento. Esta representación particular del árbol intentaría diferenciarla de la incluida en un posible documento de nombre *es_sp_ma.sem* definido para representar un árbol reconocido por los habitantes de Madrid (*ma*) que hablan el castellano (*es*) de España (*sp*), pudiendo ser representado éste último por un madroño, por ejemplo.

El proceso continúa como se apreciar partiendo de las especializaciones de un determinado entorno cultural, hacia representaciones cada vez más globales. En el ejemplo seguido, el siguiente paso será definir la representación de árbol para los hablantes del castellano (*es*) de España (*sp*) en el documento *es_sp.sem*. Este concepto de árbol puede estar definido en este nivel (representado un árbol genérico para todos los habitantes de España y que hablen castellano) o no, dejando su definición a las posibles especializaciones del entorno cultural (Asturias, Madrid, etc.) o a la generalización del mismo, es decir, al documento *es.sem*, el cual definiría la representación de un árbol para todos los hablantes del castellano en el mundo.

Como se puede apreciar, con este sencillo sistema de notación, heredado de Java y ampliado de acuerdo con las implicaciones cognitivas y perceptivas que requiere todo lenguaje, va a originar que la base de conocimiento se estructure como un gran árbol n-ario en donde la información de carácter general (idioma, cultura, etc.) se sitúa en su raíz y en las ramas superiores, mientras que la información de carácter específico (dialecto, definiciones de usuario, etc.) se localiza en sus hojas y ramas inferiores.

En tiempo de ejecución, el *Language Manager* de CodeX empezará el proceso de búsqueda por una de las hojas del árbol, la cual es especificada como punto de arranque por defecto en el Application Record (ver apartado 7.1: *Registro de Aplicaciones GADEA*). En la Figura

32, podemos ver la estructura de la base de conocimiento empleada en el ejemplo seguido en este apartado. De acuerdo con el algoritmo de búsqueda empleado por CodeX, si la hoja de búsqueda es la definida por el camino *es_sp_as_bo*, se intentará buscar la representación de un *árbol* en nodo marcado con el número 1, es decir, en un fichero de nombre *es_sp_as_bo.sem*. De encontrarse allí, el proceso de búsqueda finalizaría y CodeX enviaría la representación encontrada al módulo DEVA para que éste lo incluya en la interfaz de usuario. En caso contrario, si el fichero *es_sp_as_bo.sem* no existe o dentro de él no se encuentra definida ninguna entrada para la clave de búsqueda (el *árbol*), la búsqueda continuaría por el nodo número 2, es decir, en un fichero de nombre *es_sp_as.sem*. De forma análoga, si el proceso de búsqueda falla también en este documento, la búsqueda continuaría por el documento *es_sp.sem* (nodo 3) y luego en documento *es.sem* (nodo 4). Si llegados a este nodo, en donde es de suponer que se almacena la información genérica sobre los conceptos empleados por la aplicación, no se encuentra el concepto árbol, se procederá a mostrar el concepto manejado por los diseñadores de la aplicación, imbuido éste en el código. Si esto no es posible, simplemente se abortará el proceso, considerando la existencia de un error grave en el código de la aplicación.

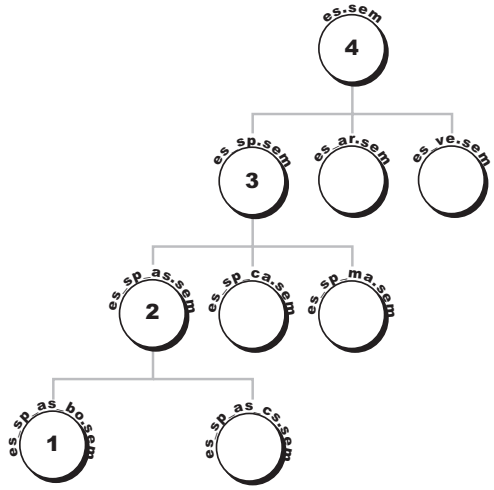


Figura 32. Árbolde búsqueda de significados en la base de conocimiento de una aplicación genérica definida para su uso por parte de botánicos e informáticos.

Será responsabilidad de los diseñadores de la aplicación el estructurar la base de conocimiento, identificando las diferentes culturas implicadas en el proceso de adaptación de acuerdo con los posibles usuarios de la aplicación. Será también responsabilidad de los autores del producto software el crear los documentos que integran la base de conocimiento y rellenarlos con la información adecuada. De este modo es posible alcanzar una separación plena entre la funcionalidad de la aplicación y su datos.

El formato de estos documentos será el de ficheros de texto en el que se especifica una línea por mensaje a traducir con el formato recogido en el Código 20. En este ejemplo se puede apreciar que a cada concepto

manejado en el modelo de interacción se le asocia un valor mediante el operador correspondiente (=). Obviamente, el lenguaje en el que están especificadas las etiquetas a traducir en el antecedente de cada operación de asignación (el concepto *árbol* en el ejemplo) han de estar escritos en el lenguaje propio manejado por los diseñadores de la aplicación y que a su vez debe ser común para todos los miembros del equipo de desarrollo. En toda aplicación desarrollada siguiendo los dictados clásicos de la ingeniería del software el significado de estos conceptos ha de quedar definido ya desde la fase de análisis [MAP (1994a)].

Código 20. Documento *en_UK.sem* en el que se especifica el significado del concepto *árbol* para usuarios de una aplicación desarrollada en castellano pero pertenecientes entorno cultural en el que se hable inglés (*en*) del Reino Unido (*UK*).

```
// Concept ::= <Value>
árbol ::= tree
```

En el ejemplo del Código 20 se puede apreciar que el formato definido para la especificación del contenido de los documentos de la base de conocimiento de una aplicación resalta por su sencillez. La idea es que estos documentos puedan ser modificables fácilmente, tanto por los creadores de la aplicación como por sus propios usuarios. De este modo, los usuarios pueden llegar a adaptar el contenido de los mismos, en función de sus necesidades o mejor aún, creando nuevos documentos y anexándolos a las hojas del árbol especificando en dichos documentos la representación particular que el usuario tiene de la información proporcionada por la aplicación.

De este modo, es el propio usuario al que tiene la capacidad última de definir su propio entorno cultural ya que en definitiva son sus experiencias previas las que condicionan la información obtenida. Así, siguiendo con el ejemplo empleado a lo largo de este apartado, si un usuario informático llamado *martín* desea personalizar la representación que la aplicación tiene del concepto *árbol*, solo tendrá que crear un documento de nombre *es_sp_as_cs_martin.sem* e indicarle a la aplicación que lo use como primer documento de sus búsquedas, especificándolo en el *Application Record*. Como se puede apreciar, la personalización no sólo es sencilla sino que no implica ninguna modificación de los documentos originales incluidos en la base de conocimiento de la aplicación .

10.2.3. Adaptación Automática

El mecanismo de reflexión de código empleado por CodeX para el registro de los procesos de usuario, canales de comunicación, diálogos interactivos, *chunks*, primitivas de datos, etc. es empleado también para realizar una adaptación automática de la semántica de la aplicación y de sus recursos. Para ello basta con emplear el nombre del concepto registrado por CodeX como clave de búsqueda en la base de conocimiento.

Así por ejemplo, cuando el punto de entrada a un proceso de usuario es detectado, el *Language Manager* empleará el nombre interno del método que hace las veces de punto de entrada y lo empleará para realizar una traducción inmediata de dicho nombre al significado final empleado por el usuario activo de la aplicación.

Para ilustrar este proceso vamos a suponer que CodeX registra la clase *Document* empleada como ejemplo en el Código 10 (capítulo 7: *Aplicaciones y Procesos de Usuario*) y que el nodo activo en la base de conocimiento de la aplicación es el incluido en el Código 21, el cual representa información particular para usuarios que hablan el castellano (*es*) de España (*sp*) en Asturias (*as*). Cuando el método *registerEveryUP* es invocado en la clase UIMS, CodeX inspecciona la clase en busca de procesos de usuario, encontrando los métodos *UP_openDocument* y *UP_closeDocument*. Empleando las claves de búsqueda *openDocument* y *closeDocument* respectivamente para cada método, el *Language Manager* encontrará los significados *Abrir Cartafueyu y Pesllar Cartafueyu* en el documento *es_sp_as.sem*, enviándoselos a DEVA, quien los incluirá como parte de la interfaz de usuario.

Código 21. Documento *es_sp_as.sem*. Ejemplo de información específica del un contexto cultural formado por los hablantes del castellano (es) de España (sp) en Asturias (as).

```
// users of the Spanish spoken in Asturias (Spain)
openDocument ::= Abrir Cartafueyu
closeDocument ::= Pesllar Cartafueyu
```

Este proceso de traducción automática es aplicable a cualquiera de los conceptos presentes en una aplicación GADEA. A medida que se identifique los distintos conceptos que forman parte de la aplicación, CodeX configurará su carga semántica con el fin de que DEVA pueda emplearlos. Para todo concepto, la información requerida estará formada por los siguientes elementos:

- **Clave de Búsqueda**. Sirve para referirse a un concepto, tanto al nivel de la metodología empleada en el desarrollo de la aplicación como a nivel de código, sirviendo de puente de enlace entre el módulo CodeX y el motor funcional de la aplicación. Su alcance está restringido al desarrollo interno de la aplicación, no trascendiendo en ningún momento al usuario a menos que esto sea estrictamente necesario (cuando en su traducción no se encuentra una definición global en la base de conocimiento). Este identificador, que podrá estar formado por cualquier secuencia continua de caracteres (empezando por una letra y sin espacios en blanco) se escribirá en el idioma del proyecto siguiendo la jerga y nomenclatura del mismo.
- **Valor Semántico**: A diferencia del identificador, esta propiedad de todo concepto indica el nombre con el cual el usuario final de la aplicación reconoce dicho concepto, formando parte de la percepción que el usuario tiene del dominio del problema y por lo tanto será susceptible de ser adaptado a su modelo cognitivo. Esta adaptación puede realizarse a varios niveles, incluyendo en primer lugar el idioma que emplea el usuario, en segundo lugar la jerga del entorno cultural al que éste pertenece y por último –y ya en

función de la experiencia personal del usuario– la adaptación puede realizarse al nivel del vocabulario empleado por el usuario.

En todo sistema de gestión de interfaces de usuario el valor semántico no puede estar limitado a la mera traducción de cadenas de caracteres sino que ha de extenderse al resto de los mensajes de la aplicación susceptibles de adaptación y que además dependen del mismo concepto o idea. Así, el concepto de *árbol* empleado en los ejemplos que ilustran este apartado puede estar formado no sólo por su representación en un idioma destino (*árbol*, *tree*, etc.) sino también por su representación iconográfica, sonora e incluso Braille. Por esta razón todo sistema de gestión de interfaces de usuario debe extender el modelo de tratamiento semántico de la información permitiendo acceder a diferentes tipos de información traducible por medio de la misma clave de búsqueda.

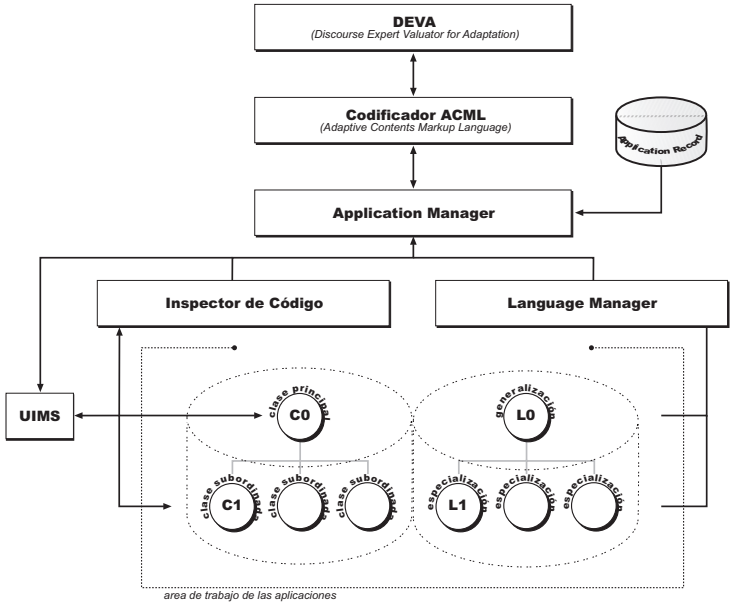


Figura 33. El diseño final de CodeX incluye la estructuración arbórea de la base de conocimiento de las aplicaciones para permitir una adaptación de contenidos en función del entorno cultural del usuario de la aplicación.

Para lograr este propósito, en función de las necesidades de información del usuario (transmitidas a través de DEVA), CodeX añade de forma automática un determinado sufijo a la clave de búsqueda, indicando con ello el tipo de información a traducir y por lo tanto el tipo de información a obtener.

En la Tabla 11 se incluye una relación completa del tipo de información susceptible de ser adaptada al entorno cultural del receptor, indicando el sufijo añadido a la clave primaria de búsqueda. Como se puede apreciar, parte de la información traducible depende del tipo de usuario que accede a la información. Tal es el caso de los usuarios con discapacidades visuales (prefijo *_BLIND_NOTATION o _VOWEL*) o para aquellos con cierto grado de veteranía en el empleo de la aplicación (prefijo *_KEYSTROKE* o *_VOWEL*).

Tabla 11. Relación de las distintas representaciones de un concepto gestionadas por el *Language Manager* de CodeX y los sufijos empleados.

SUFIJO	DESCRIPCIÓN
_SHORT_NAME	Abreviación para el nombre de un concepto. Es empleado sobretodo en el canal de comunicación auditivo.
_ICON	Dirección y nombre del fichero que contiene la representación gráfica del concepto.
_FAST_HELP	Breve descripción del concepto, desplegada como ayuda cuando el usuario señala el *widget* correspondiente al concepto mostrado.
_HELP	Dirección y nombre del fichero que contiene la descripción en detalle del concepto, el cual es empleado por el sistema de ayuda.
_BLIND_NOTATION	Descripción del concepto empleando metáforas especialmente diseñadas para usuarios ciegos o con serias discapacidades visuales.
_KEYSTROKE	Atajo de teclado para usuarios expertos.
_VOWEL	Combinación de vocales neutro empleada por usuarios expertos que dependan del canal de comunicación sonoro debido a discapacidades visuales.

En el caso de los primeros, el módulo DEVA podrá plantear métodos de despliegue de información especialmente diseñados para usuarios ciegos o con serias discapacidades visuales los cuales requieren de metáforas especiales y por lo tanto necesitarán en consecuencia una notación acorde con las mismas, tal es el caso del empleo de la conocida *metáfora de la habitación cerrada* [Savidis y Stephanidis (1995)].

En el caso de los usuarios expertos, es posible emplear medios de acceso de interacción alternativos que les permitan sacar un mayor provecho de la sesión de trabajo con la aplicación, aumentando la productividad por medio de técnicas muy simples. Algunas de estas técnicas, como aquellas del tipo de los atajos de teclado, requieren la creación de un modelo mental de las mismas, el cual es susceptible de ser adaptado [Karn et. al. (1997)].

En ambos casos, será el módulo que solicita la información (DEVA) quien especifique el tipo de información deseada en función de los datos almacenados en el modelo de usuario activo, siendo CodeX un mero intermediario entre la interfaz de la aplicación y su base de conocimiento.

En el Código 22 se ha modificado el documento *es_sp_as.sem* para incluir información adicional para los conceptos *openDocument* y *closeDocument*. Ahora estos procesos de usuario disponen de información acerca de su representación gráfica (*_ICON*), de su acelerador de teclado (*_KEYSTROKE*) y de su ayuda rápida (*_FAST_HELP*) entre otros. Para acceder a cualquiera de estos datos, el objeto solicitante solo tendrá que añadir el sufijo correspondiente a la información requerida y el *Language Manager* se encargará de buscarla, obtenerla y devolverla como resultado de la transacción. Por ejemplo si DEVA necesita el icono representativo del proceso *openDocument*, no necesitará más que solicitar al *Language Manager* que busque por la clave *openDocument_ICON* en la base de conocimiento para obtener el nombre y dirección del fichero *gif* o *jpg* que lo contiene.

Código 22. Documento de especificación semántica *es_sp_as.sem* en el que se incluyen varias representaciones de los conceptos *openDocument* y *closeDocument*.

```
// users of the Spanish spoken in Asturias (Spain)
openDocument ::= Abrir Cartafueyu
openDocument_ICON ::= ..\Resources\Images\Ast\abrir.gif
openDocument_FAST_HELP ::= Abre el cartafueyu pa la so edición
openDocument_KEYSTROKE ::= CTRL_A

closeDocument ::= Pesllar Cartafueyu
closeDocument_ICON ::= ..\Resources\Images\Ast\pesllar.gif
closeDocument_FAST_HELP ::= Peslla el cartafueyu
```

Al contrario de lo que ocurre con el significado básico de un concepto, es decir, aquel obtenido por medio de una clave de búsqueda sin sufijo, si el proceso de búsqueda en la base de conocimiento no da resultados positivos, la ejecución de la aplicación no es abortada ya que la información obtenida mediante sufijos está compuesta generalmente por información complementaria. En estos casos se pierde en gran medida el poder de adaptación proporcionado por la tecnología pero no obstante, se puede salvar la ejecución mediante el empleo de información adicional incluida en la base de conocimiento del módulo DEVA, es decir, información de carácter general para todas las aplicaciones.

Así por ejemplo, si para un concepto dado no existe sufijo *_VOWEL* pero si existe uno *_KEYSTROKE* y el usuario de la aplicación tiene problemas de discapacidad visual u auditiva, DEVA transmitirá por el canal de comunicación auditivo la vocal correspondiente al atajo de teclado haciendo uso del alfabeto internacional de la aviación civil.

En este alfabeto, las vocales son sustituidas por la palabra correspondiente incluida en la Tabla 12 relacionándose cada palabra con un vocal. Así, cuando se desea transmitir la palabra GADEA, en su lugar se emite el mensaje *Golf, Alpha, Delta, Echo, Alpha*. La ventaja que tienen estas palabras es que son muy fáciles de pronunciar, legibles por personas de todo tipo de experiencia y fácilmente distinguibles una de otra. Hay que recordar que dado el carácter internacional de este alfabeto, su diseño incluyó como requisito el que las palabras fuesen reconocibles cuando son habladas por y para personas de diferentes nacionalidades y acentos. Al emplear un vocabulario restringido, conocido tanto por el emisor como por el receptor la tasa de efectividad se incrementa [Chapanis (1965) p. 94].

Tabla 12. Alfabeto internacional adoptado por *la Organización Internacional de la Aviación Civil* (OIAC) y la *Organización de Tratado del Atlántico Norte* (OTAN) entre otras. [Fuente: Chapanis (1965) p. 94].

Alpha	***Hotel***	***Oscar***	***Victor***
Bravo	***India***	***Papa***	***Whiskey***
Charlie	***Juliet***	***Quebec***	***X Ray***
Delta	***Kilo***	***Romeo***	***Yankee***
Echo	***Lima***	***Sierra***	***Zulu***
Foxtrot	***Mike***	***Tango***	
Golf	***November***	***Uniform***	

11

11 El Application Explorer

La claridad consiste en una acertada distribución de luz y sombra. Pensadlo bien

Goethe

11.1 Introducción

El *Application Explorer* es una pequeña aplicación necesaria para ejecutar aplicaciones bajo CodeX, la cual ha sido escrita siguiendo el proceso de diseño basado en modelos de interacción descrito en el capítulo 6 (*El Modelo de Interacción*). Esta aplicación, que hace las veces del *Explorador* de la plataforma Windows [Ivens (1996)] o del Finder/MultiFinder [Apple (1987)] de la plataforma Macintosh forma parte de la distribución estándar de GADEA , actuando como mecanismo de interacción primario entre GADEA y el usuario.

Esta aplicación se carga de forma automática cuando se inicia la plataforma GADEA y permanece en activo durante toda la sesión de usuario, siendo la última aplicación en cerrarse, bien cuando se apaga el equipo (caso de tratarse del SGIU de un sistema operativo del estilo de *Oviedo-3*) o bien cuando el usuario decide terminar con su sesión de trabajo en GADEA.

Los procesos de usuario básicos de esta aplicación son los siguientes:

- Identificación de Usuario.
- Registro de Nuevos Usuarios.
- Selección de una Aplicación.
- Salir.

Al arrancar el *Application Explorer*, se presume un modelo de usuario genérico en el que todos los parámetros de interacción se encuentran iniciados al mínimo posible. Dado que se desconoce el tipo de usuario que va a emplear la aplicación, la estrategia seguida por GADEA será muy conservadora con el objetivo de alcanzar un mínimo nivel de interacción para todos los posibles usuarios hasta que sea factible identificar al usuario concreto que está intentando utilizar el sistema.

Esto implica que GADEA supondrá la existencia de un usuario disminuido físicamente en todos los sentidos, con escasa habilidad y nula experiencia con el sistema. Con esta información, DEVA adaptará en consecuencia el diálogo interactivo de este proceso de usuario, potenciando las metáforas de navegación para ciegos [Savidis y Stephanidis (1995)], disminuyendo en la medida de lo posible la carga cognitiva del diálogo, mostrando muy pocos objetos y dotándoles del mayor contraste posible (tanto auditivo como visual).

En esta situación temporal se le proporciona al usuario el mejor medio posible para darse a conocer a través de los mecanismos de registro de la aplicación y por medio de sendos procesos de usuario (11.1.1 *Identificación de Usuario* y 11.1.2 *Registro de Nuevos Usuarios*).

11.1.1. Identificación de Usuario

Cuando el usuario selecciona esta opción, se crea un diálogo interactivo simple formado por dos *chunks*, uno destinado a obtener el identificador del usuario y otro a obtener su clave de acceso al sistema. Los datos proporcionados por el usuario son enviados al gestor de accesos quien será en encargado de contrastar su veracidad en el modelo de usuario de GADEA.

Dado que GADEA prevé la distribución de modelos de usuario en una red de ordenadores, el proceso de análisis se realiza a en dos fases de ejecución, una local (primero) y una remota (después). En primer lugar se intenta obtener el modelo de usuario en un dispositivo de almacenamiento local, empleando como clave de búsqueda el identificador proporcionado por el usuario. Si la búsqueda es satisfactoria, se inicia el modelo de usuario con los parámetros guardados en el dispositivo de almacenamiento local y se procede a activar el proceso de usuario para la selección de las aplicaciones registradas en GADEA. A partir de este momento, todo el comportamiento del sistema se adapta a las características específicas del usuario activo. Dependiendo del tipo de usuario, esta adaptación puede incluir cambios drásticos en los diálogos interactivos subsecuentes, incluyendo modificaciones en las características de los

objetos perceptibles (tamaño, color, volumen, etc.) así como de su distribución en el espacio visual y auditivo, favoreciendo un modelo de interacción lo más adecuado posible a las características cognitivas, perceptivas y motoras del usuario.

En caso de fallo en el acceso local a la información sobre el modelo de usuario (bien porque el usuario no se encuentre registrado en el sistema, o bien debido a un fallo técnico), la búsqueda prosigue en el servidor central. Esta característica de GADEA favorece la movilidad de los usuarios en una red de ordenadores sin perder ni un ápice de la potencia aportada por los mecanismos de adaptación de GADEA. De hecho, en una situación ideal en la que el sistema propuesto tuviese una implantación a gran escala, sería factible que cuando un usuario emplease una aplicación GADEA en el ordenador de otra persona, dicha aplicación se adaptase por completo a las necesidades del nuevo usuario. Esto permitiría la distribución de los mecanismos de adaptación diseñados para un usuario concreto en cualquier ordenador del mundo.

De todos modos, este esquema permite el empleo de la misma aplicación en el mismo ordenador por parte de dos usuarios distintos mediante la el empleo de modelos de interacción completamente diferentes. De hecho, el comportamiento de la aplicación puede variar para un mismo usuario de una sesión a otra en la misma medida en la que sus características cognitivas, perceptivas y motoras varíen. Un ejemplo significativo de este cambio de una sesión a otra puede ser el grado de veteranía alcanzado por un usuario gracias al empleo continuado de una aplicación, todo lo cual implica el uso de técnicas de acceso adaptadas para usuarios veteranos [Borenstein p. 83-91 (1991); Karn et. al. (1997)].

Se puede llegar incluso al caso de modificar los modelos de interacción dentro de una misma sesión, sobre todo durante las primeras sesiones con el sistema, en la que el proceso de aprendizaje por parte de los agentes es más radical, al pasar de un modelo de usuario vacío uno muy específico. También hay que recordar que la capacidad de la memoria a corto plazo (MCP) [Miller (1956) p. 81-97] varía enormemente en función del esfuerzo previo realizado, por lo que un usuario que realice largas sesiones de trabajo verá mermada su capacidad de retención a corto plazo en la misma proporción. Caso aparte es el de usuarios con discapacidades visuales serias, en donde la memoria a corto plazo es el instrumento más valioso con el que cuentan para orientarse y por lo tanto debe aprovecharse con sumo cuidado. Como se verá a partir del capítulo 12 (*El Modelo Cognitivo*), la información proporcionada por ANTS servirá para ajustar aquellos puntos de un diálogo interactivo que dependan de la memoria a corto plazo.

En el caso en el que el modelo de un usuario concreto no se encuentre ni el dispositivo de almacenamiento local ni el en el remoto, este proceso de usuario (11.1.1 *Identificación de Usuario*) creará un diálogo interactivo en el que se preguntará al usuario si quiere continuar con el modelo de usuario genérico (aquel preparado para cualquier tipo de usuario y que asume que todos los parámetros del

modelo de usuario se encuentran en su nivel más bajo) o por el contrario desea registrarse en el sistema para que éste pase a guardar todos los datos obtenidos acerca de sus características cognitivas, perceptivas y motoras en un modelo de usuario individual que pueda ser recuperado entre sesiones. En caso de elegir la segunda opción se ejecutará el proceso de usuario *Registro de Nuevos Usuarios*.

Si el por el contrario, el proceso de búsqueda resulta positivo y se reconoce al usuario como miembro de la comunidad de usuarios de GADEA, se actualiza el modelo de usuario actual con los valores leídos, con lo que el próximo diálogo interactivo establecido por GADEA se diseñará de acuerdo con las características cognitivas, perceptivas y motrices del usuario actual. A continuación se desactivan los procesos de usuario 11.1.1 *Identificación de Usuario* y 11.1.2 *Registro de Nuevos Usuarios* haciendo que falso el valor de respuesta de los métodos asociados a sus *precondiciones*.

Dado que el modelo de usuario es actualizado constantemente a lo largo de una cada sesión de usuario, cuando éste decide cerrar su sesión de trabajo con GADEA, los cambios realizados en el modelo de usuario se actualizan simultáneamente en la copia que del modelo que se guarda en el dispositivo de almacenamiento local y en la que se almacena en el servidor central.

11.1.2. Registro de Nuevos Usuarios

Partiendo de la situación comentada en el apartado 7.1 (*Registro de Aplicaciones GADEA*), este proceso supone que el usuario que se está registrando dispone de un modelo de usuario genérico en el que todos los parámetros se encuentran bajo mínimos, es decir, se supone que el usuario es ciego (o al menos tiene muy disminuida su capacidad de percepción visual), presenta serias discapacidades auditivas y motrices, así como una serie disminución de su capacidad cognitiva. Es de suponer que una interfaz diseñada para este tipo de usuarios también podrá ser empleada por otros usuarios cuyos sistemas cognitivos, perceptivos y motrices se encuentren en un nivel superior. Hay que señalar sin embargo que éste no es el objetivo de GADEA, ya que un usuario de alto nivel cognitivo, perceptivo y motriz no tardará mucho en cansarse de una interfaz diseñada para un usuario de nivel menor. Por lo tanto es necesario destacar que esta modalidad es temporal y durará hasta que el usuario establezca implícitamente un modelo de usuario básico mejor adaptado a sus necesidades.

Este modelo de usuario básico se establece a partir del diálogo interactivo creado en el momento en el que se ejecuta este proceso de usuario. En este diálogo interactivo se le pregunta explícitamente al usuario por los siguientes valores:

- **Identificación**. Nombre de usuario único con el que se registrará en GADEA.
- **Clave de Acceso**. Clave con la que se validará su acceso al sistema.
- **Nombre y Apellidos**. Información de referencia sobre el usuario.

- **Fecha de Nacimiento**. Con este dato se calcula la edad del usuario, aunque también se la podría pedir explícitamente. Como se verá más adelante en el capítulo 12 (*El Modelo Cognitivo*), esta información es empleada para adaptar algunos aspectos visuales de la interfaz, como por ejemplo los tonos azulados, los cuales se perciben de modo diferente en función de la edad. Esta variable es empleada también para la configuración del tamaño de la memoria a corto plazo del individuo. El parámetro *Edad* es procesado y almacenado internamente como una variable lingüística [Zadeh (1975)] manejable por el motor de lógica difusa de DEVA. De este modo, la propiedad *Edad* pasa de ser un valor numérico a una función de pertenencia a cualquiera de los conjuntos difusos recogidos en la Tabla 13 en donde los valores numéricos representan edades. Estos conjuntos han sido elaborados en base a los parámetros obtenidos en investigaciones sobre el funcionamiento de la memoria realizados por Sternberg (1986) y Blumenthal (1991).

Tabla 13. Conjuntos difusos asociados a la variable lingüística *Edad*.

CONJUNTO	RANGO
Bebé	[0…4)
Niño	[4…10)
Joven	[10…18)
Joven Adulto	[18…30)
Adulto	[30…50)
Adulto Maduro	[50…70)
Anciano	[70…85)
Octogenario	[85…120]

- **Sexo**. Este parámetro es empleado para confeccionar el cálculo de la capacidad de la memoria a corto plazo y de la adaptación de los colores de la pantalla y de los parámetros del canal auditivo a las diferencias de precisión perceptiva entre sexos que se comentarán a partir del capítulo 12 (*El Modelo Cognitivo*).
- **Precisión Auditiva**. Indica la capacidad que tiene el usuario para distinguir entre distintos tonos audibles. Por el momento es una apreciación subjetiva del propio usuario aunque en un futuro ésta se podrá obtener mediante pruebas de alta precisión del tipo de los test de análisis de estilos cognitivos o *Cognitive Styles Analysis* (CSA) desarrollados por Riding (1991, 1994 y 1996) y aplicados a la medición auditiva por John y Boucouvalas (2000). El valor de este parámetro se expresa en términos de la función de pertenencia a una serie de conjuntos difusos que indican la calidad del aparato auditivo del usuario. Estos conjuntos difusos y su rango de pertenencia de la variable discreta asociada han sido diseñados en función de los estudios sobre percepción auditiva recogidos por Chapanis [(1965) p. 84-112: *Sistemas de Comunicación Hablada*] y son los reflejados en la Tabla 14.

Tabla 14. Conjuntos difusos asociados a la variable lingüística *Precisión Auditiva*.

CONJUNTO	RANGO
Sordo	[0%…20%)
Bajo	[20%…40%)
Normal Bajo	[40%…60%)
Normal Alto	[60…80%)
Óptimo	[80%…100%]

- **Precisión Visual**. Indica el grado de capacidad visual del individuo mediante una función de pertenencia a una serie de conjuntos difusos. Este parámetro es empleado para adaptar el contraste entre los colores de fondo y de frente empleados en el canal de comunicación visual de una interfaz, el valor de aquellos tonos de difícil apreciación (en especial los azules) y el tamaño de los objetos y la relación espacial entre los mismos. Obviamente, mientras menor sea el valor de este parámetro, mayor será el tamaño y contraste de los objetos mostrados en pantalla. El valor que puede asumir esta variable lingüística se restringe a los conjuntos difusos incluidos en la Tabla 15, cuya elaboración se ha realizado en función de los parámetros sobre percepción visual recogidos por Frova (1999), Montgomery (1991), Fernández (1985) y Chapanis [(1965) p. 47-83: *La Presentación Visual de la Información].*

Tabla 15. Conjuntos difusos asociados a la variable lingüística *Precisión Visual*.

CONJUNTO	RANGO
Ciego	[0%…10%)
Discapacidad	[10%…30%)
Bajo	[50%…70%)
Normal Alto	[70…80%)
Alto	[80%…90%)
Excelente	[90%…100%]

- **Lateralidad**. Especifica la condición de lateralidad del usuario indicando si es zurdo o diestro. Esta información es fundamental a la hora de elaborar la distribución espacial de los objetos en un campo visual, en especial de los *widgets* interactivos, es decir, aquellos objetos que puedan accederse a través del cursor, como es el caso de botones, *scrolls*, menús e incluso cuadros de diálogo.

Se espera que en versiones futuras de GADEA se puedan plantear tests en el que parte de la información aquí relacionada pueda ser obtenida de forma implícita y con un mayor nivel de precisión, en especial para la *Precisión Auditiva*, la *Precisión Visual* y la *Lateralidad*. Así por ejemplo, en el caso de la *Precisión Auditiva*, un *wizard* puede mostrar una serie de sonidos en distintas frecuencias para que el usuario

indique cual de ellas es la que reconoce con mayor claridad, determinando así con un mayor grado de exactitud el valor de este parámetro y con ello la función de pertenencia al conjunto difuso correspondiente.

Análogamente, en el caso del parámetro *Precisión Visual* un *wizard* puede elaborar directamente en pantalla los tests de detección de miopía y estrabismo clásicos [Saraux y Bias (1972) y Domingo et al. (1988)] y calcular a partir de ellos los valores del grado de precisión visual como un valor discreto que será convertido más adelante en una variable lingüística del tipo de las descritas.

En el caso de la *Precisión Visual*, se puede obtener además el cálculo realizado indirectamente al obtener el parámetro de la *Edad* del individuo por medio de técnicas de calibración del monitor. Dado que uno de los valores extraídos de la *Edad* del usuario consiste en el grado de tonos azules percibido, este último parámetro puede ser obtenido implícitamente mediante tests en los que se muestre al usuario una serie de figuras y/o fondos de diferentes tonalidad de azul, para que éste indique las que distingue con mayor claridad.

Este tipo de técnicas ya se aplican para el calibrado de monitores e impresoras en función del estado perceptivo de un usuario concreto y de las características de iluminación de su entorno de trabajo. En la Figura 34 se puede apreciar la ventana principal de trabajo del calibrador de monitores (*Gamma*) de la herramienta *Photoshop* [Adobe (2001)].

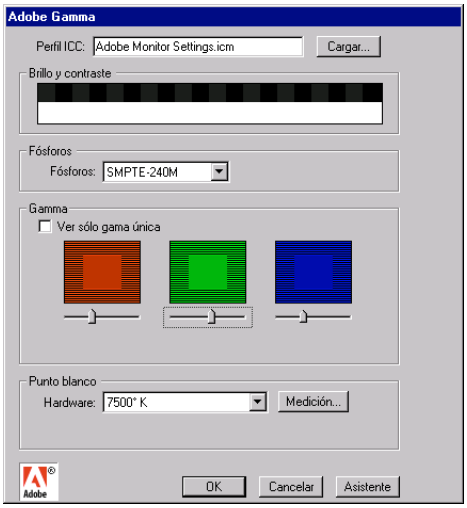


Figura 34. Calibración de un monitor con la herramienta *Gamma* de Adobe *Photoshop* 5.0. (consultar versión en color en el capítulo 24: *Imágenes en Color*, p. 445).

En el mencionado calibrador el usuario debe seleccionar los valores de las componentes roja, verde y azul del modelo RGB para conseguir un tono de gris neutro para el brillo. Naturalmente, el tono gris neutro que se desea alcanzar es un concepto abstracto cuya encarnación dependerá directamente del estado del sistema perceptivo del usuario.

Como se ha comentado, el modelo de usuario estará iniciado en un principio a los valores más bajos posibles, siguiendo una estrategia evidentemente muy conservadora. Estos valores serán los siguientes:

- *Edad*: Octogenario.
- *Sexo*: Sin especificar.
- *Precisión Auditiva*: Sordo.
- *Precisión Visual*: Ciego.
- *Lateralidad*: Diestro.

Para facilitar la interacción del usuario, los diferentes *chunks* que integran la colección de datos obtenida por el proceso de usuario *Registro de Nuevos Usuarios* es ordenada en bloques mínimos mostrados al usuario uno a uno, siguiendo el patrón de diseño de cualquier *wizard* en el que en cada ciclo de interacción intervienen muy pocos datos.

Mediante ésta técnica, el usuario especificará el valor de cada parámetro de uno en uno. Cada vez que éste proporcione el valor de un parámetro, éste pasará automáticamente al modelo de usuario actual y se confeccionará el siguiente diálogo interactivo teniendo en cuenta dicho valor. De este modo, a medida que el usuario vaya proporcionando los valores requeridos, la interfaz de este proceso de usuario se irá adaptando a los valores proporcionados.

El orden en el que el *Application Explorer* mostrará los *chunks* será el siguiente:

- *Precisión Auditiva*.
- *Precisión Visual*.
- *Lateralidad*, *Edad* y *Sexo*.
- *Identificación*, *Clave*, *Nombre* y *Apellidos*.

La justificación de este orden se encuentra en que los parámetros de mayor prioridad coinciden precisamente con aquellos que implican un cambio más drástico del modo de interacción. Adquiriendo los parámetros de adaptación con mayor peso primero se llega antes a una estabilización de los mecanismos de interacción, estabilización que obviamente resulta relativa si consideramos que el proceso de adaptación es continuo.

Debemos pensar que en la mayoría de los casos, los valores reales para estos parámetros se encuentran muy por encima de los valores asumidos al principio de este proceso (octogenario, sordo, ciego, etc.) por lo que al ir especificando su valor real siguiendo este orden concreto, la interfaz se adaptará de acuerdo con las necesidades de interacción del momento, favoreciendo las necesidades futuras.

Así por ejemplo, el primer diálogo interactivo se presentará al usuario con el canal auditivo al máximo volumen posible, lo cual puede molestar a un usuario con una *Precisión Auditiva* de valor *Normal Alto*. Una vez introducido este valor en el modelo de usuario, el volumen y los agudos del altavoz bajarán considerablemente en el siguiente

diálogo interactivo en el que se preguntará al usuario por su *Precisión Visual*. Dado que el modelo de usuario estándar asume que su valor es *Ciego*, los objetos visuales pertenecientes a este diálogo interactivo serán del mayor tamaño posible, ocupando toda la pantalla. Además, el canal auditivo transmitirá constantemente la información incluida en el canal visual. Obviamente, éste modo de interacción puede resultar irritante para un usuario dotado de una *Precisión Visual* de tipo *Alto*. Una vez expresado éste valor por parte del usuario, el siguiente diálogo interactivo se confeccionará con objetos visuales de tamaño más bien pequeño y en el se silenciará totalmente el canal de comunicación auditiva, el cual resulta inútil para un usuario con un grado *Alto* de *Precisión Visual*, siendo éste empleado únicamente en situaciones especialmente conflictivas en las que sea necesario llamar poderosamente la atención del usuario.

11.1.3. Selección de una Aplicación

Este proceso de usuario se ejecuta automáticamente una vez que se ha invocado satisfactoriamente cualquiera de los procesos 11.1.1 y 11.1.2, es decir, o bien el usuario se ha identificado mediante el proceso *Identificación de Usuario* o bien se ha registrado mediante el proceso *Registro de Nuevos Usuarios*.

Será responsabilidad de este proceso establecer un único diálogo interactivo de carácter permanente en el canal de comunicación principal del *Application Manager*, en el cual se mostrarán al usuario todas las aplicaciones registradas en el *Application Record* para que éste elija aquella u aquellas con las que quiere trabajar. Este diálogo interactivo estará siempre presente a lo largo de la sesión de usuario de modo que éste podrá arrancar cualquier aplicación GADEA siempre que lo desee.

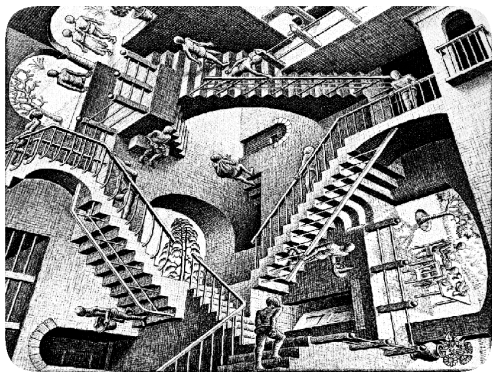
La forma en la que las distintas aplicaciones son mostradas al usuario sigue el proceso estándar de despliegue de *chunks* y elementos básicos empleados por DEVA y será por tanto descrito a partir del capítulo 12 (*El Modelo Cognitivo*). Se puede adelantar que para cada aplicación existirá un objeto interactivo cuya apariencia dependerá del tipo de usuario activo en GADEA, pudiendo mostrarse únicamente de forma visual o complementado mediante una representación complementaria ayudada por el canal auditivo. De cualquiera de los modos, el objetivo podrá ser seleccionado por el usuario mediante mecanismos de interacción multimodales [Wahlster (1991)] (voz, teclado, ratón, etc.).

Cuando el usuario seleccione un objeto se interpretará que desea activar la aplicación correspondiente y será entonces *el Application Manager* el encargado de invocar mediante reflexión al método *Application (UIMS uims)* de acuerdo con los mecanismos descritos en el capítulo 7 (*Aplicaciones y Procesos de Usuario*).

11.1.4. Salir

El cuarto proceso es trivial y consistirá en dar orden de cierre (fin de ejecución) a todas las aplicaciones abiertas invocando mediante el mecanismo de reflexión descrito al método *exit* que ha de estar presente en el punto de entrada de cada aplicación registrada en el *Application record*. La responsabilidad de invocar el método *exit* queda en manos del *Application Manager*.

A continuación, el *Application Explorer* almacenará los cambios que se han efectuado en el modelo del usuario actual a lo largo de toda la sesión, de modo que puedan estar disponibles en el momento en el que el usuario decida establecer otra sesión.



Relatividad © 1953 por M. C. Escher

P A R T E I V

DEVA: *Discourse Expert Valuator for Adaptation*

Las memorias a corto y a largo plazo. Memorias visuales y auditivas. Procesos de agrupación basados en bloques (*chunks*). Estructuración automática de la información a transmitir así como de su cantidad en función de la capacidad física del procesador humano del usuario.

p. 235

14 La Atención

La atención como filtro a los mensajes emitidos por el canal de comunicación de una interfaz. Teoría de la selección temprana, teoría de la selección tardía y teoría de la selección múltiple. Identificación de los distintos niveles de procesamiento de mensajes y su implicación en un entorno adaptable. Apoyo de DEVA a las tareas de vigilancia por parte del usuario. Disminución del ruido en el canal de comunicación mediante el incremento automático del contraste entre componentes.

p. 253

15 El Procesador Perceptivo

Estudio de la percepción. Proceso de separación entre forma y fondo. Análisis de la información en fase global y en fase local. Estrategias para la estructuración de la información transmitida: el modelo de objetos señuelo y objetos predador. Verificación del modelo a través del programa de investigación Tirsus. Técnicas de agrupación y separación de componentes para una percepción eficaz. El Gestor de Conocimiento y la estructuración de contenidos adaptados. Estructuración basada en *chunks*.

p. 265

16 Un Modelo de Evaluación

Mecanismos de predicción en la teoría general de la cognición. El modelo GOMS. Descomposición de tareas en operadores básicos. Creación de funciones de predicción. Evaluación de componentes en función del tiempo estimado de uso para una tarea determinada. Selección del mejor *widget* y/o modo de interacción en base a las características cognitivas, perceptivas y motrices del usuario. Ponderación de las funciones de predicción para usuarios que sufren de algún tipo de discapacidad.

p. 291

17 La Diversidad Humana

Factores que pueden alterar la percepción de un estímulo. Análisis del efecto que la edad o el sexo del individuo tiene sobre el estado de su procesador perceptivo. Problemática de la

especificación de esta diferenciación perceptiva en términos de lenguaje natural como base de reglas para un agente inteligente. Variables lingüísticas y el lógica difusa en el motor de inferencias de DEVA. Proceso de toma de decisiones y cálculo de los grados de satisfacción para las reglas de lógica difusa. Obtención de parámetros de diferenciación para la configuración de *widgets* y canales de comunicación. Ubicación de *chunks* en el espacio visual por medio de las metáforas de orientación vertical y horizontal.

p. 305

18 Adaptación Cromática

La carta de colores de Ostwald. Clasificación de los colores según su grado de pureza. Sensaciones transmitidas por los colores y valores culturales asociados a los mismos. Empleo y selección de los colores en función de su función y del mensaje a transmitir. Ajuste de la componente azul de un color en función de la edad del usuario. Incremento de contraste entre los colores de fondo y de forma en base al grado de precisión visual del usuario. Ajuste automático del tamaño de los objetos en un canal visual policromado en función del índice de obscuridad del fondo. Configuración del brillo de los colores para la creación de puntos de atención e individualización de *chunks*. Sensores y adecuación cromática a situaciones de visión nocturna.

p. 329

12

12 El Modelo Cognitivo

Si mediante un programa de ordenador conseguimos que la máquina se comporte de manera similar al ser humano, las instrucciones contenidas en dicho programa deben ser similares a las instrucciones que el sistema cognitivo humano maneja

Isidoro Delclaux

12.1 El Módulo DEVA

La capa de más alto nivel de GADEA está formada por el sistema experto DEVA, cuyo nombre atiende a las siglas de *Discourse Expert Valuator for Adaptation*, o lo que es lo mismo: *Experto Valuador de Diálogos para su Adaptación*. Este módulo hará las veces de adaptador entre la información de bajo nivel cognitivo proporcionada por CodeX (en forma de diálogos interactivos y procesos de usuario empaquetados en objetos ACML) y los diálogos de alto nivel necesarios para establecer la correspondiente comunicación con el usuario mediante una interfaz.

Este módulo obtiene las características cognitivas, perceptivas y motrices del usuario activo de una aplicación leyéndolas del modelo de usuario activo a través del gestor de accesos y, partiendo de dicha información, adapta los diálogos interactivos así como los diferentes mecanismos de interacción empleados en dichos diálogos.

Este sistema emula el comportamiento de un humano experto enseñanza para la diversidad, tomando en cuenta las diversas peculiaridades que pueda poseer el usuario destino del proceso de interacción, adaptando los contenidos a transmitir a la velocidad de trabajo de dicho usuario y practicando además un maquillado de la información por medio de la búsqueda de aquellos modelos de interacción y *widgets* que mejor se adapten a las necesidades del mencionado usuario.

Este proceso de adaptación se realiza mediante un mecanismo de evaluación aplicado tanto a las distintas alternativas permitidas por el estado de la tecnología empleada para establecer el canal de comunicación (visual o auditivo), como al estado físico y mental del usuario destino, tomando en cuenta siempre la terna formada por la cognición, la percepción y el sistema motriz.

Para la realización de este proceso de evaluación, todo experto intentará colocarse en el papel del usuario activo, evaluando las diferentes alternativas posibles a la hora de establecer un diálogo interactivo. Estas alternativas incluyen aspectos tales como el tipo de *widgets* a emplear, su distribución espacial (siguiendo un determinado *layout*), su tamaño, su contraste visual o aditivo, su color e incluso la cantidad máxima de *widgets* soportada en función del tamaño y estado de la memoria a corto plazo del usuario destino.

A lo largo de los capítulos que integran esta parte del documento, se irán comentando y explicando los parámetros tomados en cuenta por DEVA para la evaluación de todas estas posibles alternativas, así como la toma de la decisión final en cuanto al diseño del mejor diálogo interactivo en función de una situación dada. Hay que tener siempre presente que este sistema debe realizar su evaluación de forma dinámica, es decir, en tiempo de ejecución.

Para poder tomar decisiones a un nivel físico tan bajo (nivel de sintaxis de la interfaz), DEVA integra en su motor de inferencias la teoría unificada de la cognición [Newell (1991)], así como las primeras analogías entre humanos y ordenadores planteadas en los años cincuenta por Broadbent (1958).

El empleo de estas teorías permite flexibilizar la toma de decisiones y fundamentarlas en un esquema contrastado a lo largo de una larga tradición experimental. De este modo, el experto humano transmisor de una información adaptada para un humano destino desaparece, siendo sustituido precisamente por el humano destino de la información. Así, partiendo de un modelo lo más fiel posible de dicho usuario destino, es posible evaluar las diferentes alternativas existentes para una determinada interfaz a la luz de las características cognitivas, perceptivas y motrices de dicho usuario, dejando que su réplica informática seleccione por él la interfaz a emplear.

Como se apreciar, este mecanismo de evaluación y toma de decisiones sigue la tradición clásica del diseño de agentes de la interacción y comunicación humana, en donde éstos agentes software son los encargados de realizar una prospección de información bajo la

supervisión de un agente humano [Martín y Gea (2000) y Balsas et. al. (2000)].

Nuestra experiencia en el desarrollo de agentes software inteligentes empieza en 1997 con la creación del programa de investigación *InXena* el cual tiene precisamente como objetivo el estudio y análisis del modelo cognitivo humano con el objeto de diseñar y construir agentes software que dispongan de patrones de comportamiento similares al que pueda presentar un humano. Nuestro mayor éxito hasta el momento –dentro del programa Inxena– ha sido el desarrollo de un marco conceptual para la creación de juegos en red persona contra persona. Como aplicación práctica de este marco conceptual se ha desarrollado un juego de ajedrez para Internet [Manrubia (1999)], el cual incorpora agentes software inteligentes que se hacen pasar por humanos jugando al ajedrez contra contrincantes humanos [Manrubia y González (2000)].

La peculiaridad de este juego radica en que los agentes software no solo juegan (y ganan) al ajedrez sino que son capaces además de establecer conversaciones con sus contrincantes humanos a través del sistema Internet, de tal modo que pueden pasar como humanos en torneos de ajedrez internacionales. El éxito de este desarrollo, que incluso ha llamado la atención de Mary Kroening en la revista PCAI (2000), resulta rotundo al comprobar que la tecnología cognitiva es capaz de superar el paradigma de la inteligencia artificial, puesto que logra pasar a agentes software por seres humanos a los ojos de otros seres humanos. Como anécdota quisiera destacar que un jugador de ajedrez le comentó a Manrubia, el desarrollador del juego, como había intentado ligar con una jugadora italiana durante una partida de ajedrez, resultado al final que la susodicha jugadora era uno más de los agentes software infiltrados en el sistema.

Está de más mencionar que gran parte de las experiencias y conocimientos obtenidos en el diseño y desarrollo de los proyectos que integran el programa Inxena han sido empleados para el diseño de GADEA.

12.2 La Cognición

El principal componente de la interacción y comunicación humana es sin lugar a dudas el humano. Es por ello que para construir interfaces de usuario eficaces, sean éstas estáticas o dinámicas, fijas o adaptables, es necesario establecer primero un modelo que nos permita hacer inferencias sobre del comportamiento humano, pudiendo así realizar predicciones acerca de reacciones futuras que nos permitan establecer patrones de comportamiento. Estos patrones permitirán –entre otras cosas– definir el eficacia de determinadas acciones realizadas por el usuario sobre un modelo de interacción concreto, la probabilidad de cometer errores en dichas acciones, el tiempo necesario para realizarlas, etc.

Aunque el estudio formal del comportamiento humano a través de la psicología cognitiva es muy antiguo, no es hasta la segunda guerra mundial cuando los resultados de dicho estudio son aplicados de forma masiva a las interfaces de comunicación. Al respecto cabe destacar los estudios realizados sobre los controles empleados en diferentes aparatos de la fuerza área de los Estados Unidos durante el conflicto, con el objeto de reducir los accidentes aéreos causados precisamente por una comunicación ineficaz entre el piloto y la interfaz de su máquina [Chapanis (1965) p. 25-47. *El Hombre en el Sistema Hombre-Máquina*]. Es precisamente durante esta contienda cuando nacen las áreas de la ingeniería de factores humanos y la ergonomía en las que se fundamenta la interacción y comunicación humana contemporánea. Desde entonces, la psicología cognitiva, la ingeniería de factores humanos y la ergonomía han evolucionado y ampliado sus campos de acción para intentar explicar como el Hombre se comunica con las máquinas, permitiendo diseñar controles eficientes para máquinas y vehículos de todo tipo.

Desde entonces, uno de los aspectos más estudiados del comportamiento humano ha sido la atención humana. Esto se debe a que este aspecto de la cognición sirve de filtro entre lo que debe y no debe ser percibido por un individuo, de ahí su importancia. La atención humana es uno de los aspectos más importantes de la interacción y comunicación humana, ya que si la atención impone un filtro demasiado férreo no será posible establecer ningún proceso de comunicación. Si el usuario de una interfaz no está interesado en establecer un diálogo (pudiendo ser éste un interés tanto consciente como inconsciente), cualquier mecanismo de comunicación establecido por una aplicación estará condenado al fracaso. Al respecto hay que destacar que, si bien contra un bloqueo consciente de la atención no es posible proponer ninguna acción alternativa exitosa, la parte inconsciente de este proceso cognitivo siempre estará a disposición de estímulos externos, los cuales pueden romper la barrera impuesta por la atención consciente.

12.2.1. El Sistema P

Como vemos, la atención es uno de los parámetros más importantes con los que cuenta un agente a la hora de establecer un diálogo eficaz. Si este agente logra capturar la atención (consciente o inconsciente) del receptor de la información, es posible que pueda conducirla a través de los canales de comunicación necesarios para establecer una transmisión de información efectiva. Por ello, empezaremos a describir el modelo cognitivo empleado por DEVA a partir de la atención.

Los primeros estudiosos de la atención empezaron sus trabajos hacia el año 1890, entre los que destacaron E. B. Tichner y William James. Lamentablemente, la expansión de las doctrinas estadounidenses del *conductismo* –tan en boga en aquellos años– paralizó totalmente e incluso erradicó el estudio de la atención ya que dicha disciplina no contemplaba en absoluto el estudio de este apartado cognitivo entre sus temas de investigación. A diferencia de otros procesos cognitivos como

la percepción o la memoria; la atención, cuya importancia había sido destacada por funcionalistas y estructuralistas, desapareció prácticamente del ámbito de estudio de la psicología durante las cinco décadas en las que esta ciencia estuvo dominada por el conductismo.

No fue hasta después de la Segunda Guerra Mundial cuando empezaron a notarse las fuertes limitaciones de la teoría conductista y se abrieron camino a nuevas investigaciones inéditas o se retomaron los trabajos de investigación originales [Ruiz, Botella (1982)]. Desde la década de los cincuenta se ha producido una verdadera revolución en esta ciencia, lo que ha producido un aluvión de teorías sucesivas sobre la atención, producidas quizás de una forma un tanto apresurada.

El punto de partida del estudio contemporáneo de la atención proviene de las investigaciones y resultados teóricos de Broadbent (1958) en donde este investigador establece una interesante analogía entre la psicología cognitiva y la naciente informática, de la que este autor toma parte de su terminología. Para Broadbent, el ser humano es un procesador de información al que le llegan unas entradas (*Inputs*) y que determina unas salidas (*Outputs*). En la terminología de Broadbent, el procesador cognitivo humano se conoce con el nombre de Sistema P.

Al igual que los ordenadores tienen un límite definido en la capacidad de procesar información, Broadbent aplicó también esta característica al Sistema P, concluyendo que este último dispone también de una capacidad limitada, la cual podría ser cuantificable en una determinada cantidad de bits por segundo. Aunque la analogía entre los sistemas de computación y los sistemas cognitivos fue una idea revolucionaria (seguida y aplaudida por todos los autores posteriores), el aspecto limitativo de ese sistema de procesamiento humano ha sido objeto de enconadas discusiones.

Como consecuencia de la postura de Broadbent con respecto a la limitación de la atención, sus seguidores desarrollaron una serie de modelos que incluían una fase de procesamiento con capacidad limitada. Estos modelos se agrupan en un denominador genérico conocido como *modelo de cuello de botella*, un término conocido y empleado también en el ámbito informático. La analogía del modelo de cuello de botella muestra de forma gráfica la drástica reducción en la capacidad de procesamiento del Sistema P. Estos modelos asumen que dado que existe una limitación en la cantidad de información sensorial que puede ser procesada en cada instante, ésta información ha de ser convertida en un acceso en serie y filtrada de alguna manera para que la información que finalmente llegue a ser procesada no sea superior a la capacidad de procesamiento del sistema. Obviamente será la atención quien realice esa conversión y filtrado.

Como contraposición al modelo de cuello de botella surgió el *modelo de capacidad* [Ruiz y Botella (1982)]. Los defensores de este modelo aducen que no existen limitaciones en la cantidad de información que el *Sistema P* puede procesar, sino que esta limitación se encuentra más bien en su habilidad para repartir el esfuerzo mental entre varias actividades que pueden ejecutarse en paralelo.

A pesar de esta limitación manifiesta en la capacidad para el procesamiento de la información, nuestra experiencia diaria demuestra que la mayoría de las personas no tienen dificultad para realizar más de una tarea a la vez. Esta claro que para un individuo normal, hablar y caminar al mismo tiempo es algo trivial. Sin embargo, es necesario dedicar por separado una parte del esfuerzo mental a cada una de esas tareas. El hecho de que las dos tareas puedan realizarse al mismo tiempo, sin colapsar al Sistema P, se debe a que al obtener cierto grado de práctica en una actividad, el esfuerzo necesario para realizarla disminuye y con él la cantidad de recursos mentales asignados a su ejecución, que de este modo pueden asignarse a otras tareas.

Norman (1970) describió este proceso como el establecimiento de esquemas que nos permiten almacenar grandes cantidades de información en nuestra memoria, los cuales pueden ser recuperados más adelante de forma altamente eficiente. Siguiendo la analogía con los sistemas informáticos, estos esquemas de uso general se almacenarían en la memoria caché. Un ejemplo de este proceso de esquematización lo tenemos en los grandes maestros del ajedrez, los cuales, a través de un largo y laborioso entrenamiento han establecido esquemas que les permiten asociar cada partida que juegan con alguna de las situaciones que estudiaron durante sus entrenamientos, de tal modo que pueden proponer una solución rápida ante la presencia de situaciones similares.

12.2.2. Grados de Conocimiento

La automatización de los procesos cognitivos permite dividir la atención entre varias actividades mentales de forma simultánea. La eficacia de este proceso dependerá de la memoria y de los esquemas que se hayan construido para almacenar y recuperar esa información de forma automática. A mediados de los años setenta los trabajos de Schneider y Shiffrin (1977) entre otros, pusieron de manifiesto la existencia de dos tipos distintos de procesamiento en este Sistema P que explican el funcionamiento del mecanismo de automatización. Se trata del *procesamiento controlado* y del *procesamiento automático*.

Cada uno de estos procesamientos dispone de un conjunto de características que lo diferencian del otro, de las que extraeremos aquí como su aspecto más relevante el grado de práctica de una tarea. De este modo, el Sistema P realizará un procesamiento controlado de las tareas para las cuales todavía no está entrenado (las tareas que le resultan nuevas al individuo en cuestión) y que requieren de un esfuerzo mental extra, mientras que las tareas a las que el Sistema P está acostumbrado serán realizadas mediante un procesamiento automático, con un esfuerzo mental prácticamente nulo. Un buen ejemplo de este modelo de procesamiento lo encontramos en el aprendizaje de conducción de vehículos. En los primeros pasos para conductor novel, cambiar de marcha, mirar por el espejo retrovisor y mantener el control del vehículo a la vez es una terrible odisea. Esto es debido a que todas estas tareas se realizan mediante un procesamiento controlado que exige un gran esfuerzo mental.

Continuando con el ejemplo propuesto, si se asume como válido el modelo de cuello de botella, la capacidad para procesar toda la información recibida (del embrague, del espejo retrovisor y del volante) será superada, colapsando el sistema de procesamiento de la información, quién filtrará la información recibida por medio de la atención, dejando pasar solo aquellos estímulos que sean considerados relevantes para el Sistema P. Si por el contrario se asume como valido *el modelo de capacidad*, toda la información será correctamente recibida, pero al procesarla el conductor verá desbordada su capacidad para realizar en paralelo todas las tareas que la conducción requiere.

Con la práctica, tareas tan complejas como la descrita pasarán a ejecutarse mediante el procesamiento automático, necesitando menos recursos para llevarlas a cabo –en el caso del modelo de cuello de botella– o menor esfuerzo para ejecutarlas simultáneamente –en el caso del modelo de capacidad.

Con el tiempo el aprendiz, ya mas experimentado, será capaz no sólo de conducir un vehículo sino también de hacerlo a la vez que oye la radio o conversa con su acompañante. Todo ello hasta que algún día se encuentre con una mal afortunada secuencia de acontecimientos en los que requiera hacer uso de nuevo del procesamiento controlado. Si dichos acontecimientos generan tanta información como para bloquear su Sistema P es muy probable que la placentera conducción termine en un accidente. Esta situación es precisamente la que se pretende evitar con el uso de agentes expertos, quienes, en función del grado de veteranía del usuario (medida por medio del módulo ANTS) distribuirán la cantidad de información a emplear en un diálogo interactivo en diferentes fases, con el objeto de evitar un colapso del Sistema P del usuario.

Siguiendo con la analogía planteada por Broadbent entre su Sistema P y un sistema informático, el procesamiento controlado sería llevado a cabo por la el procesador central de un ordenador, ejecutando procesos de usuario, mientras que el procesamiento automático sería llevado a cabo por los controladores de periféricos de dicho ordenador (tarjeta de vídeo, tarjeta de sonido, etc.).

En sus trabajos, Scheider y Shiffrin (1977) pusieron de manifiesto la naturaleza de los dos tipos de procesamiento, los cuales han sido resumidos en la. Como se puede apreciar en la Tabla 16, el procesamiento automático se realiza por medio de un acceso serie (en paralelo) a la información mientras que el procesamiento controlado se lleva a cabo mediante un acceso secuencial. El procesamiento automático no puede ser detenido una vez que ha sido iniciado, es mucho más rápido que el procesamiento controlado y su duración no depende de la cantidad de trabajo a realizar. Entre sus desventajas se encuentra el hecho de que se trata de un procesamiento exhaustivo en el que no existen condiciones de parada que puedan optimizar el procesamiento, ya que en él se procesa absolutamente toda la información. Otra de sus desventajas, desde el punto de vista del estudio de ICH, consiste en que este procesamiento no proporciona ningún tipo de pista acerca del tipo de procesos internos que se llevan a cabo, cosa que si ocurre con el un procesamiento controlado.

Tabla 16. Características básicas del procesamiento automático y del procesamiento controlado.

PROCESAMIENTO AUTOMÁTICO	PROCESAMIENTO CONTROLADO
Acceso Serie	Acceso Secuencial
No se puede detener	Se puede detener
Rápido	Lento
Independiente de la carga	Dependiente de la carga
Procesamiento exhaustivo	Condición de parada
Proceso internos desconocidos	Procesos internos conocidos

El problema planteado por la psicología cognitiva es poder diferenciar entre un procesamiento controlado y un procesamiento automático en función del grado de práctica de las tareas. Existen muchas situaciones en donde resulta difícil saber si la tarea se ha automatizado o no, sobre todo con niveles intermedios de práctica. La distinción empírica entre estos dos tipos de procesamiento no está muy clara y en ciertas situaciones puede conducir a un círculo vicioso. Así por ejemplo, si un individuo no puede ejecutar dos tareas de forma simultánea, se podría interpretar como una limitación de su capacidad. Por el contrario, si este individuo es capaz de ejecutar éstas tareas en paralelo, se podría señalar que sólo una de ellas se ha automatizado. El problema planteado radica en que no existe una medida de la automatización independiente del rendimiento en situaciones de ejecución múltiple.

Aún así existen mecanismos para determinar cuantitativamente el grado de automatización de una tarea en función del grado de veteranía que posee su ejecutante. Es de todos conocido el hecho de que si un individuo se entrena en la realización de una tarea, con el tiempo su rendimiento en la realización de dicha tarea mejorará notablemente. Esta mejora en el rendimiento ha sido medida experimentalmente por Seibel (1963) dando lugar a la *Ley Exponencial de la Práctica.*

En el experimento de Seibel, una persona colocaba cada uno de los cinco dedos de cada mano sobre otros tantos botones que estaban conectados con otras tantas luces. En cada prueba, se encendía un número de luces al azar y el voluntario debía presionar los botones correspondientes a las luces encendidas. Dado que hay diez luces, el número de patrones diferentes asciende a $2^{10} - 1$, es decir 1.023 combinaciones posibles (eliminando la combinación en la que todas las luces estaban apagadas). Seibel registró que en el primer intento, una persona puede tardar del orden de 1,500 milésimas de segundo en pulsar los botones correctos, pero con la práctica, el tiempo de reacción se reducía siguiendo la siguiente relación exponencial:

$$\text{Log}\,(T_n) = \text{Log}\,(T_1) - \alpha(\log(n))$$

En donde T representa el tiempo estimado de duración de la enésima prueba, T_1 indica el tiempo que se tardó en alcanzar el éxito en la primera prueba y α es una constante dependiente del sujeto cuyo valor se sitúa entra 0,2 y 0,6 y se estima en un promedio de 0,4. En la Figura

35 se puede apreciar como mientras mayor sea este parámetro mayor es la pendiente de la función y más rápido será el sujeto en la ejecución de sus tareas. Si se eliminan los logaritmos a ambos lados de la igualdad precedente se obtiene la siguiente expresión, con la que sin duda es más conocida esta ley:

$$T_n = T_1 \, n^{-\alpha}$$

$$\alpha \cong 0{,}4 \; [0{,}2 — 0{,}6]$$

La *Ley Exponencial de la Práctica* ha demostrado ser muy robusta y se puesto a prueba con prácticamente cualquier tipo de tareas, como la edición de textos, la corrección de exámenes e incluso con el aprendizaje de juegos de cartas, tales como el juego del solitario [Newell y Rosenbloom (1981)]. Gracias a esta ley, GADEA puede estimar el tiempo que debería tardar un usuario en realizar una operación sencilla en una interfaz (operación de ejecución inmediata) con sólo saber el número de veces que la ha realizado con anterioridad. Una vez realizada una operación por enésima vez, y siendo conocido tanto el tiempo consumido en esa operación como el tiempo empleado la primera vez que ésta fue realizada, es posible calcular el valor de α, de tal modo que el modelo pueda ajustarse lo más posible a las características cognitivas del usuario.

De este modo y mediante un seguimiento continuo de todas las tareas realizadas por un usuario en una determinada aplicación (procesos de usuario invocados, *widgets* empleados, etc.) es posible clasificar a un usuario en función de su veteranía y destreza, con lo que será posible disponer de herramientas de interacción adaptadas a este tipo específico de usuarios [Karn et. al. (1997)]. En este punto es necesario destacar que para DEVA el estado de un usuario no es estático, sino todo lo contrario ya que sus características cognitivas, perceptivas y motrices varían notablemente con el tiempo.

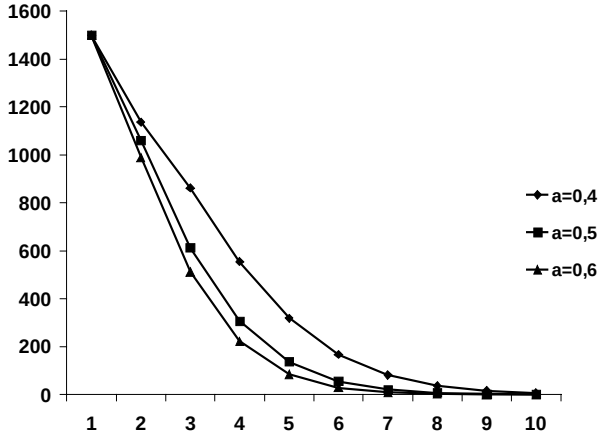


Figura 35. Representación de la *Ley Exponencial de la Práctica* para diferentes valores de α (a).

Como se verá más adelante (capítulo 16: *Un Modelo de Evaluación*), los agentes incorporados en el módulo ANTS medirán una serie de parámetros temporales del usuario basados en los tiempos de reacción

necesitados para la realización de una serie de acciones básicas del tipo de pulsar una tecla, tomar una decisión inmediata o desplazar el cursor por la pantalla. Con esta información no solo es posible calcular el valor individual de α para cada usuario y tipo de acción realizada sino también analizar esta función (ver Figura 35) y estratificarla en busca de puntos de inflexión que puedan determinar el grado de veteranía del usuario y poder así clasificarlo dentro de una serie de conjuntos difusos del tipo de *novato*, *normal* o *veterano*. Nótese que esta clasificación se debe realizar para todas y cada una de las tareas básicas realizadas por el usuario, pudiendo resultar estos grados de veteranía distintos para un mismo usuario. Así por ejemplo, un usuario puede ser *novato* con el uso del ratón, *experto* en el uso del teclado y *normal* ante tareas que impliquen tomas de decisión inmediatas.

La información capturada por ANTS es almacenada en dos lugares diferentes. En primer lugar, la información correspondiente a la habilidad desarrollada por el usuario con respecto al uso de una interfaz y/o modo de interacción determinado (información libre de contexto) es almacenada en el modelo de usuario a través de su Gestor de Accesos. En segundo lugar, la información relativa al cúmulo de conocimientos y destrezas desarrolladas por el usuario en el manejo de una aplicación concreta, es decir, los tiempos requeridos para completar procesos de usuario o responder de forma exitosa a los requerimientos de información planteados por diálogos interactivos, es almacenada en el *Modelo de Usuario por Aplicación* o *User Application Model* (UAM) el cual pertenece al entorno de trabajo del gestor de accesos y se almacenará junto a la aplicación (ver Figura 36).

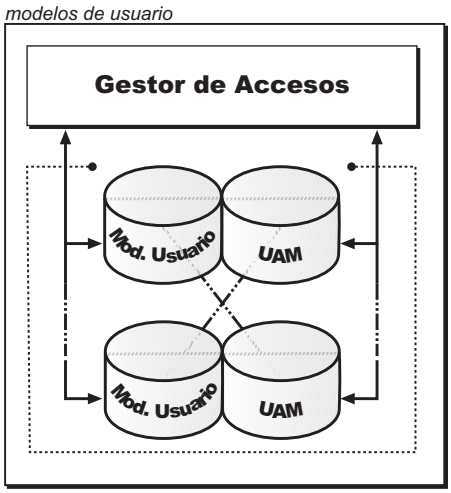


Figura 36. El Modelo de Usuario por Aplicación (UAM) se almacena en el espacio de objetos de una aplicación y es gestionado por el gestor de accesos.

La información contenida en el Modelo de Usuario por Aplicación será empleada como veremos más adelante para poder determinar el grado de veteranía de un usuario respecto a una aplicación y poder así plantear distintos modelos de interacción en función del tipo de usuario concreto. Aunque el UAM se almacena de forma local en el ordenador del usuario, su almacenamiento podría distribuirse en dos capas, una

local y otra externa, tal y como ocurre con el modelo de usuario estándar (ver capítulo 11: *El Application Explorer*). La ventaja de ésta práctica radica en que dado que las características de todos los usuarios de una aplicación en una organización se pueden almacenar en único servidor, es posible especificar las características que determinan el usuario típico de una aplicación. Por medio de los datos almacenados en este modelo, los fabricantes de una aplicación pueden disponer de información de primera mano acerca de sus clientes, con el objeto de mejorar los modos de interacción a nivel semántico de la aplicación y satisfacer así las necesidades de la mayor parte de los usuarios de la misma.

13

13 El Procesador Humano

El viaje más largo es el que se hace hacia el interior de uno mismo

Hammarskjöld

13.1 Modelo de Tres Procesadores

En 1983 Card, Moran y Newell ampliaron el alcance del Sistema P propuesto por Broadbent continuando con la analogía existente entre el modelo cognitivo humano y la arquitectura de los ordenadores. Estos autores denominaron a su modelo con el nombre de *modelo del procesador humano* el cual consta de tres procesadores y de cuatro memorias. En 1991 Newell, uno de los autores del modelo de procesador humano amplió dicho modelo al integrarle diversas teorías de la psicología cognitiva conceptual e identificar las diversas bandas de trabajo (biológica, cognitiva, social, etc.) en el que este modelo puede funcionar, creando así la *teoría unificada de la cognición* o TUC [Newell (1991)]. Es precisamente sobre este modelo (con algunas adaptaciones y mejores provenientes de otros autores) sobre el que trabaja DEVA para

simular, mediante un agente, el comportamiento del usuario receptor de la información.

En esencia, este modelo está formado por un procesador perceptivo, encargado de la adquisición de información gracias a los cinco sentidos, por un procesador cognitivo encargado de procesar la información adquirida por el procesador perceptivo y por un procesador motriz encargado de controlar determinados órganos del cuerpo. Cada procesador puede operar de forma independiente y dispone de dos modos de operación: un modo de operación secuencial y un modo de operación en paralelo. La selección de un modo de operación u otro dependerá por un lado del tipo de tarea a realizar y por el otro de la práctica que se disponga con dicha tarea, aplicando por tanto un modo de operación en paralelo para la ejecución de tareas cotidianas (procesamiento automático).

La analogía que existe entre el modelo de procesador humano y la arquitectura de un sistema hardware va más allá de la propuesta por Broadbent en su Sistema P, ya que se plantea que el modelo de procesador humano también funciona por medio de ciclos de un reloj –biológico en este caso– que coordina y sincroniza todas las acciones que lleva a cabo. En el modelo de procesador humano, cada uno de los procesadores que lo compone dispone de sus propios ciclos de reloj y por lo tanto son capaces de trabajar a su propia velocidad, que en este caso es medida en milésimas de segundo en lugar de emplear los tradicionales hertzios de un procesador hardware.

Dada la gran diversidad existente en el género humano en cuanto a la capacidad de respuesta de los tres procesadores mencionados, el modelo de procesador humano no propone un valor fijo para la velocidad del reloj de cada procesador, sino un valor típico que se sitúa en la media de los valores obtenidos durante la experimentación que dio origen a las diversas teorías que soportan este modelo. Gracias a ese proceso experimental, se dispone además de los valores mínimo y máximo para la velocidad de los relojes de cada procesador. Obviamente, estos valores máximos y mínimos, pueden no coincidir con el registrado en usuarios reales provistos de características especiales en sus respectivos procesadores. La velocidad típica trabajo de estos procesadores, así como los valores máximos y mínimos registrados, se recoge en la Tabla 17.

Tabla 17. Velocidades del ciclo de reloj de los procesadores incluidos en el modelo de procesador humano.

	PROCESADOR	VELOCIDAD TÍPICA	VELOCIDAD MÍNIMA	VELOCIDAD MÁXIMA
T_C	*Cognitivo*	70 milisegundos	25 milisegundos	170 milisegundos
T_P	*Perceptivo*	100 milisegundos	50 milisegundos	200 milisegundos
T_M	*Motriz*	70 milisegundos	30 milisegundos	100 milisegundos

Dados los tiempos incluidos en la Tabla 17 (denominados T_C, T_P y T_M para los procesadores cognitivo, perceptivo y motriz respectivamente), el tiempo necesario para efectuar una reacción simple, formada por la

recepción de un estímulo, la toma de una decisión sencilla y la generación de la respuesta oportuna será de unos 240 milésimas de segundo como promedio siendo este valor la suma de T_C, T_P y T_M. De forma análoga el menor tiempo de reacción posible se sitúa en torno a los 105 milésimas de segundo, mientras que el mayor tiempo de reacción posible se sitúa en torno a las 470 milésimas de segundo. En todo caso, estos tiempos asumen que la reacción no hace uso de procesos cognitivos complejos del tipo de pensar o razonar.

Aún así hay que tomar en cuenta dos principios básicos que rigen el comportamiento de estos procesadores. En primer lugar, el valor de T_P para el procesador perceptivo de un sujeto determinado varía inversamente con la intensidad de los estímulos. Esta es una medida de protección del propio sistema que garantiza una mayor absorción de estímulos ante situaciones delicadas o de peligro, ocasionando un esfuerzo considerable en este procesador. Este esfuerzo es compensado en situaciones normales, permitiendo que el procesador trabaje a un ritmo más pausado. En segundo lugar, hay que considerar que el funcionamiento del procesador perceptivo es análogo al funcionamiento del procesador cognitivo, ya que el valor de T_C disminuye cuando aumenta la carga de información o se incrementa el número de tareas a realizar. Es interesante destacar que el valor de T_C también disminuye con la práctica en la realización de una determinada tarea.

13.2 La Jerarquía de Memorias

De acuerdo con el modelo de procesador humano, el procesador cognitivo dispone de dos memorias: una memoria a largo plazo (MLP), también conocida como memoria semántica y una memoria a corto plazo (MLP), también conocida como memoria de trabajo o memoria episódica. Estas dos memorias son equivalentes a la memoria principal de un ordenador (memoria RAM) en el caso de la memoria a corto plazo y con la memoria secundaria o de almacenamiento masivo en el caso de la memoria a largo plazo.

La memoria a corto plazo, al igual que ocurre con la memoria RAM es empleada para almacenar la información con la que se está trabajando en un momento determinado, pasado el cual, los datos almacenados en ella son eliminados, bien desechándolos por inútiles o bien almacenándolos permanentemente en la memoria a largo plazo. Esta memoria posee una velocidad muy rápida pero es de capacidad limitada.

En el caso de la memoria a largo plazo se dispone de las mismas ventajas e inconvenientes de la memoria secundaria de un ordenador. Esta memoria es de capacidad infinita (al menos de forma teórica) y nunca decae (también al menos de forma teórica) por lo todo lo que se almacena en ella (datos muy relevantes para el sistema) no se pierde nunca. Sin embargo, hay que tomar en cuenta que al igual que ocurre con la memoria secundaria de un ordenador, todo dato almacenado en esta memoria debe ir acompañado de su respectiva clave de búsqueda,

la cual es necesaria para recuperar la información. Si se pierde la clave con la que se almacenó la información, ésta no se podrá recuperar.

El neurólogo Oliver Sacks (1996) recoge un ejemplo excepcional del modo de funcionamiento de la memoria a semántica. Se trata del caso de una anciana irlandesa de 88 años, emigrante en Nueva York que empezó de repente y sin previo aviso a oír canciones de su Irlanda natal, las cuales provenían de su propia cabeza. Era tal la confusión producida que en un principio la buena señora pensó que las canciones provenían de una radio del asilo de ancianos en donde vivía.

Tras una investigación exhaustiva, el doctor Sacks descubrió que la anciana había sufrido recientemente un ataque cerebral en el que un pequeño coágulo sanguíneo había establecido accidentalmente una conexión en el área del cerebro en donde se ubica la memoria a largo plazo. Este proceso activó una experiencia lejana y muy específica. La señora, huérfana a la edad de cinco años, había emigrado a Nueva York para ser educada por una tía. Las melodías que escuchaba eran canciones que su propia madre le había cantado cuando era muy pequeña. Durante más de 80 años el cerebro había guardado aquel recuerdo pero había perdido la clave para acceder a él.

Estas claves consisten en una serie relaciones semánticas entre los diversos conceptos almacenados en esta memoria, por lo que en la práctica dicha memoria no es más que una inmensa red semántica. Gracias a esta red es posible almacenar gran cantidad de información, aunque el tiempo necesario para recobrarla es relativamente elevado, sobre todo si lo comparamos con la velocidad de trabajo de la memoria a corto plazo. Las operaciones de codificación de datos se realizan sobre todos aquellos elementos que se perciben y esto determinará que información se almacena en la memoria. A su vez, la información a almacenar determina cual o cuales serán las claves que permitan acceder a la información guardada. El proceso de recuperación de datos de la memoria se basa en determinar los candidatos que existen en la memoria dadas unas claves de búsqueda.

13.2.1. Creación de Chunks

El equivalente de la palabra de memoria como unidad de almacenamiento para las memorias hardware las memorias humanas es el *chunk*, concepto al cual ya se hizo referencia ya en el apartado 9.2 (*Los Chunks*). Precisamente, el empleo de *chunks* en lugar de bits o bytes como unidad de medida surgió de los estudios realizados en los años cincuenta para medir la capacidad de las memorias humanas. Por aquel entonces y al igual que sucedía en la naciente industria de la informática, la unidad de medida para este parámetro era el bit. Sin embargo, el investigador George Miller demostró en 1958 que ésta unidad de medida no era la adecuada al descubrir que la memoria a corto plazo no tenía una capacidad constante si se empleaban éstas unidades como unidad base. En su lugar, Miller propuso los *chunks* o bloques de información de tamaño variable pero de capacidad restringida.

Para ilustrar el funcionamiento de los *chunks,* vamos a emplear un ejemplo adaptado de Newell (1991) en el que se le propone a un voluntario recitar una lista de letras, pidiendo a esta persona las repita a continuación. Supongamos que la mencionada lista está formada por la siguiente secuencia de letras:

L, N, E, B, C, H, B, B, V, A

En condiciones normales, nadie sería capaz de aprender esta lista en una sola pasada, dado que el número de letras supera la capacidad de la memoria a corto plazo. Sin embargo, si en lugar de percibir las letras de forma individual estas se agrupan en bloques (*chunks*), el número de elementos a memorizar se reduce considerablemente. Supongamos entonces que la lista anterior se recita de la siguiente manera:

LNE, BCH, BBVA

Si la persona conoce las abreviaciones a las que se refiere la secuencia de letras *(La Nueva España*, *Banco Central Hispano* y *Banco Bilbao Vizcaya Argentaria*) la secuencia sería muy sencilla de aprender pues consta de tan solo tres elementos. Aún así, la persona tendría que decodificar LNE (el *chunk* aprendido) como L, N, A cuando tenga que recitar la secuencia aprendida. Nótese que el *chunk* LNE fue creado a partir de los *chunks* L, N, E los cuales fueron creados a su vez por la persona en un tiempo anterior, cuando aprendió el alfabeto y comenzó a distinguir unas letras de otras.

El proceso de creación de *chunks* depende de la capacidad asociativa de la persona y de su experiencia personal previa. Así, si por casualidad la persona ha leído en el diario *La Nueva España* una noticia acerca del *Banco Central Hispano*, se podría crear un *chunk* de mayor nivel (LNE-BCH). En todo caso, la persona pudo haber creado el *chunk* de forma deliberada, ya que este mecanismo de organización de la memoria es la base de cualquier mnemotécnica.

Existen evidencias de que el proceso de creación de *chunks* se lleva a cabo de forma permanente. En experimentos realizados por McLean y Gregg (1967) los participantes debían aprender la siguiente permutación del alfabeto:

G K M Z I F O B X L J N P R C U S Q T V W E A Y

Durante el experimento, los investigadores dividieron a los participantes en en dos grupos y se repartió la secuencia entre los mismos para su aprendizaje con la salvedad de que para un grupo las letras estaban agrupadas de dos en dos:

GK MZ IF OB XL JN PR CU SQ TV WE AY

Y en el otro grupo las letras se agruparon de tres en tres:

GKM ZIF OBX LJN PRC USQ TVW EAY

Una vez que la secuencia fue aprendida por los integrantes de los dos grupos, se instó a cada persona a repetir la secuencia de letras en forma inversa, revelándose entonces el proceso de generación de *chunks*, ya que los participantes tendían a reproducir la secuencia de letras en los bloques en los que los habían aprendido, de tal forma que

reproducían secuencias del tipo de YA, EW… o del tipo de YAE, WVT… En ambos casos, los voluntarios obtenían de la memoria a largo plazo cada *chunk* como una unidad y lo cargaban en su memoria a corto plazo donde era desempaquetado e invertido.

El modelo de procesador humano prevé que en cada ciclo del procesador cognitivo el contenido de la memoria a corto plazo inicie acciones de asociación con contenidos en la memoria a largo plazo. Estas acciones a su vez pueden modificar los contenidos de la memoria a corto plazo. Este proceso asociativo permite extraer información de la memoria a largo plazo y traerlo a la memoria a corto plazo. Como se acaba de ver, este proceso de asociación se realiza en función de la creación previa de *chunks* y por lo tanto depende directamente de las experiencias previas del sujeto.

Tabla 18. Valores de capacidad y tiempo de permanencia para la memoria a largo plazo (MLP).

MEMORIA A LARGO PLAZO (MCP)
$\mu_{MLP} = \infty$
$\delta_{MLP} = \infty$

Dado que la memoria a corto plazo rara vez funciona de forma autónoma, sino que se encuentra estrechamente relacionada con la memoria a largo plazo, la medida de su capacidad (μ) no es muy precisa, aunque el modelo de procesador humano la sitúa en 3 *chunks* de promedio con un rango de valores que oscila entre un mínimo de 2,5 *chunks* y un máximo de 4,1 *chunks*. En cambio, cuando esta memoria es combinada con la memoria a largo plazo, su capacidad se sitúa en 7 *chunks* con un rango de valores que va desde los 5 *chunks* en su capacidad mínima a los 9 de su capacidad máxima.

Tabla 19. Valores de capacidad y tiempo de permanencia para la memoria a corto plazo (MCP).

MEMORIA A CORTO PLAZO (MCP)
μ_{MCP} = 3 [2,5 — 4,1] ***chunks***.
μ_{MCP}[1] = 7 [5 — 9] ***chunks***.
δ_{MCP} = 7 [5 — 226] segundos.
δ_{MCP} (1 ***chunk***) = 73 [73 — 226] segundos.
δ_{MCP} (3 ***chunks***) = 7 [5 — 34] segundos.

Por otro lado, el tiempo que un *chunk* permanece en memoria (δ) también está determinado por este modelo, siendo por lo general de 7 segundos con un intervalo definido entre los 5 y 226 segundos. Este valor depende de la cantidad de *chunks* contenidos en memoria. Como se puede apreciar en los valores recogidos en la Tabla 19, mientras

[1] Cuando la memoria a corto plazo sin combina con valores almacenados en la memoria a largo plazo.

mayor sea el número de *chunks* almacenados, menor es el tiempo que éstos permanecen en memoria.

Tabla 20. Valores de capacidad y tiempo de permanencia para la memoria visual.

MEMORIA VISUAL (MV)
μ_{MV} = 17 [7 — 17] letras.
δ_{MV} = 200 [70 — 1.000] milésimas de segundo.

Dentro de la memoria a corto plazo, el modelo de procesador humano propone la existencia de dos memorias encargadas del almacenamiento de datos específicos. Se trata de la memoria visual y de la memoria auditiva. Como se puede apreciar en la Tabla 20 y en la Tabla 21, cada una de ellas posee sus propios valores promedio, mínimo y máximo para los parámetros de su capacidad (μ) y del tiempo de permanencia (δ) de los objetos almacenados en cada uno de ellos.

Tabla 21. Valores de capacidad y tiempo de permanencia para la memoria auditiva.

MEMORIA AUDITIVA (MA)
μ_{MA} = 5 [4,4 — 6,2] letras.
δ_{MA} = 1.500 [900 — 3.500] milésimas de segundo.

En el caso de la percepción visual, cuando se percibe información de forma rápida, se capta mucha más de la que es posible almacenar. Sperling (1969) analizó esta cuestión en experimentos de rastreo visual distinguiendo el efecto de la capacidad de la memoria visual y de la limitación perceptiva en la atención, descubriendo que es posible percibir correctamente nueve letras de forma simultánea. Sin embargo, si estas letras han de ser memorizadas, solamente se almacenan cuatro. Peor aún es el resultado si es necesario recordar relaciones de orden o de cualquier otro tipo entre los objetos percibidos, ya que la captura de este tipo de información requiere un mayor nivel de atención para ser fijada. Hay que destacar por tanto que la memoria visual, al igual que ocurre con la memoria de trabajo, también está estrechamente relacionada con la memoria a largo plazo, lo cual influye notablemente en su capacidad y rendimiento.

La capacidad de almacenamiento para estas memorias se mide en letras ya que la mayoría de los experimentos encaminados a determinar su capacidad emplearon estas unidades, tal y como se puede apreciar en los experimentos recogidos por Miller (1956).

Aunque estas dos memorias específicas forman parte de la memoria a corto plazo, éstas están controladas y dirigidas por el procesador perceptivo quien se encarga de alimentarlas constantemente con nuevos datos. Por su parte, el resto de la memoria a corto plazo es gestionado por el procesador cognitivo, quien se encargará además de procesar la información almacenada tanto en la memoria visual como en la memoria auditiva, siempre que sea necesario a la hora de confeccionar una respuesta por parte del sistema.

13.3 Adaptación a la Capacidad Humana

El objetivo del agente interactivo representado por DEVA consiste en apoyar al supuesto procesador humano del usuario de las aplicaciones GADEA para no sobrepasar los límites impuestos tanto en la capacidad de sus memorias como en la velocidad de olvido o en la velocidad de los distintos procesadores implicados. De este modo, DEVA controlará la cantidad de información presente en todo momento en los distintos canales de comunicación empleados en los diálogos interactivos solicitados por el programador, limitando su alcance en función de las características concretas del usuario de la aplicación.

En ocasiones, cuando el usuario invoca a un proceso de usuario, para que ése pueda llevarse a cabo, serán necesarias cantidades ingentes de información. Naturalmente, será el usuario quien deba proporcionar tal información. En un sistema de trabajo orientado a la productividad, el primer objetivo establecido por DEVA a la hora de diseñar y crear el correspondiente diálogo interactivo será la adecuación de las solicitudes de información a las capacidades físicas y cognitivas del usuario que ha de proveer dicha información. Este sistema tendrá que examinar por tanto la cantidad de información solicitada por la aplicación a través de CodeX y contrastarla con la capacidad del usuario.

Si la cantidad solicitada es menor o igual que la capacidad de respuesta, el diálogo se podrá crear sin ningún tipo de problemas, estableciendo una relación directa entre la información solicitada y los *widgets* encargados de su obtención. Sin embargo, si la cantidad de información requerida excede a las capacidades del usuario para proveerla, será DEVA quien tenga la responsabilidad de estructurar la petición de información en *chunks* del tamaño adecuado al usuario. Otro tanto ocurrirá en el caso en el que sea la propia aplicación la que envíe información al usuario. Si la cantidad enviada excede sus capacidades, ésta tendrá que ser fragmentada en colecciones de *chunks* del tamaño adecuado, de tal modo que el usuario pueda procesar dicha información correctamente.

Así por ejemplo, DEVA emplea el límite impuesto por la memoria visual para establecer el número máximo de ítems que debe tener un menú de opciones, así como el número máximo de opciones disponibles en un grupo de botones de radio o de *checkboxes*. Los elementos visuales mencionados poseen una clara connotación de bloque, pero no son los únicos ya que los iconos de las paletas de herramientas, los grupos de botones de navegación, la información estructurada en tablas o matrices son claros candidatos para recibir idéntico tratamiento.

La adaptación de los modos de interacción y disposición de elementos en función de la capacidad cognitiva del usuario ha sido puesta a prueba por nosotros a través del programa de investigación *Cinemedia Astur*. El objetivo de este programa consiste en el análisis y diseño de herramientas de autor que faciliten la creación de aplicaciones hipermedia en entornos de colaboración, siguiendo un paradigma de desarrollo basado en la ejecución de proyectos cinematográficos [González y Cueva (1999)]. Este programa, iniciado en

1997 en el Laboratorio de Tecnologías Orientadas a Objetos del Departamento de Informática de la Universidad de Oviedo consta de varios proyectos de desarrollo que han proporcionado una evolución continua y progresiva en la potencia y flexibilidad de la herramienta.

Una de las características peculiares de CineMedia Astur es su capacidad para limitar la cantidad de información reflejada en su interfaz de acuerdo con valores estáticos para la memoria a corto plazo del usuario [González et. al. (2000)], todo ello aún cuando el número de elementos de esta interfaz se desconoce en tiempo de compilación, ya que no existe una configuración prefijada para esta herramienta, dado que su campo de acción puede ser ampliado por medio de componentes *JavaBeans* fabricados por terceras partes [Redondo (2000)].

Es por ello que CineMedia Astur dispone de un lenguaje de especificación de interfaz propio mediante el cual se puede indicar la capacidad real (ancho de banda) y efectiva (ancho de banda en relación a la capacidad del procesador humano del usuario) de sus canales de comunicación [Redondo et. al. (2000a)]. De este modo, la herramienta es capaz de realizar un proceso primitivo de adaptación en gran parte de las componentes de la interfaz [González et. al. (2000b)]. Aunque este proceso no es tan potente como el que la tecnología GADEA provee, su empleo ha dado muy buenos resultados, convirtiendo a CineMedia Astur en la herramienta de autor hipermedia con el mayor nivel de auto-adaptación de su tiempo.

La estructuración de la información en *chunks* por parte del agente interactivo se realiza de tal modo que éste simula el tamaño de los verdaderos *chunks* que emplea el usuario. Este mecanismo de estructuración es de suma importancia en sistemas como el presente, en donde es fundamental garantizar un acceso universal a la información. Nótese por ejemplo como para determinados usuarios, la capacidad de sus memorias o la velocidad de sus procesadores no coincide con los valores medios referidos en este documento.

Esta divergencia es elocuente en el caso de personas con discapacidades físicas auditivas o visuales. Para un ciego, por ejemplo, no existe memoria visual con lo que se pierde el apoyo de los diecisiete elementos como promedio que pueden almacenarse en esta memoria en tareas de reconocimiento de un canal visual que resulta inútil a todas luces. Otro tanto de lo mismo ocurre en el caso de la memoria auditiva para un usuario sordo. La pérdida de una de estas memorias implica necesariamente una disminución importante en la capacidad real de la memoria de trabajo, la cual tendrá que apoyarse de una forma más firme en el intercambio de información con la memoria a largo plazo. Para un usuario con serias discapacidades visuales, por ejemplo, la capacidad de la memoria a corto plazo empleada en tareas de interacción con una interfaz de usuario se debe situar en sus valores mínimos, ya que al tener que realizar el registro de la información activa en el canal de comunicación auditivo de forma puramente secuencial, se exige una menor velocidad de pérdida de la información (es decir, la información ha de permanecer más tiempo), lo que obliga a disminuir drásticamente la capacidad de dicha memoria.

Otro aspecto que afecta notablemente a la capacidad de la memoria de trabajo es la edad del usuario, ya que, salvo en edades muy tempranas, la capacidad de esta memoria disminuye a medida que aumenta la edad.

También hay que destacar que el modelo de procesador humano aplicado al diseño de este agente interactivo no funciona de manera estática, sino que sus valores pueden cambiar con el tiempo, incluso dentro del espacio temporal de una misma sesión de trabajo. Blumenthal (1991) comenta como situaciones de esfuerzo continuado pueden disminuir la capacidad de trabajo de esta memoria. Dado que GADEA observa en todo momento el comportamiento del usuario mientras éste trabaja con sus aplicaciones, para este sistema es relativamente sencillo poder determinar la carga de trabajo a la que está sometido un usuario, la cual se determina en función de las tareas realizadas a lo largo de su sesión de trabajo. Esta in formación almacenada por los agentes ANTS en el Modelo de Usuario por Aplicación. (UAM).

El conjunto de tareas realizadas por un usuario, puestas en relación con su destreza, puede ser empleado para determinar la disminución del rendimiento del mismo en su sesión de trabajo, reduciendo de forma dinámica los parámetros correspondientes a sus procesadores y memorias. Así por ejemplo, un usuario con capacidad típica en su memoria a corto plazo para siete elementos, puede ver disminuida su capacidad a seis elementos o incluso cinco en función del número de horas continuas dedicadas al trabajo y al número de tareas realizadas en dicho lapso con relación al número de tareas al que está acostumbrado a realizar.

Es necesario destacar que a los diseñadores o los programadores de una aplicación les resulta completamente imposible conocer el perfil cognitivo y perceptivo de sus usuarios en tiempo de diseño, por lo que la adaptación de los requisitos de interacción o del despliegue de información ha de realizarse en tiempo real. En caso contrario, si los diseñadores conocen a la perfección el estado cognitivo y perceptivo de sus usuarios, estaríamos hablando del desarrollo de aplicaciones a medida para un usuario concreto, lo cual, aunque válido es sumamente costoso y desde el punto de vista de una industria del software orientada a la producción en masa, totalmente inviable. A la hora de la verdad y mediante el enfoque dado en esta investigación, será la propia aplicación, por medio de su sistema de gestión de interfaces inteligente quien realice la estructuración de la información, evitando con ello desbordar la capacidad cognitiva y perceptiva de sus usuarios concretos. Para ello, los *widgets* empleados por DEVA han sido diseñados para tomar decisiones de forma dinámica en tiempo de ejecución y en función del usuario que los ha de emplear. Teniendo en cuenta las características actuales –reales o supuestas– del usuario activo, un *widget* encargado de desplegar una cantidad de información superior a la capacidad máxima soportada por el usuario, reestructurará dicha información para transmitirla en *chunks* del tamaño adecuado, modificando de este modo el comportamiento del propio *widget*, estableciendo así un primer nivel de adaptación.

Veamos como ejemplo el funcionamiento previsto del *widget gadea.widgets.RadioButtonGroup* de GADEA, es decir, su grupo de botones de radio. Como es bien sabido, un botón de radio es una metáfora de los botones empleados en las radios de los coches para sintonizar una determinada emisora o dial de manera rápida. En dichas radios, al presionar un botón la aguja del dial se desplaza a la posición física señalada por dicho botón. Al presionar un segundo botón, se desactiva el primero y la aguja salta a la posición señalada por el segundo botón, de tal modo que solo puede haber un botón activo. La metáfora de un grupo de botones de este tipo satisface plenamente las capacidades cognitivas y perceptivas de la gran mayoría de los usuarios, ya que el número de estos botones en una radio suele situarse en torno a cinco, cifra que representa el mínimo número de elementos con el que trabaja la memoria a corto plazo.

El excelente diseño de este control queda patente si tenemos en cuenta que éste va a ser utilizado en combinación con otras tareas (la conducción de vehículos, por ejemplo). Sin embargo, el libertinaje existente en el de diseño de las interfaces de un usuario ha dañado considerablemente la reputación de estos controles al ser empleados de modo indiscriminado en grupos de botones formados por un gran número de elementos. Obviamente si su número supera los siete elementos ya se ha superado la capacidad media de la memoria de trabajo de la mayoría de los usuarios de una aplicación.

Para garantizar un buen funcionamiento de ese control en los casos en los que por razones cognitivas, perceptivas o motrices sea aconsejado su uso (ver capítulo 16: *Un Modelo de Evaluación*) y ante un posible uso irracional del mismo, el *widget* tiene en cuenta el valor de la memoria a corto plazo del usuario (almacenado en el modelo de usuario correspondiente) no permitiendo el uso de un número mayor de botones que el que se encuentra indicado por la capacidad de dicha memoria. Si por alguna razón es necesario superar este limite, el *widget* se encargará de estructurar la información contenida en el mismo, creando una jerarquía visual de elementos que permitan acceder a dicha información sin violar la capacidad máxima del procesador perceptivo ni de sus memorias.

En concreto, el *widget* agrupará los botones de radio en bloques de tantos elementos como indique la capacidad de la memoria a corto plazo o menos aún si esto es posible. Si el número de bloques obtenido es superior a la capacidad de la memoria, éstos se agruparán a su vez en macro bloques, conteniendo cada uno de ellos tantos bloques como indique la memoria a corto plazo (o menos si esto es posible). Este proceso de agrupación se realiza de forma indefinida hasta que se logra obtener un número de macro bloques igual o inferior al valor máximo de la memoria a corto plazo del individuo. De este modo se crea un árbol conceptual en donde las ramas están formadas por bloques y las hojas por botones de radio. Como se puede apreciar, éste proceso de reestructuración de la información sigue fielmente el esquema de creación de *chunks*, permitiendo burlar la limitación espacial de la memoria a corto plazo para poder almacenar cualquier cantidad de elementos en la misma. De este modo, el agente interactivo

representado por DEVA estructura la información de forma similar a como lo haría el humano al que representa.

Una vez estructurada la información, se procede a mostrarla al usuario para su selección. Para ello se muestran los bloques de primer nivel mediante el uso de la metáfora del botón, en este caso, empleando botones normales en cuya etiqueta se hace alusión al rango de valores (semántico) contenido en el bloque representado por cada botón, es decir, indicando los bloques o botones de radio (opciones) incluidos en dicho bloque. Cuando el usuario selecciona un bloque (pulsando el botón o el número correspondiente al botón en el caso de usuarios con discapacidades visuales), este bloque se expande, permitiendo acceder a los bloques o botones de radio que contiene. Este proceso se realiza iterativamente hasta que el usuario llega a las hojas del árbol. Nótese que en cada nivel de éste árbol, el número de elementos a percibir por el usuario no excederá nunca la capacidad de su memoria de trabajo.

Para ilustrar con el máximo nivel de detalle posible el funcionamiento del *gadea.widgets.RadioButtonGroup*, vamos a suponer el caso de una aplicación en la que uno de sus procesos de usuario requiere que el usuario indique el país en el que vive, seleccionando un país entre veinticinco propuestos. La definición de este proceso de usuario puede verse en el Código 23.

Código 23. Clase ejemplo *Country* empleada par la selección de un país por parte de un usuario.

```
public class Country
{
   UIMS interface = null;

   // constructor
   Country (UIMS myInterface)
   {
      interface = myInterface;
      interface.registerEveryUP (this);
   }

   // User processes
   public boolean UP_getCountry ()
   {
      gadea.datatypes.String userCountry;
      boolean value = false;

      // Data primitives instantiation

      userCountry = new gadea.datatypes.String ("", false);

      // Components configuration
      userCountry.setRequired (true);

      gadea.datatypes.String String europeanCountries [] = {
         "Alemania", "Andorra",
         "Austria", "Bélgica", "Bulgaria",
         "Croacia", "Dinamarca", "Eslovaquia", "Eslovenia",
         "España", "Estonia", "Finlandia", "Francia", "Grecia",
         "Hungria", "Irlanda", "Islandia", "Italia", "Letonia",
         "Lituania", "Luxemburgo", "Macedonia", "Malta", "Mónaco",
         "República Checa",};


    for (int i = 0 ; i < 25 ; i++)
    {
         userCountry.addPossibleValue (europeanCountries[i]);
    }
```

```
      Dialogue dialogue = new Dialogue ("");
      Dialogue.add (new gadea.datatypes.String ("País de Origen",
      false));
      dialogue.add (userCountry);

      CommunicationChannel comChannel = new CommunicationChannel
         ("Country Selection" , CommunicationChannel.MAX_PRIORITY);

      interface.load (comChannel);
      if (comChannel.display (dialogue))
      {
         // User selected OK
         System.out.println ("Selected Country: " +
            userCountry.getFirstValue ());
         value = true;
      }
      else
      {
         // User selected CANCEL
      }

      interface.unload (comChannel);
      return (value);
}
```

Nótese como el programador ha cometido el *desliz* de incluir los veinticinco países de la lista directamente en el diálogo interactivo raíz, sin agruparlos de ninguna manera en una estructura más adecuada para la capacidad de la memoria de trabajo del usuario destino. Aunque podría haber hecho uso de la clase *Chunk* proporcionada por CodeX, en realidad no se trata de un error ya que diseñador de la aplicación no puede –ni debería– hacer ningún tipo de suposición de bajo nivel acerca del tipo de usuario de su aplicación. En todo caso podría emplear la clase *Chunk* para proporcionar algún tipo de estructuración lógica (semántica, no sintáctica) de la información a transmitir, clasificando los países de la lista de acuerdo con algún tipo de clave que pueda ser de utilidad al usuario. Por ejemplo, dado que todos los países de la lista son europeos, el diseñador podría intentar clasificar los mismos entre de acuerdo con las clases *países de la comunidad europea, países mediterráneos, países nórdicos, países del Este*, etc. Esta clasificación dependería del dominio de la aplicación o de la metáfora empleada, pero nunca antendería a consideraciones sobre la capacidad del usuario destino.

Una vez que CodeX recibe la solicitud de creación de un diálogo interactivo al invocarse el proceso de usuario *getCountry*, se convierte la petición en un documento ACML libre de sintaxis del tipo del incluido en el Código 24. Como se puede apreciar, este documento también carece por completo de *chunks* ya que en realidad se trata de obtener información acerca de un único objeto correspondiente a una cadena de caracteres (*gadea.datatypes.String*) de obtención obligatoria y que solo puede ser una de las veinticinco opciones correspondientes a otros tantos países europeos.

Código 24. Documento ACML equivalente al diálogo interactivo del proceso *getCountry* del Código 23.

```
<? ACML version = 1.0' ?>
<DIALOGUE NAME='País de Origen'>
   <STRING ISFIXED='FALSE' ISPERCEPTIBLE='TRUE' ISREQUIRED='TRUE'
      MULTIPLEVALUES='FALSE'>
      <POSSIBLE_VALUES>
         <VALUE ID = 'Alemania'/>
         <VALUE ID = 'Andorra'/>
         <VALUE ID = 'Austria'/>
         <VALUE ID = 'Bélgica'/>
         <VALUE ID = 'Bulgaria'/>
         <VALUE ID = 'Croacia'/>
         <VALUE ID = 'Dinamarca'/>
         <VALUE ID = 'Eslovaquia'/>
         <VALUE ID = 'Eslovenia'/>
         <VALUE ID = 'España'/>
         <VALUE ID = 'Estonia'/>
         <VALUE ID = 'Finlandia'/>
         <VALUE ID = 'Francia'/>
         <VALUE ID = 'Grecia'/>
         <VALUE ID = 'Hungria'/>
         <VALUE ID = 'Irlanda'/>
         <VALUE ID = 'Islandia'/>
         <VALUE ID = 'Italia'/>
         <VALUE ID = 'Letonia'/>
         <VALUE ID = 'Lituania'/>
         <VALUE ID = 'Luxemburgo'/>
         <VALUE ID = 'Macedonia'/>
         <VALUE ID = 'Malta'/>
         <VALUE ID = 'Mónaco'/>
         <VALUE ID = 'República Checa'/>
      </POSSIBLE_VALUES>
   </STRING>
</DIALOGUE>
```

Cuando el documento ACML es recibido por DEVA, éste es decodificado, eligiendo uno de los posibles *widgets* disponibles que pueden satisfacer el requerimiento de información. Para este caso concreto, se podría utilizar un campo de texto, un menú contextual, una lista desplegable o un grupo de botones de radio. Como se verá en el capítulo 16 (*Un Modelo de Evaluación*), DEVA empleará el conocimiento disponible acerca de las características cognitivas, perceptivas y motrices del usuario para seleccionar aquel *widget* que mejor se adapte a dicho usuario. Supongamos que DEVA elige un grupo de botones de radio y que además no se emplea el algoritmo de estructuración de conocimiento descrito. El resultado podría ser el ilustrado en la Figura 37.

Como se puede apreciar en la mencionada Figura 37, el número de *chunks* supera la capacidad de cualquier memoria a corto plazo. En un nivel de detalle semántico capaz de detectar las letras como tales, es decir, a un nivel de madurez que supera la infancia para una persona no analfabeta, en el canal de comunicación de la figura existen veintinueve objetos visibles. Veinticinco de ellos se corresponden con los botones de radio, dos representan las acciones *OK* y *CANCELAR*, otro objeto representa el nombre del diálogo interactivo (*País de Origen*) y un último elemento representa el número dado al *widget* grupo de botones de radio, el cual es empleado para proporcionar un acceso a dicho *widget* por medio del teclado para usuarios con discapacidades motoras.

Figura 37. Posible diálogo generado a partir del documento ACML del Código 24. La cantidad de *chunks* en el grupo de botones de radio (25) supera ampliamente la capacidad de la MCP de cualquier sujeto (consultar versión en color en el capítulo 24: *Imágenes en Color*, p. 445).

En la figura, todos los elementos con la excepción de los botones de radio pueden considerarse como objetos propios de la memoria a largo plazo ya que son habituales en cualquier canal de comunicación empleado por GADEA. Lo mismo ocurre con la propia ventana, la cual no es percibida como tal durante el proceso de introducción de información al sistema. Aún así, en el canal de comunicación quedan veinticinco elementos (los botones de radio) percibidos y almacenados por la memoria a corto plazo, la cual es desbordada aún en el mejor de los casos (capacidad igual a nueve elementos). Hay que destacar que esta vulneración de la capacidad de la memoria se presenta tanto para la memoria visual como para la memoria auditiva, puesto que los veinticinco elementos de este *widget* serán transmitidos a través del canal sonoro mediante las técnicas de barrido clásicas [Abascal et. al. (2000)].

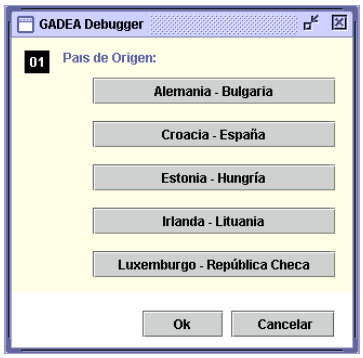


Figura 38. Diálogo adaptado para un usuario con MCP igual a 5 elementos, partiendo del diálogo interactivo incluido en el documento ACML de la Figura 37 (consultar versión en color en el capítulo 24: *Imágenes en Color*, p. 445).

Un *widget* adaptable del tipo del *widget gadea.widgets.RadioButtonGroup* empleado por GADEA, usada la información contenida en el modelo de usuario activo para estructurar la información a transmitir en función del tamaño de la memoria a corto plazo. Supongamos que para este ejemplo en concreto el usuario de la

aplicación dispone de una memoria a corto plazo con capacidad para cinco elementos. En este caso, el *widget* agrupará los veinticinco botones de radio en bloques de cinco o menos elementos. El total de *chunks* obtenidos por este mecanismo asciende a un total de 25/5 chunks, es decir, cinco elementos. Dado que este número es menor o igual que la capacidad de la memoria a corto plazo, no será necesario reagrupar los cinco bloques obtenidos en nuevos macro bloques. El componente se limitará entonces a mostrar al usuario los cinco bloques obtenidos en lugar de los veinticinco originales. En la Figura 38 Se muestra el resultado de esta agrupación y de las opciones presentadas al usuario. Hay que recordar que igual número de opciones son presentadas a través del auditivo en el caso en el que el valor de los parámetros cognitivos, perceptivos y motrices del usuario así lo aconsejen.

Como se puede apreciar en la Figura 38, no sólo se disminuye drásticamente el área de rastreo sino que ésta cabe por completo tanto en la memoria a corto plazo, como en cualquiera de las dos memorias incluidas en ésta (la memoria visual y la memoria auditiva). De este modo, la capacidad de las memorias no es superada y al no producirse este evento, tampoco se disminuye el tiempo de duración del ciclo de reloj del procesador perceptivo, ya que éste no se ve saturado por un bombardeo excesivo de información. Al no superarse la carga máxima admitida por el procesador perceptivo en un ciclo de trabajo, tampoco se supera la capacidad del procesador cognitivo, al menos como consecuencia de un exceso de información proveniente de este *widget*. El resultado de todo ello es un flujo continuado de la información a una velocidad lenta, sin producirse ninguna sobrecarga, con lo cual se facilita el establecimiento de estrategias de procesamiento automático y se libera al proceso perceptivo de gran parte de su carga cognitiva.

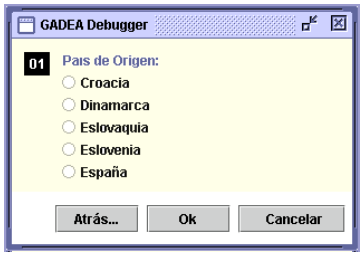


Figura 39. Aspecto del *widget* de la Figura 38 al seleccionarse el segundo botón por arriba. Gracias al mecanismo de adaptación, el número de *chunks* empleado sigue sin superar la capacidad de la MCP del sujeto (consultar versión en color en el capítulo 24: *Imágenes en Color*, p. 445).

Supongamos que el país de origen del usuario es *España*. Para elegir este opción, el usuario realizará un reconocimiento visual y lineal de la columna de botones que representan las opciones, seleccionado el segundo botón empezando por arriba, con lo que el *widget* asumirá el aspecto recogido en la Figura 39. En el caso del canal de comunicación auditivo, la información contenida en los botones de la Figura 38 será transmitida de forma constante mientras el usuario se encuentre en el *chunk* etiquetado con el número 1 (el único existente) siguiendo la técnica de barrido comentada. Cuando la opción *Croacia – España* sea

transmitida por el canal, el usuario la seleccionará y se pasará entonces a la versión auditiva de la información recogida en la Figura 39.

En dicha figura se puede apreciar como el mecanismo de adaptación del grupo de botones de radio sigue respetando la capacidad de la memoria a corto plazo al transmitir tan solo cinco *chunks*, cifra que iguala a la de la memoria de trabajo del supuesto usuario. Tanto en el caso del canal de comunicación visual como en el del canal de comunicación auditivo, la búsqueda de la opción deseada se efectúa por medio un proceso secuencial, el cual concluye una vez encontrado el objeto deseado. En el ejemplo concreto que estamos empleando, será necesario recorrer todos los elementos (cinco) antes de dar con *España*.

La tecnología cognitiva con la que ha sido diseñado este *widget* no solo permite una adaptación sencilla y segura a las limitaciones de la capacidad humana sino que facilita además un acceso rápido a la información para personas con discapacidades visuales. Nótese que la estructuración en bloques permite desechar hasta un máximo de $\mu - 1$ ramas del árbol por nivel. En el ejemplo, un usuario ciego que desease seleccionar *España* como opción siguiendo el comportamiento estándar de la Figura 37, tendría que esperar el barrido de diez países antes de llegar a la opción elegida (ya que se trata de un barrido secuencial), mientras que con la versión adaptable del *widget*, sólo tendría que esperar seis barridos. Si se deseara seleccionar la última opción de todas (la *República Checa*) siguiendo el modo de funcionamiento habitual de estos componentes, un usuario ciego tendría que esperar el barrido de veinticinco elementos, mientras que con el empleo de la técnica de adaptación propuesta, el barrido se reduce a tan solo diez elementos, lo cual representa una reducción del orden del 60%.

14

14 La Atención

Todo método consiste en el orden y disposición de aquellas cosas hacia las cuales es preciso dirigir la agudeza de la mente

Descartes

14.1 Las Señales

La atención se sitúa en un nivel periférico del procesador humano actuando como regulador de los estímulos procesados, seleccionado aquellos que son relevantes para el sujeto o para la tarea que éste está realizando. Estos estímulos pasan a la consciencia para un procesamiento en detalle. Gracias a la atención es posible rechazar los estímulos irrelevantes, evitando así la sobrecarga del sistema.

En los dos modelos de limitación de capacidad de procesamiento propuestos, tanto en el modelo de cuello de botella como en el modelo de capacidad, se identifica la atención con la consciencia. Esto es debido en parte a que la metodología utilizada en la época en que fueron postulados estos modelos empleaba parámetros totalmente conscientes, tales como la calidad del recuerdo de tareas realizadas o la eficacia con la se desarrollaban las mismas. Resulta obvio por tanto que la medición

de estos parámetros impedía probar o refutar la existencia de un procesamiento inconsciente de la información, también conocido como procesamiento subliminal. Por aquel entonces los autores no solían plantearse la existencia de procesamientos inconscientes o percepciones subliminales por lo que se vieron abocados a considerar al Sistema P como un canal de información similar al de una línea telefónica. Dado que las informaciones sensoriales recibidas son múltiples, se necesitaba un mecanismo capaz de filtrar la información y dejar pasar a la consciencia únicamente a datos relevantes. Este mecanismo era la atención.

14.1.1. Selección Temprana y Selección Tardía

En el modelo de línea telefónica, el rango de las señales sensoriales se descompone en señales físicas y en señales semánticas. Mientras que las primeras son fáciles de analizar y requieren pocos recursos para ello, las segundas son elaboradas a partir de las señales físicas y por lo tanto deberían emplear más tiempo y recursos para ser decodificadas. Es el propio Broadbent (1958) quien propone la primera explicación para el procesamiento de este tipo de señales, la cual se conoce por el nombre de *Teoría de la Selección Temprana* o *Early Selection Theory*. En esta teoría, Broadbent llega a la conclusión de que el análisis de los estímulos se realiza primero sobre las señales físicas, pasando a continuación a tratar las señales semánticas. Su razonamiento parte del ahorro energético que se consigue con el tratamiento exclusivo de las señales físicas –más fáciles y baratas de procesar que las señales semánticas– ya que un procesamiento temprano y completo de todos los estímulos (tanto físicos como semánticos) supone un gasto de energía inútil. De acuerdo con esta teoría, si las señales físicas han pasado a la consciencia tras el primer filtro impuesto por la selección temprana, será entonces cuando se analizarán las señales semánticas. Para este modelo, el análisis semántico no forma parte de la atención.

Sin embargo, en los años siguientes, los trabajos de Moray (1959) y Treisman (1960) demostraron serias contradicciones en el modelo de Broadbent. Esto originó la *Teoría de la Selección Tardía* (*Late Selection Theory*) propuesta por Deutsch y Deutsch (1975) en la que la información se selecciona en función de su significado (señales semánticas). Para los defensores de esta teoría, la realización de un proceso de selección practicado exclusivamente en función de señales físicas puede llevar al organismo a poner en peligro su supervivencia debido a que pasaría por alto algunos estímulos potencialmente peligrosos para él. Según esta teoría, no tiene ningún sentido ahorrar unas pocas unidades de energía mental si ello lleva consigo la aniquilación completa del sistema.

De acuerdo con los dos planteamientos enfrentados, o la selección se realiza en función de señales físicas (caso de la teoría de la selección temprana) o se lleva a cabo usando sólo las señales semánticas (caso de teoría de la selección tardía). Como suele suceder, los dos planteamientos anteriores son demasiado estrictos y tajantes como para prosperar sin una fusión, surgiendo así una nueva teoría, la *Teoría de la*

Selección Múltiple. En esta teoría, el sistema cognitivo selecciona los estímulos a procesar (de naturaleza física o semántica) en función de la situación.

Si bien el esfuerzo mental implicado en el procesamiento de las señales semánticas es superior al necesario para decodificar las señales físicas, los trabajos de Duncan (1981) han demostrado que esa diferencia no es lo suficientemente significativa como para fortalecer la posición de una selección temprana, más bien al contrario. En los experimentos realizados por este investigador, a los voluntarios se les mostraba brevemente una parrilla en forma de cruz en la que se incluían cuatro signos. Los voluntarios debían responder alzando o bajando un brazo (o los dos) cuando identificaban caracteres alfanuméricos en la cruz. Cuando las respuestas debían ser indicadas con un solo brazo, no se detectaba sobrecarga alguna aunque los estímulos fuesen múltiples. Justo lo contrario ocurría cuando la respuesta implicaba el movimiento de los dos brazos. Los experimentos realizados por este investigador vinieron a demostrar que el rendimiento no se ve afectado por la cantidad de estímulos a analizar, incluso cuando éstos estímulos están formados exclusivamente por señales semánticas. La verdadera sobrecarga viene dada por el número de respuestas simultáneas que el Sistema P debe elaborar, dado que el elemento clave en la capacidad de procesamiento no es la aparición del estímulo sino la confección de la respuesta.

Estos resultados vienen a imponer una limitación más al modelo de procesador. Esta limitación consiste en que dicho procesador sólo podrá realizar un procesamiento controlado a la vez. Aunque un sujeto puede recibir varios estímulos de la interfaz (limitados en todo caso a la capacidad de su memoria de trabajo) sólo podrá procesarlos para ejecutar con ellos una tarea a la vez. De este modo es necesario diseñar cuidadosamente el contenido de la información solicitada o mostrada al usuario a través de cada uno de los canales de comunicación con el objeto de evitar que se supere esta limitación. Sólo cuando la ejercitación y entrenamiento en la ejecución de la tarea han convertido su puesta en práctica en un proceso automático, será posible asignar una nueva tarea al mecanismo de procesamiento controlado.

14.1.2. Procesamiento Avanzado

Actualmente, con una teoría de la selección temprana refutada por numerosos estudios –entre los que destacan los de Duncan (1986)– el debate se centra entre la teoría de la selección múltiple y una teoría de la selección tardía modificada, que ya no plantea un procesamiento exclusivo de las señales semánticas, sino un proceso de análisis selectivo a nivel semántico de todos los estímulos. Dicho proceso selectivo basa sus decisiones de forma indistinta en las características físicas y / o semánticas de los estímulos.

Sin embargo, los trabajos de Stroop (1935, citado en [Bauer (1997)]) aportan pruebas extraordinarias de que las señales semánticas son procesadas por la atención humana, incluso en situaciones límite. Los experimentos realizados por Stroop tenían por objeto analizar la

interferencia de informaciones presentadas simultáneamente a un grupo de sujetos. Estas experiencias consistieron en mostrar a un grupo de observadores trozos de papel de diversos colores así como unas etiquetas en la que estaban escritos los nombres de dichos colores. El objetivo del experimento consistía en que los observadores fuesen capaces de pronunciar en voz alta el nombre del color mostrado (bien a través del papel de color o por medio de la etiqueta).

En la primera fase del experimento se enseñaron las etiquetas y los papeles de colores a los observadores, midiendo el tiempo de reacción empleado en nombrar el color que se les presentaba. Como se puede apreciar, en esta primera fase del experimento se instaba a los voluntarios a identificar la misma información, pero por medio de dos tipos de señales distintas: las señales físicas representadas por los papeles de colores y las señales semánticas representadas por las etiquetas con los nombres de los colores.

En la segunda parte del experimento, Stroop escribió el contenido de las etiquetas (nombre de cada color) en los papeles de colores, pero con la salvedad de que el color escrito no coincidía con el color del papel, es decir, el contenido de las señales físicas y semánticas transmitidas por un mismo trozo de papel no coincidía. A continuación se pidió a los observadores que nombraran el color de los papeles, es decir, la señal física, detectándose entonces una tasa significativa de fallos en las respuestas de los voluntarios. En dichas respuestas erróneas, los observadores tendían a nombrar el color escrito en el papel en lugar de referirse al color del papel.

Como se puede apreciar, las señales semánticas no solo fueron procesadas en su totalidad (superando el filtro de la atención) sino que este proceso se llevó a cabo con prioridad sobre las señales físicas. En este experimento se puede ver claramente como los mecanismos de procesamiento de las señales semánticas se activan aun en los casos en que el objetivo del sujeto se concentra en analizar las señales físicas solamente.

En base a estos resultados y teniendo en cuenta que el procesamiento primario de estímulos semánticos por parte de la atención no parece sobrecargar ni sobrepasar la limitada capacidad del procesador humano, DEVA dará prioridad al envío de señales semánticas a través de los canales de comunicación activos en detrimento de la transmisión de señales puramente físicas.

Siempre que la información correspondiente a un concepto se encuentre en la base de conocimiento de la aplicación, DEVA transmitirá dicho concepto por medio de todas las representaciones posibles aportadas por el *Language Manager* de CodeX (ver capítulo 10: *El Lenguaje*) incluyendo su representación iconográfica y descripción breve, dejando la apariencia externa y la disposición física del concepto en un segundo nivel de prioridad. Así por ejemplo, el nombre dado a un botón (señal semántica) tendrá prioridad sobre otras consideraciones como su tamaño, forma o color (señal física), subordinando éstas últimas a garantizar la correcta transmisión de la carga semántica del botón. De este modo, el color del botón y la fuente tipográfica del

mismo serán modificados para garantizar la máxima legibilidad de su contenido aunque éstas operaciones vayan en detrimento de otros factores tales como la consistencia o la estética de la interfaz.

14.2 Niveles de Procesamiento

De acuerdo con Ballesteros (1994) existen tres niveles de atención en el sistema perceptivo humano, los cuales dependen directamente de la energía mental necesaria para la captación de los distintos estímulos. En un primer nivel se sitúa la atención subliminal en la que algunas características de los estímulos se perciben sin intención de ser percibidas aunque éstas no pasan directamente a la consciencia. En un segundo nivel se sitúa la atención distribuida en la que es posible percibir múltiples elementos de forma consciente, los cuales pueden ser tratados en paralelo por medio de un procesamiento automático. En el último nivel se situaría la atención focal en la que algunas características de los estímulos percibidos requieren de la atención consciente del individuo para ser captadas y por lo tanto consumen gran cantidad de recursos del procesador cognitivo y del procesador perceptivo.

Un estudio realizado por Norman (1988) vino a demostrar la existencia de la separación entre los estímulos importantes y aquellos que lo son menos. Para comprobarlo, se llevaron a cabo experimentos en los que se sometía a un observador a una activa conversación con un interlocutor mientras, paralelamente, otra pareja mantenía una conversación similar a su lado. Llegado un momento determinado, se pedía al observador que intentase recordar algo de la conversación de la pareja vecina, a la que obviamente no estaba atendiendo. Si bien los observadores se sentían incapaces para recordar apenas nada de dicha conversación a la que no prestaban atención, sorprendentemente llegaban a recordar algo. Incluso se llegó a determinar que la cantidad de palabras recordadas del canal de comunicación no atendido se situaba entre las cinco y las siete, resultado que no contradice los parámetros asignados para la memoria a corto plazo y para la memoria auditiva recogidos en el capítulo 13 (*El Procesador Humano*).

Es muy importante tener en cuenta el nivel de percepción en el que el usuario realiza un determinado procesamiento de la información ya que éste determinará de forma activa el tipo de atención que será requerida por parte del usuario, pudiendo ser ésta consciente o inconsciente. Es por ello que DEVA hace uso de los resultados de estas investigaciones para distribuir la información a transmitir por medio de diferentes canales de comunicación, proyectando la información con carácter prioritario a través de los canales de comunicación principales y transmitiendo la información de carácter auxiliar a través de los canales de servicio asociados a dichos canales principales, o en todo caso, empleando los vacíos dejados en el canal de comunicación principal para rellenarlos con información auxiliar. Así, los resultados de los experimentos de Stroop sirven para configurar la capacidad de

los canales de comunicación auxiliares, que como se ha visto, se corresponde básicamente con la capacidad de la memoria de trabajo.

En el experimento de Stroop se verifica la existencia de una importante distinción entre las tareas importantes y las menos importantes para el procesador cognitivo, las cuales poseen distintos grados de prioridad. Siguiendo con la analogía existente entre la informática y el procesador humano, se puede hablar de la existencia de varios hilos de computación a los que se asignan distintas prioridades de ejecución.

También hay que destacar la existencia un activo proceso de selección por parte de la atención ante la existencia de estímulos secundarios. Se puede observar como los estímulos supuestamente irrelevantes (las palabras de la conversación de segundo plano) son percibidos, seleccionados e incluso procesados por la atención inconsciente. Nótese que si alguna de las palabras de la conversación secundaria es relevante para el sujeto, ésta pasará inmediatamente a la atención consciente, la cual dejará de atender a la conversación primaria.

Sabiendo que un sujeto atiende a estímulos procedentes de canales secundarios –aunque su atención consciente se encuentre centrada en los estímulos que provienen de un canal principal– es posible emplear dichos canales secundarios para favorecer la comunicación. Por ejemplo, es posible utilizar un canal de sonido con información permanente que ayude al usuario a finalizar la tarea que está realizando, aún cuando el usuario no se encuentra atendiendo directamente al mencionado canal de sonido. Este es por ejemplo, el mecanismo básico empleado por DEVA para ayudar a los usuarios que aún no han superado las primeras curvas de aprendizaje del sistema.

El empleo de sonido no es la única técnica mediante la cual DEVA puede transmitir información auxiliar, ya que los canales de comunicación visuales también son válidos. Es por ello que en determinados diálogos interactivos, DEVA emplea *widgets* especiales basados en texto deslizante o texto fundido, los cuales son colocados en el propio canal de comunicación principal de GADEA para que los mensajes a transmitir mediante los textos se encuentren siempre presentes en la interfaz, sin llegar a ocupar un lugar destacado dentro de la composición visual. Con ello se pretende que el canal secundario no se convierta a su vez en un canal principal. Mediante esta técnica es posible transmitir información relevante sobre el modo de operar recomendado para realizar la tarea prevista el canal principal, emitiendo mensajes de forma corta y sesgada. De este modo el usuario centra su atención consciente en la tarea principal mientras que por medio del canal secundario su atención inconsciente es alimentada con los mensajes mencionados.

Esta técnica ha sido probada con éxito en uno de los proyectos del programa de investigación Inxena. En este proyecto, desarrollado por Aparicio (1999), consiste en un sistema de edición de anuncios publicitarios para Internet capaz de integrarse tanto en sistemas de conversación o *chats* como en páginas *web*. Cada uno de los anuncios

desarrollados con esta herramienta dispone de una serie de palabras clave que lo identifican, las cuales son almacenadas en una base de datos para su consulta por medio de agentes inteligentes software. Dichos agentes se cargan en el *chat* o en el explorador de Internet del usuario y observan el tipo de información empleada por el usuario en sus conversaciones y búsquedas en Internet, intentando detectar sus preferencias en cuanto al producto que le puede ser de interés. Una vez encontrado éste, el anuncio correspondiente es transmitido por un canal de comunicación secundario. De este modo, los anuncios transmitidos son aquellos con la configuración semántica más parecida a aquella empleada por el usuario en el momento actual, reduciendo así la carga cognitiva de la atención del usuario y disminuyendo la probabilidad de que un anuncio sea rechazado por el filtro impuesto por la atención.

14.3 Atención Continuada

Cuando el estímulo transmitido a través de un canal de comunicación logra ser percibido de forma consciente por la atención usuario, queda todavía pendiente la tarea más difícil y complicada del proceso de transmisión de información, la cual no es otra que mantener la atención del sujeto sobre la el estímulo todo el tiempo que sea necesario para cumplir el propósito encomendado a dicho estímulo.

Dentro del estudio de la atención existe un apartado de especial importancia dedicado a la investigación del mecanismo que el procesador humano emplea para el mantenimiento de la atención sobre un estímulo determinado. Este mecanismo es una forma especial de atención conocida como vigilancia. El estudio de la vigilancia comenzó en Inglaterra a partir de los problemas de atención que surgieron entre los operadores de radar durante la Segunda Guerra Mundial. En las largas y aburridas horas de guardia, los operadores notaban que al cabo de un tiempo se cansaban rápidamente y disminuía su capacidad para detectar las señales de radar que rebotaban sobre los aviones enemigos.

A partir de estas observaciones Macworth (1950) desarrolló algunos experimentos para investigar los problemas asociados con el mantenimiento de la atención durante largos períodos de tiempo. Para ello diseñó un reloj dotado de una manecilla, la cual se movía a intervalos regulares aunque, de vez en cuando, hacía una doble pulsación. El objetivo de sus experimentos era precisamente hacer que los voluntarios observasen con cuidado el giro de la aguja del reloj y notificasen al experimentador cada vez que percibían el evento de la doble pulsación. Para evitar toda referencia espacial durante la ejecución del experimento, la esfera del reloj era totalmente blanca, sin señales de ninguna especie.

Los resultados registraron una caída total de la atención transcurrida la primera media hora de experimentación. A partir de este momento los observadores registraron solo un 25% de las dobles pulsaciones emitidas. El problema planteado a raíz de estos resultados era determinar si la caída del rendimiento se debía a una fatiga del sistema sensorial o a un fallo en la atención. La respuesta a esta incógnita vino

de la mano de las investigaciones de Broadbent (1971) y de Hilgard (1978), quienes demostraron que la razón de la caída del rendimiento no era debida a la fatiga de los órganos sensoriales sino a un cambio en la disposición del observador para dar respuesta a la señal. Al parecer existe un cierto grado de vigilancia óptimo y el usuario necesita cierto tiempo para alcanzarlo. Una vez superado este grado máximo, el procesador cognitivo deja de prestar atención al evento aún cuando los órganos sensoriales siguen transmitiendo información de forma correcta y continua.

A medida que la estimulación de la atención crece, ésta se centra cada vez más en el estímulo principal, mientras que otras informaciones son ignoradas. Esto significa que niveles crecientes de estimulación de la atención pueden ser contraproducentes y llevar a peores rendimientos. La relación entre la estimulación de la atención y el rendimiento se conoce como la *Curva de Yerkes-Dobson* [Brehm y Self (1989)]. La representación gráfica de esta función puede apreciarse en la Figura 40. En dicha figura se observa como a medida que se estimula la atención, los rendimientos cosechados aumentan en una buena proporción hasta llegar aun punto medio en el que se alcanza el máximo rendimiento posible. A partir de este momento, cualquier estímulo adicional de la atención será contraproducente y conducirá a una disminución progresiva en el rendimiento.

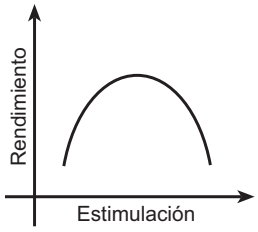


Figura 40. Curva de Yerkes-Dobson.

Estos resultados indican en que el prestar intensa atención a un estímulo externo requiere un esfuerzo mental extra, el cual no es realizado si no se favorece de alguna manera. Existen ciertas técnicas de estimulación psicológica que se realizan de forma automática en situaciones de tensión, tales como la adopción de ciertas posturas corporales o la tensión de ciertos músculos. Cuando se presta atención a algo, normalmente se orientan los órganos sensoriales hacia la fuente del estímulo, ya sea para que el objeto proyecte su imagen dentro de la retina (estímulo visual) o para captar un sonido con más claridad (estímulo auditivo).

Si a un sujeto se le proporciona una señal de alerta justo antes de que se produzca el estímulo al que ha de prestar atención, el rendimiento aumentará considerablemente. Además se ha demostrado que el tiempo óptimo para emitir esa señal se sitúa entre las 500 y las 200 milésimas de segundo antes de que se produzca el estímulo. Si por el contrario se da la señal y se prolonga la llegada del estímulo más allá del mencionado período de tiempo (entre las 500 y las 200 milésimas de segundo antes de producirse el evento), el nivel crítico de la atención se extingue y el rendimiento de este proceso se ve seriamente perjudicado.

En DEVA, el resultado de estos estudios sobre la vigilancia se concreta en el empleo de estímulos especiales que notifican al usuario la existencia de un nuevo diálogo interactivo o canal de comunicación en la interfaz. Estas señales se transmiten a través de canales de comunicación secundarios respecto al canal de comunicación en el que se emite un nuevo diálogo interactivo y se establecen por medio de un modo de interacción inverso a aquel correspondiente a la señal principal. Así, si el canal de comunicación es sonoro, la señal de alerta se enviará a través del de comunicación visual (el cual hace las veces de canal de comunicación secundario) empleando por tanto una señal visual y viceversa, ya que cuando el canal de comunicación principal es el visual, la señal de alerta se transmitirá a través del canal de comunicación sonoro. De este modo, ante toda novedad en el canal de comunicación activo, por pequeña que ésta sea, el usuario es advertido a través del canal de comunicación secundario, permitiendo que este proceso de percepción de nueva información no se realice por medio de un sistema de vigilancia activa, sino empleando un mecanismo de vigilancia pasiva que libere gran parte de la carga cognitiva de la atención.

Estas señales de alerta se envían por el canal de comunicación correspondiente medio segundo antes de que la señal principal sea transmitida. Naturalmente, la señal de alerta no dura más de medio segundo de tal modo que ésta se extingue antes de que la información principal con la que se relaciona esta señal aparezca en el canal de comunicación principal. Para aumentar la efectividad de estas señales de alerta, su contraste aumenta en el lapso de tiempo en el que éstas son transmitidas. Así, las señales auditivas ganarán en intensidad y volumen, mientras que las señales visuales ganarán en intensidad de brillo y tamaño, resaltando aún más su existencia para dar cuenta al usuario del evento que está a punto de producirse.

El empleo de estas señales de alerta se hace imprescindible dado que la separación total entre la funcionalidad de toda aplicación GADEA y su interfaz implica un desconocimiento total acerca de la duración de determinados procesos ejecutados a partir de la invocación de otros tantos procesos de usuario. Algunos de estos procesos tendrán una duración despreciable en términos perceptivos y por lo tanto la información obtenida o requerida a partir de ellos será transmitida de forma instantánea a través del canal de comunicación activo. Sin embargo, en otros casos la duración de los procesos puede ser los suficientemente elevada como para que el retraso en el despliegue de información a través del canal de comunicación elegido para tal fin provoque un cambio en la disposición del receptor de la información al sobrepasar el punto óptimo de la Curva de Yerkes-Dobson. Es por ello que es necesario advertir al usuario de la finalización de dicho proceso para que éste pueda retomar la ejecución de su tarea –suspendida al invocar al proceso de usuario– con el mayor grado de atención posible.

14.4 Contraste Visual

De los cinco sentidos que posee el ser humano (vista, olfato, tacto gusto y oído), la vista es considerado como el más importante desde el punto de vista del diseño de interfaces de usuario, ya que cerca del 82% de los estímulos percibidos se obtienen a través de este sentido sensorial. La vista ha sido además el favorito de psicólogos en sus experimentos sobre el estudio de la percepción ya que su formidable precisión y la uniformidad de respuestas obtenidas por diversos observadores ante un estímulo común hacen de este sentido el ideal para su uso en este tipo de investigaciones.

La vista, al contrario de lo que ocurre con el oído, es un sentido en el que los dos órganos participan de forma sincronizada y deben orientarse continuamente hacia la posición de donde proviene el estímulo para que el objeto de interés proyecte su imagen en la retina. Este proceso se ha estudiado experimentalmente sometiendo a los observadores a tareas de rastreo visual. En estos experimentos matrices se suelen crear matrices compuestas por diversos objetos (letras o iconos), en donde el observador deberá buscar un objeto concreto a petición del experimentador. En la Figura 41 se pueden observar dos ejemplos de este tipo de matrices en donde al usuario se le puede pedir que busque por ejemplo la letra *d* o el icono .

Figura 41. Matrices de búsqueda en tareas de rastreo visual.

Al igual que ocurre en el caso de otras tareas realizadas por el procesador humano, en el grado eficiencia alcanzado en el proceso de búsqueda influye notablemente la práctica desarrollada por los observadores en la realización de esta tarea, observándose que con la práctica los sujetos reducen sensiblemente el tiempo de búsqueda. Sin embargo, en las conclusiones estos experimentos destaca el hecho de que el contenido propia matriz también presenta gran influencia en la eficiencia de la búsqueda. En concreto, resulta mucho más difícil localizar un objeto dentro de la matriz cuando éste está rodeado por otros de características similares. Nótese por ejemplo la dificultad que se presenta al intentar percibir la letra *d* o el icono en las matrices de la Figura 42 al poseer cada una de ellas gran cantidad de objetos similares.

Figura 42. Matrices de búsqueda cuyos objetos son muy similares entre sí. Nótese la dificultad presente al individualizar cada uno de los objetos de estas matrices con respecto a los objetos incluidos en las matrices de la Figura 41.

Como podemos apreciar, las similitudes y diferencias entre señales visuales, tanto a un nivel físico como a un nivel semántico, pueden alterar la habilidad de un sujeto para dirigir su atención hacia un aspecto de la información que intenta percibir. Es por ello que es necesario emplear un máximo contraste entre objetos visuales transmitidos a través de todos los canales de comunicación de idéntica tipología, de modo que se facilite en la medida de los posible la percepción individual de los objetos. De este modo, todos aquellos objetos visuales que han de ser transmitidos juntos dentro de un mismo canal de comunicación, son configurados para que contengan el mayor número posible de características físicas distintas, intentando con ello individualizar su aspecto con respecto a los objetos vecinos.

Aunque el procesador humano tiende a agrupar los objetos gráficos que cumplen con una tarea similar (por ejemplo los botones de alineación en un procesador de texto incluidos en la Figura 43), para facilitar la búsqueda de un objeto por parte del usuario, estos objetos visuales deben diseñarse desde un principio para que adopten formas distintas, premisa que obviamente ha de ser asumida por los diseñadores de la aplicación a la hora de incluir dicha representación visual en la base de conocimiento de la misma. En todo caso, DEVA realiza una pequeña adaptación de bajo nivel modificando ligeramente los tonos de color y brillo empleado en aquellas representaciones gráficas que han de transmitirse muy juntas por medio del mismo canal de comunicación visual. Con ello se logra que dichos objetos visuales se diferencien lo más posible de aquellos objetos que tengan en su vecindad.

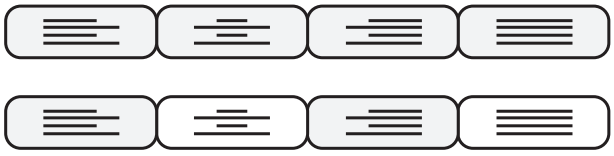


Figura 43. Posibles iconos empleados por una aplicación de procesador de texto para mostrar sus distintas opciones de alineación [Arriba]. En este caso, la diferenciación entre los iconos es difícil y requiere un esfuerzo mental notable. Para facilitar la labor del usuario en la diferenciación, la imagen inferior incorpora una pequeña variación del tono de gris de los iconos.

Sin embargo, no todos los objetos visuales de un canal de comunicación visual disponen de una representación iconográfica sobre la cual pueda aplicarse el proceso de adaptación descrito. En realidad los objetos visuales dotados de representación iconográfica suelen ser minoría en una interfaz. Por ejemplo, los campos de edición de texto, menús, etiquetas, etc. pueden considerarse como elementos visuales de un interfaz y sin embargo carecen de representación visual. En los casos mencionados, DEVA aplicará el proceso de incremento de contraste visual al fondo sobre el que se sitúan los elementos, realizando

pequeñas variaciones sobre su brillo y tono, cuyo proceso de elaboración será descrito en el capítulo 18: *Adaptación Cromática*. De este modo, tal y como se puede apreciar en el ejemplo de la figura Figura 44, es posible reforzar el contraste entre los elementos consecutivos de un menú, proporcionado un mecanismo sencillo para la individualización de sus elementos constitutivos que favorezca un proceso de búsqueda efectivo.

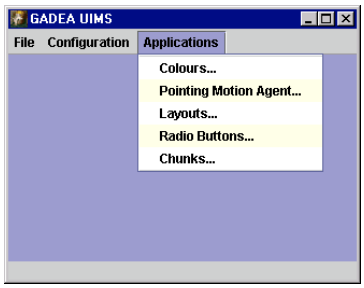


Figura 44. Incremento de contraste visual entre elementos de un menú DEVA para facilitar la búsqueda e identificación de dichos elementos (consultar versión en color en el capítulo 24: *Imágenes en Color*, p. 445).

El grado de contraste a alcanzar entre dos objetos consecutivos en el canal dependerá directamente de la precisión visual del usuario activo de la aplicación. Naturalmente, mientras peor sea la capacidad de un usuario para percibir objetos visuales, mayor será el grado de contraste perseguido por DEVA.

En la mayoría de los casos, las letras o iconos que rodean al objeto visual buscado en las matrices de rastreo visual son percibidos por los observadores como ruido o simplemente no las ven. Según Neisser (1981), esto implica que los observadores no están procesando la información letra a letra o icono a icono, sino que están tomándola en bloques. Dentro de estos bloques, solo destacan las características distintivas del objetivo, las cuales sí son percibidas. Esto viene dar mayor énfasis a la necesidad de un fuerte contraste entre objetos visuales, ya que la similitud entre el objetivo y el resto de la información contenida en el bloque hace más difícil las tareas de rastreo visual.

Si DEVA no efectuara este proceso de aumento de contraste para objetos visuales muy cercanos entre sí, se estaría forzando la capacidad de la atención visual del usuario pues se obligaría a que ésta realizase un gasto innecesario de energía mental en los procesos secuenciales de búsqueda de elementos visuales, ya que ésta búsqueda se realiza en un océano de objetos de características similares.

15

15 El Procesador Perceptivo

Las cosas no se ven como son, las vemos como somos

Askasubi

15.1 La Percepción

Con independencia del modelo empleado para explicar la limitación de la capacidad humana en el procesamiento de información (modelo de cuello de botella o modelo de capacidad), la psicología experimental ha venido demostrando que los humanos no podemos atender de forma eficaz a varias tareas a la vez, por lo que la atención humana parece tener una naturaleza fundamentalmente selectiva.

Heinz (1969) define la atención humana en sus trabajos como el aspecto selectivo de la percepción, con el carácter limitativo que dicha definición conlleva. Será por tanto la atención quien controle que el procesador humano no se exceda de un determinado límite, bien porque ese límite represente la capacidad máxima de procesamiento de información por parte del individuo –como propone el modelo de cuello de botella– o bien debido a que dicho límite represente nuestra la

incapacidad para repartir los recursos mentales entre varias actividades simultáneas –tal y como propone el modelo capacidad.

Para ello, la atención analiza constantemente los estímulos recibidos, tanto a nivel de señales físicas como a nivel de señales semánticas, actuando como selector, de modo que solo aquellos estímulos que representen algún tipo de relevancia para el procesador pueden pasar a la consciencia para un tratamiento en detalle. Dado que la captación de los estímulos es lo primero que realiza el procesador humano, se ajustará el engranaje del resto del sistema en función de la calidad y precisión de este proceso perceptivo.

La llegada de una serie de estímulos a la atención, para su procesamiento y selección se realiza a través del procesador perceptivo. Este procesador se encarga de recoger la información obtenida por todos y cada uno de los órganos sensoriales (vista, oído, olfato, tacto y gusto) para su tratamiento por parte del procesador cognitivo.

La actividad del procesador perceptivo comprende la aplicación de un proceso de toma de decisiones mediante el cual se deduce el objeto o la sensación recibida a través del estímulo. Por medio de las experiencias previas y del aprendizaje, se obtiene información adicional sobre lo percibido, pudiendo establecerse relaciones entre las distintas sensaciones. Una captación errónea de los estímulos percibidos llevará asociado un tratamiento equivocado por parte de la atención, la cual puede desechar estímulos importantes captados de forma incorrecta y dejar pasar a la consciencia estímulos totalmente irrelevantes pero que han sido maquillados como importantes tras un proceso fallido de percepción.

Resulta inapropiado hacer referencia al proceso de la percepción humana sin mencionar los estudios de la *Gestalt*. Esta escuela alemana y las posteriores ramas en las que derivó, ha sido sin lugar a dudas la más relevante e influyente en el campo de la psicología perceptiva, pues hace de su estudio su bandera. La denominación *Gestalt* es de origen alemán y se traduce al castellano por *Forma* o *Estructura* por lo que a esta escuela se la conoce también como *Escuela de la Forma*.

La *Gestalt* fue fundada por el físico alemán Wertheimer (1880 – 1934) como reacción contra los fundamentos teóricos de la psicología del siglo XIX: el Atomismo y el Elementalismo principalmente. Para la *Gestalt*, la realidad entendida como un sistema de partes que se asocian resulta de una simplicidad y falsedad evidente. Según los partidiarios de esta escuela, la atención no percibe átomos psíquicos, sino estructuras organizadas o totalizadas, es decir, se trata de *formas*. Las ideas expuestas por los defensores de la *Gestalt* llamaron poderosamente la atención de la comunidad científica del momento. Esta escuela produjo una verdadera fascinación y ya para 1927, en el congreso de psicología de Bonn se pudieron contar 535 ponencias relacionadas con el estudio de la forma. La *Gestalt* comprueba sus teorías experimentalmente llegando a realizar numerosos trabajos sobre la percepción y afirmando que los hechos psíquicos son formas, es decir, unidades orgánicas que se individualizan y que se limitan al campo espacial y temporal de la percepción. En este sentido, son los seguidores de la *Gestalt* los

primeros en negar la explicación aditiva en la que el todo es igual a la suma de sus partes; proponiendo una explicación novedosa en la que el todo es *algo más* que la suma de sus partes, entendiendo la noción del todo (o forma) como un sistema auto-subsistente.

Uno de los derivados de la *Gestalt* que mayor éxito ha alcanzado ha sido la *Escuela de Graz*, representada principalmente por Meinong, Witasek, Höfler y Benushi. Esta escuela también es conocida por el nombre de *Escuela Dualista*, ya que separa el fenómeno perceptivo en dos partes claramente diferenciadas. Por un lado se encuentra la *sensación*, la cual proviene del emisor y por el otro lado se encuentra la *forma*, la cual no es un elemento propio de la sensación sino que constituye un producto de la mente, una representación ajena a los órganos sensoriales. Por ejemplo, en el caso de la percepción de una melodía, el oído humano percibe los sonidos, no la melodía en si misma. En ciertas condiciones, los sonidos pueden variar sin que la melodía cambie pues, por ejemplo, se puede oír la misma melodía en *Do Mayor* o en *Sol Mayor*. La conclusión a la que llega la Escuela de Graz respecto a este hecho es que son dos los objetos captados por el sujeto: los sonidos por una parte (sensación) y la melodía (forma) la otra.

En el campo visual, Rubin diseñó un experimento ya clásico para demostrar esta teoría, empleando una figura ambigua en la cual se pueden percibir dos formas totalmente distintas [Lindzey et. al. (1985) p. 115. *Percepción de la Forma*]. Se trata de la Figura 45, en donde se puede percibir o bien una copa o bien un par de perfiles humanos enfrentados. En la ilustración, el negro y el blanco pueden ser percibidos indistintamente como fondo o como forma. En este caso los colores representan la sensación y las figuras las formas.

Figura 45. Copas o rostros. En función de cómo se perciban las sensaciones (colores blanco y negro) las formas percibidas serán distintas.[Fuente: Lindzey et. al. (1985)]

Los estudios de la forma de Rock (1985) defienden la teoría de que la percepción de formas requiere siempre de una atención consciente. Esto significa que sin la atención no puede lograrse ningún tipo de percepción. Rock propone la idea de que la atención no se dirige al espacio de forma indiscriminada, sino que actúa sobre objetos concretos. La información contenida en dichos objetos sería analizada y procesada por la atención de modo que solo llegasen a la consciencia los datos verdaderamente relevantes para el sujeto. De acuerdo con éste modo de operar, el proceso previo de análisis de información desarrollado por la atención humana se lleva a cabo en dos fases: una fase inicial de análisis global de la situación, en la que se seleccionan los

objetos relevantes, seguida de una fase final en la que se analizan en detalle los objetos seleccionados por la primera fase.

De este modo la atención humana realiza un análisis desde un punto de vista global y después –si procede– desde un punto de vista local. Esto significa que la atención analiza la información general recibida y luego procede a extraer de ella la información detallada que requiere para efectuar un análisis completo. De este modo, el procesador central humano (siguiendo la analogía impuesta por el Sistema P) descargaría parte de la carga cognitiva en el procesador perceptivo en donde la atención actuaría a modo de filtro.

Los estudios de Daneman y Mericle (1996) y de Merikle (2000) han confirmado que el ejercicio de análisis global realizado en la primera fase de la percepción puede inhibir la ejecución de la segunda fase de análisis local. Este fenómeno acontece en situaciones en las que existe información global interferente. En el caso contrario, cuando existe información local irrelevante, la presencia de la citada información no afectará en ningún modo a la fase de análisis global, ya que dicha fase se realizará primero.

Los estudios efectuados por los citados investigadores demuestran que existe una clara supremacía del análisis global en la percepción de los estímulos. El filtro impuesto por la primera fase de percepción hace que el individuo no malgaste energías en analizar información irrelevante para sus intereses. A menos que la información de carácter general enviada a través de los diversos canales de comunicación de una aplicación sea relevante para el usuario–y por lo tanto supere la primera fase de la percepción y filtrado– la carga semántica incluida en el detalle de la información transmitida por dichos canales no llegará a su destino, ya que el procesador perceptivo del usuario la habrá rechazado por irrelevante o lo que es peor aún, ni siquiera la habrá tomado en cuenta.

15.1.1. Señuelos y Predadores

La implicación directa que tiene este modo de operar para el agente interactivo representado por DEVA será la de aprovechar, de la mejor manera posible el ancho de banda de los canales de comunicación disponibles, intentando transmitir por ellos información de carácter general, sin saturarlos de una información detallada que no será aprovechada. Todo ello, dentro del margen de maniobra que permita el nivel sintáctico en el que se mueve GADEA.

Para alcanzar este objetivo, DEVA clasifica los *chunks* a transmitir a través de cualquier canal de comunicación en dos categorías de objetos: *objetos señuelo* y *objetos predador*. Los objetos señuelo servirán –como indica su nombre– para capturar la atención del usuario y llevarla a un terreno favorable para el intercambio de información, lugar en donde actuarán los objetos predador, quienes darán cuenta de ella.

Los objetos señuelo serán *chunks* capaces de transmitir una información de alcance global, sencilla y fácil de interpretar. Dada la existencia del filtro perceptivo bifásico recién comentado, no tiene

sentido emplear *chunks* provistos de información extremadamente detallada como objetos señuelo, si la información en ellos contenida no será percibida por el usuario en caso de que el *chunk* sea desechado durante la primera fase de percepción.

Como consecuencia, durante las primeras tomas de contacto del usuario con una aplicación concreta o con cada uno de sus procesos de usuario y diálogos interactivos, DEVA empleará la totalidad o al menos una gran parte del ancho de banda disponible en el canal de comunicación activo para transmitir objetos señuelo, de modo que éstos abarquen el máximo espectro posible de las expectativas del usuario. Por ancho de banda entendemos naturalmente el mínimo entre la capacidad máxima del canal de comunicación y la capacidad máxima del procesador humano del usuario destinatario de la información.

Dada la alta carga semántica concentrada en estos objetos señuelo, su transmisión masiva durante las primeras fases de la toma de contacto con una aplicación o documento, facilitará la comprensión de la base de conocimiento con la se enfrenta el usuario al actuar éstos objetos como resumen del contenido de dicha base de conocimiento. Al aumentar las posibilidades de que uno de estos objetos señuelos logre capturar la atención del usuario, existen más posibilidades de que éste continúe actuando sobre ella y por lo tanto progrese en el proceso de exploración de la misma [Furnas (1997)].

Si el usuario muestra interés por uno de los objetos señuelo, el canal de comunicación pasará a transmitir los objetos predador correspondientes al objeto señuelo seleccionado por el usuario, intentando cubrir con ellos todo el ancho de banda disponible. Estos objetos predador –que ya poseen el control de la atención del usuario humano captada gracias al objeto señuelo correspondiente– proporcionarán la información detallada que el usuario necesita, conduciéndolo hasta la consecución de los objetivos globales del sistema.

En función del tipo de *chunk* correspondiente a un objeto señuelo, existirán situaciones en las que será imposible descomponerlo en objetos predador directamente. Por ejemplo, si el objeto señuelo dispone de gran carga semántica, la descomposición se hará entonces en un conjunto de objetos señuelo de carga semántica baja o moderada en lugar de hacerlo directamente en objetos predador. Cada uno de estos objetos señuelo hijos del objeto señuelo original podrán descomponerse a su vez en otros objetos señuelo o en objetos predador, creando así un árbol semántico en el cual, en la raíz estará la información concentrada y en las hojas se encontrará la misma información pero una versión muy detallada. Este esquema sigue de manera fiel el modelo empleado para la creación de *chunks* descrito en el apartado 13.2 (*La* Jerarquía de Memorias) del capítulo 13 (*El Procesador Humano*) y se puede apreciar claramente en la Figura 46. Nótese que el número máximo de objetos en cada nivel se calcula dinámicamente en tiempo de ejecución y en función de la capacidad del canal de comunicación y de la propia capacidad del usuario destinatario de la información.

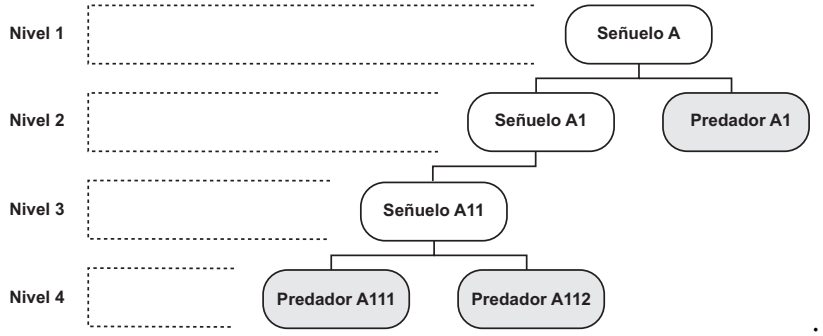


Figura 46. Estructura jerárquica de objetos señuelo y predador. En blanco están representados los objetos señuelo y en gris claro sus correspondientes objetos predador. Nótese el caso del Señuelo A, quien descompone su carga semántica en el Señuelo A1 y el Predador A1.

Al descomponer un canal de comunicación siguiendo la pauta de las dos fases en las que se divide el percepción humana, se hará un uso realmente óptimo de su ancho de banda, además de proporcionar una estructura natural a al diseño de los diálogos interactivos a nivel sintáctico. Nótese que si se rellena el ancho de banda de un canal de comunicación de forma indiscriminada con objetos señuelo y con objetos predador, se estará desperdiciando todo aquel espacio del ancho de banda empleado por los últimos, ya que en una primera fase de la percepción el usuario se fijará únicamente en los objetos señuelo, provistos de información de carácter general.

Figura 47. Canal de comunicación con capacidad para 4 *chunks*. En el momento I se envían 2 objetos señuelo (A y B) y dos objetos predador (A1 y A2).

En la Figura 47 se puede apreciar un caso evidente de desaprovechamiento de un canal de comunicación. Aunque el canal tiene capacidad para transmitir cuatro *chunks* sólo los objetos señuelo A y B tienen alguna posibilidad de ser percibidos durante el momento I. Los objetos predador A1 y A2 que detallan al *chunk* A tienen muy poca probabilidad de ser percibidos en un primer momento, ya que de tener algún interés para el usuario, éste centrará su atención primero en el objeto señuelo A, el cual contiene información de carácter general sobre el tema en cuestión. Sólo si el usuario percibe dicho objeto señuelo y está interesado en la información por él transmitida, mostrará algún interés por los predadores A1 y A2. En este ejemplo, los objetos con mayor probabilidad de ser percibidos serán los señuelos A y B.

Si se desea aprovechar al máximo la limitada capacidad de este canal de comunicación (cuatro elementos), una buena gestión de dicho canal plantearía transmitir cuatro objetos señuelo en lugar de sólo dos. Esto ocurre en la Figura 48 en donde se transmite en un primer momento los objetos señuelo A, B, C y D. De este modo, la probabilidad de que cualquiera de estos objetos sea percibido rondará el 25%. Sólo cuando el usuario muestre algún interés por un objeto señuelo se pasará a transmitir los correspondientes objetos predadores por el canal de

comunicación. Este es el caso del momento II en la misma figura, en el que se empiezan a transmitir los predadores A1, y A2 ante el interés mostrado por el usuario con respecto al objeto señuelo A.

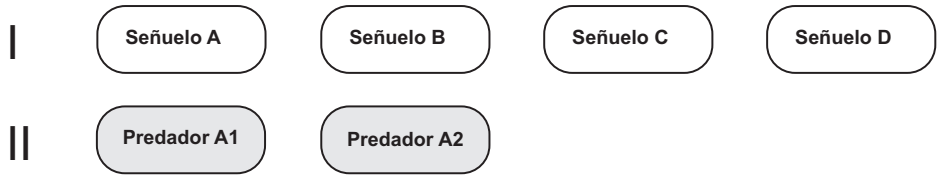


Figura 48.El canal de comunicación de la Figura 47 incrementa su rendimiento al transmitir cuatro objetos señuelo en el momento I. Sólo si el usuario muestra interés por el objeto señuelo A se pasará a transmitir los objetos predador A1 y A2. Esto ocurrirá en el momento II.

Esta técnica se traduce en la descomposición de la interfaz en una estricta jerarquía semántica de canales de comunicación, en cuya capa superior se encontrará la información general, todo lo cual permitirá guiar al usuario rápidamente hacia información más específica.

El diseño y desarrollo de la tecnología de comunicación basada en objetos señuelo y objetos predador ha sido probado por medio del programa de investigación Tirsus y su serie de proyectos relacionados. Este programa de desarrollo nace en 1998 en Laboratorio de Tecnologías Orientadas a Objetos del Departamento de Informática de la Universidad de Oviedo, teniendo como objetivo primordial el estudio y análisis de distintos mecanismos de navegación hipermedia sobre bases de conocimiento dinámicas, así como las bases para el establecimiento de una comunicación eficaz entre dichas bases de conocimiento y usuarios de todo tipo. En los proyectos desarrollados por este programa, se emplea como base de conocimiento la Historia, ya que por un lado, su naturaleza dinámica en continua evolución permite ensayar distintos modelos de navegación y por el otro lado, su interés por parte de diversos colectivos de usuarios permite analizar los requerimientos de diversos modelos de interacción en función de las necesidades de cada uno de ellos.

Dentro de los proyectos pertenecientes a este programa, el más significativo con respecto al modelo de objetos señuelo y objetos predador es el proyecto Tirsus II [Sánchez (2000a)], una herramienta multimedia que recrea la historia antigua de Asturias, desde el principio de la Edad de Hierro hasta el final del Imperio Romano.

En varias de las secciones que componen esta herramienta, el volumen de información a transmitir es tal que el canal de comunicación disponible, restringido a una sola pantalla por nodo resulta demasiado pequeño. Como solución preliminar a este problema, se reestructuraron los nodos de forma tal que las diversas secciones que aparecían en los nodos originales se transformaron en nuevos nodos. Una vez reestructurados los nodos el problema persistía ya que no se podía combinar toda la información que se deseaba ofrecer con un diseño estético y de impacto visual, problema identificado como patrón por Schwabe et. at. (1998).

La solución pasó por el diseño de varios objetos señuelo que pasaron a ocupar el canal de comunicación principal, seguido de varios objetos predador accesibles a petición del usuario sólo cuando los objetos señuelo son capaces de capturar la atención del usuario [González et al. (2000c)].

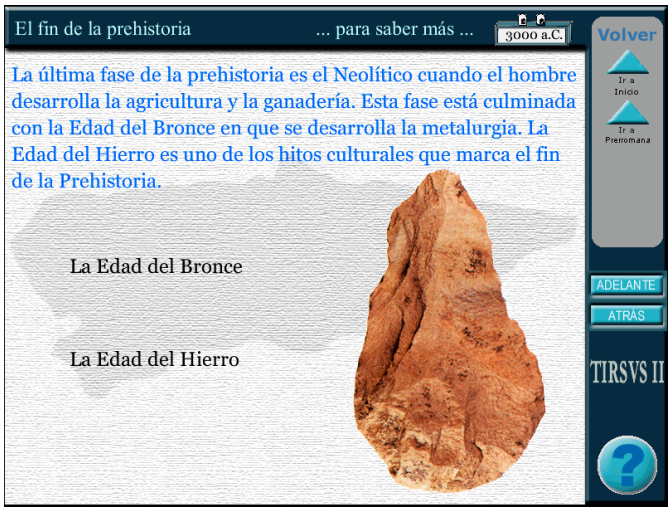


Figura 49. La información visual y la información sonora poco detalladas actúan como objetos señuelo en Tirsus II (consultar versión en color en el capítulo 24: *Imágenes en Color*, p. 445).

En la Figura 49 se puede observar el diseño de uno de estos nodos en Tirsus II. La información general se transmite por medio del canal de comunicación visual (primario) y de un canal auxiliar sonoro (secundario) mediante el cual se complementa la información transmitida por el canal de comunicación principal. Si el usuario está interesado en la información aportada por el nodo y desea obtener un conocimiento más profundo sobre el tema, elegirá entonces la opción …*para saber más*… De este modo el usuario obtiene el objeto predador correspondiente, el cual complementará la información proporcionada por los señuelos transmitidos en primer lugar, empleando como soporte metafórico un panel de información auxiliar, el cual puede contemplarse en la figura Figura 50 [Sánchez et. al. (2000b)].

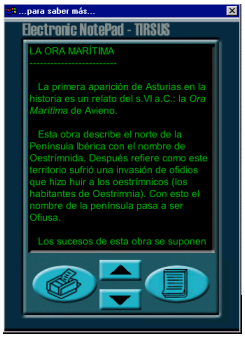


Figura 50. Objeto predador en Tirsus II encapsulado en un canal de comunicación auxiliar (consultar versión en color en el capítulo 24: *Imágenes en Color*, p. 445).

Tirsus II ha sido sometido severos tests de usabilidad con una muestra representativa de la población de usuarios de aplicaciones hipermedia, clasificada por edades y por grados de conocimiento en base al uso de aplicaciones informáticas y en base al uso de la propia base de conocimiento [Sanchez et. al. (2000)]. El resultado obtenido con respecto al modelo de objetos señuelo y objetos predador fue satisfactorio ya que los usuarios hicieron uso del mismo cuando la situación planteada por las pruebas así lo requiso, si bien hay que destacar que un grupo numeroso del colectivo de usuarios con un grado de conocimiento bajo en informática no llegaron a emplear los objetos predador por desconocer como acceder a los mismos.

Esta limitación es superada por GADEA al simular el comportamiento de un usuario experto con DEVA. De este modo, será el agente interactivo quien proporcione pistas al usuario humano acerca de cómo acceder a la información contenida en un objeto predador, bien a través del canal de comunicación principal o bien por medio de cualquiera de los canales de comunicación secundarios establecidos a tal fin.

15.1.2. Agrupación

Una de las particularidades del procesador perceptivo humano es su capacidad para analizar un conjunto de formas totalmente individuales como si se tratase de una sola forma más sencilla y global. Es decir, este procesador tiende a agrupar formas, simplificando el universo que le rodea mediante la construcción de niveles de abstracción superiores.

Del planteamiento anterior se deduce que los grupos de formas son candidatos a constituir objetos señuelo, actuando las propias formas como objetos predador. Esta técnica es perfectamente aplicable en el caso en el que el ancho de banda del canal de comunicación visual utilizado sea lo suficientemente amplia como para no racionalizar su uso mediante una aplicación estricta de la técnica propuesta, es decir, transmitiendo los objetos señuelo y los objetos predador a la vez.

Si se emplea el mecanismo de agrupación apropiado, puede transmitirse de forma conjunta un grupo formado por objetos predador dotados de características semánticas estrictamente individuales de tal modo que éstos pueden ser percibidos como un único objeto señuelo durante la primera fase de análisis perceptivo. Por ejemplo, en la Figura 39, todos los botones de radio del grupo, por proximidad y similitud pueden ser percibidos como un único objeto señuelo aún cuando cada uno de los botones actúa de forma individual como objetos predador.

La agrupación permite por tanto unificar objetos de carga semántica sencilla en grupos provistos de una carga semántica compleja, siguiendo una vez más el mecanismo de asociación basado en *chunks*. Al poder con agrupar varios objetos predador de bajo nivel en un único objeto señuelo de alto nivel, es posible reducir la cantidad de objetos transmitidos por medio de un canal de comunicación, con lo cual es posible superar su capacidad física, tanto en función de su ancho de banda como en base a la capacidad que tiene la memoria de trabajo del

usuario para absorber el volumen de información distribuida a través de dicho canal.

Supongamos por ejemplo que la capacidad máxima de una canal de comunicación se sitúa en cuatro elementos, siendo ésta la combinación del ancho de banda del canal y de la capacidad perceptiva del usuario activo de la información. En teoría, en una primera fase de transmisión solo es posible la distribución de cuatro objetos señuelo a través de dicho canal. Sin embargo, por medio de la agrupación de elementos, es posible empaquetar varios de esos objetos señuelo para crear otro de mayor nivel semántico. Supongamos en el ejemplo que en cada paquete se puedan incluir cuatro objetos señuelo.

En este caso, la capacidad efectiva del canal de comunicación pasará de los cuatro objetos previstos inicialmente a dieciséis, consiguiendo con ello un incremento del 400% en la capacidad del canal, de ahí la importancia de emplear técnicas de agrupación efectivas en la distribución de información a través de los distintos canales de comunicación empleados por cada aplicación GADEA.

Continuando con el ejemplo, cuando el usuario reciba los cuatro objetos señuelo de primer nivel, procederá a descomponer cada uno de ellos en los cuatro objetos señuelo que los integran, obteniendo los dieciséis objetos originales. Nótese como los objetos señuelo transmitidos en primer lugar actúan como verdaderos señuelos de los cuatro objetos predador que contiene.

De acuerdo con Tomlinson (1984) y Ballesteros (1994), el procesador humano tiende a percibir los elementos provenientes de un canal de comunicación como un bloque, relacionándolos y agrupándolos entre sí y agrupados por medio de las siguientes leyes básicas:

- **Regla de la Proximidad**: Según esta regla, se tiende al agrupamiento de todas aquellas formas que se encuentran relativamente cerca unas de otras. El cumplimiento de esta ley afecta especialmente a los grupos de botones de una interfaz, los cuales, por regla general, contienen información poco relevante por ser repetitiva, por lo que es conveniente agrupar estos *widgets* por bloques empleando esta regla, de tal modo que el usuario los perciba como si se tratase de uno solo. Esta regla es empleada por DEVA para convertir bloques de botones de radio o *checkboxes* en objetos señuelo.

  Por el lado contrario, esta regla atañe negativamente a los objetos predador representados por las opciones de los menús. Al situarse estas opciones de forma consecutiva en un área visual muy limitada, dichas opciones tienden a ser analizadas como un solo grupo, lo cual no el objetivo buscado, dado que es necesario conseguir el máximo contraste posible entre cada una de las opciones para facilitar su reconocimiento por parte de la atención, según se comentó en el apartado 14.4 (*Contraste Visual*). Para estos casos, DEVA logra la individualización de los objetos situado en zonas físicas cercanas aplicando el efecto contrario por medio de la regla de similitud que citaremos a continuación, ya que es imposible realizar lo propio con la regla de la proximidad.

Figura 51. La proximidad entre los elementos básicos de la forma (círculos negros), permite percibirlos como una unidad (regla de la proximidad). En este caso se perciben dos grupos de círculos en lugar de círculos individuales (izquierda y derecha).

- **Regla de la Similitud**: De acuerdo con esta regla, se agrupan aquellas formas que son de un mismo tipo o que comparten características comunes aún a pesar de que estas formas se encuentren en zonas alejadas entre sí. Esta regla es empleada por DEVA para agrupar los elementos que forman parte de un mismo *chunk*, dotándolos de atributos similares, tales como el color, la forma, el tamaño, la duración, el volumen, la fuente tipográfica etc. Aún cuando el diseño de un diálogo interactivo puede dificultar la agrupación de formas por proximidad, siempre es posible dotarlas de atributos comunes para formar un grupo, aunque estas formas se encuentren localizadas puntos diametralmente opuestos del canal de comunicación.
- **Regla del Destino Común**: Esta regla formaliza la tendencia a agrupar formas que se mueven en la misma dirección y a la misma velocidad. Por medio de esta regla, DEVA facilita la selección y arrastre de más de un objeto visual, ya que cuando el usuario desplace el cursor por la pantalla durante un proceso de arrastre, percibirá que todos los objetos seleccionados se mueven al unísono. De este modo, toda la colección de objetos seleccionados se convertirá automáticamente en un solo objeto dentro del modelo mental del usuario. Nótese que para el usuario será fácil controlar múltiples elementos de la interfaz que se muevan al unísono, ya que la simplificación de su esquema mental convertirá tales formas en una sola.

A pesar de la importancia que tiene la agrupación para GADEA, existen situaciones en las que su empleo es improcedente, principalmente cuando el objetivo buscado es eliminar toda posibilidad de agrupación de formas, destacando con ello las características peculiares de cada una de ellas.

En el apartado 14.4 (*Contraste Visual*) se llegó a la conclusión de que un máximo contraste entre las formas facilitaba el proceso selectivo de la atención. Es por ello que el proceso de agrupación se evita en situaciones en las que prima la captación de detalles sobre el procesamiento global de grandes cantidades de información. Otra situación que requiere de una ruptura intencionada del proceso de agrupación de formas, viene dada cuando tiene prioridad la percepción de ciertos atributos de las formas sobre la percepción de las propias formas. Los trabajos de Duncan (1977), demostraron que resulta más sencillo percibir dos atributos de una misma forma que si estos atributos pertenecen a formas distintas. Esto implica que en estados avanzados del proceso de descomposición de *chunks*, cuando la mayor

parte de la información transmitida se corresponde con objetos predador, en donde la información transmitida debe ser analizada con gran detalle, los procesos de agrupación de formas no son aplicados por DEVA, pues dificultarían la percepción de los atributos (detalles) de las estructuras individuales.

Cuando se transmiten objetos predador, su representación se hace por medio de formas individuales no agrupables, a diferencia de los objetos señuelo, los cuales son transmitidos a través del canal de comunicación por medio de grupos. Dado que para aprovechar al máximo el ancho de banda disponible en un canal de comunicación se transmite más de un objeto predador de forma simultánea, es de vital importancia evitar que dichos objetos predador puedan generar algún tipo de agrupación en el procesador perceptivo del receptor, so pena de generar una cantidad considerable de ruido en la comunicación, ruido que interferirá de forma negativa en el proceso de identificación de los atributos de cada objeto predador.

15.2 Generación de Señuelos y Predadores

A la hora de diseñar y construir un diálogo interactivo, DEVA analiza la información contenida en el objeto ACML proporcionado por CodeX y asigna *widgets* a primitivas de datos básicas (tal y como se analizará en el capítulo 17: *La Diversidad Humana*), obteniendo por tanto un *widget* por cada dato a mostrar y/u obtener por parte del usuario. Estos *widgets* pueden actuar en solitario, como objetos predadores, o ser agrupados en *chunks* que hacen las veces de objetos señuelo. Para ello, DEVA analizará las relaciones semánticas entre componentes indicada en el código binario de la aplicación a través de la clase *Chunk* proporcionada por CodeX.

Una vez que éstos objetos son agrupados en los *chunks* definidos por el programador de la aplicación y siguiendo el esquema de prioridades definido éste, DEVA intentará transmitir la mayor cantidad de componentes a través del canal de comunicación activo, determinando el número máximo de componentes en función de tres parámetros:

- Experiencia del usuario con el proceso o diálogo interactivo activo.
- Capacidad perceptiva del usuario.
- Capacidad física del canal de comunicación.

La prioridad asignada a cada uno de estos parámetros viene dada por el orden en el que éstos son incluidos en la relación anterior. Como se puede apreciar, siempre se intenta transmitir un número de objetos igual o inferior a la capacidad del procesador humano del usuario sin hacer ninguna concesión a una posible capacidad superior del canal de comunicación, quien es relegado a un tercer plano.

En función del grado de experiencia del usuario activo de la aplicación con el proceso de usuario invocado o con el diálogo interactivo don el que desea trabajar es posible suponer ciertas

destrezas que aumenten la capacidad de su memoria de trabajo. Así por ejemplo, el uso continuo de una serie de diálogos interactivos puede hacer que se establezca una relación entre la información que éstos contienen y la memoria a largo plazo del sujeto, de tal modo que mediante el proceso interno de creación de *chunks* descrito anteriormente, la capacidad neta de la memoria de trabajo aumenta puesto que parte de los objetos percibidos no pasan por ella.

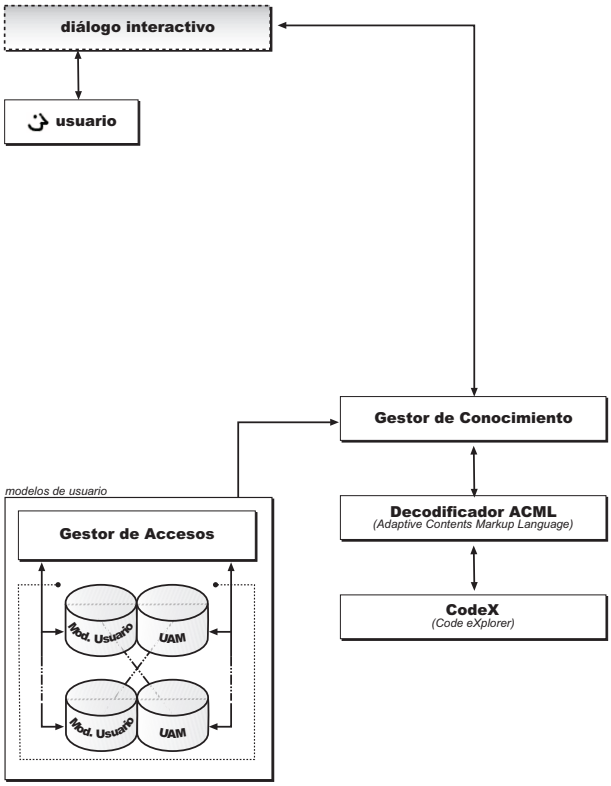


Figura 52. Esquema básico de DEVA. La información obtenida acerca del modelo de usuario es contrastada con el diálogo interactivo aportado por CodeX a través de un objeto ACML una vez que éste es decodificado. Esta información es empleada para diseñar un diálogo interactivo primitivo en base a los *chunks* identificados por el gestor de conocimiento.

De este modo, el número de objetos predador transmitidos para usuarios entrenados será sensiblemente superior al transmitido en el caso de usuarios novatos, permitiendo superar la capacidad de la memoria de trabajo del usuario en cuestión.

Para describir como DEVA emplea su conocimiento acerca de la capacidad del usuario en la estructuración de la información a nivel sintáctico, se va a emplear el ejemplo citado en el capítulo 6 (*El Modelo de Interacción*) en el que en un punto determinado de una aplicación era necesario obtener información acerca los datos personales del usuario, información que comprendía su *DNI*, *nombre*, *apellidos*, *calle*, *código postal*, *ciudad* y *país* (ver *Tabla 2*), es decir, siete datos

Para iniciar el ejemplo, supondremos que la capacidad de la memoria a corto plazo del usuario activo de la aplicación se limita a cinco elementos en el momento en que el que el diálogo es establecido y que el programador desea adquirir los siete datos en un solo bloque, es

decir, sin el establecimiento de chunks que puedan dar pistas a DEVA acerca de la mejor agrupación posible de la información en base al conocimiento que se tiene de la misma. El fragmento de código necesario para establecer este diálogo interactivo sería uno similar reflejado en el Código 25.

Código 25. Proceso de usuario *infoPersonal* empleado para recoger siete primitivas de datos en un solo bloque.

```
public class Userinfo
{
   UIMS interface = null;

   // constructor
   Userinfo (UIMS myInterface)
   {
      interface = myInterface;
      interface.registerEveryUP (this);
   }

   // User processes
   public boolean UP_infoPersonal ()
   {
      gadea.datatypes.String dni, nombre, apellidos, calle, cp,
      ciudad, pais;
      boolean value = false;

      // Data primitives instantiation
      dni = new gadea.datatypes.String ("", false);
      dni.setRequired (true);

      nombre = new gadea.datatypes.String ("", false);
      nombre.setRequired (true);

      apellidos = new gadea.datatypes.String ("", false);
      apellidos.setRequired (true);

      calle = new gadea.datatypes.String ("", false);
      calle.setRequired (true);

      cp = new gadea.datatypes.String ("", false);
      cp.setRequired (true);

      ciudad = new gadea.datatypes.String ("", false);
      ciudad.addPossibleValue ("Caracas");
      ciudad.addPossibleValue ("La Guaira");
      ciudad.addPossibleValue ("Oviedo");
      ciudad.addPossibleValue ("Cangas de Onís");
      ciudad.setRequired (true);

      pais = new gadea.datatypes.String ("", false);
      pais.addPossibleValue ("Venezuela");
      pais.addPossibleValue ("España");
      pais.setRequired (true);

      Dialogue dialogue = new Dialogue ();
      dialogue.add (new gadea.datatypes.String ("DNI: ", true));
      dialogue.add (dni);
      dialogue.add (new gadea.datatypes.String ("Nombre: ", true));
      dialogue.add (nombre);
      dialogue.add (new gadea.datatypes.String ("Apellidos: ",
         true));
      dialogue.add (apellidos);
      dialogue.add (new gadea.datatypes.String ("Calle: ", true));
      dialogue.add (calle);
      dialogue.add (new gadea.datatypes.String ("CP: ", true));
      dialogue.add (cp);
      dialogue.add (new gadea.datatypes.String ("Ciudad: ", true));
      dialogue.add (ciudad);
      dialogue.add (new gadea.datatypes.String ("País: ", true));
      dialogue.add (pais);


      CommunicationChannel comChannel = new CommunicationChannel
```

```
        ("Información Personal",
        CommunicationChannel.MAX_PRIORITY);

     interface.load (comChannel);
     if (comChannel.display (dialogue))
     {
        // User selected OK
        value = true;
     }
     else
     {
        // User selected CANCEL
     }

     interface.unload (comChannel);
     return (value);
}
```

Cuando el usuario invoca al proceso de usuario *infoPersonal* en el Código 25, CodeX registra la petición de información de la aplicación por medio de un diálogo interactivo, realiza una adaptación automática de contenidos por medio del *Language Manager* y generaría un documento ACML del tipo del ilustrado en el Código 26. Este documento será procesado por DEVA para establecer una relación directa entre las primitivas de datos incluidas en él y los *widgets* necesarios para establecer comunicación con el usuario, analizando el modelo de usuario para seleccionar aquellos *widgets* que mejor se adaptan a su perfil cognitivo, perceptivo y motriz. En este ejemplo supondremos que la selección de *widgets* es la incluida en la tabla Tabla 22.

Código 26. Documento ACML correspondiente al diálogo interactivo definido en el código Código 25.

```
<? ACML version = 1.0' ?>
<DIALOGUE NAME=''>
   <STRING ISFIXED='TRUE' ISPERCEPTIBLE='TRUE' ISREQUIRED='FALSE'
      MULTIPLEVALUES='FALSE'>
      <VALUES>
         <VALUE ID ='DNI:'/>
      </VALUES>
   </STRING>
   <STRING ISFIXED='FALSE' ISPERCEPTIBLE='TRUE' ISREQUIRED='TRUE'
      MULTIPLEVALUES='FALSE'>
      <VALUES/>
   </STRING>

   <STRING ISFIXED='TRUE' ISPERCEPTIBLE='TRUE' ISREQUIRED='FALSE'
      MULTIPLEVALUES='FALSE'>
      <VALUES>
         <VALUE ID ='Nombre:'/>
      </VALUES>
   </STRING>
   <STRING ISFIXED='FALSE' ISPERCEPTIBLE='TRUE' ISREQUIRED='TRUE'
      MULTIPLEVALUES='FALSE'>
      <VALUES/>
   </STRING>

   <STRING ISFIXED='TRUE' ISPERCEPTIBLE='TRUE' ISREQUIRED='FALSE'
      MULTIPLEVALUES='FALSE'>
      <VALUES>
         <VALUE ID ='Apellidos:'/>
      </VALUES>
   </STRING>
   <STRING ISFIXED='FALSE' ISPERCEPTIBLE='TRUE' ISREQUIRED='TRUE'
      MULTIPLEVALUES='FALSE'>
      <VALUES/>
```

```
   </STRING>

   <STRING ISFIXED='TRUE' ISPERCEPTIBLE='TRUE' ISREQUIRED='FALSE'
      MULTIPLEVALUES='FALSE'>
      <VALUES>
         <VALUE ID ='Calle:'/>
      </VALUES>
   </STRING>
   <STRING ISFIXED='FALSE' ISPERCEPTIBLE='TRUE' ISREQUIRED='TRUE'
      MULTIPLEVALUES='FALSE'>
      <VALUES/>
   </STRING>

   <STRING ISFIXED='TRUE' ISPERCEPTIBLE='TRUE' ISREQUIRED='FALSE'
      MULTIPLEVALUES='FALSE'>
      <VALUES>
         <VALUE ID ='CP:'/>
      </VALUES>
   </STRING>
   <STRING ISFIXED='FALSE' ISPERCEPTIBLE='TRUE' ISREQUIRED='TRUE'
      MULTIPLEVALUES='FALSE'>
      <VALUES/>
   </STRING>

   <STRING ISFIXED='TRUE' ISPERCEPTIBLE='TRUE' ISREQUIRED='FALSE'
      MULTIPLEVALUES='FALSE'>
      <VALUES>
         <VALUE ID ='Ciudad:'/>
      </VALUES>
   </STRING>
   <STRING ISFIXED='FALSE' ISPERCEPTIBLE='TRUE' ISREQUIRED='TRUE'
      MULTIPLEVALUES='FALSE'>
      <VALUES/>
      <POSSIBLE_VALUES>
         <VALUE ID = 'Caracas'/>
         <VALUE ID = 'La Guaira'/>
         <VALUE ID = 'Oviedo'/>
         <VALUE ID = 'Cangas de Onís'/>
      </POSSIBLE_VALUES>
   </STRING>

   <STRING ISFIXED='TRUE' ISPERCEPTIBLE='TRUE' ISREQUIRED='FALSE'
      MULTIPLEVALUES='FALSE'>
      <VALUES>
         <VALUE ID ='País:'/>
      </VALUES>
   </STRING>
   <STRING ISFIXED='FALSE' ISPERCEPTIBLE='TRUE' ISREQUIRED='TRUE'
      MULTIPLEVALUES='FALSE'>
      <VALUES/>
      <POSSIBLE_VALUES>
         <VALUE ID = 'Venezuela'/>
         <VALUE ID = 'España'/>
      </POSSIBLE_VALUES>
   </STRING>
</DIALOGUE>
```

Una vez determinados los *widgets* necesarios para recoger la información solicitada por el diálogo interactivo, DEVA determinará la experiencia del usuario en el empleo de dicho diálogo, calculando la coordenada en la que el se encuentra en la curva de aprendizaje, aplicando la *Ley Exponencial de la Práctica* en función de valores recogidos en el uso de dicho diálogo en experiencias previas del usuario. Si el usuario no ha superado el punto crítico de la curva de aprendizaje y por lo tanto puede considerarse novato, la cantidad de información a transmitir será mínima, no superando en ningún caso a la capacidad de la memoria de trabajo del individuo, ya que la concepción del diálogo interactivo no se encuentra todavía asimilada en su memoria a largo plazo.

Suponiendo entonces que el usuario es novato en el empleo del diálogo solicitado, DEVA considerará que la capacidad máxima del canal de comunicación se sitúa en los cinco elementos que posee la memoria a corto plazo del sujeto. Es por ello que resultará imposible transmitir los siete elementos del diálogo en una sola fase, siendo necesarias dos fases. En cada una de ellas el número de elementos transmitidos será igual o inferior a cinco.

Tabla 22. Posible relación de *widgets* necesarios para satisfacer la demanda de información expresada en el Código 26.

NOMBRE	TIPO	WIDGET
DNI	STRING con plantilla	Campo de Texto
Nombre	STRING	Campo de Texto
Apellidos	STRING	Campo de Texto
Calle	STRING	Campo de Texto
Código Postal	INTEGER con plantilla	Campo de Texto
Ciudad	STRING	Lista Desplegable
País	STRING	Lista Desplegable

En este momento el agente interactivo tendrá que elegir el número de elementos a transmitir en cada fase, intentando siempre balancear la carga a la que se somete el canal de comunicación. Dado que el número de objetos a transmitir es de siete elementos y la capacidad del canal cinco, DEVA transmitirá 7 / 2= 3,5 objetos (~3 objetos) en la primera fase y otros tantos (4 objetos) en la segunda fase.

A los tres objetos transmitidos durante la primera fase habrá que añadir un cuarto objeto, el cual actuará como objeto señuelo de los objetos que se transmitirán en la segunda fase, de tal modo que el usuario sea consciente de que hay una segunda fase de transmisión, así como también del tipo de información que obtendrá en esta segunda fase. Dado que el programador no ha estructurado la información contenida en el diálogo interactivo del proceso de usuario *infoPersonal*, DEVA no podrá usar ningún tipo de información acerca de las relaciones semánticas entre los componentes del diálogo para crear el objeto señuelo correspondiente, teniendo que limitarse a la creación de un objeto señuelo dotado de una carga semántica muy somera, del tipo de *más información*…, tal y como se puede observar en diálogo interactivo resultante, representado en la Figura 53.

En dicha figura se puede apreciar la existencia de tres objetos predador correspondientes a las primitivas de datos *DNI*, *nombre* y *apellidos*, así como el objeto predador *más información*…, el cual dá pistas al usuario acerca de la existencia de una segunda fase de transmisión, representada en la Figura 54. Cuando el usuario invoca al comando *más información*… dará inicio entonces a la segunda fase de transmisión del diálogo interactivo, la cual será imprescindible para completar el intercambio de información planteado. En esta fase el usuario novato podrá regresar a la fase de transmisión inicial por medio del comando *atrás*…

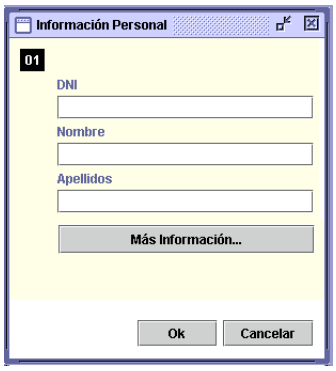


Figura 53. Diálogo interactivo del proceso de usuario *infoPersonal* diseñado para un usuario novato y para una capacidad de transmisión de cinco objetos (consultar versión en color en el capítulo 24: *Imágenes en Color*, p. 445).

Supongamos ahora que el usuario ha superado el punto en la curva de aprendizaje del diálogo interactivo en cuestión que permite clasificarlo en el conjunto borroso de los usuarios expertos con un alto grado de pertenencia. En este caso, dado el amplio conocimiento del diálogo interactivo por parte del usuario, se da por supuesto que los diversos objetos que lo componen son conocidos por el mismo, siendo todos ellos percibidos como objetos predador, dado que la atención del usuario se dirigirá directamente a los detalles de dichos objetos. También se da por supuesto que la concepción espacial y física del diálogo interactivo está ya fuertemente asentada en la memoria a largo plazo del sujeto.

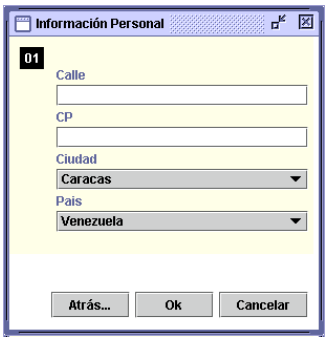


Figura 54. Segunda fase del proceso de comunicación abierto al invocar al proceso de usuario *infoPersonal* (consultar versión en color en el capítulo 24: *Imágenes en Color*, p. 445).

Debido a estas razones, el número de elementos a transmitir no dependerá ya de la capacidad de la memoria a corto plazo del usuario sino de su capacidad a largo plazo, la cual, como sabemos, es infinita. Obviamente, esto no significa que el número de elementos a transmitir sea infinito, dado que queda todavía por dilucidar la capacidad física real del medio de transmisión. Dado que el canal de comunicación es visual y la herramienta de transmisión es la ventana, DEVA puede asumir que realmente la capacidad de transmisión es infinita, ya que el espacio visual representado puede ser ampliado mediante el uso de *scrolls*. Nótese que esta capacidad infinita no siempre está presente, ya que existen canales de comunicación, como el auditivo, que no

permiten una transmisión infinita de información, al menos desde el punto de vista operativo para usuarios con discapacidades.

En el ejemplo que se está siguiendo y suponiendo una capacidad física infinita para el canal de comunicación visual, será posible ahora transmitir toda la información requerida por la aplicación, tal y como puede apreciarse en la Figura 55. Aunque el cambio hacia la nueva modalidad de transmisión de información en una sola fase puede afectar a la eficacia del trabajo del usuario y disminuir su rendimiento y destreza en el uso del diálogo , los resultados de nuestra investigación en el diseño y adaptación progresiva de modelos de navegación como parte del programa Tirsus (referido en el apartado 15.1.1: *Señuelos y Predadores*), nos indican que esta posible turbación es efímera y de efectos despreciables, ya que el cambio de modalidad no implica un cambio en el modo de acceder a la información, sino en la ubicación y configuración con la que ésta es representada. Dado que la representación de la información y el modelo de interacción empleado por la nueva estrategia de transmisión coincide con la que el usuario ha asentado en su memoria a largo plazo, el proceso de reclasificación de la información perceptiva es mínimo y la curva de aprendizaje apenas se resiente.

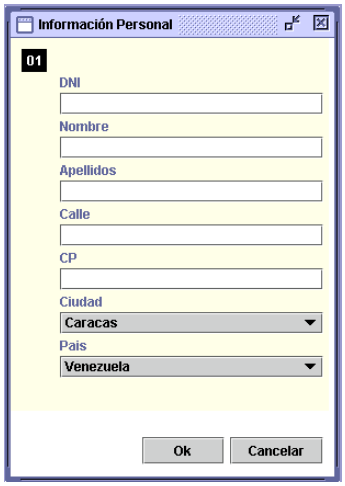


Figura 55. Construcción del diálogo interactivo correspondiente al documento ACML incluido en el Código 26, diseñado dinámicamente para usuarios avanzados (consultar versión en color en el capítulo 24: *Imágenes en Color*, p. 445).

15.2.1. Organización Basada en Chunks

En el ejemplo anterior, el programador no hace uso de ningún tipo de organización de la información en función del posible significado que puede tener para el usuario. Aunque, como acabamos de ver, esta organización o clasificación previa de la información no es estrictamente necesaria para que DEVA se encargue de su gestión, su empleo en combinación con DEVA puede mejorar significativamente el proceso de interacción.

A continuación se repetirá la experiencia del ejemplo anterior para describir como DEVA emplea la información contenida en los *chunks* definidos por la aplicación –e identificados por CodeX– en la estructuración y clasificación del conocimiento, intentando que ésta estructuración tenga un reflejo lo más fiel posible en la sintaxis de la interfaz generada.

Supondremos entonces que en la fase de análisis de la aplicación y en función del dominio de ésta y de la tipología de sus usuarios los diseñadores desean agrupar las siete primitivas de datos anteriores (*DNI*, *nombre*, *apellidos*, *calle*, *código postal*, *ciudad* y *país*) en tres categorías distintas: *identificación personal* (integrado por *DNI*, *nombre* y *apellidos*), *domicilio* (formado por *calle* y *código postal*) y *ciudad* (formado por *ciudad* y *país*). Para definir esta clasificación, el programador deberá definir tres *chunks*, –uno por categoría– en tiempo de edición, registrando en ellos las primitivas de datos correspondientes.

Código 27. Versión del proceso de usuario *infoPersonal* incluyendo los *chunks identificación personal*, *domicilio* y *ciudad*.

```
public class Userinfo
{
   UIMS interface = null;

   // constructor
   Userinfo (UIMS myInterface)
   {
      interface = myInterface;
      interface.registerEveryUP (this);
   }

   // User processes
   public boolean UP_infoPersonal ()
   {
      gadea.datatypes.String dni, nombre, apellidos, calle, cp,
      ciudad, pais;
      boolean value = false;

      // Data primitives instantiation
      dni = new gadea.datatypes.String ("", false);
      dni.setRequired (true);

      nombre = new gadea.datatypes.String ("", false);
      nombre.setRequired (true);

      apellidos = new gadea.datatypes.String ("", false);
      apellidos.setRequired (true);

      calle = new gadea.datatypes.String ("", false);
      calle.setRequired (true);

      cp = new gadea.datatypes.String ("", false);
      cp.setRequired (true);

      ciudad = new gadea.datatypes.String ("", false);
      ciudad.addPossibleValue ("Caracas");
      ciudad.addPossibleValue ("La Guaira");
      ciudad.addPossibleValue ("Oviedo");
      ciudad.addPossibleValue ("Cangas de Onís");
      ciudad.setRequired (true);

      pais = new gadea.datatypes.String ("", false);
      pais.addPossibleValue ("Venezuela");
      pais.addPossibleValue ("España");
      pais.setRequired (true);

      Dialogue dialogue = new Dialogue ();

      Chunk ipChunk = new Chunk ("Identificación Personal");
      ipChunk.add (new gadea.datatypes.String ("DNI: ", true));
```

```
      ipChunk.add (dni);
      ipChunk.add (new gadea.datatypes.String ("Nombre: ", true));
      ipChunk.add (nombre);
      ipChunk.add (new gadea.datatypes.String ("Apellidos: ",
         true));
      ipChunk.add (apellidos);

      Chunk domicilioChunk = new Chunk ("Domicilio");
      domicilioChunk.add (new gadea.datatypes.String ("Calle: ",
      true));
      domicilioChunk.add (calle);
      domicilioChunk.add (new gadea.datatypes.String ("CP: ",
      true));
      domicilioChunk.add (cp);

      Chunk ciudadChunk = new Chunk ("Ciudad");
      ciudadChunk.add (new gadea.datatypes.String ("Ciudad: ",
      true));
      ciudadChunk.add (ciudad);
      ciudadChunk.add (new gadea.datatypes.String ("País: ", true));
      ciudadChunk.add (pais);

      dialogue.add(ipChunk);
      dialogue.add(domicilioChunk);
      dialogue.add(ciudadChunk);

      CommunicationChannel comChannel = new CommunicationChannel
         ("Información Personal" ,
         CommunicationChannel.MAX_PRIORITY);

      interface.load (comChannel);
      if (comChannel.display (dialogue))
      {
         // User selected OK
         value = true;
      }
      else
      {
         // User selected CANCEL
      }

      interface.unload (comChannel);
      return (value);
}
```

Como se puede apreciar en el Código 27, los objetos *ipChunk* *domicilioChunk* y *ciudadChunk* de la clase *Chunk* sirven para crear las categorías descritas, proporcionándoles además la representación semántica con la que éstas serán representadas en la interfaz. Una vez que éstos objetos son creados y configurados, pueden ser cargados en el canal de comunicación activo para su transmisión. El documento ACML correspondiente al Código 27 se encuentra recogido en el Código 28. Se puede observar como la estructura arbórea de este documento representa claramente los tres *chunks* definidos. La prioridad de que dispone cada uno de estos *chunks* en el canal de comunicación activo está determinada por el orden en el que se incluyen en este documento.

Código 28. Versión ACML del Código 27. Nótese la existencia de los *chunks* que sirven para agrupar a las primitivas de datos.

```
<? ACML version = 1.0' ?>
<DIALOGUE NAME=''>
   <CHUNK NAME='Información Personal'>
      <STRING ISFIXED='TRUE' ISPERCEPTIBLE='TRUE' ISREQUIRED='FALSE'
         MULTIPLEVALUES='FALSE'>
         <VALUES>
            <VALUE ID ='DNI:'/>
         </VALUES>
```

```
    </STRING>
    <STRING ISFIXED='FALSE' ISPERCEPTIBLE='TRUE' ISREQUIRED='TRUE'
       MULTIPLEVALUES='FALSE'>
       <VALUES/>
    </STRING>

    <STRING ISFIXED='TRUE' ISPERCEPTIBLE='TRUE' ISREQUIRED='FALSE'
       MULTIPLEVALUES='FALSE'>
       <VALUES>
          <VALUE ID ='Nombre:'/>
       </VALUES>
    </STRING>
    <STRING ISFIXED='FALSE' ISPERCEPTIBLE='TRUE' ISREQUIRED='TRUE'
       MULTIPLEVALUES='FALSE'>
       <VALUES/>
    </STRING>

    <STRING ISFIXED='TRUE' ISPERCEPTIBLE='TRUE' ISREQUIRED='FALSE'
       MULTIPLEVALUES='FALSE'>
       <VALUES>
          <VALUE ID ='Apellidos:'/>
       </VALUES>
    </STRING>
    <STRING ISFIXED='FALSE' ISPERCEPTIBLE='TRUE' ISREQUIRED='TRUE'
       MULTIPLEVALUES='FALSE'>
       <VALUES/>
    </STRING>
 </CHUNK>

 <CHUNK NAME='Domicilio'>
    <STRING ISFIXED='TRUE' ISPERCEPTIBLE='TRUE' ISREQUIRED='FALSE'
       MULTIPLEVALUES='FALSE'>
       <VALUES>
          <VALUE ID ='Calle:'/>
       </VALUES>
    </STRING>
    <STRING ISFIXED='FALSE' ISPERCEPTIBLE='TRUE' ISREQUIRED='TRUE'
       MULTIPLEVALUES='FALSE'>
       <VALUES/>
    </STRING>

    <STRING ISFIXED='TRUE' ISPERCEPTIBLE='TRUE' ISREQUIRED='FALSE'
       MULTIPLEVALUES='FALSE'>
       <VALUES>
          <VALUE ID ='CP:'/>
       </VALUES>
    </STRING>
    <STRING ISFIXED='FALSE' ISPERCEPTIBLE='TRUE' ISREQUIRED='TRUE'
       MULTIPLEVALUES='FALSE'>
       <VALUES/>
    </STRING>
 </CHUNK>

 <CHUNK NAME='Ciudad'>
    <STRING ISFIXED='TRUE' ISPERCEPTIBLE='TRUE' ISREQUIRED='FALSE'
       MULTIPLEVALUES='FALSE'>
       <VALUES>
          <VALUE ID ='Ciudad:'/>
       </VALUES>
    </STRING>
    <STRING ISFIXED='FALSE' ISPERCEPTIBLE='TRUE' ISREQUIRED='TRUE'
       MULTIPLEVALUES='FALSE'>
       <VALUES/>
       <POSSIBLE_VALUES>
          <VALUE ID = 'Caracas'/>
          <VALUE ID = 'La Guaira'/>
          <VALUE ID = 'Oviedo'/>
          <VALUE ID = 'Cangas de Onís'/>
       </POSSIBLE_VALUES>
    </STRING>

    <STRING ISFIXED='TRUE' ISPERCEPTIBLE='TRUE' ISREQUIRED='FALSE'
       MULTIPLEVALUES='FALSE'>
       <VALUES>
          <VALUE ID ='País:'/>
       </VALUES>
```

```
      </STRING>
      <STRING ISFIXED='FALSE' ISPERCEPTIBLE='TRUE' ISREQUIRED='TRUE'
         MULTIPLEVALUES='FALSE'>
         <VALUES/>
         <POSSIBLE_VALUES>
            <VALUE ID = 'Venezuela'/>
            <VALUE ID = 'España'/>
         </POSSIBLE_VALUES>
      </STRING>
   </CHUNK>
</DIALOGUE>
```

Cuando DEVA decodifica el documento ACML y tras asignar *widgets* a primitivas de datos (siguiendo la conversión directa determinada en la Tabla 22 para este ejemplo), éste agente asignará una cuota del ancho de banda a cada uno de los objetos a transmitir pero, a diferencia del ejemplo anterior en donde todos los objetos predador (*widgets*) disponían de prioridad similar, ahora el tipo de objetos a considerar serán los propios *chunks*, los cuales actuarán como verdaderos objetos señuelo, representando a los objetos predador que contienen (los *widgets*).

Como consecuencia, DEVA se enfrenta ahora a un problema trivial: el de transmitir tan sólo tres objetos señuelo (los *chunks*) por medio de un amplio canal de comunicación con capacidad para cinco objetos (la capacidad de la memoria episódica del usuario). Como resultado, los tres *chunks* se transmitirán de forma directa, de modo similar al indicado en la Figura 56 en donde cada objeto señuelo es representado por un botón. Nótese como ahora existe una mayor ganancia en la señal y disminuye el ruido ya que la carga semántica de los mensajes transmitidos por éstos objetos (*información personal…*, *domicilio…* o *ciudad…*) es sensiblemente mayor a la transmitida en circunstancias similares en el ejemplo anterior (*más información…).*

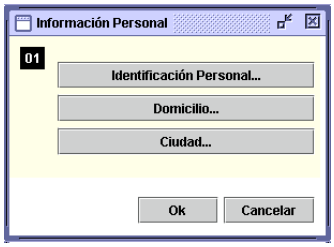


Figura 56. Diálogo interactivo creado a partir del documento ACML del Código 28 para un usuario novato (consultar versión en color en el capítulo 24: *Imágenes en Color*, p. 445).

Naturalmente, cuando el usuario seleccione alguna de las opciones relativas a los *chunks* u objetos señuelo, podrá acceder directamente a los objetos predador contenidos en ellos, por medio de una estrategia de transmisión de información bifásica (ver Figura 57).

A medida que el usuario emplea el diálogo interactivo correspondiente al proceso *infoPersonal*, dicho usuario irá aumentando su destreza y rendimiento y por ende también su grado de veteranía. Al igual que ocurría en el ejemplo anterior, una vez que el usuario haya alcanzado un nivel de veteranía aceptable y por lo tanto haya aumentado el ancho de banda efectivo del canal de comunicación, será

posible modificar la estrategia adoptada, de tal modo que los objetos predador puedan ser transmitidos directamente.

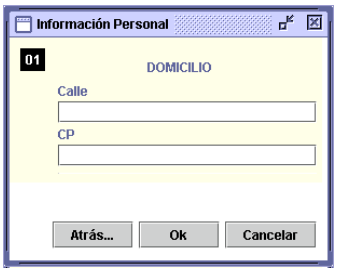


Figura 57. Representación de los objetos predador correspondientes al objeto señuelo *domicilio* de la Figura 56 (consultar versión en color en el capítulo 24: *Imágenes en Color*, p. 445).

Sin embargo, gracias a la estructuración aportada por el establecimiento de diferentes categorías semánticas para los objetos, es posible modificar la estrategia de acceso a la información de un modo gradual, permitiendo bajar progresivamente por los diferentes niveles del árbol semántico en el que se estructura la información (ver Figura 46). Así, partiendo de un objeto señuelo concreto se puede permitir el acceso directo a los hijos de dicho objeto, a los nietos, a los bisnietos, etc. siempre en función del grado de veteranía concreto alcanzado por el usuario.

En este ejemplo, para un usuario de experiencia intermedia, sería posible permitir un acceso total a los predadores del primer objeto señuelo (*identificación personal*) manteniendo todavía un acceso bifásico a la información contenida en los otros dos objetos señuelos (objetos *domicilio* y *ciudad*) tal y como podemos ver en la Figura 58. En caso en el que el grado de experiencia del usuario sea elevado, se podría permitir un acceso directo a todos los predadores del diálogo interactivo (Figura 59). Nótese como en el primer caso (Figura 58) el número de elementos transmitidos se limita a cinco (tres objetos predador y dos objetos señuelo), coincidiendo con el tamaño de la memoria a corto plazo del sujeto.

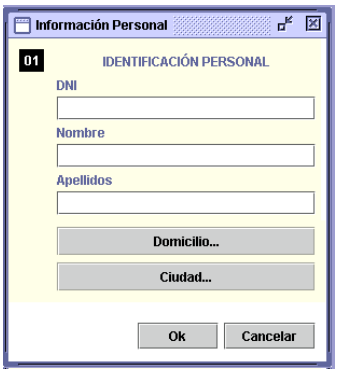


Figura 58. Diálogo interactivo de la Figura 56 adaptado para un usuario con un grado moderado de veteranía (consultar versión en color en el capítulo 24: *Imágenes en Color*, p. 445).

A medida que el nivel de veteranía alcanzado por el usuario aumenta, el número de objetos transmitidos crece y supera la memoria a corto plazo del usuario, será necesario el empleo de medidas especiales para garantizar que la clasificación en *chunks* impuesta por el programador en la fase de diseño se siga manteniendo en la interfaz en tiempo de ejecución. Para ello se emplean técnicas de agrupación, manteniendo unidos todos aquellos *widgets* correspondientes a un *chunk* por medio de las reglas de agrupación descritas en el apartado 15.1.2 (*Agrupación*). Tal y como se puede apreciar en la Figura 59, todos los objetos de un *chunk* ocupan un mismo nicho espacial en la representación final del diálogo interactivo y comparten atributos comunes, empleando sobre todo el color de fondo como elemento unificador. En la figura Figura 59 se puede apreciar como a pesar de la disposición consecutiva de los distintos *chunks* que forman parte del diálogo interactivo, es posible identificar cada uno de ellos por medio del color de fondo.

Figura 59. Versión para usuarios veteranos del diálogo interactivo del documento ACML incluido en el Código 28. Nótese como el color de fondo aumenta la cohesión perceptiva de la información contenida en cada *chunk* (consultar versión en color en el capítulo 24: *Imágenes en Color*, p. 445).

16

16 Un Modelo de Evaluación

Lo que el estilo es a la persona, la estructura es a la obra

Luis Goytisolo

16.1 Comportamiento Dirigido por Tareas

Uno de los aspectos más importantes de la teoría unificada de la cognición (TUC) [Newell (1991)] y de su precedente inmediato, el modelo de procesador humano [Card et. al. (1983)], es su capacidad para realizar predicciones del comportamiento humano con un nivel elevado de eficacia, lo cual ha permitido emplear estas teorías en el diseño y evaluación de nuevos modelos de interacción así como de técnicas aplicadas a distintos campos [Lucas et. al. (2000)]. El mecanismo de predicción de la TUC está basado en el modelo GOMS, el cual es una representación del conocimiento práctico requerido por un usuario para realizar determinadas tareas con un dispositivo [Falgueras y Guevara (2000)].

El nombre que recibe este modelo es un acrónimo formado por los conceptos *Goals* (objetivos), *Operators* (operadores), *Methods* (métodos) y *Selections* (selecciones). En el modelo GOMS, los objetivos representan

la tarea que el usuario desea realizar, los operadores indican los actos cognitivos, perceptivos o motrices llevados a cabo en la ejecución de la tarea, los métodos describen la secuencia de pasos necesarios para alcanzar el objetivo requerido y por último, las selecciones representan el conjunto de reglas empleadas (seleccionadas) con el propósito de dirigir el control de la operación.

El agente interactivo representado por DEVA asume el papel del usuario, analizando el comportamiento requerido por éste cuando ha de intercambiar información con el sistema por medio de un diálogo interactivo. Para DEVA, los objetivos (G) de todo diálogo están claramente definidos, puesto que al partir de un marco conceptual basado en un sistema de comunicación, el objetivo perseguido será establecer dicha comunicación con el menor nivel de ruido posible y permitir así un intercambio fluido de información. Para este agente, los métodos (M) y las selecciones (S) también se encuentran claramente definidas y delimitadas, ya que la información a transmitir se encuentra estructurada en bloques que dependen de la capacidad de procesamiento del usuario, tal y como se analizó en el capítulo 15 (*El Procesador Perceptivo*). Por lo tanto, el conocimiento almacenado por el usuario para cada tarea (los métodos), así como el mecanismo mediante el cual se recupera y aplica dicho conocimiento (las selecciones) está determinado a partir del modo en el que el gestor de conocimiento estructura la información en *chunks*.

Queda por tanto pendiente la definición de los operadores (O) en el modelo GOMS adoptado por DEVA, es decir, evaluar los actos cognitivos, perceptivos y motrices que el usuario debe llevar a cabo para alcanzar los objetivos planteados en cada proceso de usuario y diálogo interactivo.

16.1.1. Operadores

Como ya se ha comentado en los capítulos precedentes, la primera tarea realizada por DEVA cuando recibe un documento ACML de CodeX será su decodificación para un análisis preliminar de la información que ha de contener el diálogo interactivo contenido en dicho documento (ver Figura 52). Una vez analizada esta información y en función de los *chunks* contenidos en el diálogo, así como de la experiencia del usuario, el gestor de conocimiento puede definir el número de fases de transmisión que componen el diálogo interactivo, así como las primitivas de datos que irán incluidas en cada una de ellas. Realizado este procesamiento preliminar de alto nivel, será necesario determinar que *widgets* actuarán como agentes para cada primitiva de datos, alcanzando así una definición sintáctica a bajo nivel del diálogo interactivo.

Sin embargo, la selección de un *widget* para una primitiva de datos no es un proceso trivial, ya que en muchas ocasiones, para una determinada primitiva existe más de un *widget* que podría satisfacer los requisitos de interacción planteados –desde un punto de vista puramente técnico. Supongamos por ejemplo que en un momento dado de una aplicación, el programador desea obtener del usuario el nombre

de un color de entre siete posibles. Naturalmente, siguiendo el esquema de desarrollo planteado a partir del capítulo 5 (*Diseño Centrado en el Usuario*), el diseñador de la aplicación construirá un diálogo interactivo para tal fin y empleará el mecanismo de reflexión de CodeX para que dicho diálogo sea mostrado al usuario. En el Código 29 se incluye el código necesario para establecer este diálogo.

Código 29. Proceso de usuario *selectColor* para la selección de un color entre siete posibles.

```
public class Colors
{
   UIMS interface = null;

   // constructor
   Colors (UIMS myInterface)
   {
      interface = myInterface;
      interface.registerEveryUP (this);
   }

   // User processes
   public boolean UP_selectColor ()
   {
      gadea.datatypes.String color;
      boolean value = false;

      // Data primitives instantiation
      color = new gadea.datatypes.String ("", false);
      color.setRequired (true);
      color.addPossibleValue ("Amarillo");
      color.addPossibleValue ("Azul");
      color.addPossibleValue ("Blanco");
      color.addPossibleValue ("Naranja");
      color.addPossibleValue ("Negro");
      color.addPossibleValue ("Rojo");
      color.addPossibleValue ("Verde");

      Dialogue dialogue = new Dialogue ();
      dialogue.add (new gadea.datatypes.String ("Seleccione un
         color: ", true));
      dialogue.add (color);

      CommunicationChannel comChannel = new CommunicationChannel
         ("Colores" , CommunicationChannel.MAX_PRIORITY);

      interface.load (comChannel);
      if (comChannel.display (dialogue))
      {
         // User selected OK
         value = true;
      }
      else
      {
         // User selected CANCEL
      }

      interface.unload (comChannel);
      return (value);
}
```

Como se puede apreciar en el Código 29, la primitiva de datos ideal para obtener el nombre del color es el componente *gadea.datatypes.String*, configurando dicho componente para que sólo se pueda introducir en él un valor perteneciente a la lista de valores posibles. En el ejemplo, la lista está formada por los colores amarillo, azul, blanco, naranja, negro, rojo y verde. Nótese como el componente *color* es configurado sin ningún valor por defecto y se especifica expresamente que tras haber finalizado el diálogo interactivo en el que

se incluye, el componente debe contener un valor válido especificado por el usuario.

Código 30. Versión ACML del requerimiento de información incluido en el Código 29.

```
<? ACML version = 1.0' ?>
<DIALOGUE NAME=''>
   <STRING ISFIXED='TRUE' ISPERCEPTIBLE='TRUE' ISREQUIRED='FALSE'
      MULTIPLEVALUES='FALSE'>
      <VALUES>
         <VALUE ID ='Seleccione un color'/>
      </VALUES>
   </STRING>

   <STRING ISFIXED='FALSE' ISPERCEPTIBLE='TRUE' ISREQUIRED='TRUE'
      MULTIPLEVALUES='FALSE'>
      <VALUES/>
      <POSSIBLE_VALUES>
         <VALUE ID = 'Amarillo'/>
         <VALUE ID = 'Azul'/>
         <VALUE ID = 'Blanco'/>
         <VALUE ID = 'Naranja'/>
         <VALUE ID = 'Negro'/>
         <VALUE ID = 'Rojo'/>
         <VALUE ID = 'Verde'/>
      </POSSIBLE_VALUES>
   </STRING>
</DIALOGUE>
```

En el Código 30 se incluye el documento ACML obtenido por CodeX cuando el programador carga el diálogo interactivo en el sistema por medio de un canal de comunicación. Cuando DEVA recibe este documento, lo decodifica empleando un decodificador de ACML, obteniendo así una estructura de datos dinámica en memoria capaz de ser analizada en tiempo de ejecución. Cuando DEVA examina el segundo objeto del documento y sin tomar en cuenta para nada el perfil cognitivo, perceptivo o motriz del usuario, determinará que el componente en cuestión podrá ser satisfecho por cualquier *widget* capaz de adquirir una cadena de caracteres (el nombre del color) comprobando que dicha cadena de caracteres coincida con una cualquiera de las especificadas ente los posibles valores del componente, es decir, de las incluidas en la etiqueta *<POSSIBLE_VALUES>*. Tal y como se aprecia en la Figura 60, en la plataforma Java, existen al menos cuatro tipos distintos de *widgets* capaces de realizar esa operación. Se trata de los *widgets javax.swing.Jbutton, javax.swing.JradioButton, javax.swing.JtextField* y *javax.swing.JComboBox*, y, siendo el primero de ellos utilizado en un grupo de botones mutuamente excluyente.

Desde un punto de vista puramente técnico, cualquiera de los *widgets* incluidos en la Figura 60 cumple a la perfección con su cometido pero, naturalmente, desde un punto de vista de la interacción hay ciertos *widgets* que se adaptarán mejor a la capacidad del modelo de procesador humano del usuario activo y otros que se adaptarán peor. El papel jugado por una agente interactivo adaptable será precisamente el de seleccionar de una forma inteligente aquel *widget* que, cumpliendo con la función básica para la que se la pretende usar, se adapte mejor a las características cognitivas, perceptivas y motrices del usuario. Por ello, el cometido principal de DEVA será evaluar todas las posibilidades existentes en un momento dado para satisfacer un requerimiento de interacción. Precisamente, de este proceso de evaluación para la

adaptación viene el nombre de este módulo ya que DEVA es un acrónimo que significa *Discourse Expert Valuator for Adaptation* o *Experto Valuador de Diálogos para su Adaptación*.

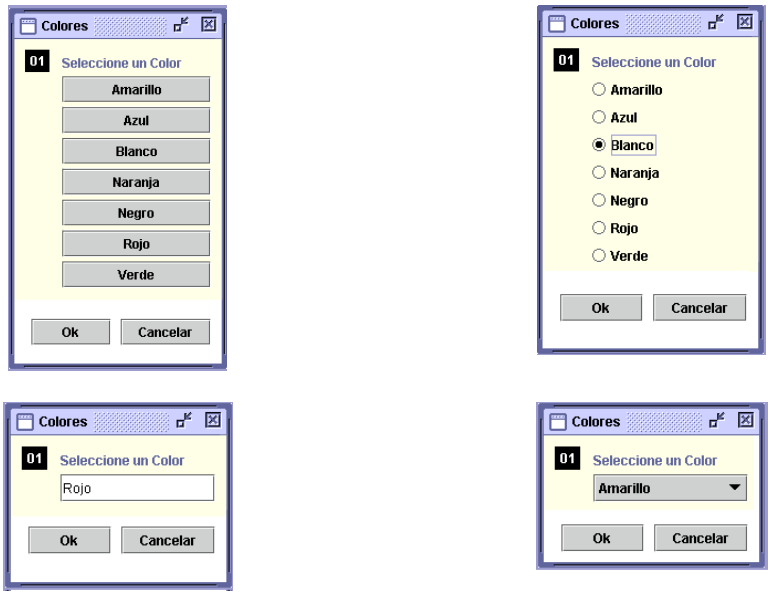


Figura 60. Representación de los cuatro posibles *widgets* candidatos a satisfacer los requisitos de interacción del segundo objeto incluido en el documento ACML del Código 30 (consultar versión en color en el capítulo 24: *Imágenes en Color*, p. 445).

16.1.2. Evaluación de Componentes

Para cada uno de las primitivas de datos incluidas en un diálogo interactivo, DEVA determina el conjunto de los posibles *widgets* que satisfacen el requerimiento interactivo del diálogo en una primera fase, seguida de una segunda fase en la que se evalúa la funcionalidad del *widget* con respecto a la capacidad cognitiva, perceptiva y motriz del usuario. Para ello, DEVA analiza el modo de interacción de cada *widget* asignándole una puntuación, seleccionando entonces aquel *widget* que alcance la mejor puntuación como el idóneo para ser empleado por el usuario.

Para ello, DEVA se basa una vez más en la teoría unificada de la cognición y en especial en el modelo de predicción de esta teoría, planteado por primera vez por Card et. al. (1983). Este modelo es una herramienta de ingeniería empleado en interacción y comunicación humana para determinar de forma aproximada el tiempo requerido por un usuario en la realizaría tareas básicas, lo cual permite comparar técnicas de interacción para la realización de valuaciones preliminares de interfaces de usuario. Una de las novedades de su empleo en DEVA radica en su aplicación continua y en tiempo real por parte de una agente interactivo software.

Este modelo ha sido validado por sus autores (Card, Moran y Newell) en la evaluación de editores de texto, herramientas de diseño gráfico y utilidades para sistemas operativos, experimentando un nivel

de eficacia del 20% en todas ellas, parámetro muy elevado para un modelo de predicción de comportamiento humano a nivel cognitivo. Este modelo de predicción esta basado en una serie de premisas que se recogen a continuación:

- El usuario conoce como realizar la tarea sobre la que se hará la predicción, así como la secuencia de operadores a aplicar.
- El usuario no cometerá errores en la realización de la tarea.
- Los parámetros cognitivos, perceptivos y motrices del usuario son conocidos.

Dadas las premisas anteriores, el modo de operar de este modelo consiste en analizar el método empleado para realizar la tarea, examinando los operadores básicos a emplear. Estos operadores se remiten a una serie limitada de operaciones que se aplican de forma secuencial para realizar una tarea. La lista completa de los operadores empleados en este modelo se incluye en la Tabla 23 y como tal y como se puede apreciar, consisten en cinco tipos de operaciones básicas.

Tabla 23. Parámetros empleados para la medición de operadores básicos en el modelo de predicción de la TUC.

ID	PARÁMETRO	NIVEL	DESCRIPCIÓN
K	Tecleado (*Keystroking*)	Motriz	Operación necesaria para pulsar una tecla en un teclado, *track-ball*, *joystick*, ratón u otro dispositivo.
P	Señalización (*Pointing*)	Motriz/Perceptivo	Desplazamiento del cursor en la pantalla hasta siutarlo sobre el objeto requerido.
H	Cambio de Dispositivo (*Homing*)	Motriz/Perceptivo	Operación de cambio de dispositivo de interacción, moviendo por ejemplo las manos desde el teclado al ratón.
D	Dibujo con puntero (*Drawing*)	Motriz/Perceptivo	Operador requerido al dibujar segmentos con el puntero o en operaciones de arrastre con un dispositivo de señalización (ratón, *joystick*, *track-ball*, etc.).
M	Proceso mental (*Mental*)	Cognitivo/Perceptivo	Procesos mentales para la toma de decisiones inmediatas.

En función de la tarea a realizar, el número y secuencia de operadores varía, pudiendo determinarse el tiempo requerido para llevar a termino la tarea en función de los operadores implicados así como en la eficacia con la que un usuario determinado ejecuta dichos operadores. Como se puede apreciar en la Tabla 24, la eficacia de cada uno de estos operadores es cuantificable en una medida temporal –habitualmente milésimas de segundo– por lo que la eficacia de la tarea se mide de forma sencilla sumando el tiempo empleado por todos y cada uno de los operadores necesarios para su correcta ejecución.

Tabla 24. Tiempos de ejecución medios para la ejecución de operadores básicos. El tiempo real será obtenido por ANTS mediante la aplicación de las reglas incluidas en esta tabla.

ID	TIEMPO	PARÁMETRO	TIEMPO MEDIO	OBTENCIÓN
K	T_k	Tiempo medio necesario para pulsar una tecla del teclado.	0,2 segundos.	Regularidades de Salthouse.

K	T_{km}	Tiempo medio requerido para pulsar un botón del ratón.	0,2 segundos.	Regularidades de Salthouse.
P	T_p	Tiempo medio de desplazamiento del puntero desde un punto origen hasta un punto destino.	1,1 segundos.	Ley de Fitt.
H	T_h	Tiempo medio requerido para cambiar de dispositivo de entrada de datos.	0,4 segundos.	Medición directa.
D	T_d	Tiempo medio para realizar operaciones de arrastre con el puntero.	1,5 segundos.[2]	Ley de Fitt adaptada.
M	T_m	Tiempo medio para operaciones de toma de decisiones inmediata.	1,35 segundos.	Ley de Hick, Principio de la Incertidumbre.

Dado que el modo de empleo de empleo de un *widget* requiere de una combinación especial de operadores, es posible predecir el tiempo medio requerido por cada posible *widget* de un diálogo. De este modo es posible evaluar de forma directa la eficacia del diálogo, seleccionado aquel *widget* cuyo empleo requiera la menor cantidad de tiempo posible para un usuario dado. En el modelo de predicción original, se emplean estimaciones del tiempo necesario por un usuario medio para aplicar cada uno de los operadores. Sin embargo, DEVA emplea valores reales para cada parámetro y usuario. La clave radica en los agentes incluidos en el módulo ANTS, cuyo funcionamiento será detallado a partir del capítulo 19 (*Espías*). Estos agentes observan el comportamiento del usuario activo de la aplicación en cada uno de los diálogos interactivos establecidos por DEVA, haciendo uso de conocidas reglas de la psicología cognitiva para la determinación de los tiempos requeridos por cada operación. De este modo, la precisión del modelo de predicción empleado por DEVA aumenta considerablemente y permite así determinar el tiempo medio necesitado por un usuario concreto del sistema para efectuar una tarea completa sobre un *widget*.

16.1.3. Evaluación Aplicada

Para ilustrar el mecanismo de evaluación de *widgets*, vamos a emplear como ejemplo el proceso de selección empleado por DEVA a la hora de ofrecer al usuario las distintas opciones activadas por la aplicación en un momento y contexto determinado, es decir, el conjunto de procesos de usuario que se encuentran registrados por CodeX en el estado actual de una aplicación y en los que la evaluación de sus respectivas *precondiciones* ha resultado positiva. En la Tabla 25 se recogen los procesos de usuario a emplear en el ejemplo, así como la adaptación semántica con la que aparecerán en la interfaz de usuario.

Cuando DEVA ha de mostrar la bandeja de opciones al usuario, el problema de la selección del modelo de interacción presentado es idéntico al planteado a la hora de seleccionar un *widget*, existiendo por tanto varias opciones para satisfacer la tarea desde un punto de vista

[2] Para un solo segmento de unos 10 pixels de largo.

operacional o técnico, teniendo que seleccionar la opción apropiada empleando criterios cognitivos, perceptivos y motrices.

Tabla 25. Procesos de usuario y sus respectivas adaptaciones semánticas en un ejemplo de selección de estilo de interacción.

PROCESO	CONTENIDO
OpenDoc	*Abrir Documento*
EasyInteger	*Representar Entero*
EasyCounter	*Cuenta Atrás*
InfoPersonal	*Añade tus Datos*
SelectColor	*Elige un Color*
CloseDoc	*Cerrar Documento*

En el caso presentado, DEVA puede optar bien por un modelo de interacción CLI (*Command Line Interface*) o bien por un modelo GUI (*Graphic User Interface*). En el primer caso, especialmente recomendado para usuarios con discapacidades motrices serias o moderadas, la información se presenta mediante un menú formado por elementos de texto numerados que hacen las veces opciones y que el usuario selecciona pulsando el número correspondiente a su elección, como puede verse en la Figura 61. En el segundo caso, las distintas opciones se insertan en un menú incluido en la barra de menús de la ventana correspondiente a la aplicación, menú que se activa cuando el usuario sitúa el cursor sobre el nombre del mismo, provocando su despliegue para la selección de la opción deseada, tal y como puede verse en la Figura 62.

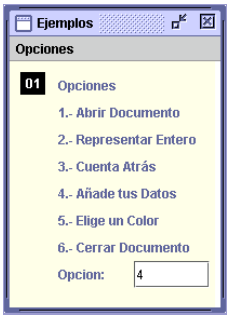


Figura 61. Modo de interacción CLI clásico empleado para satisfacer la transmisión de información determinada por los parámetros del la Tabla 25 (consultar versión en color en el capítulo 24: *Imágenes en Color*, p. 445).

En el modo de interacción basado en CLI, el usuario puede seleccionar la opción adecuada empleando sólo dos teclas, sin hacer uso de mecanismos de desplazamiento del puntero (ratón, joystick, track-ball, etc.) con lo que el proceso de interacción global se simplifica. Por el contrario, el modo de interacción planteado por la interfaz GUI se fundamenta en el uso de técnicas basadas en la señalización y arrastre del puntero. Resulta obvio pensar que el tipo de operadores necesarios para llevar a cabo la ejecución de la misma tarea en los dos modos de

interacción es distinto, así como la secuencia en la que éstos operadores debe ser empleada.

Figura 62. Menú GUI empleado para mostrar las opciones de la Tabla 25. Este menú se puede emplear en combinación con un menú CLI tal y como se puede apreciar en la figura (consultar versión en color en el capítulo 24: *Imágenes en Color*, p. 445).

La secuencia de operadores necesarios para ejecutar una tarea de selección simple para cada uno de los modos de interacción, así como el tiempo medio estimado para la ejecución de dichas tareas se encuentra recogido en la Tabla 26. Nótese como para cada uno de los modos de interacción, el usuario requerirá aplicar el operador M para efectuar el proceso mental que le permita seleccionar la opción deseada. Aunque a nivel de predicción, el orden en el que se aplica este operador es indistinto, es necesario mencionar que en el modo de interacción CLI. El operador M es empleado al principio del proceso, en cuanto las opciones disponibles son mostradas al usuario, mientras que en el modo de interacción GUI, este operador es aplicado en las etapas intermedias del proceso, justo en el momento en el que el menú de opciones es desplegado.

Tabla 26. Operadores y función de tiempos estimados para los modos de interacción CLI y GUI.

MODO	OPERADORES	TIEMPO ESTIMADO
CLI	M H K_k K_k	$T_m + T_h + 2T_k$
GUI	M H P D K_m	$T_m + T_h + T_p + T_d + T_{km}$

El siguiente operador que se aplica en los dos casos descritos es el cambio de contexto (H) en el que el usuario selecciona el dispositivo de interacción que necesita para llevar a cabo el proceso (el ratón o el teclado). Dado que los agentes incorporados en el módulo ANTS inspeccionan constantemente el comportamiento del usuario durante la sesión de trabajo con GADEA, es posible conocer en determinados casos el dispositivo de interacción con el que está actuando el usuario, por lo que el valor T_h podría ser nulo en alguna de las funciones de predicción. Así por ejemplo, si el usuario ha empleado el ratón en su último proceso de interacción y el lapso de tiempo transcurrido entre la finalización de dicho proceso y el presente es moderadamente corto, DEVA puede asumir que el usuario está actuando con el ratón,

asignando un valor T_h nulo para la función de evaluación GUI en los casos en los que el siguiente operador a aplicar sea P o D.

El siguiente operador aplicado en el proceso de selección depende directamente del modo de interacción. En el caso del modo CLI, el usuario deberá aplicar el operador Kk dos veces, una para elegir la opción y otra para confirmarla por medio de la tecla *intro*. En el caso del modo GUI, el número y variedad de operadores a aplicar es mucho más extenso, ya que será necesario aplicar el operador P para desplazar el puntero del lugar original a la barra de menús (concretamente al lugar que ocupa el menú deseado), seguido de una operación de arrastre (D) para desplazar el puntero por la lista de opciones hasta alcanzar la deseada. El proceso finaliza con una operación K_{km} para seleccionar la opción elegida por medio de una pulsación con el botón del ratón.

16.1.4. Evaluación Ponderada

En función de las operadores necesarios para ejecutar una determinada acción, así como de las características específicas de los mismos para un usuario concreto en los niveles cognitivos (M), perceptivos (M, P, D) y motrices (H, P, D, K_k y K_{km}) de dichos operadores el proceso de evaluación realizado por DEVA favorecerá a aquel estilo de interacción más eficiente. En el ejemplo propuesto, si el usuario activo de la aplicación en un momento dado presenta un rendimiento superior con en el uso del ratón (o de cualquier dispositivo de desplazamiento mecánico del puntero), que en el uso del teclado, los parámetros de la función de predicción del estilo GUI poseerán valores más bajos que aquellos empleados en la función de predicción del estilo CLI por lo que, al ser menores los tiempos estimados para el primer estilo de interacción, será éste el elegido.

Sin embargo, dado que el empleo de un determinado estilo de interacción no entra en conflicto con el uso del otro estilo, ambos estilos pueden ser empleados de forma concurrente tal y como se observa en la Figura 62, en donde las opciones recogidas en el modo de interacción CLI incluidas en el panel de la ventana representada, se recogen además en un menú de opciones de tipo GUI. Este empleo concurrente de los dos estilos de interacción puede efectuarse durante las primeras fases de uso de la operación que ha dado origen a la evaluación y selección del estilo, de tal modo que ambos estilos puedan ser empleados de forma indistinta por parte del usuario. Por medio de los agentes ANTS, es posible observar las preferencias de iteración adoptadas por el usuario con respecto a los dos estilos, cuantificando la magnitud de sus preferencias.

De este modo, DEVA puede ponderar cada una de las funciones de predicción en función del conocimiento previo que se dispone sobre las preferencias del usuario, aumentando en este caso el poder de adaptabilidad del sistema en dos frentes distintos: la productividad del modo de interacción y el grado de aceptación por parte del usuario. En estos casos, la ponderación es realizada por medio de los parámetros U_{CLI} y U_{GUI}, los cuales representan la proporción de uso del modo de interacción CLI y la proporción de uso del modo GUI respectivamente.

Las funciones ponderadas para la estimación del tiempo de empleo de los modo CLI y GUI se encuentran recogidas en la Tabla 27.

Tabla 27. Funciones de predicción ponderadas para estilos de interacción CLI y GUI.

MODO	OPERADORES	FUNCIÓN PONDERADA
CLI	M H K_k K_k	$(1 - U_{CLI})\ (T_m + T_h + 2T_k)$
GUI	M H P D K_m	$(1 - U_{GUI})\ (T_m + T_h + T_p + T_d + T_{km})$

Ante decisiones de diseño comprometidas, como por ejemplo una disminución del ancho de banda del canal de comunicación, se puede optar por emplear uno solo de los modos de interacción propuestos en el ejemplo. Será en estos casos donde la función de predicción ponderada muestre toda su valía.

Tabla 28. Valores de operadores y unidades de ponderación de un posible usuario de GADEA.

ID	TIEMPO/FACTOR CORRECTOR	VALOR
K	T_k	0,2 segundos
K	T_{km}	0,1 segundos
P	T_p	0,4 segundos
H	T_h	0,6 segundos
D	T_d	0,5 segundos
M	T_m	1,25 segundos
CLI	U_{CLI}	25%
GUI	U_{GUI}	75%

Supongamos que en el ejemplo anterior, los valores para los tiempos de ejecución de los operadores de un usuario de GADEA son los reflejados en la Tabla 28. Si dichos valores se emplean para calcular el valor de las funciones no ponderadas incluidos en la Tabla 26 veremos como, desde un punto de vista de la productividad y basado en parámetros de bajo nivel (M, H, P, etc.), el modelo de interacción CLI se ve favorecido ya que obtiene unos tiempos de ejecución sensiblemente menores que los obtenidos por el modo GUI (ver Tabla 29).

Tabla 29. Tiempos estimados de ejecución no ponderados para los modos de interacción CLI y GUI en base a los valores recogidos en la Tabla 28.

MODO	FUNCIÓN	EVALUACIÓN
CLI	$T_m + T_h + 2T_k$	1,25 + 0,6 + 2(0,2) = 1,85 + 0,4 = 2,25 segundos
GUI	$T_m + T_h + T_p + T_d + T_{km}$	1,25 + 0,6 + 0,4 + 0,5 + 0,1 = 2,85 segundos

Sin embargo, si se aplican los valores de ponderación recogidos en la Tabla 28, en donde el historial de interacción del usuario con GADEA refleja unas preferencias claras por el empleo del modo de interacción

GUI, el peso específico de las funciones de predicción se invierte, favoreciendo el empleo de éste último modo de interacción, pues presenta un valor ponderado menor al obtenido por el modo CLI (ver Tabla 30).

Tabla 30. Ponderación de las funciones de predicción CLI y GUI en base a los valores recogidos en la Tabla 28.

MODO	FUNCIÓN	EVALUACIÓN
CLI	$(1 - U_{CLI})\ (T_m + T_h + 2T_k)$	(1 – 0,25)(1,25 + 0,6 + 2(0,2)) = 1,85 + 0,4 = (0,75) 2,25 = 1,6875 segundos.
GUI	$(1 - U_{GUI})\ (T_m + T_h + T_p + T_d + T_{km})$	(1 – 0,75)(1,25 + 0,6 + 0,4 + 0,5 + 0,1) = (0,25) 2,85 = 0,7125 segundos.

Naturalmente, éste proceso de evaluación no se limita a la selección de un estilo de interacción determinado, sino que se aplica además a la selección de *widgets* para satisfacer requerimientos de información sobre primitivas de datos de la forma más efectiva posible.

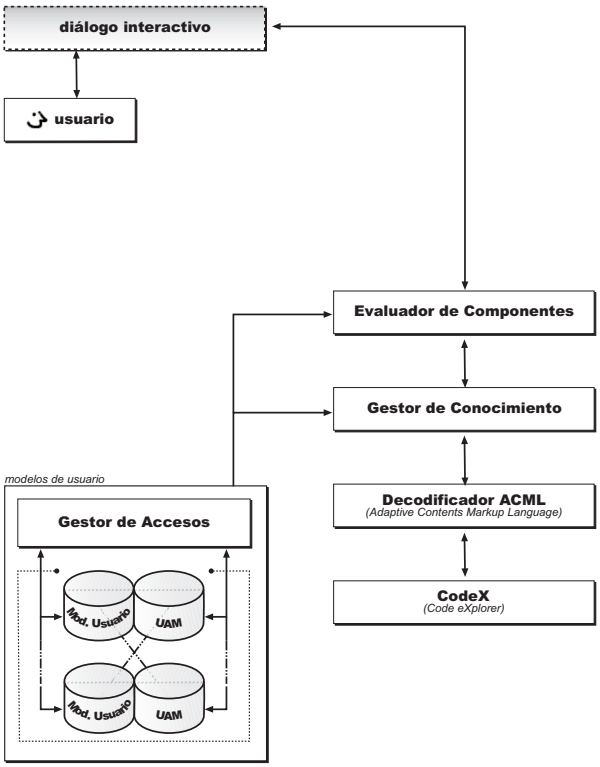


Figura 63. DEVA aprovecha la información recogida por ANTS y almacenada en el modelo de usuario individual de cada usuario para seleccionar en *widget* y modo de interacción apropiado por medio del valuador de componentes.

Por último, cabe destacar que en un entorno totalmente adaptable y basado en un acceso universal a la información, existen ocasiones en donde el mecanismo de evaluación y selección de componentes comentado no puede ser aplicado de forma directa, debido a que pueden no existir valores correctos para los parámetros de tiempo de ejecución derivados de los operadores básicos, o a que dichos

operadores no pueden ser aplicados por determinados individuos debido principalmente a que sufren de algún tipo de discapacidad.

Así por ejemplo, el uso de cualquier dispositivo de desplazamiento del puntero en la pantalla del ordenador carece totalmente de sentido si éste va a ser empleado por un usuario que sufre de discapacidades visuales graves o totales. En este caso concreto, los parámetros P y D no pueden ser aplicados por el usuario para la consecución de la tarea deseada. Nótese que dado que el usuario no puede emplear estos operadores, ANTS no puede –ni debe– medir sus correspondientes parámetros de tiempo de ejecución (T_p y T_d).

En casos como estos, en los que DEVA requiere evaluar una serie de parámetros que no pueden ser medidos por ANTS, dichos parámetros son bloqueados e iniciados al mayor valor posible, de tal modo que mientras se encuentre presenten los factores que impiden su medición, los agentes incluidos en ANTS se abstengan de tomar las correspondientes mediciones. Por otro lado, los parámetros que miden el nivel de uso (U) de un determinado *widget* o modo de interacción que requiere el empleo exclusivo de técnicas de acceso que dependen de los operadores prohibidos, también serán bloqueados por ANTS, para impedir su actualización siendo iniciados al valor nulo.

De este modo, no sólo se gana en eficiencia al evitar la medición de parámetros inútiles, sino que además, el proceso de evaluación realizado por DEVA se aplica por igual a cualquier tipo de usuario, sin realizar excepciones. Nótese que, con valores de uso (1 - U) nulos y parámetros de tiempo de ejecución (T) de valor infinito se evita por completo que el proceso de evaluación permita la selección de *widgets* o de técnicas de interacción de empleo imposible por parte de un usuario discapacitado.

17

17 La Diversidad Humana

Las personas cambian y generalmente se olvidan de comunicar dicho cambio a los demás

Lillian Hellman

17.1 Diversidad y Adaptación

Una vez que el modo de interacción ha sido elegido, así como los distintos *widgets* que satisfarán los requisitos de interacción previstos en el diálogo interactivo, DEVA iniciará la tarea del diseño del diálogo, estableciendo las características físicas –perceptibles– que han de poseer los *widgets* seleccionados, así como su ubicación espacial dentro del canal de comunicación activo.

La definición de las características de estos *wigdets*, así como su configuración, depende directamente del perfil cognitivos, perceptivos y motriz del propio usuario. En este perfil se encuentra definido en el modelo de usuario individual de cada usuario de la aplicación y su empleo concreto en el proceso de adaptación se encuentra matizado por una serie de factores que afectan a la diversidad del proceso perceptivo en el género humano.

17.1.1. Variación de la Percepción con Respecto a la Edad

Las características del procesador humano, tanto las del procesador cognitivo como las del procesador perceptivo y motriz no son estáticas sino todo lo contrario, ya que se encuentran en un estado de permanente evolución a lo largo de toda la vida del individuo. Durante esta evolución, el procesador cognitivo se enriquece con todas las experiencias que se van adquiriendo en la misma medida en la que el procesador perceptivo y el procesador motriz van perdiendo eficacia a causa del cansancio y envejecimiento de los órganos sensoriales y motores. A medida que una sujeto se hace mayor, si bien se detecta un descenso gradual en la eficiencia de los órganos sensoriales y motrices, se presenta una mejora importante de su habilidad para la discriminación perceptiva.

En el estudio del procesador humano, los niños han jugado un papel primordial ya que éstos son a buen seguro los humanos sobre los cuales existen menos problemas para establecer condiciones objetivas de experimentación. A pesar de las limitaciones que su edad impone al tipo de tareas que ellos pueden realizar y sobre las que se puede experimentar, el estudio de la infancia es ideal, pues en él se pueden encontrar las esencias más puras del comportamiento humano. Es por ello que el mayor número de estudios sobre la percepción humana se ha realizado con niños y también es por ello que la mayor cantidad de información disponible sobre la percepción se debe al estudio del comportamiento humano en sus primeras fases de vida.

En el estudio de la evolución perceptiva de los bebés en cuanto a su capacidad para la discriminación, reconocimiento y procesamiento de los estímulos externos, el factor fundamental de estudio ha sido la observación continua del modo en que éstos seleccionan la información que les viene del exterior. El mecanismo de experimentación más sencillo para detectar esta selección ha sido observar el movimiento de los ojos de los niños cuando éstos son estimulados visualmente.

En experimentos de reconocimiento visual, los bebes de aproximadamente un mes de edad se quedan literalmente *enganchados* a la primera característica distintiva de la forma que se les muestra. Los niños de estas edades observan solamente los contornos de las formas, mientras que los posibles elementos interiores permanecen totalmente ignorados. No es hasta los dos meses de edad cuando empiezan a explorar los elementos interiores de las formas, ahora con una precisión casi obsesiva, puesto que tan solo realizan una breve inspección de la zona exterior. Este patrón de exploración se mantiene generalmente hasta la edad de cuatro años, en la que se retoma el hábito de explorar las zonas externas de las formas con más frecuencia, sin menospreciar por ello las zonas interiores.

No es hasta edades comprendidas entre los seis y los siete años cuando se completa el ciclo de ajuste del procesador perceptivo y los niños empiezan a realizar exploraciones sistemáticas de las zonas externas de la forma, detectando se movimientos ocasionales de los ojos hacia la parte interior de la figura. Esta será la estrategia empleada por personas adultas, la cual es compatible con los modelos de atención

bifásicos descritos anteriormente y de donde se deriva el modelo de objetos señuelo y de objetos predador empleado por DEVA. Sin embargo, es necesario destacar que en el caso de usuarios pertenecientes a los conjuntos difusos *bebé* o *niño* descritos en el apartado 11.1.2 (*Registro de Nuevos Usuarios*), es necesario modificar el comportamiento de este modelo de objetos señuelo y objetos predador, ya que como acabamos de ver, éstos usuarios fijan su atención primero en el detalle (el interior de las formas u objetos predador) y luego en los aspectos más generales de la forma (el contorno u objetos señuelo).

Un interesante estudio que confirma este hecho ha sido el realizado por Vurpillot (1985) en su serie de experimentos con niños. En sus experimentos, ésta investigadora presentaba a una serie de niños de distintas edades las figuras de dos casas prácticamente idénticas, salvo por pequeños detalles en las ventanas. El objetivo del experimento consistía en que los niños encontraran lo antes posible las diferencias entre las dos casas, mientras el movimiento seguido por sus ojos era analizado hasta que éstos alcanzasen el objetivo.

Vista la tarea a realizar desde un punto de vista práctico, lo más lógico en términos de eficiencia sería practicar una comparación sistemática entre las dos casas hasta detectar alguna diferencia y concluir así el proceso de búsqueda. Sin embargo, se detectó que los niños de menor edad no realizaban una búsqueda sistemática, sino que paseaban su mirada una y otra vez sobre las dos casas e incluso sobre las dos ventanas sin hallar la diferencia. Los niños de edades más avanzadas, provistos ya de esquemas perceptivos sistemáticos obtuvieron resultados más precisos. Aparte de la ausencia de esquemas perceptivos diseñados para tareas de búsqueda, los niños de menor edad basan su proceso de búsqueda en aspectos muy detallados, sin descartar la información superflua, lo cual introduce grandes cantidades de ruido en la comunicación. En sus búsquedas pasan de la información local a la global, lo cual representa un nivel de búsqueda altamente ineficaz, pues supone analizar uno a uno todos los elementos individuales de las formas, sin descartar bloques de objetos.

En el esquema perceptivo de los niños no cabe la noción de aplicar el clásico *divide y vencerás* analizando la información de las formas mediante la agrupación de los elementos individuales de las mismas. Así, en el experimento de Vurpillot (1985), los niños no agrupan los elementos de las casas en sus componentes básicos (puertas, ventanas, chimenea, etc.) para luego analizarlos de forma individual, como sería lo más lógico. Desde una perspectiva adulta, la estrategia más eficaz consistiría en dividir el problema inicial (diferencia entre dos casas) en problemas más pequeños y fáciles de resolver, como por ejemplo: diferencias entre dos ventanas o diferencias entre dos chimeneas.

De este tipo de comportamiento se deduce que una aplicación directa del modelo de objetos señuelo y objetos depredador descrito anteriormente a un usuario bebé o niño resulta totalmente ineficaz, ya que este tipo de usuarios carecen totalmente de niveles de abstracción suficientes como para realizar un análisis global. En este escenario, solo tienen sentido los objetos predador, que serán los objetos utilizados por DEVA para este tipo de usuarios. Dado que los bebés y los niños

realizan un procesamiento detallado y minucioso de la información sin practicar un agrupamiento de la misma en abstracciones globales, para este tipo de usuarios será necesario mostrar toda la información al máximo nivel de detalle en lugar de transmitir la información global que les permita seleccionar después la información detallada.

A medida que el niño crece tiende a considerar aspectos más globales de las formas y de sus elementos con respecto a los elementos más detallados. Otro ejemplo muy interesante [Weiner y Elkind (1983)] empleado para detectar este factor es el de la figura conocida como el *pájaro vegetal*. En este experimento, se dibuja un pájaro cuyo tronco y extremidades están formados por figuras que representan frutas y hortalizas. En el ejemplo clásico ilustrado en la Figura 64, la cabeza es una naranja, el pico un par de zanahorias, el cuerpo una pera, las patas un par de cerezas y la cola un manojo de zanahorias.

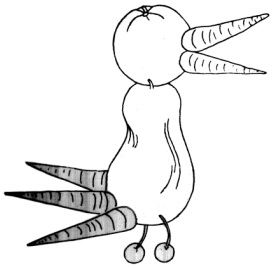


Figura 64. Pájaro vegetal empleado en experimentos basados en la distinción de formas por parte de niños de corta edad.

En este experimento se muestra el pájaro vegetal a un conjunto de niños de diversas edades quienes tienen que indicar todo lo que ven. Los niños con edades comprendidas entre los cuatro y los cinco años, sólo perciben las partes del pájaro, es decir, la frutas y hortalizas (objetos predador). A los siete años, los niños perciben indistintamente tanto los figuras individuales como el propio pájaro (objetos señuelo y/u objetos predador). Por último, a la edad de ocho a nueve años, los niños informan en términos tanto de las partes individuales, como de la organización global de la figura, es decir, mencionan la existencia de un pájaro (objeto señuelo) hecho de frutas y hortalizas (objetos predador).

La facilidad con la que los adultos practican complejas discriminaciones visuales se explica mediante la multitud de estrategias perceptivas desarrolladas a lo largo de los años. Esta habilidad se debe por lo tanto a un extenso proceso de aprendizaje y no a una evolución fisiológica del sistema perceptivo. Con la experiencia, la percepción se relaciona más con las propiedades físicas reales del estímulo ya que es menos propensa a caer en ilusiones perceptivas [González y Cueva (2001)]. Sin embargo, existen situaciones en las que los adultos se ven superados por los niños en la identificación de figuras verticales, sobre todo cuando a un adulto se le muestra un estímulo visual con una orientación distinta de la habitual [Carey (1985)]. Por ejemplo, un adulto tendrá serias dificultades para reconocer un rostro invertido, ya que cuando éste se invierte, pierde gran parte de su apariencia como tal. Este fenómeno acontece incluso con los rostros de personas cuyos rasgos son familiares. Sin embargo, la eficacia en el reconocimiento de

los rostros por un niño de seis años no se ve afectada por la orientación. Su habilidad será la misma independientemente de que la figura se le presente en su orientación normal o en una orientación invertida. Es solo a partir de los diez años cuando el reconocimiento facial se estanca y se asemeja al de un adulto. Por ello, la sensibilidad de los adultos a las diferencias de orientación tienden a llevarlos hacia un peor rendimiento en la discriminación de figuras, respecto de la que se encuentra en un niño.

Como es lógico, a lo largo de la vida los receptores sensoriales envejecen y la eficiencia del sistema perceptivo disminuye, perdiendo parte de su sensibilidad. Al envejecer se reduce la sensibilidad perceptiva de los colores con longitud de onda corta, como los azules. Este fenómeno se traduce en una especial adaptación a la percepción visual en condiciones de obscuridad. Aunque las personas de mayor edad tienen menor precisión en su percepción visual si se las compara con personas jóvenes, presentan en cambio una mejor percepción en condiciones escasas de luz, al aumentar el contraste de lo percibido a obscuras, precisamente debido a esa disminución en la eficiencia de sus receptores sensoriales.

Como se verá en el capitulo 18 (*Adaptación Cromática*) DEVA hará uso de esta especial sensibilidad de las personas mayores hacia los colores de longitud de onda corta adaptando el tono de estos colores para evitar situaciones en las que la recepción de un tipo determinado de información dependa de combinaciones de colores de difícil percepción.

17.1.2. Variación de la Percepción con Respecto al Sexo

El sexo de un individuo es otro factor determinante de su sistema perceptivo ya que existen notables diferencias en cuanto a la percepción de estímulos por parte de ambos sexos. Tanto los hombres como las mujeres segregan conjuntamente hormonas masculinas y femeninas. Es la proporción segregada de cada una de ellas la que establece la diferencia sexual. En estado embrionario el cerebro aún es bisexual. Solo con la segregación hormonal comienza el proceso de diferenciación sexual del cerebro, tanto en su estructura como en su funcionamiento, en un proceso que dura desde la época prenatal hasta la adolescencia. La presencia de determinadas hormonas en un individuo (masculinas o femeninas), pueden alterar no sólo su carácter sino también la concepción espacial del universo que le rodea [Ruiz (1991)].

Como es bien sabido, el cerebro humano consta de dos hemisferios conectados entre sí por fibras nerviosas. El hemisferio derecho es el encargado de gestionar las aptitudes espaciales y es el que los varones desarrollan primero. Es por ello que los hombres disponen de una mayor capacidad espacial que las mujeres. Por su parte, las mujeres desarrollan antes el hemisferio izquierdo, el cual controla la inteligencia verbal y procesa la información de forma lógica. Es por ello que la capacidad verbal femenina supera con creces a la masculina. Está comprobado que existen cuatro veces más niños disléxicos que niñas y

muchos más tartamudos varones. A la edad de seis meses, las niñas contestan con balbuceos más diferenciados a las carantoñas de sus padres y a los dos años hablan más y mejor que los varones. A partir de los once, queda claramente patente la superioridad femenina en la comprensión y expresión verbal.

Aunque las mujeres disponen de mayor capacidad verbal, éstas no utilizan mejor el hemisferio izquierdo. Para resolver un problema, en el cerebro de las mujeres intervienen los dos hemisferios juntos, ya que en el cerebro femenino las capacidades del hemisferio izquierdo se encuentran parcialmente replicadas en el derecho y viceversa. Esto no solo hace a las mujeres más receptivas sino que también las hace menos vulnerables a los traumatismos cráneo-encefálicos. Esta diferencia en el modo de utilización de los hemisferios cerebrales tiene una fuerte relación con la habilidad innata en la realización de tareas espacio-visuales.

Para probar esta relación se han practicado una serie de experimentos en los que el observador (masculino o femenino) ha de reconocer una forma dentro de distintos contextos poco contrastados, intentado confundir al observador empleando en la aplicación del proceso de reconocimiento de formas [Ruiz (1991)]. En todos los casos, completar el experimento satisfactoriamente involucra muchas habilidades espaciales aprendidas, que de acuerdo con los resultados obtenidos, parecen favorecer al hombre. De acuerdo con Peterson (1983) los distintos resultados obtenidos en la solución de estos problemas se deben a diferencias fisiológicas entre los dos sexos. Independientemente del origen de estas diferencias, lo realmente relevante es que estas diferencias existen y que el hombre suele responder mejor que la mujer ante la resolución de tareas espacio visuales.

La influencia de las hormonas en la diferenciación de los sexos también está presente en el funcionamiento de los sentidos y por lo tanto también lo está en el procesador perceptivo. Con respecto al gusto por ejemplo, las mujeres suelen poseer una apreciación mucho más intensa que los hombres en los sabores dulces. También el sentido del olfato las mujeres es más agudo que el de los hombres, precisión que varía en relación directa con su ciclo hormonal. En otros sentidos como el tacto y la audición, la mujer parece tener también una cierta preponderancia. Es bien conocido que la mujer posee mayor sensibilidad táctil que el hombre y que esa sensibilidad se reparte por todo su cuerpo. Las mediciones del espectro auditivo también muestran una especial sensibilidad auditiva en las mujeres y esta diferencia aumenta cuando se trata de frecuencias agudas, haciéndose mucho mas patente con la edad.

En cuanto al sentido de la vista, los hombres presentan una mayor agudeza visual con buenas condiciones de luz. Este sentido se ve influenciado altamente por la presencia o ausencia de hormonas ya que se ha detectado que la precisión visual de las mujeres varía según las fases del ciclo menstrual.

DEVA hace uso de estas diferencias para configurar las diferentes características de los canales de comunicación, diálogos interactivos y *widgets*, en función de que el usuario sea hombre o mujer, ponderando al alza o a la baja el empleo de diferentes modos de interacción en función del sexo del usuario.

Así por ejemplo, si el usuario de una aplicación es una mujer, se ponderarán a la baja los parámetros de su modelo de usuario respectivos a su *Precisión Visual* y a la alta los parámetros correspondientes a su *Precisión Auditiva*, lo que traerá como resultado un mayor contraste visual y un menor contraste auditivo en relación a un diálogo idéntico creado para un usuario de sexo masculino. En el caso del canal de comunicación visual, el tamaño de los objetos a transmitir a un usuario mujer será ligeramente mayor que el de los objetos transmitidos a un usuario hombre. Un tratamiento similar será provisto para la intensidad de los colores empleados, la cual será mayor para interfaces destinados a mujeres. En el caso del canal auditivo, se dará preponderancia al empleo de tonos más agudos para las mujeres y un contraste auditivo menor del empleado para usuarios varones. En apartado 17.2.1 (*Un Modelo Borroso*) se verá como se realiza este tipo de adaptación para el canal de comunicación visual.

17.2 Adaptación de Canales

Tal y como se ha especificado en varios de los capítulos precedentes, gran parte de la información adquirida por el procesador perceptivo de un usuario está determinada por una secuencia de valores relativos en lugar de una colección de valores absolutos. Así por ejemplo, expresando lingüísticamente nuestras percepciones de una interfaz podríamos decir que *El botón obscuro es grande*, en donde *obscuro* y *grande* son conceptos que denotan conjuntos sin límites precisos, los cuales no pueden ser medidos ni caracterizados por medios matemáticos [Velarde (1994)]. En el caso de GADEA por ejemplo, el valor del parámetro *Precisión Auditiva* incluido en nuestro modelo de usuario (ver capítulo 11: *El Application Explorer*) es determinado como una gradación (*Sordo, Bajo, Normal Bajo y Normal Alto*) relacionada con otros parámetros incluidos también en el mismo modelo de usuario (*Edad* y *Sexo*).

17.2.1. Un Modelo Borroso

Cuando una agente interactivo del tipo de DEVA ha de tomar la decisión de definir las características perceptibles de un objeto que mejor se ajustan al modelo perceptivo del usuario al que simula, deberá tomar en cuenta esta relación entre parámetros y sobre todo el hecho de que éstos parámetros no son absolutos sino relativos. Esta es una de las razones que aconsejan que el motor de inferencias empleado por DEVA para la toma de decisiones esté basado en lógica difusa [Zadeh (1965)], en lugar de en lógica de predicados.

Otra razón muy importante que recomienda el uso de sistemas difusos en DEVA radica en que una elevada proporción del conocimiento manejado por el estado del arte actual en el estudio de la interacción y comunicación humana es proporcionado a través del lenguaje natural. El problema que presenta el empleo directo del lenguaje natural en la definición de las reglas y hechos de la base del conocimiento sobre la que el agente interactivo basará sus decisiones radica en la ambigüedad e imprecisión de dicho lenguaje, factor que no es tolerado por un motor de inferencias basado en lógica de predicados pero que si son contemplado por la lógica difusa a través de sus variables lingüísticas [Zadeh (1975)].

Dado que la base de conocimiento empleada por DEVA no es más que un almacén de conocimiento humano y puesto que gran parte de dicho conocimiento es impreciso por naturaleza, las reglas y hechos en él son almacenadas no son ni totalmente ciertas ni totalmente consistentes. Resulta pues mucho más apropiado gestionar esta parte de conocimiento –vago e impreciso– mediante el empleo de conjuntos difusos en lugar de usar conceptos rígidos [Zadeh (1983)]. Debemos tomar en cuenta además que buena parte de la lógica del razonamiento humano no está basada en la lógica tradicional, sino en un lógica con valores de verdad difusos, con conectivas difusas y con reglas de inferencia difusas [Velarde (1992 y 1991a)]. Por todo ello, el empleo de un motor de inferencias basado en lógica difusa se aproxima más al objetivo final, que no es otro que emular el comportamiento humano –a bajo nivel– por medio de un agente interactivo.

El sistema de lógica difusa empleado por DEVA se basa en el motor de inferencias *FuzzySys* desarrollado en Java por Chung y Wai (2000). Aunque existen otros sistemas de lógica difusa (Cox, 1994) la mayoría de ellos no son orientados a objetos ni alcanzan el nivel de flexibilidad de *FuzzySys*. Tal y como se puede apreciar en la figura Figura 65, la arquitectura de este sistema, como la de muchos otros [Klir (1995), Zimmermann (1996)], se fundamenta en un base de reglas, una base de conjuntos difusos y en tres módulos encargados del tratamiento secuencial de la información obtenida por el sistema.

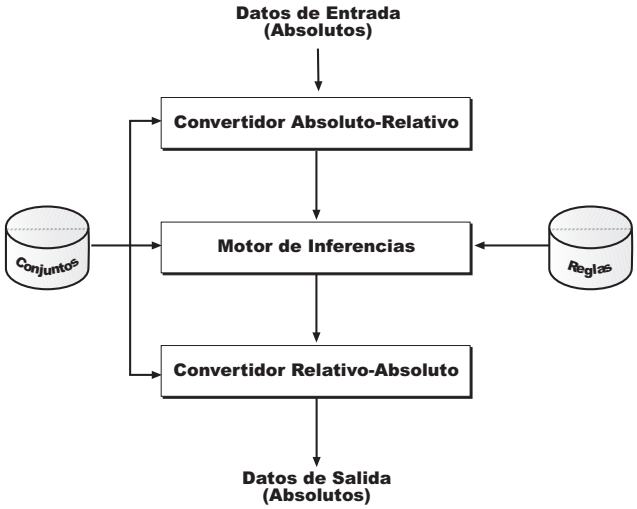


Figura 65. Módulos básicos de un sistema de lógica difusa.

Los tres módulos que integran el sistema de lógica difusa son el convertidor absoluto-relativo, el motor de inferencias y el convertidor relativo-absoluto. En el proceso habitual de toma de decisiones de DEVA, la información de carácter absoluto recibida a través de los agentes del módulo ANTS es convertida a una medida de carácter difuso por el convertidor absoluto-relativo. Este proceso de conversión se basa en el grado de pertenencia del valor absoluto recibido con respecto al conjunto difuso correspondiente a la variable lingüística que éste representa [Zadeh (1975)]. Esta información difusa es procesada por el motor de inferencias, empleando para ello las reglas contenidas en la base de reglas, obteniendo como resultado un conjunto de valores difusos, los cuales son convertidos de nuevo a un valor absoluto por medio del convertidor relativo-absoluto. Los valores absolutos así obtenidos pueden ser empleados por DEVA para configurar el aspecto perceptible de un determinado *widget*, así como especificar las características de un canal de comunicación.

Para emplear este sistema de lógica difusa, DEVA crea una instancia de cada uno de los módulos descritos, ordenando la ejecución del proceso de toma de decisiones invocando a los métodos *inputProc*, *procRules* y *outputProc* de las clases asociadas a los módulos convertidor absoluto-relativo, motor de inferencias y convertidor relativo-absoluto respectivamente. Naturalmente, cada uno de estos métodos es invocado por separado cuando la situación así lo requiere.

El convertidor absoluto-relativo es capaz de gestionar más de una entrada de información a la vez, de tal modo que puede controlar el estado de mas de una variable absoluta contenida en el modelo de usuario, verificando cualquier cambio en su valor por medio de la observación de los eventos generados por los agentes encargados de su actualización. Cuando DEVA invoca al método *inputProc* de este módulo, las variables registradas son analizadas para determinar si se ha producido algún cambio en su valor. De producirse éste, se procede a convertir el valor de la variable absoluta en su correspondiente valor difuso en función de la información contenida en la base de conjuntos difusos. A continuación se procede a realizar un filtrado del valor obtenido para iniciar a cero el valor de aquellas variables lingüísticas con un pequeño grado de pertenencia con respecto al al correspondiente conjunto difuso [Velarde (1991) p. 111: *Análisis gnoseológico de la posibilidad frente a la probabilidad*].

Tabla 31. Conjuntos difusos asociados a la variable lingüística *Tamaño* de un objeto.

CONJUNTO	RANGO
Nulo	[0%...25%)
Bajo	[25%...50%)
Medio	[50%...75%)
Grande	[90%...100%]

Una vez que la información absoluta contenida en las variables lingüísticas ha sido convertida en valores difusos, entra en juego el

motor de inferencias de DEVA para llevar a cabo un proceso de toma de decisiones siguiendo las normas establecidas en la lógica difusa [Zadeh (1965)]. Las lógica necesaria para realizar este proceso se establece en la base de reglas por medio de un documento de texto en el que se indican explícitamente las reglas a emplear usando un lenguaje muy cercano al natural, tal y como se puede apreciar en el Código 31, el cual recoge parte de las reglas empleadas por DEVA para configurar diferentes aspectos de un canal de comunicación.

Código 31. Selección de reglas de lógica difusa empleadas por DEVA para la configuración de *widgets* y canales de comunicación.

```
//OBJETO PASIVO
(weigth 1.2) If Precision Visual of User is Discapacidad then
Tamano of Objeto Pasivo is grande

If Precision Visual of User is Bajo then Tamano of Objeto Pasivo is
Medio

If Precision Visual of User is Alto then Tamano of Objeto Pasivo is
Bajo

If Precision Visual of User is Excelente then Tamano of Objeto
Pasivo is Nulo

// OBJETO ACTIVO
If Precision Visual of User is Discapacidad then Tamano of Objeto
Activo is Grande

If Precision Motriz of User is Baja AND Precision Visual of User is
Bajo then Tamano of Objeto Activo is Grande

If Precision Motriz of User is Normal AND Precision Visual of User
is Bajo then Tamano of Objeto Activo is Grande

If Precision Motriz of User is Alta AND Precision Visual of User is
Bajo then Tamano of Objeto Activo is Medio

If Precision Motriz of User is Excelente AND Precision Visual of
User is Bajo then Tamano of Objeto Activo is Bajo

If Precision Motriz of User is Baja AND Precision Visual of User is
Alto then Tamano of Objeto Activo is Grande

If Precision Motriz of User is Normal AND Precision Visual of User
is Alto then Tamano of Objeto Activo is Medio

If Precision Motriz of User is Alta AND Precision Visual of User is
Alto then Tamano of Objeto Activo is Nulo

If Precision Motriz of User is Excelente AND Precision Visual of
User is Alto then Tamano of Objeto Activo is Nulo

If Precision Motriz of User is Baja AND Precision Visual of User is
Excelente then Tamano of Objeto Activo is Normal

If Precision Motriz of User is Normal AND Precision Visual of User
is Excelente then Tamano of Objeto Activo is Medio

If Precision Motriz of User is Alta AND Precision Visual of User is
Excelente then Tamano of Objeto Activo is Nulo

// SONIDO
(weigth 1.2) If Precision Auditiva of User is Discapacidad then
tono of Canal is maximo

(weigth 1.2) If Precision Auditiva of User is Discapacidad then
volumen of Canal is maximo

If Edad of User is Octogenario AND Precision Auditiva of User is
Bajo AND Sexo of User is Hombre then Contraste Auditivo of Canal is
Maximo
```

If Edad **of** User **is** Octogenario **AND** Precision Auditiva **of** User **is** Alto **AND** Sexo **of** User **is** Hombre **then** Contraste Auditivo **of** Canal **is** Medio Alto

If Edad **of** User **is** Octogenario **AND** Precision Auditiva **of** User **is** Excelente **AND** Sexo **of** User **is** Hombre **then** Contraste Auditivo **of** Canal **is** Medio Bajo

If Edad **of** User **is** Adulto **AND** Precision Auditiva **of** User **is** Bajo **AND** Sexo **of** User **is** Hombre **then** Contraste Auditivo **of** Canal **is** Alto

If Edad **of** User **is** Adulto **AND** Precision Auditiva **of** User **is** Alto **AND** Sexo **of** User **is** Hombre **then** Contraste Auditivo **of** Canal **is** Normal

If Edad **of** User **is** Adulto **AND** Precision Auditiva **of** User **is** Excelente **AND** Sexo **of** User **is** Hombre **then** Contraste Auditivo **of** Canal **is** Medio Bajo

If Edad **of** User **is** Nino **AND** Precision Auditiva **of** User **is** Bajo **AND** Sexo **of** User **is** Hombre **then** Contraste Auditivo **of** Canal **is** Medio Alto

If Edad **of** User **is** Nino **AND** Precision Auditiva **of** User **is** Alto **AND** Sexo **of** User **is** Hombre **then** Contraste Auditivo **of** Canal **is** Medio Bajo

If Edad **of** User **is** Nino **AND** Precision Auditiva **of** User **is** Excelente **AND** Sexo **of** User **is** Hombre **then** Contraste Auditivo **of** Canal **is** Bajo
--
If Edad **of** User **is** Octogenario **AND** Precision Auditiva **of** User **is** Bajo **AND** Sexo **of** User **is** Hombre **then** Contraste Auditivo **of** Canal **is** Maximo

If Edad **of** User **is** Octogenario **AND** Precision Auditiva **of** User **is** Alto **AND** Sexo **of** User **is** Mujer **then** Contraste Auditivo **of** Canal **is** Medio

If Edad **of** User **is** Octogenario **AND** Precision Auditiva **of** User **is** Excelente **AND** Sexo **of** User **is** Mujer **then** Contraste Auditivo **of** Canal **is** Bajo

If Edad **of** User **is** Adulto **AND** Precision Auditiva **of** User **is** Bajo **AND** Sexo **of** User **is** Mujer **then** Contraste Auditivo **of** Canal **is** Medio Alto

If Edad **of** User **is** Adulto **AND** Precision Auditiva **of** User **is** Alto **AND** Sexo **of** User **is** Mujer **then** Contraste Auditivo **of** Canal **is** Medio Bajo

If Edad **of** User **is** Adulto **AND** Precision Auditiva **of** User **is** Excelente **AND** Sexo **of** User **is** Mujer **then** Contraste Auditivo **of** Canal **is** Bajo

If Edad **of** User **is** Nino **AND** Precision Auditiva **of** User **is** Bajo **AND** Sexo **of** User **is** Mujer **then** Contraste Auditivo **of** Canal **is** Medio

If Edad **of** User **is** Nino **AND** Precision Auditiva **of** User **is** Alto **AND** Sexo **of** User **is** Mujer **then** Contraste Auditivo **of** Canal **is** Bajo

If Edad **of** User **is** Nino **AND** Precision Auditiva **of** User **is** Excelente **AND** Sexo **of** User **is** Mujer **then** Contraste Auditivo **of** Canal **is** Nulo

//CONTRASTE VISUAL
(weigth 1.2) If Precision Visual **of** User **is** Discapacidad **then** Contraste Visual **of** Canal **is** maximo

If Edad **of** User **is** Octogenario **AND** Precision Visual **of** User **is** Bajo **AND** Sexo **of** User **is** Hombre **then** Contraste Visual **of** Canal **is** Maximo

If Edad **of** User **is** Octogenario **AND** Precision Visual **of** User **is** Bajo **AND** Sexo **of** User **is** Hombre **then** Contraste Visual **of** Canal **is** Alto

If Edad **of** User **is** Octogenario **AND** Precision Visual **of** User **is** Alto **AND** Sexo **of** User **is** Mujer **then** Contraste Visual **of** Canal **is** Medio Alto

```
If Edad of User is Octogenario AND Precision Visual of User is Alto
AND Sexo of User is Hombre then Contraste Visual of Canal is Normal

If Edad of User is Octogenario AND Precision Visual of User is
Excelente AND Sexo of User is Mujer then Contraste Visual of Canal
is Normal

If Edad of User is Octogenario AND Precision Visual of User is
Excelente AND Sexo of User is Hombre then Contraste Visual of Canal
is Medio Bajo

If Edad of User is Adulto Maduro AND Precision Visual of User is
Bajo AND Sexo of User is Mujer then Contraste Visual of Canal is
Alto

If Edad of User is Adulto Maduro AND Precision Visual of User is
Bajo AND Sexo of User is Hombre then Contraste Visual of Canal is
Medio Alto

If Edad of User is Adulto Maduro AND Precision Visual of User is
Alto AND Sexo of User is Mujer then Contraste Visual of Canal is
Normal

If Edad of User is Adulto Maduro AND Precision Visual of User is
Alto AND Sexo of User is Hombre then Contraste Visual of Canal is
Medio Bajo

If Edad of User is Adulto Maduro AND Precision Visual of User is
Excelente AND Sexo of User is Mujer then Contraste Visual of Canal
is Medio Bajo

If Edad of User is Adulto Maduro AND Precision Visual of User is
Excelente AND Sexo of User is Hombre then Contraste Visual of Canal
is Bajo

If Edad of User is Joven Adulto AND Precision Visual of User is
Bajo AND Sexo of User is Mujer then Contraste Visual of Canal is
Medio Alto

If Edad of User is Joven Adulto AND Precision Visual of User is
Bajo AND Sexo of User is Hombre then Contraste Visual of Canal is
Normal

If Edad of User is Joven Adulto AND Precision Visual of User is
Alto AND Sexo of User is Mujer then Contraste Visual of Canal is
Medio Bajo

If Edad of User is Joven Adulto AND Precision Visual of User is
Alto AND Sexo of User is Hombre then Contraste Visual of Canal is
Bajo

If Edad of User is Joven Adulto AND Precision Visual of User is
Excelente AND Sexo of User is Mujer then Contraste Visual of Canal
is Bajo

If Edad of User is Joven Adulto AND Precision Visual of User is
Excelente AND Sexo of User is Hombre then Contraste Visual of Canal
is Nulo

If Edad of User is Nino AND Precision Visual of User is Bajo AND
Sexo of User is Mujer then Contraste Visual of Canal is Normal

If Edad of User is Nino AND Precision Visual of User is Bajo AND
Sexo of User is Hombre then Contraste Visual of Canal is Medio Bajo

If Edad of User is Nino AND Precision Visual of User is Alto AND
Sexo of User is Mujer then Contraste Visual of Canal is Bajo

If Edad of User is Nino AND Precision Visual of User is Alto AND
Sexo of User is Hombre then Contraste Visual of Canal is Nulo

If Edad of User is Nino AND Precision Visual of User is Excelente
AND Sexo of User is Mujer then Contraste Visual of Canal is Nulo


// CANTIDAD DE AZUL PERCIBIDO
```

```
(weigth 1.2) If Edad of User is Nino then Azul Percibido of Canal
is Máximo

If Edad of User is Joven Adulto then Azul Percibido of Canal is
Alto

If Edad of User is Adulto Maduro then Azul Percibido of Canal is
Normal

If Edad of User is Octogenario then Azul Percibido of Canal is
Medio Bajo
```

Cuando se invoca al método *procRules* de este módulo, se toma el vector de variables lingüísticas de contenido difuso obtenido por medio del convertidor absoluto-relativo y se procede a calcular el valor de todas las premisas del antecedente de las reglas, empleando el valor mínimo para el operador *and* y el valor máximo para el operador *or*, obteniendo entonces el grado de satisfacción del precedente o DOF (*Degree of Fulfillment*) [Berkan (1997)]. Como se puede apreciar en el Código 31, cada DOF puede ser ponderado para aumentar o minimizar su peso en el proceso de toma de decisiones por medio del descriptor *weight*.

Tabla 32. Conjuntos difusos asociados a las variable lingüísticas *Contraste Visual*, *Contraste Auditivo*, *Tamaño* y *Azul Percibido* de un canal de comunicación.

CONJUNTO	RANGO
Nulo	(0%…15%)
Bajo	[15%…30%)
Medio Bajo	[30%…45%)
Normal	[45%…60%]
Medio Alto	[60%…85%)
Alto	[85%…95%)
Máximo	[95%…100%]

Una vez obtenido el valor del DOF para el precedente de cada regla, éste es asignado como valor a la variable lingüística que actúa como consecuente de la misma. Cada una de estas variables puede recibir el valor de más de un DOF en situaciones en las que existan más de una regla en la que la mencionada variable actúe como consenciente. En estos casos, el valor asumido por la variable se corresponderá con el máximo de los grados de satisfacción del precedente, ya que operador que actúa entre las reglas es el operador *or*. Como ejemplo de este caso particular, en el Código 32 se han recogido varias reglas que eventualmente asignarían un valor *grande* al tamaño de un objeto. Como se acaba de mencionar, el grado de satisfacción para el valor *grande* del objeto será el máximo de los grados de satisfacción correspondientes a todas y cada una de las reglas incluidas en el mencionado Código 32.

Código 32. Selección de reglas para la asignación de un tamaño *grande al Contraste Visual* de un canal de comunicación..

```
//CONTRASTE VISUAL
If Edad of User is Octogenario AND Precision Visual of User is Alto
AND Sexo of User is Hombre then Contraste Visual of Canal is Normal

If Edad of User is Octogenario AND Precision Visual of User is
Excelente AND Sexo of User is Mujer then Contraste Visual of Canal
is Normal

If Edad of User is Adulto Maduro AND Precision Visual of User is
Alto AND Sexo of User is Mujer then Contraste Visual of Canal is
Normal

If Edad of User is Joven Adulto AND Precision Visual of User is
Bajo AND Sexo of User is Hombre then Contraste Visual of Canal is
Normal

If Edad of User is Nino AND Precision Visual of User is Bajo AND
Sexo of User is Mujer then Contraste Visual of Canal is Normal
```

Por último, el convertidor relativo-absoluto convierte el valor difuso de una determinada variable lingüística en un valor absoluto cuando DEVA invoca al método *outputProc* de este módulo, adecuando el grado de pertenencia de la variable al correspondiente conjunto difuso, de acuerdo con métodos tradicionales de conversión recogidos por Zimmermann (1996).

Tabla 33. Conjuntos difusos asociados a la variable lingüística *Precisión Motriz*.

CONJUNTO	RANGO
Nula	(0%...10%)
Baja	[10%...40%)
Normal	[40%...75%)
Alta	[75...90%)
Excelente	[90%...95%)

17.2.2. La Base de Reglas

En el Código 31 se puede observar una selección significativa de las reglas de inferencia basadas en lógica difusa aplicadas por DEVA para la toma de decisiones. En ella se incluyen entre otras, los factores tenidos en cuenta por el agente interactivo para definir la magnitud del tamaño de los objetos visuales, el contraste de los canales de comunicación e incluso la cantidad de tonos de azul que percibirá el usuario en función de su edad. La aplicación de estas reglas es realizada por el módulo Diseñador de Diálogos de DEVA en el momento de configurar los componentes que integran un diálogo interactivo, permitiendo una adaptación automática de la interfaz al estado cognitivo, perceptivo y motriz del usuario, adaptación que, tal y como se puede apreciar, se realiza a nivel de sintaxis.

Con respecto a las reglas empleadas por DEVA, cabe destacar la diferencia que éste agente interactivo hace entre los objetos denominados *pasivos* y los objetos denominados *activos*. Para GADEA, los objetos pasivos son todos aquellos objetos visuales empleados para mostrar información al usuario pero que no permiten que éste pueda actuar directamente sobre ella mediante el empleo de las técnicas

clásicas de desplazamiento del puntero por la pantalla. Se trata de *widgets* de tipo etiqueta o de grandes colecciones de texto diseñadas para transmitir información única y exclusivamente desde la aplicación a la interfaz. Por el otro lado, los objetos activos son aquellos que no solo cumplen con la labor de transmisión de información desde el interfaz al usuario sino que recogen además información proporcionada por el mismo a través de operaciones con dispositivos de señalización (ratón, track-ball, joystick, etc.) Se trata por tanto de objetos en donde la acción del usuario es activa y requiere de operaciones que implican el uso de su procesador motriz.

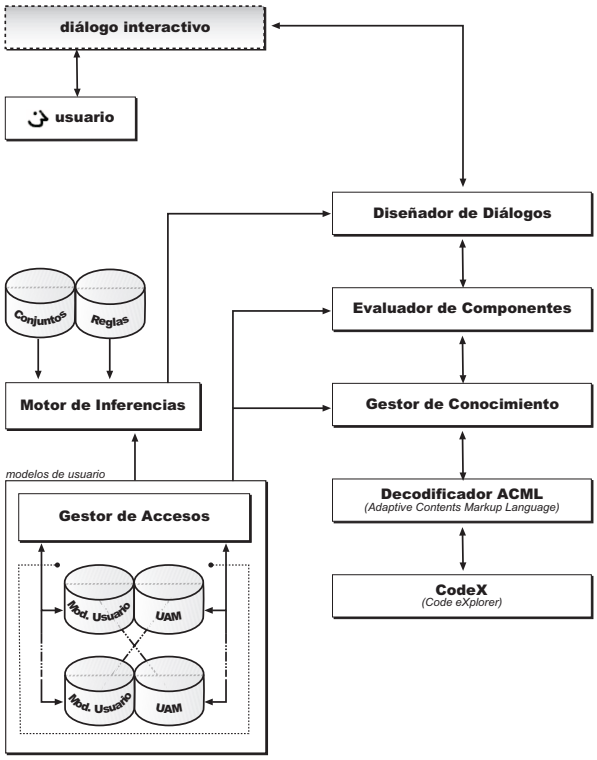


Figura 66. La adaptación de los *widgets* de un diálogo es realizada por el Diseñador de Diálogos en base a los resultados obtenidos por el motor de inferencias al aplicar las reglas de lógica difusa sobre los hechos recopilados por los agentes ANTS.

Los factores tomados en consideración para la determinación del tamaño físico de los objetos visuales dependen del tipo de objeto a emplear, ya que en el caso de los objetos pasivos, su configuración sólo depende del estado del procesador cognitivo y perceptivo del usuario, mientras que en el caso de los objetos activos, es necesario considerar además el estado del procesador motriz, quien jugará un papel fundamental en el uso que se le otorgue al *widget* a configurar. Mientras que en el caso de los objetos pasivos el tamaño de los objetos depende directamente de la *Precisión Visual* del usuario, el tamaño de los objetos activos se ve afectado además por la habilidad con la que el usuario es capaz de manejar dispositivos de señalización, parámetro al que se ha denominado *Precisión Motriz* y que será calculado por los agentes ANTS (ver capítulo 19: *Espías*). Los valores de los conjuntos difusos establecidos para este parámetro se encuentran recogidos en la Tabla 33.

Si el estado del procesador perceptivo y del procesador motriz de un usuario es bueno, el tamaño elegido para un *widget* será relativamente pequeño. Lo contrario ocurrirá en las situaciones en las que el procesador perceptivo o el procesador motriz presenten un rendimiento bajo, configurándose entonces un *widget* de tamaño relativamente grande, el cual será más fácil de percibir y de acceder que uno pequeño. En líneas generales, mientras mejor sea el rendimiento del procesador perceptivo y del procesador motriz menor será el tamaño de los objetos y viceversa. Un razonamiento similar se aplica a el caso del canal de comunicación auditivo, en donde el tono y volumen del sistema de lectura en voz alta empleado por DEVA [IBM (2001)] se define en función de la *Precisión Auditiva* del usuario, aumentando el tono de la voz a medida que la precisión del procesador perceptivo es menor y viceversa.

En cualquier caso resulta necesario destacar que las reglas que permiten definir el tamaño relativo de un objeto son matizadas de acuerdo con los parámetros *Edad* y *Sexo* almacenados en el modelo de usuario activo en la forma en la que se ha determinado en el apartado 17.1 (*Diversidad y Adaptación*), dado que tanto el valor de la *Precisión Visual* como el de la *Precisión Auditiva* o el de la *Precisión Motriz* han de ser ponderados en relación al efecto que la *Edad* o el *Sexo* del usuario tienen sobre ellos.

En la Tabla 32, se recogen –entre otros– los valores de adecuación correspondientes al tamaño de un objeto (sea este activo o pasivo) dentro del marco de conjuntos difusos al que pertenece este parámetro. Como se puede apreciar, el valor absoluto del tamaño de un objeto para los distintos conjuntos difusos no es una medida real aplicable en pixels, ya que se trata de una medida ideal especificada en porcentajes. Este porcentaje representa la tasa aplicada para aumentar el tamaño base de un objeto.

Este tamaño base es calculado función a un usuario con parámetros de comportamiento óptimos para su procesador cognitivo, perceptivo y motriz. De este modo, el porcentaje absoluto obtenido tras la ejecución del proceso de toma de decisiones, servirá para aumentar el tamaño del objeto en función del desfase existente entre estos valores ideales y los valores reales almacenados en el modelo de usuario del usuario activo. Para ilustrar este proceso, supondremos que DEVA está determinando el valor del tamaño final de un botón cuyo tamaño y aspecto base se encuentra definido en la Tabla 34.

Tabla 34. Posibles dimensiones de partida y aspecto de un botón antes de la configuración de su tamaño.

ANCHO BASE	ALTO BASE	TAMAÑO FUENTE BASE	ASPECTO BASE
75 pixels	25 pixels	12 puntos	Botón

Dado que el objeto en cuestión es un botón y por lo tanto se trata de un objeto activo, las reglas relacionadas con el cálculo de su tamaño serán todas aquellas relacionadas con este parámetro, las cuales se han sido

recogidas en el Código 33. Como se puede apreciar, los factores empleados para la determinación del tamaño son la *Precisión Visual* y la *Precisión Motriz* del usuario, los cuales son proporcionados a DEVA por el Gestor de Accesos del modelo de usuario. Mientras que –al menos en esta versión de GADEA– el valor del primer parámetro es un valor explícito proporcionado por el usuario en su proceso de registro en el *Application Explorer* (ver 11: *El Application Explorer*), el segundo valor es calculado de forma continua por los agentes ANTS.

Código 33. Selección de reglas para la determinación del tamaño de un objeto activo.

```
If Precision Visual of User is Discapacidad then Tamano of Objeto
Activo is Grande

If Precision Motriz of User is Baja AND Precision Visual of User is
Bajo then Tamano of Objeto Activo is Grande

If Precision Motriz of User is Normal AND Precision Visual of User
is Bajo then Tamano of Objeto Activo is Grande

If Precision Motriz of User is Alta AND Precision Visual of User is
Bajo then Tamano of Objeto Activo is Medio

If Precision Motriz of User is Excelente AND Precision Visual of
User is Bajo then Tamano of Objeto Activo is Bajo

If Precision Motriz of User is Baja AND Precision Visual of User is
Alto then Tamano of Objeto Activo is Grande

If Precision Motriz of User is Normal AND Precision Visual of User
is Alto then Tamano of Objeto Activo is Medio

If Precision Motriz of User is Alta AND Precision Visual of User is
Alto then Tamano of Objeto Activo is Nulo

If Precision Motriz of User is Excelente AND Precision Visual of
User is Alto then Tamano of Objeto Activo is Nulo

If Precision Motriz of User is Baja AND Precision Visual of User is
Excelente then Tamano of Objeto Activo is Normal

If Precision Motriz of User is Normal AND Precision Visual of User
is Excelente then Tamano of Objeto Activo is Medio

If Precision Motriz of User is Alta AND Precision Visual of User is
Excelente then Tamano of Objeto Activo is Nulo
```

El primer paso a realizar por el motor de inferencias de DEVA será convertir el valor absoluto de las variables *Precisión Visual* y *Precisión Motriz* en un valor relativo determinado el grado de pertenencia del valor absoluto de estas variables con respecto a los diferentes conjuntos difusos empleados por las mismas (Tabla 15 y Tabla 33 respectivamente). A continuación, se calcula los grados de satisfacción (DOF) de cada una de las reglas incluidas en el Código 33, el cual se obtiene determinando el mínimo de los grados de pertenencia de cada una de las variables empleadas en el antecedente en cada regla con respecto al conjunto difuso especificado en la mencionada regla, ya que en todas las reglas –salvo la primera– se emplea el operador *and*. Así por ejemplo, para calcular el DOF de la segunda regla, será preciso determinar el mínimo entre el grado de pertenencia del valor absoluto de la variable *Precisión Motriz* con respecto al conjunto *Baja* y del grado de pertenencia del valor absoluto de la variable *Precisión Visual* al conjunto *Bajo*. Por último, el máximo entre los grados de satisfacción calculados (operador *or*) determinará el valor relativo del tamaño del objeto activo.

Supongamos que el máximo grado de satisfacción se corresponde con la tercera regla del Código 33. Entonces, el valor de la variable *tamaño* del objeto activo será *grande*. Qué tan grande será este tamaño será determinado por el valor numérico del grado de satisfacción que dio origen a la selección de la regla, existiendo por tanto diversos matices numéricos para dicho valor *grande*. En todo caso, el valor absoluto para el valor de la variable tamaño está determinado en la Tabla 31 y se sitúa entre un 90% y un 100%. Esto quiere decir que el valor numérico absoluto de la variable se situará entre estos valores, acercándose al 100% mientras mayor sea el DOF del precedente de la regla. Es decir, mientras más *normal* sea la *Precisión Motriz* del usuario y más *bajo* sea la *Precisión Visual* del mismo, más *grande* debería ser el tamaño del botón.

Suponiendo que el resultado absoluto obtenido para la variable *tamaño* sea un 93%, el tamaño por defecto incluido en la Tabla 34 debería aumentarse en igual proporción, dando como resultado las dimensiones del botón recogido en la Tabla 35, las cuales serán empleadas finalmente para mostrar el *widget* al usuario. En la Tabla 35 el ancho y alto del botón, así como el tamaño de la fuente del texto del mismo han sido aumentados un 93%.

Tabla 35. Dimensiones y aspecto adaptado del botón de la Tabla 34.

ANCHO FINAL	ALTO FINAL	TAMAÑO FUENTE FINAL	ASPECTO FINAL
140 pixels	47 pixels	22 puntos	

17.2.3. Reparto del Ancho de Banda

Una vez que DEVA ha seleccionado y configurado los *widgets* a transmitir en un canal de comunicación, este agente interactivo ha de establecer la correspondiente asignación del ancho de banda del canal entre los distintos *widgets*. Esta asignación dependerá de la prioridad asignada a cada uno de los *chunks* por parte de la aplicación, según éstos hagan el papel de objetos señuelo o de objetos predador.

En el caso del canal de comunicación auditivo, la asignación es directa, dado que aquellos objetos dotados de la máxima prioridad se transmitirán primero, dejando la transmisión de los objetos de baja prioridad para las últimas fases. De este modo, la prioridad asignada a cada objeto se refleja claramente en el orden de aparición en escena. El canal de comunicación visual sin embargo, presenta más problemas puesto que éste no es lineal como el canal de comunicación auditivo, sino que presenta varias dimensiones (ancho, largo y profundidad), entrando en juego las diferentes metáforas de orientación que podrá emplear el usuario para explorar la información recibida [Lakoff (1980)].

En un espacio en dos dimensiones como es el caso de las ventanas y paneles en donde se han de situar físicamente los *chunks* que componen la información de un canal de comunicación visual, es posible aplicar al

menos dos tipos de metáforas de orientación: las metáforas de orientación vertical y las metáforas de orientación horizontal. En cada una de estas metáforas, la posición de un objeto en el eje de coordenadas cartesiano determinará parte de la relevancia que éste tiene para el observador, relevancia que será aprovechada por DEVA para asignar valores de prioridad en base a la posición asignada a cada objeto.

La ubicación vertical de un objeto en cualquier espacio (tridimensional o en dos dimensiones), se relaciona directamente con la metáfora *arriba es más* y *abajo es menos* [Shulz et. al. (1997)]. Esta metáfora se fundamenta en propiedades físicas que la mayoría de los humanos experimentan en su ambiente físico a lo largo de su vida y con las que se encuentran familiarizados, tales como la gravedad. En medios naturales y sociales en donde exista una clara distinción entre el fondo y la forma de una figura, los términos arriba y abajo han dado origen a numerosas metáforas de orientación e incluso a símbolos basados en ellas, existiendo significados universales para éstos, los cuales son independientes de los entornos en el que fueron creados. La influencia de la gravedad, presente en todas las culturas de la Tierra, hace que los conceptos de *arriba* y *abajo*, así como la percepción de la verticalidad sean fácilmente identificables en cualquier parte del mundo [Liungman (1992)].

Sin embargo, la clara distinción que se hace entre arriba y abajo no puede trasladarse a los conceptos de izquierda y derecha ya que en las metáforas horizontales no siempre se puede realizar asociaciones del tipo de *izquierda es menos* y *arriba es más*. Esto es debido a que los humanos suelen presentar mayores dificultades a la hora de distinguir entre los conceptos *izquierda* y *derecha* puesto que en determinadas situaciones la izquierda puede ser la derecha y viceversa. Por ejemplo, si dos personas se miran una a la otra, la izquierda de una persona es la derecha de la otra y viceversa. Dado que no existen condicionantes naturales que determinen el valor de estos conceptos (izquierda y derecha), la asignación de contenidos a estos conceptos ha dependido siempre del entorno cultural que los emplea [Shulz et. al. (1997)]. Así por ejemplo, en algunas culturas, se emplea la izquierda para representar el pasado y la derecha para representar el futuro (ver Figura 67) y en otras culturas ocurre justo lo contrario [Liungman (1972)].

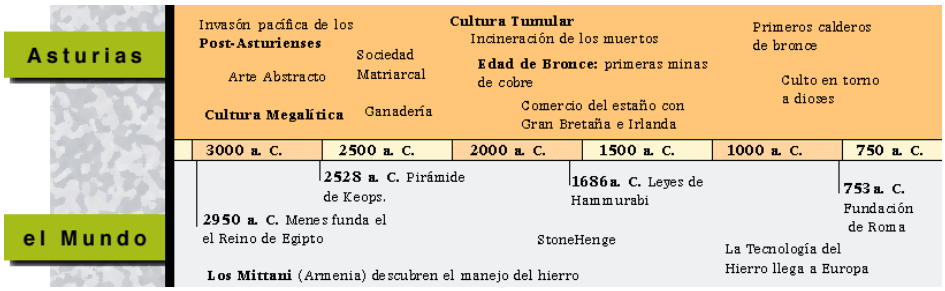


Figura 67. Representación cronológica de la Historia de Asturias. En este caso, a la izquierda se representa el pasado y a la derecha el futuro. [Fuente: ARQUEOASTUR (2001)] (consultar versión en color en el capítulo 24: *Imágenes en Color*, p. 445).

Con la invención de la escritura en plena Edad del Bronce, no solo se ha definido un estándar para el almacenamiento de las ideas, sino que se establece también una definición formal para la percepción visual de las mismas por medio de las imágenes. No se debe olvidar que los caracteres de un alfabeto, son al fin y al cabo imágenes de un tipo peculiar. Lamentablemente y como suele suceder cuando varios grupos humanos trabajan por separado sobre una misma invención, cada entorno cultural diseñó su propio sistema de escritura y con él definió también su propio mecanismo de percepción visual

El mecanismo de interpretación de una imagen –y con ello también el de un espacio visual– es idéntico al empleado para leer un texto. Desde el punto de vista de la cultura occidental, cuando se lee una imagen, se empieza a hacerlo por su esquina superior izquierda, de modo similar a cuando se empieza a leer la página de un libro. A un nivel inconsciente, se recorre el primer *renglón* de la imagen de izquierda a derecha, pasando luego a leer el siguiente renglón situado un poco más abajo del primero y así sucesivamente hasta terminar en la esquina inferior derecha de la imagen.

En líneas generales, el mecanismo de percepción visual rastreará la imagen por medio de una diagonal trazada desde la esquina superior izquierda de la misma hasta su esquina inferior derecha (ver Figura 68). En el caso de los pueblos semitas, en donde el sistema de escritura se basa en renglones escritos de derecha a izquierda, el patrón de visualización de la imagen irá desde la esquina superior derecha de la misma hasta la esquina inferior izquierda. En las culturas orientales, como la china o la japonesa, en donde la lectura se realiza en líneas verticales, el patrón seguido es análogo.

Dentro del marco de la cultura occidental, lo primero que se observa de una imagen es su esquina superior izquierda y por lo tanto será éste su punto de máxima atracción, el cual se denomina *foco*. La esquina inferior derecha de la imagen es lo último que se ve y por lo tanto la parte menos relevante de la imagen. Este punto se conoce como *fuga*. Si se desea destacar un elemento concreto de una imagen sobre el resto, deberá colocarse en un punto cercano al foco. Por el contrario, los aspectos poco relevantes, deberán ubicarse en las cercanías del punto de fuga.

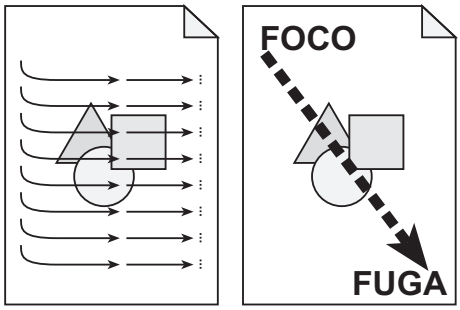


Figura 68. El mecanismo inconsciente de análisis de una imagen mediante el rastreo individual de las líneas que la componen (izquierda) se puede abstraer mediante una línea general de rastreo (derecha) que irá desde el foco hasta la fuga.

Este concepto tan simple es de vital importancia a la hora de obtener buenas imágenes o de diseñar los mecanismos de una interfaz de comunicación. Un ejemplo elocuente de la relevancia que tiene la correcta utilización de éste concepto es el diseño del sistema de circulación de vehículos en carretera. Mientras que por las carreteras británicas los vehículos circulan por su izquierda, el resto del mundo adoptó un convenio mucho más natural, circulando por la derecha. En el caso británico, el conductor se sienta a la derecha del vehículo, por lo que el foco de su visión se sitúa aproximadamente en la mitad de su coche, desperdiciando de este modo la mejor parte de su campo visual en una zona irrelevante para la conducción mientras que con su fuga controla el borde derecho del vehículo, que en su caso es la parte más importante que debería vigilar, ya que por esa zona es por donde vienen los vehículos en sentido contrario. En el caso del resto del mundo, la interfaz es más segura, ya que como el conductor se sienta a la izquierda del vehículo controla con su foco la parte izquierda de su coche (la zona más relevante para la conducción) y deja su fuga para el centro del vehículo (la menos relevante).

Con este ejemplo no se pretende demostrar que el conducir por la izquierda sea especialmente contraproducente, puesto que este modo de conducción ya ha sido asimilado culturalmente por los británicos, sino que la adopción de ese sistema de circulación no es coherente con su entorno cultural. Se trata pues de un fallo en el diseño de la interfaz de conducción, debido principalmente a que en su momento no se tuvieron en cuenta los aspectos culturales de los usuarios de dicha interfaz.

Otro aspecto a tomar en cuenta –como consecuencia directa de nuestro peculiar mecanismo de visualización– es la percepción del movimiento. Debido a que leemos de izquierda a derecha y de arriba hacia abajo, se percibe mayor sensación de velocidad en el movimiento en estos sentidos que en sus opuestos. Este aspecto ha sido muy explotado en la creación de vídeos publicitarios para la promoción de vehículos. Cuando se quiere vender la potencia y velocidad de un coche, éste siempre aparece en el anuncio desplazándose de izquierda a derecha, dando la sensación de que es más rápido de lo que realmente es. Si por el contrario, se quiere vender la seguridad del vehículo, éste aparecerá viajando de derecha a izquierda, dando por tanto una sensación de tranquilidad y estabilidad. También es todo un clásico la promoción de vehículos de tracción en las cuatro ruedas representados subiendo montañas y salvando obstáculos mientras se desplazan de derecha a izquierda y de abajo hacia arriba para dar la impresión de que ningún obstáculo los puede detener, ni siquiera nuestra propia concepción del movimiento.

Se pueden ver otros buenos ejemplos de nuestro peculiar sentido del movimiento en las agujas del reloj o en los indicadores de velocidad de un vehículo. Siempre se mueven de izquierda a derecha, potenciando la sensación del paso del tiempo en el primer caso y de la velocidad en el segundo. El cine y la televisión también han sabido valorar este concepto. En los *westerns*, cuando la situación es desesperada y el asalto final de los indios es inminente, siempre aparece el *Séptimo de Caballería*

avanzando por la izquierda de la pantalla para –desplazándose a toda velocidad hacia la derecha– poner en fuga a los agresores. El cine bélico tampoco es una excepción. Al inicio de *Sin Novedad en el Frente* (1968), las tropas alemanas realizan un desesperado asalto sobre las posiciones francesas, el cual tiene que ser finalmente abortado debido al cuantioso número de bajas. Naturalmente, el asalto se realiza de derecha a izquierda, añadiendo otro impedimento a los atacantes. Además de las alambradas, el barro y el fuego cruzado del enemigo, los alemanes tienen que enfrentarse al obstáculo de realizar un movimiento antinatural.

Un buen ejemplo de la aplicación de esta técnica en el diseño de interfaces de comunicación es su empleo para el diseño de barras de desplazamiento en las que se indica la cantidad relativa de trabajo realizado en operaciones costosas en tiempo de ejecución. El desplazamiento de estas barras siempre se hace de izquierda a derecha para dar la impresión de realizarse más rápido.

Por todo ello, siempre que sea posible DEVA distribuye los objetos visuales siguiendo una orientación vertical, ubicando aquellos objetos dotados de la más alta prioridad en la parte superior del canal de comunicación (*arriba es más, abajo es menos*). En el caso en el que esto no sea posible, los objetos se alinearán de izquierda a derecha en función del sistema de escritura del lenguaje empleado por el usuario. Aunque en la versión actual de GADEA sólo se trabaja con lenguas basadas en el latín (lenguas escritas de izquierda a derecha) se prevé que en versiones futuras de este sistema que den soporte a lenguas semíticas como el árabe y el hebreo (escritas de derecha a izquierda), GADEA alineará los objetos visuales de derecha a izquierda, siguiendo las normas de diseño de acceso universal definidas al respecto [Apple (1992) p. 19-24].

Sin embargo, tal y como se ha comentado en el apartado 17.2.1 (*Un Modelo Borroso*), DEVA hace una clara distinción entre los objetos pasivos (aquellos objetos cuyo único cometido es proporcionar información al usuario) y los objetos activos (los cuales pueden ser empleados por el usuario para proporcionar información a la aplicación por medio de técnicas de desplazamiento y señalización con el puntero). A la hora de asignar espacios en un canal de comunicación visual, los objetos pasivos son alineados de acuerdo con el sistema de escritura empleado por el usuario (de izquierda a derecha en lenguas latinas) mientras que los objetos activos serán alineados de acuerdo con la lateralidad del usuario, alineando dichos objetos a la izquierda siempre que el usuario sea zurdo y a la derecha en caso contrario.

En experimentos realizados durante el desarrollo de esta investigación y comentados previamente en el capítulo 3 **(***La Individualidad***)**, se ha detectado que cuando un elemento de interfaz activo es duplicado y situado en los extremos izquierdo y derecho de un espacio visual, los usuarios tienen a emplear aquellos *widgets* localizados en su espacio de lateralidad ignorando los *widgets* situados en la ubicación opuesta [González y Vidau (2000e)]. Esta estrategia de alineación no solo favorece a la comunicación entre los dos entes involucrados (aplicación y usuario), sino que disminuye sensiblemente

los tiempos de desplazamiento del puntero (T_p) con lo que el modelo de predicción definido y descrito en el capitulo 16 (*Un Modelo de Evaluación*) permite obtener resultados con un mayor grado de precisión, redundando todo ello en un mayor nivel de adaptación a las necesidades cognitivas, perceptivas y motrices del usuario.

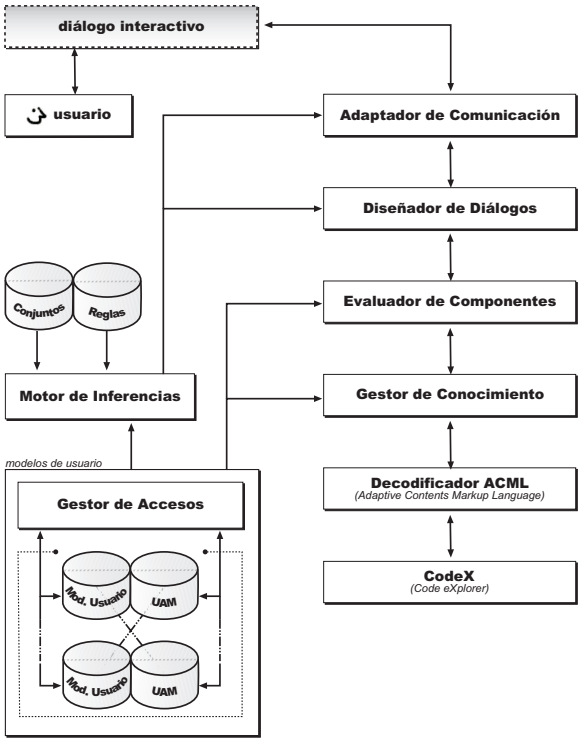


Figura 69. En DEVA, el Adaptador de Comunicación se encarga de la asignación de ancho de banda a los distintos *chunks* transmitidos, así como de la asignación de espacio en el canal de comunicación en función de la prioridad de cada *chunk*.

18

18 Adaptación Cromática

El arte de la pintura consiste en aclarar y obscurecer los tonos sin decorarlos

Pierre Bonnard

18.1 Teoría del Color

Aparte de la distribución espacial de los objetos en un canal de comunicación, otro de los parámetros importantes que DEVA es capaz de adaptar en un canal de comunicación visual es el de la composición de los colores empleados en el mismo.

Debido a su carga semántica, los colores juegan un papel fundamental en la configuración de un canal de comunicación visual, ya que, como se ha visto en el apartado 14.1.2 (*Procesamiento Avanzado*), los colores transmiten información de forma directa por medio de señales físicas. Además de su importante papel semántico, los colores se emplean como importantes elementos de diferenciación entre *chunks*, tal y como se vio también en el apartado 14.4 (*Contraste Visual*).

A lo largo de la historia muchos han sido los investigadores que han intentado explicar la naturaleza la luz (y con ella la del color) con el objetivo de diseñar una teoría global que explicase todos los fenómenos

relacionados con el espectro luminoso. Platón, Aristóteles, Descartes, Newton, etc. fueron solo algunos de ellos. Sin embargo, todos estos investigadores basaron sus observaciones en el aspecto físico de la luz –como ente incorpóreo primero y como onda después– y su capacidad para ser absorbida o reflejada por los objetos, recreando así los colores.

No fue hasta 1921 cuando el químico y físico alemán Wilhelm Ostwald diseña una teoría del fenómeno cromático desde un punto de vista práctico, utilizable por tanto en el desarrollo de las técnicas artísticas y en el estudio de los modelos cognitivos. Ni que decir tiene que su teoría ha tenido enorme éxito y, aunque ha sido revisada posteriormente, sigue siendo válida y ampliamente utilizada en multitud de disciplinas; las cuales van desde la producción industrial de tintes cromáticos hasta el diseño gráfico en general.

En su *Farbenfibel* [Eco Productions (2001)] –conocida en castellano como c*arta de Colores*– Ostwald define el color como una sensación del ojo humano, producida por los estímulos luminosos en la retina o por las reacciones del sistema nervioso. Para Oastwald existe una distinción clara entre una gama *de colores neutros* (el negro, el blanco y la gama de grises intermedios) y otra de a la que denominó *colores activos*, los cuales pueden ser mezclados entre sí para producir colores intermedios.

De todo el espectro de colores, solo existen tres colores puros que no resultan de la mezcla de los otros, siendo además la base para todas las mezclas cromáticas. A estos tres colores se les conoce *como colores primarios* y están formados por el amarillo, el cian y el magenta (ver Figura 70). La suma de estos tres colores en su máxima saturación produce el color negro, lo cual es muy útil a la hora de obscurecer los colores y las mezclas resultantes.

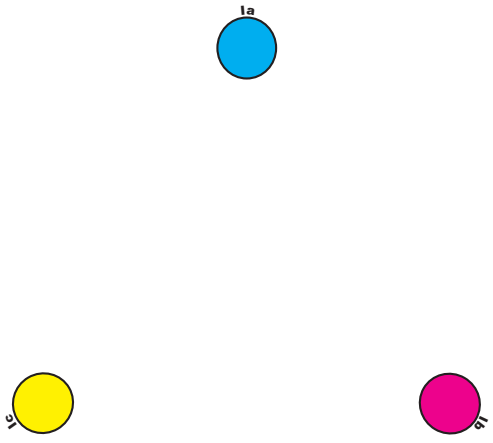


Figura 70. Colores primarios empleados en el modelo de color de Ostwald. Estos colores son el cian (Ia), el magenta (Ib) y el amarillo (Ic). (consultar versión en color en el capítulo 24: *Imágenes en Color*, p. 445).

Si se mezclan los colores primarios entre sí por parejas, se obtiene otro trío de colores, los cuales son conocidos como *colores secundarios* o *colores complementarios* pues estos colores complementan a los primarios. Así, la mezcla del amarillo y del cian dará origen al verde, la

del amarillo y el magenta al rojo naranja y por último la del cian y del magenta originará un azul violáceo (ver Figura 71).

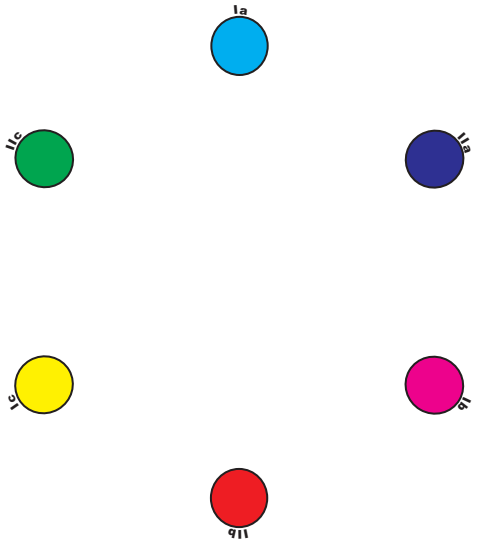


Figura 71. De la mezcla de los tres colores primarios se obtienen tres colores secundarios, a saber, el azul violáceo (IIa), el rojo naranja (Iib) y el verde (Iic). (consultar versión en color en el capítulo 24: *Imágenes en Color*, p. 445).

Estos seis colores integran el espectro cromático de la luz, el cual es conocido comúnmente con el nombre de arco iris. Hay que destacar que el orden que siguen los colores en dicho espectro es el aquí descrito para obtener las mezclas. Así en el arco iris, el orden de los colores será el siguiente: azul violáceo, cian, verde, amarillo, rojo naranja y magenta (ver Figura 72). Si estos colores se distribuyen en un círculo perfecto en el que el color azul violáceo esté adyacente al cian, se crea lo que Oastwald denominó *círculo cromático.*

Figura 72. Representación del espectro cromático de la luz o arco iris formado por los tres colores primarios y los tres colores secundarios (consultar versión en color en el capítulo 24: *Imágenes en Color*, p. 445).

De la mezcla entre los colores primarios con los secundarios adyacentes en el círculo cromático se obtiene una nueva gama de colores conocida como colores terciarios. Así por ejemplo, de la suma del amarillo primario con el rojo naranja secundario se obtiene el naranja, de la mezcla del cian primario con el verde secundario se obtiene el verde esmeralda y así sucesivamente. En definitiva, como producto de esta segunda fase mezcla se obtienen los siguientes colores terciarios: rojo carmín, naranja, verde claro, verde esmeralda, azul ultramar y violeta (ver Figura 73).

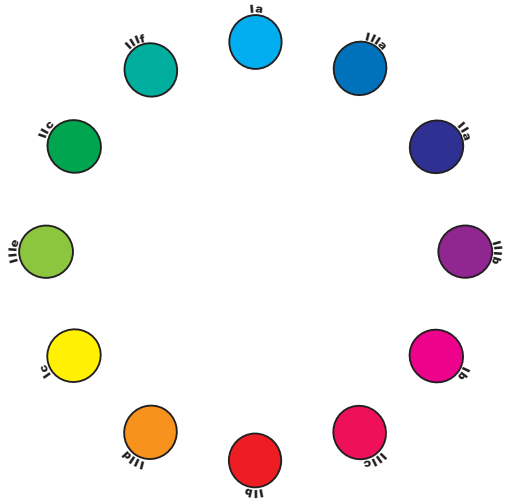


Figura 73. Circulo cromático básico con tres colores primarios, tres colores secundarios y seis colores terciarios (consultar versión en color en el capítulo 24: *Imágenes en Color*, p. 445).

De un modo análogo, los colores cuaternarios se obtienen mezclando cada color terciario con el color primario y secundario adyacente en el círculo cromático. Con este proceso se obtienen doce nuevos colores, completando con ello el círculo cromático original de Ostwald, el cual está formado por los veinticuatro colores descritos, de los cuales tres son primarios, tres son secundarios, seis son terciarios y doce son cuaternarios (ver Figura 74).

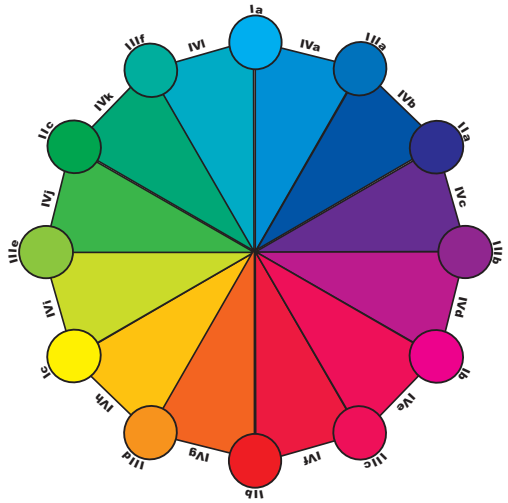


Figura 74. Círculo cromático de Ostwald formado por un total de veinticuatro colores (consultar versión en color en el capítulo 24: *Imágenes en Color*, p. 445).

Como es lógico, se puede seguir extendiendo este mecanismo hasta el infinito siguiendo procesos de mezcla análogos. Sin embargo, los veinticuatro colores del círculo cromático original se conocen también como *colores activos* o *colores puros* pues son tonalidades limpias de impurezas al haber sido obtenidos mediante combinaciones coherentes entre tonos situados entre dos parejas de colores primarios [Gash (1991) p. 42. *Teoría del Color y sus Mezclas*].

Siguiendo este modelo, el círculo cromático podrá dividirse en tres áreas, las cuales estarán delimitadas por los colores primarios. Cualquier mezcla entre colores de una misma área, es decir, entre colores comprendidos entre dos primarios, dará como resultado un color activo. Por ejemplo, si se mezcla cualquier pareja de colores situada entre el cian y el magenta, se obtendrá un nuevo color activo libre de impurezas. Sin embargo, si se saltan estas fronteras y se mezclan colores pertenecientes a sectores distintos del círculo cromático (con la excepción de las mezclas entre los propios colores primarios) se obtendrán tonos sucios y grisáceos conocidos como *colores neutros*. Por ejemplo, si se mezcla el naranja perteneciente al sector magenta-amarillo con el color verde esmeralda perteneciente al sector amarillo-cian se obtendrá un color neutro.

La importancia de los colores activos para GADEA reside en su luminosidad, gracias a su pureza. Los colores luminosos aumentan el contraste con los colores que les rodean y por lo tanto se distinguen mejor. Hay que recordar que el contraste es uno de los elementos fundamentales que intervienen en el proceso de captación de la atención. De esto se deduce que los colores activos son los indicados para emplearse en la confección visual de los objetos señuelo, ya que actúan como ganchos importantes para la captación de la atención (ver 15.1.1: *Señuelos y Predadores*). Evidentemente, los colores neutros (negro, grises, marrones, etc.) no entran en la categoría de colores llamativos y por lo tanto su probabilidad de llamar la atención es baja. Es por ello que rara vez se les emplea con este sentido en los anuncios publicitarios. Sin embargo, debido a factores fisiológicos, los colores neutros cansan menos la vista y por lo tanto permiten mantener una atención prolongada. Por esta razón son ideales para emplearlos en la construcción de objetos predador, pues una vez que la atención ha sido capturada por los colores puros de los objetos señuelo, ésta se puede mantener por mas tiempo mediante el uso de colores neutros.

18.1.1. Colores Fríos y Colores Cálidos

Tradicionalmente se ha establecido un concepto de frío y cálido para los colores que tiene mucho que ver con la presencia de estos colores en la naturaleza. Así por ejemplo, los colores verdes y azules suelen representan tonalidades frías y los amarillos y rojos tonalidades calientes. Es fácil por ejemplo relacionar el verde con las aguas, hierbas y árboles y su sensación de frescor, mientras que el azul se relaciona con el agua e incluso con la nieve y el hielo. Por su parte, el paralelismo existente entre los amarillos y los rojos con el fuego y el sol es inmediato. Esta claro que esta asociación cognitiva entre colores y temperaturas, al igual que las metáforas de orientación vertical vistas en el capítulo 17 (*La Diversidad Humana*) se debe a la percepción continua de las fuerzas de la naturaleza. Esta asociación existe a un nivel tan profundo que llega a afectar incluso al sistema fisiológico humano, resultando interesante comprobar por ejemplo, como se tiende a elevar el nivel de la calefacción en una habitación pintada con colores fríos, mucho más que en otra pintada con colores cálidos, intentando así compensar físicamente la sensación visual térmica percibida.

Entre los colores primarios hay dos colores cálidos (el magenta y el amarillo) y uno frío (el cian). El amarillo es el color del sol y por lo tanto el más luminoso. El magenta es el color del fuego y el cian representa el color de la noche y por lo tanto de la lejanía y de la obscuridad. Si en el círculo cromático se traza una línea que deje a un lado el magenta y el amarillo y al otro el azul, todos los colores del lado de este último serán fríos y los que se sitúen del lado del amarillo y del magenta serán cálidos (ver Figura 75). Los colores en la frontera de esta línea (verdes y violetas) disfrutarán de simultáneamente las dos características, pudiendo ser a la vez fríos o cálidos según contengan más o menos cantidad de cian. Evidentemente, mientras mayor sea la proporción de cian en la mezcla, mayor será la sensación de frío que transmita el color resultante. Por su parte, los colores neutros (grises) conservan también su sentido neutro en la transmisión de sensaciones térmicas, aunque están más cerca de los colores fríos, sobre todo en determinadas mezclas de colores tales como los violetas y los marrones. De todos modos, estos colores pueden cambiar rápidamente hacia cálidos con solo variar la proporción de las mezclas.

La aplicación práctica del uso de colores fríos y cálidos ha sido inmediata en el mundo de las artes plásticas, aplicación que simula e imita el agente interactivo DEVA. Así, cuando se plantea la composición sintáctica de un diálogo interactivo, se pueden resaltar las formas del fondo de varias maneras, pero como ya hemos visto al tratar la atención y la percepción, una de las más importantes es el contraste. En el caso que nos ocupa, se pueden emplear colores de tono neutro para los fondos y claro para las formas o viceversa. Además se puede utilizar un tono frío o neutro para los fondos y tonos cálidos para las figuras, llegando al máximo contraste al emplear un fondo oscuro y de tonalidad fría junto a un tono claro pero cálido para las formas.

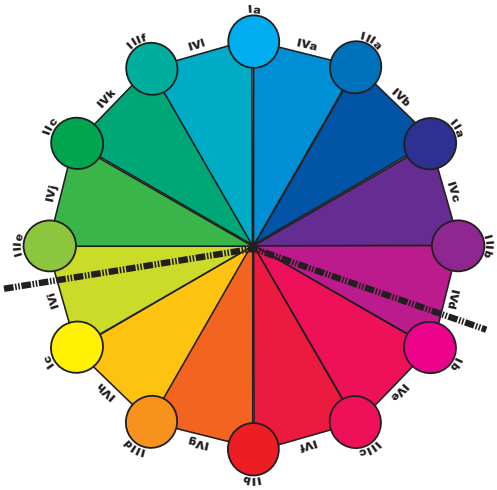


Figura 75. Regiones cálidas y frías del círculo cromático. Todos los colores por encima de la línea divisoria (tonos verdes y azules) se consideran fríos. Los que se encuentran por debajo (tonos amarillos y naranjas) se perciben como colores cálidos (consultar versión en color en el capítulo 24: *Imágenes en Color*, p. 445).

Se ha de tener en cuenta que los fondos o ambientes suelen ser secundarios en las escenas y por lo tanto se deben emplear en ellos

preferentemente los tonos grises y pálidos, independientemente de que sean fríos o cálidos. En cualquier caso, el empleo de los colores dependerá obviamente de lo que se desea expresar con la escena en cada momento.

La temperatura perceptiva del color es empleada por DEVA para mucho más que transmitir sensaciones relacionadas con la temperatura. Por ejemplo, esta característica influye en la percepción de la distancia, ya que debido a la pérdida de detalle visual en la percepción a medida que un objeto se aleja del receptor, la coloración de dicho objeto se tiende al azul, el color frío por excelencia. Dado que los objetos en la lejanía van adquiriendo una tonalidad azulada, como norma general, aquellos objetos que presentan coloraciones azuladas y frías parecen encontrarse a mayor distancia del receptor que aquellos provistos de colores cálidos. De hecho, los objetos representados con colores cálidos parecen avanzar hacia el observador. Esta característica es empleada por DEVA para adaptar el color de fondo empleado en los diálogos interactivos.

18.1.2. Aspectos Cognitivos y Culturales del Color

La sensación cálida o fría transmitida por los colores también afecta a otros factores cognitivos relacionados con experiencias previas del receptor. Así, los colores cálidos se relacionan con las estaciones cálidas como la primavera y el verano o con la actividad diurna, símbolo universal de la vida y en cierto modo de la felicidad; mientras que por el contrario, los colores fríos se relacionan culturalmente con las estaciones frías como el otoño o el invierno y con la noche, símbolo de la muerte, la tristeza y la derrota. Es por ello que los colores cálidos proporcionan sensaciones de cercanía además de cordialidad, alegría, vitalidad y victoria, mientras que los colores fríos dan la impresión de tristeza, melancolía o derrota.

Estos son factores con los que se juega activamente en el mundo de la publicidad para dar personalidad propia a un producto, que se venderá como serio y efectivo si en su presentación se emplean colores de tonalidades frías o como alegre y divertido si por el contrario se usan tonos cálidos. Un buen ejemplo de esta manipulación cromática a la que todos estamos expuestos es la publicidad de vehículos. Por lo general, cuando se intenta vender un vehículo de gama alta, pensado para personas de alto nivel adquisitivo y por lo tanto supuestamente adultos, éste se promociona empleando colores fríos, no solo en el aspecto de la publicidad, sino también en el propio color del vehículo ofertado, el cual aparece siempre con tonos azulados, verdosos o incluso neutros, llegando al paroxismo al emplear tonos grises y negros. De este modo se consigue transmitir la imagen de seriedad, eficacia y alta tecnología que requiere el consumidor de ese tipo de vehículos. Sin embargo, cuando se pretende vender un vehículo joven, los colores tienden a ser extremadamente cálidos con el objeto de transmitir una fuerte sensación de alegría con la que se pueda identificar el consumidor joven, alegre y despreocupado para el que ese vehículo fue concebido.

A pesar de que la percepción del color es una experiencia básicamente sensorial, existen otros factores no sensoriales que afectan a la percepción de un objeto y que tienen una fuerte relación con experiencias previas del entorno cultural en donde vive el componente humano receptor de la sensación cromática. En función del patrón cultural, la percepción de los colores ya que puesto que el color recordado de un objeto difiere a menudo de su color real. Así por ejemplo, en la cultura occidental se tiende a buscar el máximo contraste en la percepción, mientras que en las culturas orientales, se tiende a disminuir la percepción del componente amarillo en los colores, es decir, a la gente de raza amarilla no les gusta verse amarillos. Este hecho es de sobra conocido por los fabricantes de película fotográfica y por los fotógrafos profesionales en general. De hecho, los primeros han practicado una profunda segmentación en el mercado de las películas, fabricando distintos tipos de material en función de la ubicación geográfica de sus consumidores, mientras que los segundos adquieren un determinado tipo de material fotográfico en función de sus clientes. En este sentido, los fabricantes de película fabrican un modelo especial destinado al mercado occidental en el que la emulsión fotosensible está diseñada para capturar gran cantidad de matices en las zonas claras y oscuras y pocos en las medias luces, logrando así reproducir unos tonos más ricos y contrastados de los que aparecen realmente en la naturaleza. Del mismo modo, el modelo de película que se vende en el mercado asiático dispone de una emulsión insensible a ciertas gamas de amarillo, de modo que los objetos y sobre todo las personas así retratadas tienden a mostrar colores basados en el cian y en el magenta solamente.

Si se experimenta con este factor y se le da a un observador occidental una muestra de color para que luego la identifique dentro de un carta de colores, éste se equivocará sistemáticamente al escoger colores más brillantes –cuando el que se le mostró era un color brillante– o más obscuros –cuando el que se le mostró era un color obscuro. Si se le pide que identifique en la carta de colores, el tono de objetos familiares de colores característicos como tomates, manzanas o naranjas, el observador tiende a recordar colores más vivos y en general más intensos. Así, se recuerdan los tomates más rojos, las manzanas más verdes y las naranjas obviamente más naranjas [Álvarez (1995)].

Otro efecto interesante relacionado con la percepción cromática es el de la variación en la percepción del tamaño de una forma al emplear distintos colores para el fondo sobre el que ésta se sitúa así como los usados para pintar la propia forma. En general siempre que se emplea un fondo de color, las formas se perciben más pequeñas en cuanto el fondo sea más obscuro y más grandes en la medida en la que se emplea un fondo más luminoso. Esta variación en la percepción del tamaño se debe al contraste que existe entre el color de la forma y el del fondo. A medida que aumenta el contraste, el color obscuro del fondo se sobrepone contra las débiles fronteras que delimitan la forma, las cuales son de color claro. Esto hace que dichas fronteras se acoplen visualmente con el fondo y no sean percibidas, por lo que la forma parece más pequeña de lo que realmente es al perder sus fronteras. En el caso contrario, cuando la forma es de color obscuro y el fondo es de

color claro, las fronteras de la forma, se expanden a costa de las fronteras del fondo, haciendo que ésta se perciba con un tamaño mayor del que realmente tiene. Este efecto se lleva al extremo cuando se emplean formas de color azul sobre fondo blanco, ya que las formas pintadas con el primer color tienden a retroceder ante los colores adyacentes (percibirse como más pequeñas) y las formas coloreadas de blanco tienden a crecer (percibirse como mayores).

DEVA toma muy en cuenta este efecto a la hora de adaptar el tamaño de ciertos componentes una vez que se ha decidido el color del fondo y de la forma. Así, DEVA adapta el contraste visual entre fondo y forma determinando el tamaño percibido de una composición, el cual no tiene por qué coincidir con el tamaño físico de la misma. Si este tamaño percibido no se ajusta al tamaño determinado para un objeto en función de la configuración obtenida por el por el motor de inferencias de DEVA, el agente aumentará el tamaño del objeto hasta que se logre un ajuste perfecto entre el tamaño esperado y el tamaño percibido.

Desde un punto de vista fisiológico, el color hace algo más que proporcionar información adicional acerca de los estímulos, ya que llega a afectar emocional y físicamente al receptor, provocando sensaciones de tranquilidad, ansiedad, excitación, etc. Es notorio el estudio de los efectos producidos por una estimulación sensorial excesiva del color rojo en los fotógrafos profesionales que trabajan en laboratorios de revelado de películas en blanco y negro. Dado que la emulsión fotosensible de este tipo de películas es insensible al color rojo, los fotógrafos se ayudan de una lámpara de este color como única fuente de iluminación mientras dura el proceso de ampliado de negativos. El fotógrafo suele trabajar entonces durante horas en un entorno dominado por una componente roja que absorbe el resto de los colores. Se ha demostrado que este ambiente provoca excitación, un aumento en las pulsaciones cardíacas y tras períodos de exposición largos, se llega a incluso a provocar náuseas, mareos y dolor de cabeza [Ripota (1991)].

Tabla 36. Valores simbólicos de algunos colores en el marco de la cultura occidental [Fuentes: Álvarez (1994) y González 1995)].

COLOR	VALOR SIMBÓLICO
Amarillo	Comunica calor, luz y cierto sosiego. En sus tonos cálidos, con la intervención del rojo en su composición, es un color que transmite alegría, sin embargo, cuando el rojo no interviene (amarillo limón), el amarillo transmite sensaciones negativas de enfermedad o decaimiento. En el entorno supersticioso del mundo teatral es un color de mala suerte asociado con el diablo desde la Edad Media por lo que no suele ser utilizado.
Azul	Simboliza la fidelidad, el frío y en sus tonalidades claras, la fe. En la naturaleza representa el mar y el cielo. Con su serenidad inspira calma, tristeza o sofocación. Es un color muy conservador, incluso en la política. Desde los años setenta, el cine erótico también se relaciona con este color.
Blanco	Simbólicamente se asocia con ideas de pureza, inocencia, virginidad y limpieza. Es el color de las nubes, el cielo (junto al azul) y los ángeles. También está asociado con la timidez. En algunas culturas orientales es símbolo de luto y por lo tanto se asocia también con la muerte.
Naranja	Representa el calor del fuego y por su vistosidad se le asocia con señales de precaución. Es un color positivo y de alegría que en la naturaleza se asocia con las frutas cítricas y con el crepúsculo. Aunque comunica placidez, a medida que aumenta el tono rojo en su

	composición, crece en violencia.
Negro	Transmite elegancia y fuerza, pero representa a la vez connotaciones mágicas relacionadas con la brujería y las fuerzas demoníacas. En algunas culturas este color se relaciona con la muerte y por ello es empleado como señal de luto.
Rojo	Inspira celos, miedo, amor apasionado y pecado. Este color se asocia también con el fuego, el peligro, la sangre y en general simboliza la violencia. Es un color que indica movimiento y dinamismo hasta el paroxismo representado por la revolución a la que también representa. Sin embargo, su empleo como señal de emergencia, advertencia e incluso prohibición (caso de los semáforos) también indica el concepto de detención súbita. En algunas culturas orientales este color es símbolo de pureza.
Verde	En tonalidades cálidas, este color transmite sensación de descanso e incluso de cierta euforia. En tonalidades frías (con mucha intervención del azul) puede producir cierta sensación de angustia. Se le emplea como símbolo de la esperanza. Actualmente se le ha vinculado al movimiento ecologista y por lo tanto se le relaciona con cosas buenas y saludables. También se le relaciona con desplazamiento (semáforos), movimiento y potestad para hacer algo (*tienes luz verde*).
Violeta	Produce sensación de melancolía y en sus tonalidades obscuras puede inspirar incluso miedo. En su tonalidad púrpura representa riqueza, salud, opulencia, honor y a la realeza.

Pero no es la fotografía el único entorno en donde el uso masivo de un color puede alterar fisiológicamente al individuo. En los hospitales por ejemplo, se ha ido substituyendo el color blanco típico en los uniformes de los cirujanos de principios de siglo por el color verde actual, más tranquilizador para los pacientes, ya que el blanco los excitaba, amén del desagradable contraste producido entre la sangre roja sobre el blanco.

Además de las particularidades físicas, los colores poseen diversas características simbólicas que cada cultura, al paso de los milenios, ha hecho suya. En la mayoría de los casos, esta simbología cromática ha permitido establecer un lenguaje de los colores particular en cada entorno cultural. Sirva como ejemplo los valores simbólicos recogidos en la Tabla 36.

18.1.3. Empleo del Color

La presencia del color en un sistema de comunicación influye sobre el resto de los componentes visuales y a la vez es influido por ellos. Es por ello que un tono cromático se ve distinto en función del color o colores vecinos (ver Tabla 37). El uso correcto del color ha de tener en cuenta este fenómeno, conocido como *contraste cromático*.

Tabla 37. Selección de las armonías cromáticas entre colores empleadas en el diseño gráfico. [Fuente: Lester y Morrish (1994)].

COLOR	USO RECOMENDADO
Azul	Es el más frío y el más débil de los colores. En el diseño gráfico es el color comodín pues armoniza con cualquier color. Sin embargo tiene una fuerte tendencia a retroceder ante otros colores, es decir, las formas coloreadas de azul se perciben más pequeñas de lo que realmente son.
Blanco	Este color tiende a crecer ante otros colores adyacentes provocando que las formas dibujadas en cualquier color sobre un fondo blanco se perciban como más pequeñas. El blanco no toma reflejos del color adyacente sino de su complementario. Por ejemplo, un blanco al lado del azul aparece ligeramente anaranjado, mientras que al

	lado del rojo se percibe ligeramente verdoso.
Gris	Es un color neutro que armoniza con cualquier tono. Si se emplea como fondo muy claro en lugar de un blanco puro, suaviza el contraste violento producido por los colores activos contra el mencionado fondo blanco.
Marrón	Al ser un color neutro (combinación del naranja y gris o negro) es aburrido y se emplea principalmente para contrarrestar el efecto de los colores activos y dar un descanso a la vista. Es por ello un color especialmente indicado para cubrir áreas extensas del canal de comunicación y para su empleo en los fondos.
Naranja	Se obtiene de la combinación de dos colores primarios cálidos: el rojo y el amarillo. Es por ello que este color es el más cálido de todos. Existen combinaciones de fuerte impacto para la atención con rojos y marrones, pero provocan cansancio en el receptor con cierta facilidad.
Negro	Este color, combinación de los tres primarios (amarillo, cian y magenta) tiene la característica única de aprovechar la luminosidad de los colores sobre los que es colocado en beneficio propio.
Rojo	Es un color cálido y dinámico que expresa movimiento en todo el sentido de la palabra, es decir, se percibe en un movimiento continuado de acercamiento al receptor, por lo que es muy útil para llamar la atención (objetos señuelo).
Verde	Compuesto por un color cálido (amarillo) y por otro frío (azul). Su armonización con el rojo es posible, pero difícil por lo que no es aconsejable.
Violeta	Se obtiene por combinación de rojo y azul. Se armoniza bien con los tonos amarillos y amarillos verdosos que son sus complementarios.

Resulta obvio que para lograr un buen contraste cromático es necesario que dos colores vecinos dispongan de una apreciable diferencia de tono, entendiendo por tono la intensidad del color. El valor mínimo de esa diferencia ha de estar situado en torno a un 30%, siendo un 50% un valor aceptable aunque, naturalmente, el valor del mayor contraste se obtiene con un 100% de diferencia, es decir, utilizando tonos en blanco (0%) y de negro (100%) [Gordon (1993)]. Si no queda más remedio que emplear dos colores cercanos de intensidades similares conviene separarlos mediante un color neutro o en todo caso, neutralizar ambos colores mezclándolos con colores neutros [Álvarez (1994)]. Esto equivale a practicar alternancias de colores activos (luminosos) y de colores neutros.

Si se desea alcanzar el máximo contraste posible, éste se conseguirá empleando colores puros. Aún así no debe forzarse demasiado el contraste con la luminosidad ya que, aunque se consigue el máximo contraste y por lo tanto el máximo nivel de atención, esta técnica cansa pronto el sistema de percepción visual y por lo tanto se pierde rápidamente la atención del receptor.

Para facilitar la comprensión del diálogo interactivo, DEVA emplea siempre que puede el mismo esquema de color. El empleo de un mismo esquema cromático proporciona continuidad y normalización al sistema, permitiendo que sus usuarios se identifiquen fácilmente con él. Al igual que ocurre con otros aspectos de la interfaz, el número máximo de colores empleado por DEVA está limitado por la capacidad del procesador humano del usuario activo. Siempre que DEVA emplea más de un color en un mismo canal de comunicación, el color se balancea con extensas áreas de blanco, un color puro excelente para aislar el resto de los colores gracias a su altísimo nivel contraste. Si en lugar de blanco

es necesario emplear algún otro color para el fondo (como ocurre en el caso de la diferenciación de *chunks*), es preferible utilizar un color pálido y claro, evitando siempre los colores obscuros ya que limitan la legibilidad.

Debido a las características del entorno cultural occidental en donde los textos son leídos de arriba abajo, DEVA emplea como fondo una pátina de color que se va degradando en esa dirección, siguiendo la metáfora de orientación vertical descrita en el apartado 17.2.3 (*Reparto del Ancho de Banda*). De este modo se facilita el movimiento de los ojos a través del fondo en busca de las formas superpuestas. El efecto se ve incrementado pasando de colores cálidos a colores fríos.

Siempre que es necesario colorear áreas muy amplias, como ocurre en el caso de los fondos, DEVA emplea tonalidades pálidas opacas. Esto es debido a que los colores muy brillantes y las áreas de color muy extensas cansan la vista del receptor muy rápidamente. Es necesario por tanto emplear los tonos brillantes con precaución y sólo en aquellos lugares del canal de comunicación visual en los que se necesite destacar un mensaje mediante un énfasis especial (objetos señuelo con la máxima prioridad).

En el caso de la transmisión de mensajes en forma de textos escritos, la precaución debe llevarse a límites extremos ya que los textos extensos tienden a consumir rápidamente la atención del receptor, máxime si estos textos están escritos con colores brillantes. Es necesario evitar los colores muy brillantes en áreas de texto muy largas o complicadas que requieran un esfuerzo extra de atención por parte del receptor. En estos casos se emplean colores de tono muy suave, aumentando el tamaño y grosor de la fuente para reforzar el contraste.

18.2 El Adaptador de Comunicación

Una vez analizada la teoría en la que se fundamenta la adaptación cromática llevada a cabo por DEVA se procederá a describirla en detalle paso a paso. DEVA parte del principio de que la selección de los colores a emplear en una aplicación, tanto en las formas como en los fondos, depende directamente del significado atribuido a dichos colores y comentado a lo largo de este capítulo, por lo que este agente interactivo asume que la elección de dichos colores es responsabilidad de los diseñadores de la aplicación. Nótese que el valor del significado de un determinado color sólo tiene sentido a nivel semántico y no al nivel sintáctico en el que trabaja GADEA. De hecho, el trabajo que realizará DEVA no será la selección de los colores sino su adaptación a las necesidades cognitivas, perceptivas y motrices de un usuario concreto, aunque en los casos en los que la aplicación no especifique ningún color concreto para formas y fondos, será DEVA quien seleccione dichos colore tomando la máxima visibilidad como criterio de la selección.

18.2.1. Reducción Selectiva del Azul

Una vez que los colores para la forma y fondo del diálogo interactivo o canal de comunicación visual han sido definidos, el primer paso realizado por DEVA será aplicar un filtro que posibilite la reducción de la cantidad de azul del modelo RGB con el que se especifican dichos colores. Tal y como se analizó en el apartado 17.1.1 (*Variación de la Percepción*), las personas ancianas sufren pérdidas en la sensibilidad perceptiva de los colores azulados por lo que será necesario evitar parte de este tono en los colores, aumentando el de las otras componentes del color para facilitar la lectura de la información transmitida por éste. Como se vio en el apartado 17.2 (*Adaptación de Canales*), la cantidad de azul a eliminar es determinada por el motor de inferencias de DEVA a partir de una serie de reglas definidas a tal fin.

Figura 76. Proceso de adaptación cromática típica para un usuario octogenario y con *Precisión Visual* baja (Eliminación de azul: 40% y *Contraste Visual* del 80%). En la primera fase, se retira el 40% del azul del color de fondo (azul) y del color de la forma (blanco) seleccionado por la aplicación, obteniendo una variante fría del primero (verde) y una variante cálida del segundo (amarillo). A continuación se aumenta el contraste, consiguiendo un color de fondo aún más frío. En la tercera fase se aumenta el tamaño de la fuente al ser el fondo demasiado obscuro y por último se muestran tres variantes de los colores obtenidos en la tercera fase. (consultar versión en color en el capítulo 24: *Imágenes en Color*, p. 445).

Cuando el valor de la variable difusa *Azul Percibido* es definido y convertido a un valor absoluto, el filtro aplicado por DEVA actuará de forma distinta dependiendo de si el color a filtrar es un fondo o una forma. En el primer caso se intentará que toda la proporción de azul retirada se añada en la misma proporción a la componente verde del color, de tal modo que el tono del mismo siga conservando su temperatura cromática fría, es decir, la componente fría retirada (representada por el azul) se revierte en el color por medio de otra componente (el verde). De modo análogo y para facilitar la transmisión de la información, conviene que la forma mantenga su temperatura cálida para aumentar su contraste sobre el fondo y facilitar así su percepción. Por ello, la cantidad de azul retirada del color de la forma es añadida en igual proporción en la componente roja del color, compensando así sobre un color cálido, la cantidad de color frío retirado.

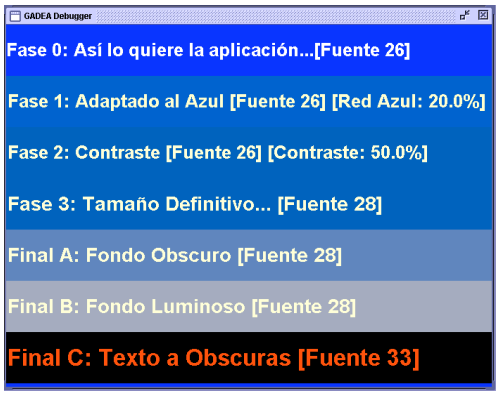


Figura 77. Adaptación visual del ejemplo de la Figura 76 cuando el usuario es más joven y por lo tanto la cantidad de azul a retirar es menor (20%). Nótese como tras el proceso de eliminación de parte del azul en la primera fase de adaptación, los resultados no son tan drásticos como en la Figura 76, ya que el color de fondo sigue siendo azul (en un tono más pálido) y no verde (consultar versión en color en el capítulo 24: *Imágenes en Color*, p. 445).

Tras este proceso, el color de fondo será frío (si no lo era en un principio) y lo contrario ocurrirá con el color de la forma que pasará a ser cálido. Tal y como se analizó en el apartado 18.1.1 (*Colores Fríos y Colores Cálidos*), el empleo de formas de color cálido hace que éstas se perciban mejor sobre fondos de colores fríos. En los ejemplos que ilustran este apartado puede verse como se aplica este efecto en diferentes situaciones. Quizás la más drástica de todas ellas es la representada por la Figura 78 en donde la elevada cantidad de azul retirada (40%) convierte al color azul de fondo de partida en un color verde y al color blanco de la forma en un color amarillo. Este ejemplo ilustra como se sigue manteniendo intacta la transmisión de la temperatura asociada a los colores aún cuando la eliminación del azul es elevada.

18.2.2. El Contraste Visual

El siguiente paso en el proceso de adaptación consiste garantizar el valor del *Contraste Visual* definido en la variable homónima durante el proceso de toma de decisiones del motor de inferencias de DEVA. Para ello se analizan las diferencias de tono entre todos y cada uno de los colores implicados en el modelo RGB, comprobándo que éstas diferencias sean mayores o iguales a las definidas en el *Contraste Visual*. En el caso en el que alguna de estas diferencias sea menor de la permitida, se incrementará la proporción del color predominante y se reducirá la proporción del color de menor tono hasta lograr el contraste esperado.

Para ilustrar este proceso, supongamos para un usuario determinando es necesario alcanzar un *Contraste Visual* igual o superior al 80%. Supongamos también que tras el proceso de filtrado de azul, la componente roja de la forma tiene un valor de intensidad de 151 y la misma componente del fondo tiene un valor de 251. Dado que la máxima variación posible que puede alcanzar una componente

cromática en el modelo RGB es de 255 unidades, un contraste visual del 80% implica una diferencia de al menos (255 * 80) / 100 unidades entre los dos colores, o lo que es lo mismo, una diferencia de 204 unidades. Comparando esta diferencia máxima con respecto a la que ya existe (251 – 151 = 100), se aprecia que el contraste actual entre las dos componentes no es el requerido y que debería incrementarse en al menos otras 204 – 100 = 104 unidades.

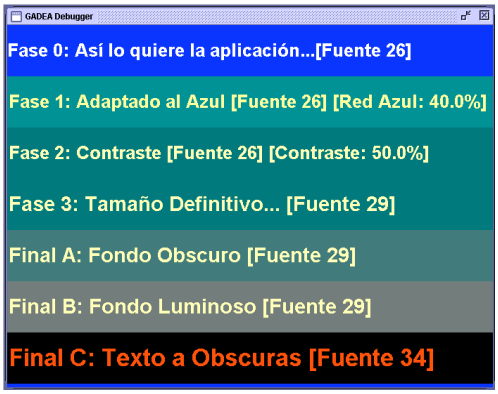


Figura 78. Adaptación cromática de los colores elegidos en la Figura 76 (blanco para la forma y azul para el fondo) para un usuario dotado de una *Precisión Visual* mayor. En este caso el *Contraste Visual* exigido es de tan solo el 50%. Nótese como el color verde claro de fondo obtenido en la primera fase del proceso de adaptación (al retirar un 40% de azul) es convertido en un verde más obscuro para aumentar el contraste con respecto al color de la forma (amarillo). (consultar versión en color en el capítulo 24: *Imágenes en Color*, p. 445).

Para garantizar el *Contraste Visual* necesario, se repartirán equitativamente las 104 unidades de menos entre cada color, tocando 52 unidades al color de la forma y otras tantas al color de fondo. Como el color de la forma tiene menos intensidad en la componente en cuestión que el color del fondo (251 > 151), se le retirarán 52 unidades a este color quedando entonces en 151 - 52 = 99 unidades. Por el contrario, al color del fondo se le añadirán las otras 52 unidades. Sin embargo, durante este proceso de aumento de intensidad, la componente roja del fondo alcanzará el máximo permitido, pasando de 251 unidades a las 255 tope. Es decir, sólo es posible añadir 4 unidades (255 – 251) de las 52 en discordia a este color, quedando por tanto 48 (52 - 4) unidades pendientes por asignar con la premisa de un *Contraste Visual* del 80%. Para garantizar este *Contraste Visual*, DEVA eliminará esas 48 unidades, pasando entonces de 99 a 51 unidades (99 – 48 = 51).

Como se puede apreciar, en este proceso se ha pasado de 251 unidades en el color de fondo a 255 y de 151 unidades en el color de la forma a 51 con lo que se consigue la diferencia de 204 unidades (255 - 51 = 204) que exige un *Contraste Visual* del 80%. Un proceso análogo se realiza para las componentes verde y azul con la salvedad de que el proceso descrito no se aplica íntegramente a éste último color ya que esta componente ha sido previamente procesada por el filtrado correspondiente. Para esta componente concreta, el *Contraste Visual* a alcanzar será ponderada en función del *Azul Percibido*.

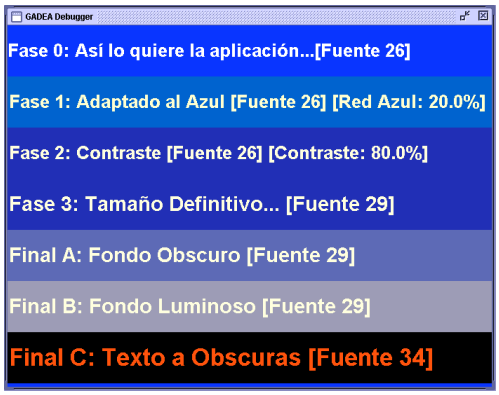


Figura 79. Aumento de contraste para el ejemplo iniciado en la Figura 76, suponiendo ahora que los colores van a ser empleados por un usuario joven (cantidad de azul a retirar igual al 20%) pero con *una Precisión Visual* baja, ya que requiere un *Contraste Visual* del 80%. Aunque el color obtenido tras la primera fase sigue siendo un tono azul, éste es obscurecido durante esta fase de aumento de contraste (consultar versión en color en el capítulo 24: *Imágenes en Color*, p. 445).

En los ejemplos que ilustran este apartado se puede comprobar como el proceso de aumento de contraste tiende a obscurecer el color de fondo y aclarar el color empleado para las formas. En el ejemplo de la Figura 83 hay que destacar como el incremento de contraste necesario es tan elevado que se llega a modificar el tono del color. Sin esta drástica modificación, DEVA no garantizaría una correcta transmisión la información contenida en los colores. En la Figura 80 se puede ver un ejemplo opuesto en donde los colores de partida presentan un nivel de contraste tan elevado que no es necesario modificar sus características, permaneciendo intactas.

18.2.3. Adaptación del Tamaño de la Forma

El tercer paso en el proceso de adaptación cromática consiste en determinar el tamaño del objeto percibido en función del contraste entre existente entre el fondo y la forma. Si este tamaño percibido no se adapta a las exigencias marcadas por el proceso de toma de decisiones del motor de inferencias de DEVA, en el que se define el tamaño de un objeto (ver capítulo 17: *La Diversidad Humana*) será necesario obrar en consecuencia aumentando o reduciendo el tamaño físico del objeto. Nótese que el tamaño percibido no necesariamente coincide con el tamaño físico real del objeto.

Para ello, DEVA define y confecciona un índice para medir el grado de obscuridad de color de fondo, partiendo de la suposición de que el color de la forma se encuentra ya suficientemente contrastado tras aplicar el segundo paso de adaptación. Dado que en el modelo de color RGB, mientras mayor sea la cantidad de color incluida en cada una de las componentes, más se acercará el color resultante al blanco y por lo tanto más brillante será el color resultante, el índice de obscuridad del color de fondo será el inverso de la suma de todas las intensidades de un color. Este índice se ponderada con respecto a la suma máxima de

esas intensidades (255 + 255 + 255 = 765). El valor obtenido será por tanto una medida porcentual del grado de obscuridad de un color.

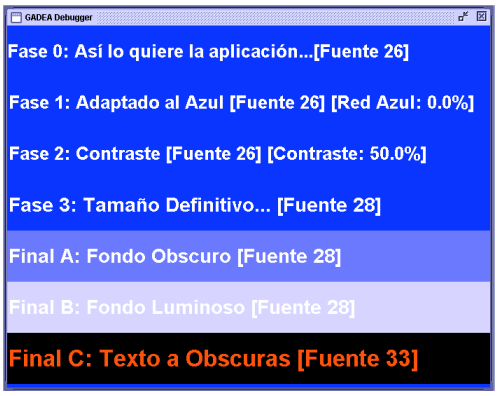


Figura 80. Adaptación cromática del ejemplo de la Figura 76 para un usuario muy joven (en donde no es preciso retirar cantidad alguna de azul) y dotado de una visión relativamente buena ya que solo requiere de un contraste visual del 50%. A pesar de ello, se puede observar como es necesario aumentar el tamaño físico de la fuente tipográfica en la tercera fase, pasando de los 26 puntos originales a 28 puntos. Esto se debe a que el color de fondo (azul) es demasiado obscuro (consultar versión en color en el capítulo 24: *Imágenes en Color*, p. 445).

En función del valor obtenido para el color de fondo, DEVA procederá a aumentar el tamaño del objeto que actúa como forma en pequeños incrementos y de forma inversamente proporcional a la obscuridad del fondo. Así, mientras más obscuro sea el color de fondo, mayor será el incremento del tamaño de la forma. Además, esta tasa de incremento dependerá del grado de agudeza visual del usuario de la aplicación. Mientras peor sea esta agudeza visual, mayor será el incremento y viceversa.

En todos los ejemplos que ilustran este apartado (Figura 76, Figura 77, Figura 78, Figura 79, Figura 80, Figura 83 y Figura 84), salvo en la Figura 85, el proceso de adaptación del tamaño de la forma resulta positivo, aumentando el tamaño de la fuente tipográfica de los ejemplos en incremento del orden de dos a tres puntos. En la Figura 85, el color de fondo es tan claro que no afecta al tamaño percibido de las formas, por lo que no es necesario ningún incremento en la fuente tipográfica, permaneciendo ésta en los 26 puntos originales.

18.2.4. El Brillo

Llegado a este punto en el proceso de adaptación, DEVA habrá obtenido dos colores base (fondo y forma) con componente azul y contraste adaptados a la capacidad del procesador perceptivo del usuario así como un tamaño final para los *widgets* decorados con dichos colores base, el cual se encuentra adaptado también a las necesidades del usuario. Será entonces sobre estos dos colores base sobre los que DEVA pueda realizar las variaciones necesarias para garantizar el contraste requerido para una correcta percepción de los distintos *chunks* contenidos en un diálogo interactivo.

Dado que el mecanismo de distribución de objetos señuelo en el espacio de un canal de comunicación visual está basado en una metáfora de orientación vertical, los *chunks* se distribuyen uno a continuación del otro en una secuencia que va desde la zona superior del espacio visual hasta la inferior. Esta sucesión de *chunks* permite alternar las características físicas de los mismos por parejas, con el objeto de diferenciarlos visualmente unos de otros, empleando para ello tan solo dos variaciones sobre cualquier característica distintiva. Es posible por tanto emplear la variación A para el primer *chunk* de la sucesión, la variación B para el segundo, volver a emplear la variación A para el tercero, la variación B para el cuarto y así sucesivamente a lo largo de todo el eje vertical del espacio visual asignado al diálogo interactivo (ver capítulo 17: *La Diversidad Humana*).

Precisamente, una de las características distintivas empleadas por DEVA para individualizar cada *chunk* es su color de fondo. Por ello, a partir del color de fondo base especificado por la aplicación, se generan dos colores, uno para los *chunks* impares y otro para los *chunks* pares. Estos dos colores de fondo, son empleados por DEVA en cualquier otra situación en la que sea necesario realizar una distinción visual efectiva entre dos objetos. Así por ejemplo, en el diálogo ilustrado en la Figura 59, los *chunks* impares (el primero y el tercero en el ejemplo) emplean un mismo color de fondo (amarillo suave). Lo mismo ocurre con los *chunks* pares (el segundo).

Por otro lado, existen situaciones en las que es necesario destacar una determinada forma sobre el resto para favorecer un determinado modelo de interacción. Así por ejemplo, cuando es necesario llamar la atención del usuario sobre un determinado evento o dirigir dicha atención sobre un objeto concreto, DEVA requiere emplear un tono de color especial para la forma, destacándola sobre el resto, tono que obviamente deberá ser distinto del tono empleado para las figuras adyacentes a la destacada. Es por ello que a partir del color de la forma base, DEVA obtiene también dos colores, uno para su empleo habitual en todas las formas y otro, dotado de unas características especiales de brillo, que permiten usarlo en las formas a resaltar.

En cualquiera de los casos, para obtener los dos colores de fondo y los dos colores de las formas, DEVA manipula los valores de brillo y de saturación de los dos colores iniciales, creando dos variaciones para cada uno de los colores. Resulta obvio mencionar que éstas variaciones se generan de forma coherente con respecto al modelo cognitivo en el que se fundamenta este agente interactivo.

El sistema de percepción del brillo se ha estudiado intentando diseñar herramientas para su simulación de la percepción partiendo únicamente de los aspectos fisiológicos que se llevan a cabo en la retina. Para ello se han desarrollado modelos matemáticos y sistemas basados en la inhibición lateral de neuronas que intentan predecir cambios en el brillo a partir de las relaciones espaciales existentes entre las células de la retina. Sin embargo, la percepción del brillo es demasiado compleja para ser simulada empleando únicamente consideraciones fisiológicas, al entrar en juego aspectos del tipo cognitivo. Se ha demostrado por ejemplo que al igual que ocurre con otras constantes perceptivas, los

valores del brillo percibidos se ajustan en función de las experiencias previas de las que dispone el observador respecto a la naturaleza del mundo que le rodea.

De este modo, el procesador perceptivo realiza suposiciones acerca de la cantidad de luz que alcanza la superficie de una forma, ajustando lo percibido para que el brillo aparente se limite a las experiencias previas, en lugar de ajustarse a la cantidad real de luz que alcanza el ojo. Este efecto se detecta claramente al dibujar una forma plana que representa a un cilindro, pintando toda la forma con el mismo tono de gris. En este efecto, representado en la Figura 81, el tono gris del interior del cilindro se percibe más claro que el del exterior, aun cuando se trate del mismo tono.

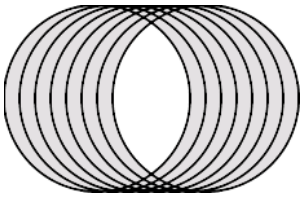


Figura 81. El interior del cilindro (que no es tal) tiene el mismo tono de gris que el exterior, sin embargo el gris del interior se percibe más obscuro.

Coren y Girgus (1978) sugieren que esta diferencia de brillo se debe a un ajuste cognitivo basado en lo que se presume acerca del entorno. En este caso, es de suponer que si la forma cilíndrica fuese un tubo en el mundo real, recibiría más luz en el exterior que en su interior aún cuando la cantidad de luz que alcanza al ojo sea la misma. Así mismo, existen muchos casos en los que las predicciones hechas a partir de la mera inhibición lateral en un modelo fisiológico no se cumplen. Para ilustrar estos casos, sirva el ejemplo recogido en la Figura 82. En esta representación se emplean dos imágenes similares, una formada por una serie de líneas verticales, en donde se alternan los colores blanco y gris y otra similar en donde se emplean los colores negro y gris. Aunque el tono de gris empleado en las dos imágenes es exactamente el mismo, este color parece más claro en la imagen de la izquierda, en donde este color se combina con el blanco, que en la imagen de la derecha, en donde el gris se combina con el negro.

Figura 82. Asimilación del brillo. Aunque el tono de gris es el mismo en las dos figuras, se percibe un gris más obscuro en la imagen de la derecha.

Según el modelo fisiológico de inhibición lateral, cuando una neurona es estimulada, esta retrae la estimulación de las neuronas adyacentes con el objeto de conseguir una mayor definición y contraste

en la sensación percibida. El efecto producido por el experimento anterior realiza justo lo contrario, pues de acuerdo con este modelo, el blanco debería obscurecer al gris (buscando el contraste) y no al contrario. Este efecto inverso se conoce como asimilación del brillo.

Siguiendo un razonamiento basado en el modelo cognitivo, las líneas verticales de color negro de la Figura 82 representan objetos opacos que se oponen a la luz y por lo tanto dan la sensación de dar sombra, obscureciendo de este modo el tono gris. Las líneas de color blanco sugieren objetos transparentes o translúcidos que al dejar pasar la luz, no crean ningún tipo de sombra sobre las líneas grises y por lo tanto éstas no son obscurecidas

Coren et al (1979) advirtieron que la parte del campo visual al que se presta mayor atención muestra mayor contraste en el brillo, extendiendo esta observación para explicar la asimilación del brillo. Estos investigadores demostraron que las zonas a las que se presta atención muestran un fuerte contraste en el brillo respecto de las que las rodean, mientras que aquellas a las que no se atiende presentan asimilación del brillo. Así, las líneas de la Figura 82, al tratarse de formas definidas capturan la atención visual y por tanto sufren contraste del brillo, apareciendo las líneas blancas más blancas de lo que son y las negras más negras. Sin embargo, el fondo gris recibe menos atención al no ser una figura definida y por ello sufre asimilación de brillo, pareciendo ser más blanco en la imagen dotada de líneas blancas y más obscuro en la de las líneas negras. Es por ello que un cambio voluntario en la atención, que concentrase ésta en el fondo gris en lugar de en las líneas, haría que el fondo sufriese contraste en lugar de asimilación y por lo tanto parecería más obscuro en la imagen de las líneas blancas y más claro en la imagen de las líneas negras.

Dado que la atención se centra en aquellas zonas de un espacio visual dotadas del mayor contraste de brillo, DEVA manipulará la cantidad de brillo de los colores de fondo y de forma, generando las dos parejas de colores comentadas (dos colores para el fondo y dos colores para la forma) intentando que exista el mayor contraste posible entre dichas parejas y reduciendo el contraste existente entre dos elementos de la misma pareja. Con ello, este agente interactivo persigue que exista un proceso de asimilación de brillo en los colores de fondo con respecto con respecto a los colores de la forma (destacando éstos últimos sobre los primeros). De este modo se refuerza el papel prominente jugado por las formas, haciendo que éstas se perciban como tales.

Para realizar este proceso de adaptación, es necesario convertir el modelo de partida pasando del modelo RGB a un modelo HSB que permita manipular el tono (H), la saturación (S) y el brillo (B). Una vez realizado este paso, DEVA generará los dos colores de fondo empleados para la distinción de *chunks* reduciendo la saturación (S) del color de fondo base en dos niveles, uno para cada color. De forma similar, DEVA aumentará la proporción de brillo (B) existente en el color base asignado a la forma en dos niveles, obteniendo así un color básico para la forma (el de menor nivel de brillo) y un color especialmente resaltado (el de mayor brillo) empleado en situaciones en las que sea necesario llamar la atención del usuario sobre una forma determinada.

Al reducir el brillo el color de fondo base se consigue reducir considerablemente el poder de captura de la atención de los colores generados, con lo que éstos pasan a jugar un papel secundario en la composición global del canal de comunicación visual. Aún así, dado que los dos colores poseen un nivel de saturación distinto, éstos serán percibidos como dos colores ligeramente diferentes aún cuando parten de un mismo color. Con todo ello se alcanzan tres objetivos. En primer lugar, se obtienen colores con muy bajo nivel de brillo, muy poco llamativos y por lo tanto ideales para su uso como color de fondo. En segundo lugar, las diferencias de saturación entre los dos colores son lo suficientemente elevadas como para ser percibidas y por lo tanto pueden ser empleadas para individualizar la percepción de los *chunks* a los que están asociados, diferenciando así dos *chunks* situados en franjas consecutivas del espacio visual. En tercer lugar, la variación de tono entre los dos colores resultantes es mínima y por lo tanto, a pesar de las manipulaciones realizadas por DEVA, se sigue conservando la carga semántica asociada al tono del color de fondo elegido por los diseñadores de la aplicación durante la fase de análisis del modelo de interacción de la misma.

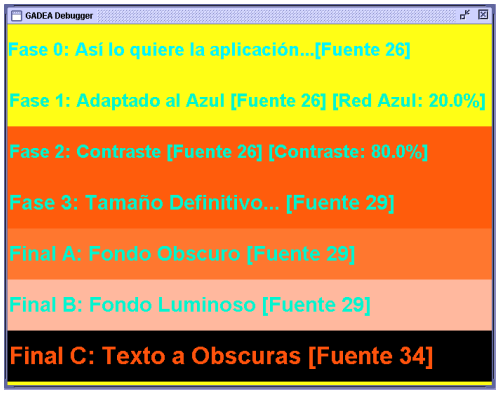


Figura 83. Proceso de adaptación cromática de DEVA ante una situación difícil. El azul claro original de la forma presenta muy poco contraste con respecto al amarillo del fondo. Tras eliminar un 20% de azul y convertir el azul original en un verde claro, el proceso de aumento de contraste cambia el amarillo de fondo por un rojo, el cual es finalmente convertido en un naranja claro y un rosa para su uso como fondo de los *chunks* (final A y final B). (consultar versión en color en el capítulo 24: *Imágenes en Color*, p. 445).

Por otro lado, en el caso de los colores que servirán para configurar las formas, La existencia de valores de brillo de elevada magnitud persigue como es lógico la asimilación del brillo de las formas con respecto al fondo, de tal modo que los objetos configurados con estos colores sean claramente separados del color de fondo. Además, la variación de la intensidad de brillo en ambos colores logra destacar fácilmente aquel color dotado del mayor nivel de brillo.

Las proporciones de reducción de la saturación del color de fondo o de aumento del brillo del color de la forma son parámetros que dependen directamente de la capacidad del procesador perceptivo del usuario y son calculados en base a su *Precisión Visual* (ver capítulo 11: *El Application Explorer*). El proceso de cálculo de estos índices de

adaptación se realiza de forma análoga al del resto de las variables borrosas empleadas por DEVA en otros apartados del proceso de adaptación, es decir, mediante la aplicación de determinadas reglas de lógica difusa sobre los hechos o datos de partida almacenados en el modelo de usuario (ver capítulo 17: *La Diversidad Humana*). En líneas generales, mientras peor sea la *Precisión Visual* del usuario mayor será el valor asignado a estos índices obteniendo así un mayor contraste existente tanto entre el fondo y la forma como entre la pareja de colores empleada para configurar fondos y formas. En estas condiciones en las que la *Precisión Visual* es baja, es necesario aumentar al máximo el contraste entre los objetos, siempre y cuando este contraste se encuentre dentro de los límites permisibles y no afecte a otras áreas del diálogo interactivo.

Figura 84. Situación imposible ya que el programador pretende emplear un mismo color para el fondo y para la forma (el blanco). DEVA transformará el blanco del fondo en amarillo y más tarde en azul (durante el proceso de aumento de contraste). El color de la forma pasará de blanco a un tono de amarillo pálido, garantizando la legibilidad (*final A* y *final B*). Durante el proceso de ajuste también se aumenta el tamaño de la fuente, la cual pasa de 26 a 28 puntos (consultar versión en color en el capítulo 24: *Imágenes en Color*, p. 445).

En los ejemplos que ilustran este apartado se puede apreciar el efecto de manipulación del brillo llevado a cabo por DEVA en las franjas horizontales cinco y seis de cada ejemplo (empezando desde arriba), las cuales están etiquetadas como *Final A y Final B* respectivamente. El color de fondo de estas franjas se corresponde con el color de fondo de los *chunks* a las que representan. Nótese como el brillo del color de fondo de estas franjas es mucho menor que el color empleado en las formas. El contraste entre los colores de fondo de las etiquetas *Final A* y *Final B* es suficientemente elevado como para distinguir unas de otras a simple vista.

Hay que destacar que este mecanismo de adaptación es válido para cualquier pareja que haga las veces de colores de fondo y de forma, no teniendo porque coincidir necesariamente con el color de fondo y de forma de todo el canal de comunicación, pudiendo referirse tan solo al color de fondo y de forma de un *widget* específico. También es importante resaltar, que este mecanismo permite evitar posibles errores provocados por los diseñadores de la aplicación al codificar ésta

empleando parejas de color (fondo y forma) con un contraste difícil o imposible de usar en función de la situación.

En la Figura 83 se puede apreciar como una selección de colores poco contrastados (azul sobre amarillo) es adaptada por DEVA para mejorar el contraste y la calidad de la información transmitida por la combinación de estos colores. Otro tanto sucede en la Figura 84 en donde se intenta aplicar el mismo color para el fondo y para la forma (blanco). Como se puede apreciar, el algoritmo de adaptación funciona perfectamente al detectar el error de forma automática en el proceso de aumento de contraste y al seleccionar una alternativa coherente con el modelo cognitivo respecto a la planificada por los programadores, alternativa en donde, en el caso concreto del ejemplo, se empleará el azul como color de fondo y el amarillo como color de la forma.

18.2.5. Condiciones de Obscuridad

El mecanismo de adaptación cromática planteado por DEVA podría extenderse aún más para adaptarse a las condiciones del ambiente de trabajo en las que el usuario va a realizar su tarea. En este sentido, el proceso de adaptación cromático se adaptaría (valga la redundancia) a las condiciones de luminosidad de la habitación o espacio físico en el que se va a emplear el diálogo interactivo.

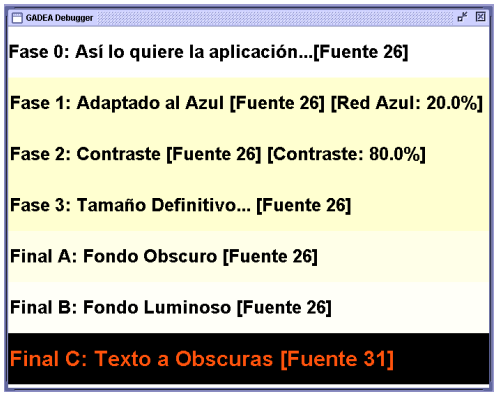


Figura 85. El color blanco para el fondo y el negro para las formas son los colores por defecto de DEVA. En esta ilustración se observa el proceso de adaptación de estos colores para un usuario relativamente joven (eliminación de azul del 20%) y de *Precisión Visual* baja (*Contraste Visual* requerido: 80%). El resultado (*Final A* y *Final B*) ha sido el empleado para configurar los ejemplos de diálogos interactivos incluidos en prácticamente todo este documento (consultar versión en color en el capítulo 24: *Imágenes en Color*, p. 445).

Dado que en el proceso de investigación llevado a cabo para diseñar y codificar el proceso de adaptación descrito no se ha podido contar con sensores para la medición de las condiciones luminosas, en la presente versión de DEVA se supone la existencia de valores normales y de valores extremos de iluminación, los cuales son simulados de forma artificial, creando estados de iluminación que el usuario ha de activar manualmente, indicando de forma explícita el nivel de iluminación con el que desea trabajar.

En este sentido, DEVA contempla tan solo dos tipos de iluminación, una iluminación natural y constante en la que se desenvuelve todo el proceso de adaptación hasta ahora descrito y una ausencia total iluminación, es decir, la obscuridad absoluta, para la cual se necesita la ayuda de los mecanismos de adaptación especiales que se van a comentar a continuación.

La retina del ojo humano dispone de dos terminales fotosensibles distintos: los llamados conos y los bastones. Ambas terminaciones requieren un período apreciable de tiempo para alcanzar su máxima sensibilidad después de haber sido expuestos a luces brillantes. En este sentido, los bastones son especialmente lentos pues pueden llegar a requerir hasta 30 minutos para alcanzar su nivel operativo máximo tras haber sido deslumbrados por luces intensas.

Mientras que los conos funcionan principalmente bajo condiciones de iluminación diurna, los bastones están especializados en condiciones de obscuridad extrema, aproximadamente bajo el nivel de la luz que alcanza una noche de luna llena) [Chapanis (1965). p. 79. *La Presentación Visual de la Información*]. Otra diferencia importante que existe entre estas dos terminaciones nerviosas radica en que los conos son especialmente sensibles a las longitudes de onda larga del espectro cromático (los tonos rojos), mientras que los bastones son especialmente sensibles a las longitudes de onda corta (los tonos azules). Gracias a esta diferencia en la sensibilidad, si se emplea luz roja para permitir que los conos vean en la obscuridad, los bastones podrán descansar., en estas mismas condiciones.

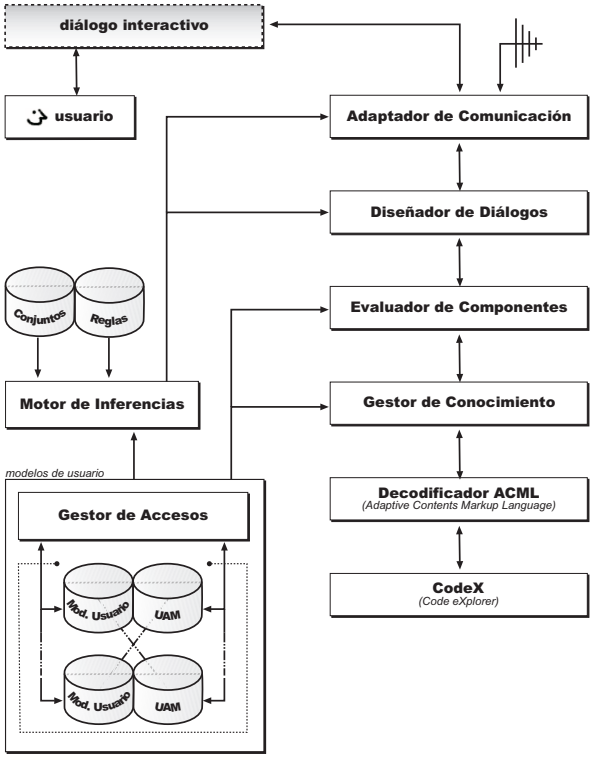
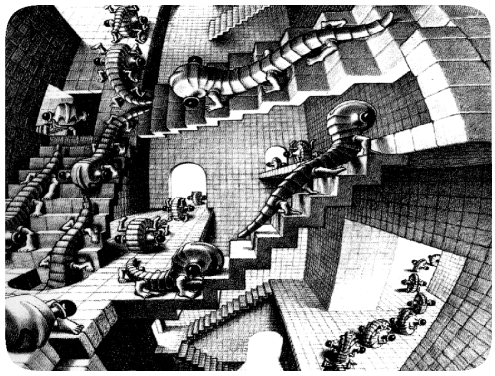


Figura 86. El empleo de sensores externos por parte del Adaptador de Comunicación permitirá ajustar al máximo la información cromática transmitida, tanto en función de las necesidades cognitivas y perceptivas del usuario como de las condiciones de su ambiente de trabajo.

El uso de luz roja en la obscuridad presenta la ventaja de poder emplear iluminación (roja) en estas condiciones de iluminación sin llegar a deslumbrar al receptor. Así, este podrá tener acceso a indicadores y controles iluminados preservando su adaptación a la obscuridad. Este fenómeno es la base del diseño de numerosos artefactos empleados en condiciones de obscuridad extrema. Así, este color es el empleado habitualmente para iluminar los indicadores y tableros de control de máquinas, vehículos, pantallas de radar, etc. Los cuartos de preparación a bordo de portaaviones e instalaciones aéreas militares terrestres suelen estar equipados con iluminación roja, de tal forma que los pilotos de combate puedan estar preparados para llevar a cabo sus misiones de vuelo nocturno con sus ojos adaptados a la obscuridad. Otro buen ejemplo es el de las salas de control aéreo o los laboratorios fotográficos en blanco y negro en el que la iluminación también es roja. Si algún operador necesita abandonar la sala y desea conservar el nivel de adaptación a la obscuridad, suele colocarse un par de lentes de color rojo antes de salir de la sala. De este modo, al regresar a la misma, los bastones de sus ojos presentarán un nivel de adaptación a la obscuridad prácticamente absoluto.

En DEVA, cuando el usuario indica que desea trabajar en condiciones de obscuridad absoluta, el agente interactivo ordena una abstracción total de la información cromática transmitida con el objeto de poder hacer llegar al usuario la mayor cantidad de información posible, dadas las especiales circunstancias en las que la comunicación va a ser establecida. Nótese la gran cantidad de ruido que va a afectar al canal de comunicación visual activo. En estos casos, además de potenciar el uso de otros canales de comunicación (como el auditivo), el agente interactivo determina el color rojo como color de las formas y el color negro como color de fondo.

En este proceso de adaptación especial se inhiben las dos primeras fases (eliminación de tonalidades azules y aumento del contraste) permitiendo tan solo la ejecución de la tercera fase, determinando el tamaño de las formas tomando en consideración el máximo grado de obscuridad alcanzado por el color de fondo (negro). En todos los ejemplos que ilustran este apartado, puede verse como en la última franja *(Final C*) la fuente tipográfica alcanza un tamaño máximo, aumentando ésta unos siete u ocho puntos con respecto a su tamaño original.



Cubo de Escalera © 1951 por M. C. Escher

P A R T E V

ANTS: Automatic Navigability Testing System

19

19 Espías

Mientras mejor conozca un ordenador a su usuario, mejor podrá servirle.

Jon Orwant

19.1 El Módulo ANTS

La materia prima con la que se alimenta el módulo DEVA para la adaptación de los diálogos interactivos y canales de comunicación proviene en gran medida del propio usuario. De hecho, la calidad de los procesos de adaptación efectuada por DEVA depende linealmente de fiabilidad del conocimiento del que dispone este agente interactivo inteligente. Es por ello mientras mayor sea la calidad de la información almacenada en los dos modelos de usuario empleados por el agente (el modelo de usuario individual y el UAM), más efectiva será la adaptación realizada por lo tanto mayor será la calidad de GADEA.

En este punto crítico se sitúa el módulo ANTS, el cual constituye la parte del sistema de gestión de interfaces de usuario encargado de valor por una correcta y continua actualización de la información almacenada tanto en el modelo de usuario individual como en el UAM. Para garantizar este objetivo, ANTS comprende una serie de familia de

agentes especializados en la captura de un determinado tipo información, los cuales se infiltran cual agentes secretos en los distintos diálogos interactivos generados por DEVA, observando al usuario sin que éste se percate de ello, permitiendo así obtener información en *estado puro*, es decir, aquella información generada de forma espontánea por el usuario y en la que el *factor observación* no ha introducido ningún tipo de cambio. Obviamente, existen determinados tipos de información empleados por DEVA que no pueden ser obtenidos implícitamente mediante un proceso de observación encubierta. Pertenecen a esta categoría datos tales como el nombre del usuario, su clave de acceso, su edad o su sexo. Naturalmente en estos casos a los agentes no les queda más remedio que inquirir directamente al usuario en busca de esta información.

La información en bruto obtenida mediante las dos modalidades descritas es procesada cuando procede por los propios agentes, contrastándola con otras informaciones mediante un proceso de observación continuo hasta refinarla en un grado aceptable, depositándola entonces en el modelo de usuario correspondiente, el cual puede estar almacenado localmente en la máquina en la que se ejecuta GADEA o en un servidor central remoto. A partir de este momento, DEVA se encuentra en capacidad de tomar decisiones sobre esta información. Sin embargo, este agente inteligente no será el único ente capaz de beneficiarse de estos datos, ya que parte de la información recopilada por los agentes está destinada a los propios diseñadores de una aplicación compatible con GADEA. Esta información comprende datos acerca de la actividad realizada por un usuario cualquiera de la aplicación (conservando su anonimato, eso sí) sobre los procesos de usuario, canales de comunicación y/o diálogos interactivos de la misma. Esta valiosa información permite obtener una visión global a nivel semántico de la calidad del modelo de interacción planteado por el diseñador de la aplicación para su interfaz de usuario, lo cual permitirá a este llevar a cabo las oportunas modificaciones que permitan incrementar la calidad de dicho modelo en futuras versiones de su aplicación.

19.1.1. La Metáfora de Diseño

El diseño de ANTS está basado en el paradigma cliente – servidor siguiendo una metáfora de diseño basada en la vida de una colonia de hormigas (*ants* en inglés). Estas hormigas se representan mediante agentes formados por un puñado de clases de muy pequeño tamaño, lo cual facilita su distribución y transmisión por cualquier tipo de red [González y Álvarez (2000a)]. Siguiendo con la metáfora del mundo real, estas hormigas salen de su hormiguero (el servidor central) en búsqueda de comida (la información) en todo picnic disponible (cualquier diálogo interactivo definido por DEVA). Una vez que la hormiga ha conseguido su comida, ésta vuelve al hormiguero y almacena cuidadosamente la información recolectada en una de sus galerías o almacenes (la bases de datos del modelo de usuario individual del UAM).

Esta metáfora de diseño está basada en el paradigma de la colonia de hormigas planteado originalmente por Dorigo et al (1996) para resolver problemas complejos representados sobre grafos, en especial en problemas relacionados con el tráfico de agentes móviles por una red. Aunque se trata todavía de una tecnología joven, ésta ha sido empleada con éxito en la resolución de problemas complejos antiguamente resolubles mediante algoritmos no polinómicos, como por ejemplo la selección de la ruta a seguir por parte de una serie de paquetes de datos en una red de telecomunicaciones [Dicaro y Dorigo (1997), Gallego-Schmid (2000)].

En el paradigma de la colonia de hormigas, un grupo de estos agentes se mueve a lo largo de un grafo formado por una colección de nodos y enlaces, el cual que representa el problema a resolver. Cada una de estas hormigas (agentes) representa un ente autónomo capaz de realizar tomas de decisión por su cuenta y riesgo pero siempre teniendo en cuenta el beneficio común de la colonia. Para ello, las acciones desarrolladas por una hormiga en un punto cualquiera del grafo, podrán ser continuadas más tarde por sus compañeras, mediante el establecimiento de un complejo lenguaje basado en señales dejadas en dichos puntos de trabajo (lo mismo ocurrirá con las hormigas reales). Nótese como esta estrategia encaja perfectamente en la tesis planteada en el capítulo 4 (*La Comunicación*), en donde se preconizaba supeditar el beneficio individual de los componentes usuarios de un sistema al objetivo global del sistema.

Al igual que ocurre con las hormigas del mundo real, las hormigas artificiales modifican el entorno en el que se mueven a medida que lo exploran, recolectando hojas (información) o cortando tallos (dejando señales), de tal modo que, tras sucesivas excursiones el hábitat de las hormigas puede ser completamente distinto. A medida que las hormigas artificiales se mueven por el grafo, éstas construyen soluciones parciales, modificando el problema, añadiendo la información recogida y actualizando los nodos. En el caso de estas hormigas artificiales, el objetivo perseguido al modificar el entorno, será la reducción de la complejidad del grafo que representa el problema hasta que ésta alcance un nivel lo suficientemente bajo como para que la solución del problema resulte trivial.

En el caso que nos ocupa, el comportamiento mostrado por un usuario cualquiera cuando explora el modelo de navegación se asemeja bastante al comportamiento de la comunidad de hormigas descrita, puesto que la concepción que dicho usuario tiene del modelo de navegación no es estática, sino que ésta se va modificando a medida que se desarrolla el proceso de exploración [González (2000g)]. A medida que el usuario visita nuevos nodos, el esquema mental que representa al entorno conceptual que está visitando es modificado continuamente mediante la creación progresiva de nuevos *chunks* (ver capítulo 15: *El Procesador Perceptivo*) capaces de representar, clasificar y agrupar la nueva información adquirida. Cuando el usuario visita un nodo cualquiera de un modelo de navegación e incorpora su representación mental en su memoria semántica, en realidad está dejando una pista a su mismo sobre la concepción especial, temporal y

estructural del mismo, pista que será recogida la próxima vez que visite dicho nodo. En otras palabras, a medida que un usuario conoce los nodos de un grafo, la representación sintáctica de los mismos pasará a convertirse en claves de búsqueda para la información previamente almacenada por el usuario en su memoria semántica acerca de estos nodos .

Cuando el usuario accede al diálogo interactivo creado a partir del punto de partida de un proceso de usuario (el proceso *UP_* en CodeX), en realidad puede estar accediendo a todo un grafo, ya que dicho proceso puede estar formado a su vez por una red de procesos de usuario y/o de procesos internos de la aplicación. Puede apreciarse por tanto como la ejecución de un simple proceso de usuario se convierte en un proceso de navegación en el que la asimilación de contenidos es equiparable al desarrollo la excursión de hormigas descrita. Desde este punto de vista, los agentes hormiga de ANTS no solo observan al usuario en cada una de sus acciones, sino que emularán además su comportamiento en un modo similar a la simulación realizada por DEVA (capítulo 12: *El Modelo Cognitivo*); simulación que se efectuará naturalmente a una escala mucho menor.

En la arquitectura de módulos diseñada para GADEA, cuando DEVA crea un nuevo diálogo interactivo, los agentes hormiga son automáticamente incorporados a éste antes de que sea mostrado al usuario. Al respecto cabe decir que dichos agentes son incorporados exactamente del mismo modo en el que se incluyen otros recursos multimedia estándar, como pueden ser los textos, imágenes y sonidos. Esta tarea puede es realizada de forma automática por el módulo DEVA, mediante solicitudes explícitas al módulo ANTS en función del tipo de *widgets* contenidos en el diálogo y en base también al modelo de interacción empleado.

No obstante, al igual que ocurre con el módulo CodeX y con el módulo DEVA, la arquitectura ANTS ha sido diseñada para que éste módulo pueda funcionar de manera totalmente autónoma con respecto a los demás módulos, de cara siempre a su utilización como producto individual. Nótese que gracias a este diseño modular, ANTS puede ser empleado como herramienta para la realización de pruebas de usabilidad automáticas orientadas a la navegación, no en balde el nombre de este módulo es un acrónimo para *Automatic Navigability Testing System* o *Sistema de Pruebas de Navegación Automático*.

Puesto que la plataforma Java en la que está desarrollado ANTS y GADEA en general está presente en prácticamente cualquiera de las plataformas multimedia del mercado, incluyendo Unix, Windows y Macintosh, así como en una en una amplia colección de exploradores de Internet como Microsoft Explorer o Netscape Communicator, es posible emplear este módulo por separado para realizar pruebas de usabilidad automáticas sobre los modelos de navegación de cualquier tipo de producto hipermedia disponible en el mercado. Esta generalización incluye tanto a las aplicaciones hipermedia dependientes de una plataforma concreta, como a los sitios web en Internet [González et al (2000g)], tanto estáticos como dinámicos (generados mediante Servlets

y JSP), la cual es uno de los principales objetivos de esta herramienta [González y Álvarez (2000d)].

Dado que los agentes cumplen íntegramente la especificación de los componentes *JavaBeans* de Sun (1996), las propiedades públicas de las hormigas pueden ser configuradas fácilmente tanto por una herramienta de desarrollo como manualmente por el programador de un producto hipermedia. Una vez que las hormigas han sido incluidas en los nodos de información del modelo de navegación de la aplicación, la versión de prueba del producto está lista para ser distribuida entre los voluntarios participantes en las pruebas de usabilidad. De este modo, cualquier herramienta para el desarrollo de productos hipermedia basada en *JavaBeans* puede beneficiarse de esta tecnología, caso de JEDI [Transtools (1998)] o CineMedia Astur [Redondo et al (2001)].

Cuando un usuario visita un nodo durante su sesión de navegación por un diálogo interactivo, la hormiga o hormigas asociadas al mismo son descargadas desde hormiguero (el servidor ANTS) junto con el resto de los recursos multimedia y *widgets* que integran la estructura del dialogo, los cuales, en función del tipo de información pueden provenir de distintas fuentes como por ejemplo servidores web, administradores de recursos, el *Language Manager*, etc.

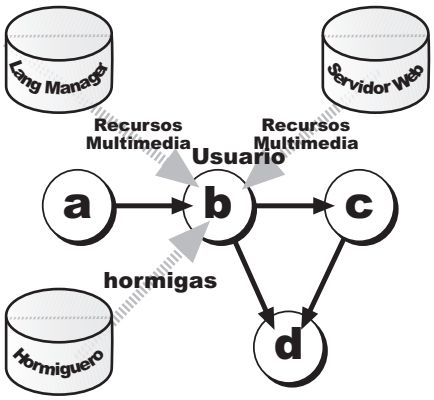


Figura 87. Los *widgets* asociados a un diálogo interactivo son descargados junto con los agentes hormiga en el mismo momento en el que el usuario accede al diálogo.

Una vez que la hormiga llega a la máquina del usuario sea su situación local o remota con respecto a la ubicación de su hormiguero, este agente establece un canal de comunicación con el hormiguero bien mediante *sockets* o bien mediante invocación remota de métodos o RMI (*Remote Method Invocation*). A partir de este momento, el canal de comunicación será empleado para enviar la información recolectada al servidor de origen de la hormiga (ver Figura 88). La amplia variedad de protocolos de comunicación proporcionados por Java (*sockets*, RMI, CORBA, etc.) ha sido un factor de gran ayuda en el diseño y posterior desarrollo de ANTS, ya que esta variedad permite la selección dinámica del mejor canal de comunicación disponible en tiempo de ejecución, dadas unas necesidades concretas. Así por ejemplo, a la hora de establecer el canal de comunicación las hormigas intentan hacerlo

primero mediante *sockets* y si la apertura del canal de este comunicación falla, se intentará emplear RMI en su lugar [González y Álvarez (2000h)]. Esta estrategia permite comunicación con el servidor de la forma más rápida posible (*sockets*) pero menos fiable, y si esta estrategia falla se intenta emplear una estrategia más lenta (el uso de RMI implica la descarga de extra de clases *stub*) pero más segura, ya que RMI suele ser inmune a los cortafuegos basados en *proxies* al disfrazar la conexión mediante el puerto 8080 usado normalmente por el protocolo *HTTP*, normalmente activado por el administrador del sistema de cualquier cortafuego [Eckel, (2000)].

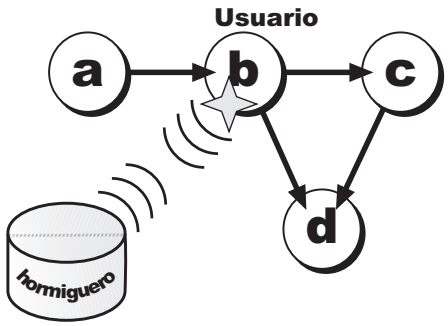


Figura 88. El agente-hormiga (la estrella de color gris) en el nodo B observa el comportamiento del usuario y envía un informe al hormiguero a través de su canal de comunicación.

Un objeto de tipo asistente-hormiga ubicado en el servidor (ver Figura 89) será el encargado de mantener activo el canal de comunicación mientras dure la sesión de navegación. Cada una de la sesiones abiertas por el usuario dispone de su propio asistente-hormiga de modo que cada agente que se encuentra participando en una sesión de navegación deberá contactar con su propio asistente-hormiga. Este objeto recogerá la información enviada por su hormiga asociada y una vez clasificada, la enviará al almacén para que ésta sea guardada en el registro correspondiente del modelo de usuario individual o del UAM.

Los objetos asistentes-hormiga son responsables además de mantener el canal de comunicación abierto por la hormiga en perfecto estado operativo, ya que el hormiguero verifica periódicamente el estado de cada canal de comunicación de cada hormiga que se encuentre activa activos. Esta operación evita la existencia de asistentes-hormiga conectados a hormigas muertas. Así, si el ordenador del usuario se cuelga o la comunicación falla, la hormiga muere, ya que a partir de ese momento se verá incapaz de proporcionar nuevos datos al servidor. De forma periódica, toda hormiga debe enviar información de control a su asistente-hormiga. Si la hormiga deja de enviar esta información durante un período de tiempo razonable, se considera que la hormiga esta muerta y su correspondiente asistente-hormiga es eliminado de la posición que ocupa en la memoria del servidor.

El control periódico del estado de cada hormiga conectada al servidor es particularmente importante cuando se prueban prototipos de aplicaciones GADEA. Hay que destacar que la fase de pruebas de los modelos de navegación coincide normalmente con la fase de pruebas

del código del producto. Dado que el prototipo en pruebas suele ser todavía una versión sin depurar del todo, lo cual incrementa la probabilidad de que dicho prototipo presente algún tipo comportamiento anómalo que pueda conducir a un agotamiento de recursos del ordenador del usuario y como consecuencia a la muerte de la hormiga asociada.

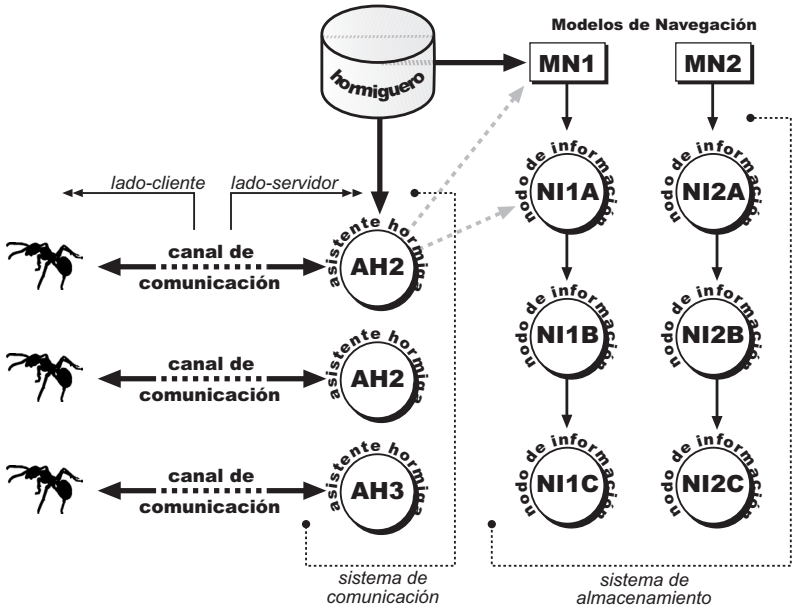


Figura 89. Los asistentes-hormiga están en contacto con las hormigas remotas (izquierda) por medio de un canal de comunicación. La información recibida por los agentes-hormiga es enviada al sistema de almacenamiento, donde es organizada y guardada en el nodo correspondiente al modelo de usuario en estudio..

19.1.2. Creación Manual de Hormigas

Cuando el hormiguero recibe información proveniente de una hormiga, dicha información es guardada en uno de los almacenes del hormiguero. Como la naturaleza multitarea del hormiguero permite el control en paralelo de para varios modelos de navegación a la vez, la información recogida debe ser organizada en categorías, dependiendo del modelo que está siendo probado por cada agente. Para saber en que modelo es necesario almacenar la información recogida, el agente debe notificarle a su asistente en el lado servidor el modelo de navegación concreto está probando. Para poder realizar esta tarea, un código de identificación es asignado a cada modelo de navegación bajo observación.

Sin embargo, esta información no es suficiente, dado que el servidor debe conocer el nodo exacto que el agente está probando, de tal modo que se requiere también un código que identifique cada nodo del modelo de navegación. Los dos identificadores mencionados han de ser únicos y deben ser asignados durante la fase de diseño del proyecto en el caso de una prueba manual de la aplicación o de forma automática y dinámica en el caso de una ejecución bajo el control total de GADEA.

En el primer caso, los diseñadores pueden asignar estos códigos manualmente, aunque lo recomendado es que sea la propia herramienta de desarrollo la que asigne éstos identificadores de forma automática.

En el Código 34 se muestra como configurar los parámetros de una hormiga para probar con ANTS una aplicación hipermedia cualquiera no compatible con GADEA. Como se puede apreciar, la configuración de la hormiga tiene lugar dentro del constructor del nodo del nodo que se quiere observar. La identificación del modelo de navegación y del nodo es proporcionada como parte de los parámetros del constructor. Los últimos dos parámetros del constructor de la hormiga son dependientes del diseño de ANTS y representan el nombre de la máquina que actúa como servidor (hormiguero) y el número puerto de comunicación empleado por los *sockets* en la transmisión de la información. En el ejemplo, la hormiga ha sido diseñada para capturar también eventos de relacionados con el uso del ratón. En concreto, la hormiga informará al servidor cada vez que el usuario pulse el botón declarado al principio del constructor.

Código 34. Configuración manual de un hormiga para la observación de un nodo en una aplicación no GADEA.

```
/**
constructor del nodo

@param theModel identificación del modelo de navegación
@param theNode identificación del nodo
*/

public node (String theModel, String theNode, …)
{
   // Configuración de otros recursos multimedia.
   ...
   java.awt.Button miBotón = new Button ("esto es un botón");
   ...

   // Configuración de la hormiga
   hormiga = new gadea.antas.Ant (theModel, theNode,
polar.uniovi.es, 1932);

   hormiga.addActionListener (miBotón);

   // Otras operaciones ha ser realizadas por el constructor
   ...
}
```

El fragmento de código HTML incluido en el Código 35 muestra la configuración equivalente de la misma hormiga para probar un sitio web. El parámetro MODEL representa el código asignado al modelo de navegación (un sitio web denominado *Tirsus*). El parámetro ID proporciona el código que representa a una página web concreta, identificada como *Portal*. En este ejemplo, el nombre de la máquina que hace las veces de hormiguero, así como el puerto de comunicación empleado por el *socket* es opcional, dado que forman parte del código interno del *applet* que representa al agente. La clase *gadea.ants.AdvertApplet* muestra una señal visual (un *banner* diseñado mediante una imagen animada) que indicando que el *applet* contiene una hormiga en su interior, de tal modo que el usuario sabe que la página web que está visitando está siendo probada mediante ANTS.

Código 35. Configuración de una hormiga destinada a observar acciones realizadas sobre una página web.

```
<applet
 code="gadea.ants.AdvertApplet.class"
 codebase = "http://www.di.uniovi.es/~martin/BinJava"
 width=100
 height=32>
  <param name="MODEL" value="Tirsus">
  <param name="ID"  value="Portal">
</applet>
```

Cuando la hormiga establece el canal de comunicación con el hormiguero, debe especificarle quien será su asistente-hormiga en el lado servidor (ver Figura 89). Dado que existe un asistente-hormiga por cada sesión de navegación, se requiere un código que identifique a cada una de estas sesiones. Este código dependerá del tipo de proceso de usuario o diálogo interactivo que se está probando y en función de ello será generado de forma automática por la propia hormiga. Para páginas web, una combinación de la dirección IP (*Internet Protocol*) y del nombre de la máquina del usuario pueden ser suficientes. Para una aplicación GADEA, los agentes pueden complementar la información anterior con el nombre del usuario, el número de licencia del producto, o incluso mejor aún, con un código especial proporcionado por los propios diseñadores en función del nivel de detalle requerido en las pruebas.

19.1.3. Información Recolectada por ANTS

La información recogida por los agentes consiste principalmente en una identificación completa de los puntos de referencia empleados por el usuario para navegar a través de los nodos que integran un proceso de usuario incluyendo una referencia al modelo de navegación correspondiente. Esta información es completada con la hora de partida o salida de/hasta el nodo. Las hormigas son capaces de obtener esta información temporal incluso cuando el usuario realiza una tarea diferente a la pretendida, es decir, cuando el usuario trabaja concurrentemente con otra aplicación, una practica corriente en plataformas de trabajo multitarea. En el caso de los agentes que se ejecutan dentro de páginas web, este evento es detectado por medio de los métodos *start* y *stop* de la clase *applet* modificados mediante herencia en la clase en la *clase.gadea.ants.AdvertApplet*.

Si el gestor de seguridad de la máquina virtual de Java lo permite, las hormigas son capaces además de obtener información acerca del propio usuario, incluyendo su nombre, la plataforma de trabajo que emplea o la versión de su sistema operativo (Weber 1997). Naturalmente, este modelo genérico, pensado para su uso en la realización de pruebas de navegación automáticas de cualquier tipo de interfaz, es extendido por GADEA para la realización de pruebas de usabilidad avanzadas en las que es necesario extraer información más detallada. Dado que todo agente es en realidad un *JavaBean* [Sun (1996)], estos se pueden subscribir como *listeners* o escuchador de los eventos estándar producidos cualquier *widget* incluido en el mismo diálogo interactivo al que pertenece el agente. De este modo, los agentes

pueden obtener información acerca de cualquier tipo de actividades relacionada con estos *widget* y realizada por el usuario en cada una de sus sesiones de navegación. Estas actividades pueden incluir por ejemplo, el conjunto de comandos invocados, la lista de las zonas de una ventana en donde el usuario a pulsado con el ratón, el conjunto de *widgets* activos seleccionados o la colección de componentes que pueden estar visibles u ocultos en el diálogo interactivo en un momento dado.

19.2 Especialización de las Hormigas

Los agentes pueden trabajar en combinación con otros componentes especialmente diseñados para las pruebas de usabilidad, como por ejemplo, aquellos destinados a medir la habilidad del usuario al usar el teclado [Salthouse (1986)], o sus habilidades en el reconocimiento de formas visuales [Sternberg (1969 y 1975)]. El tipo de información capturada por los agentes dependerá obviamente de la información requerida por los diseñadores a la hora de mejorar la calidad de sus productos.

Continuando con la metáfora de la colonia de hormigas planteada para el diseño de ese módulo, cada tipo de agente empleado por ANTS se encuentra especializado en la captura de un tipo determinado de información. Así, mientras en el reino animal existen hormigas obreras, hormigas niñera, hormigas reina y hormigas soldado, etc., en este módulo existen una clase de hormigas especializada en la obtención de la *Lateralidad* del usuario, una clase de hormigas diseñada para la medición de su *Precisión Motriz*, una clase específica para la obtención de su tiempo de reacción medio (Tm) etc., así hasta cubrir todos y cada uno de los parámetros incluidos en el modelo de usuario individual (ver la Tabla 38) y en el UAM (ver Tabla 39). En algunos casos y por razones de eficiencia existen hormigas capaces de obtener más de un tipo de dato, como es el caso por ejemplo de la clase de hormiga encargada de obtener una medida de la habilidad del usuario al emplear un teclado, la cual es capaz de obtener los valores T_k y T_{km} correspondientes al tiempo medio de pulsación en el teclado y en el ratón respectivamente.

Tabla 38. Valores contenidos en el modelo de usuario empleado por GADEA.

PARÁMETRO	DESCRIPCIÓN	MEDIO DE OBTENCIÓN
Identificación	Código de usuario único empleado para identificar a éste en un entorno GADEA	Explícito
Nombre	Nombre del usuario (válido únicamente a efectos descriptivos)	Explícito
Apellidos	Apellidos del usuario (válido únicamente a efectos descriptivos)	Explícito
Clave de Acceso	Clave que permite a un usuario de un entorno GADEA identificarse como tal	Explícito

Edad	Grado de pertenencia a conjuntos difusos: Bebé, Niño, Joven, Joven Adulto, Adulto, Adulto Maduro, Anciano y Octogenario	Explícito
Sexo	Valor discreto: Masculino o Femenino	Explícito
Precisión Auditiva	Grado de pertenencia a conjuntos difusos: Sordo, Bajo, Normal Bajo, Normal Alto y Óptimo	Explícito susceptible de ser automatizado
Precisión Visual	Grado de pertenencia a conjuntos difusos: Ciego, Discapacidad, Bajo, Normal Bajo, Normal Alto, Alto y Excelente	Explícito susceptible de ser automatizado
Precisión Motriz	Grado de pertenencia a conjuntos difusos: Nula, Baja, Normal, Alta y Excelente	Ley de Fitt adaptada
Lateralidad	Valor discreto: Zurdo o Diestro	Explícito susceptible de ser automatizado
T_k	Tiempo medio necesario para pulsar una tecla del teclado	Regularidades de Salthouse
T_{km}	Tiempo medio requerido para pulsar un botón del ratón	Regularidades de Salthouse
T_p	Tiempo medio de desplazamiento del puntero desde un punto origen hasta un punto destino	Ley de Fitt
T_h	Tiempo medio requerido para cambiar de dispositivo de entrada de datos	Medición directa
T_d	Tiempo medio para realizar operaciones de arrastre con el puntero	Ley de Fitt adaptada
T_m	Tiempo medio para operaciones de toma de decisiones instantánea	Ley de Hick, Principio de la Incertidumbre

A pesar de esta especialización de las hormigas en clases, hay que destacar que todas ellas comparten un patrón de comportamiento común que las hace capaces de detectar los parámetros relacionados con la navegación por el grafo que representa el modelo generado a partir de un proceso de usuario. Aunque esta funcionalidad añadida no es empleada por todas las hormigas, se ha conservado teniendo en cuenta ampliaciones futuras de la capacidad de adaptación sintáctica de la interfaz en función de un contexto de navegación determinado. De este modo, la información proporcionada por un agente especializado por ejemplo en la captura de la *Precisión Motriz* del usuario, puede acompañar esta información con datos acerca del punto en concreto del modelo de navegación en donde éstos han sido capturados, de tal modo que sea posible detectar determinados cambios en la habilidad del usuario con respecto al contexto, cambios que obviamente necesitarán

de un proceso de adaptación selectivo si se quiere equilibrar la carga cognitiva de las operaciones sintácticas realizadas en toda la aplicación.

Tabla 39. Valores contenidos en el UAM (Modelo de Usuario por Aplicación) por proceso o diálogo interactivo.

PARÁMETRO	MEDIO DE OBTENCIÓN	DESCRIPCIÓN
N	Inspección Automática	Número de veces que el usuario ha invocado el proceso de usuario o ha empleado el diálogo interactivo
T1	Inspección Automática	Tiempo empleado en el proceso de usuario o diálogo interactivo la primera vez que éste fue invocado.
Tn	Inspección Automática	Tiempo empleado en el proceso de usuario o diálogo interactivo la última vez que éste fue invocado.
Alfa	Cálculo Automático	Medida de la destreza del usuario en esta operación. Desplazamiento de la curva de aprendizaje con respecto a lo esperado en un usuario normal.
Experiencia	Cálculo Automático	Grado de pertenencia a los conjuntos difusos *novato* y *experto*

La obtención de la mayoría de los parámetros reflejados en la Tabla 38y en la Tabla 39 es inmediata mediante la aplicación de un simple proceso de medición de tiempos, el establecimiento de contadores para determinadas acciones o mediante la realización de preguntas sencillas pero directas al usuario. Sin embargo, existen determinados parámetros en donde no basta con una simple obtención de datos en bruto, sino que es necesario realizar un cuidadoso proceso de análisis del flujo de datos obtenido si se desean obtener datos fiables a partir de los cuales sea posible realizar predicciones de comportamiento. La importancia que tienen estos procesos de análisis, así como de las reglas y teorías de la psicología cognitiva implicados en ellos, requiere una explicación adicional en este documento y para ello se empleará el agente *gadea.ants.MotionAccuracy* encargado de determinar la *Precisión Motriz* de cada usuario.

El modo de interacción planteado por GADEA para el acceso a los *widgets* activos contempla el uso del ratón u otro dispositivo mecánico de naturaleza similar (joystick, trackball, etc.) para el movimiento del puntero hasta los mencionados *widgets*. Analizando y evaluando el modo en el que estos dispositivos son manejados por el usuario, así como su patrón de uso, es posible extraer información suficiente como para obtener un parámetro de medida de la *Precisión Motriz* bastante preciso.

El movimiento del brazo o de la mano necesario para desplazar el puntero por la pantalla del ordenador está definido por la Ley de Fitt [Newell (1991)], la cual proporciona una medida muy fiable del tiempo requerido por un humano para realizar este tipo de desplazamientos desde un punto inicial hasta un objeto destino. De acuerdo con esta ley, el lapso de tiempo requerido para que el procesador perceptivo gestione este desplazamiento está determinado por la distancia que existe entre la posición inicial y el objeto destino y el tamaño del objeto.

El primer parámetro –la distancia– se suele denotar con la letra D, mientras que al segundo –el tamaño– se le designa con la letra S, considerando que esta medida representa indistintamente el ancho o el alto del objeto destino. Dada esta situación, el tiempo necesario para alcanzar el objeto destino viene dado por la siguiente igualdad:

$$T = I \log_2 (D / S + 0{,}5)$$

En donde T es el tiempo de desplazamiento e I es la velocidad del procesador motriz individual de cada usuario, situada entre las 30 y las 100 milésimas de segundo (ver Tabla 17). Como se puede apreciar, el tiempo de desplazamiento es inversamente proporcional al tamaño del objeto.

La Ley de Fitt es extremadamente robusta y se cumple tanto si se trata de un movimiento repetitivo (adelante y atrás) como si se trata de un solo movimiento. También se cumple con independencia de si se realiza con un dedo, la mano o empleando el ratón para desplazar un puntero por la pantalla. Esta ley se cumple también para el movimiento de una pierna, un pie o incluso si se mueve un puntero con la boca. Al parecer, esta ley también se cumple bajo el agua o cuando se observa a través de un microscopio y se acede la muestra con un micro manipulador [Keele (1986)]. La versión original de la Ley de Fitt se basaba en tareas realizadas en una sola dimensión, pero también puede aplicarse sobre tareas basadas en espacios en dos y hasta en tres dimensiones, teniendo en cuenta siempre que el desplazamiento debe realizarse a lo largo de una línea recta. En el caso de trabajar en dos y tres dimensiones deberán realizarse las correcciones necesarias para que los parámetros D y S puedan ser medidos a lo largo de la línea recta seguida desde la posición inicial hasta la posición mantenida por el objeto destino. En todos los casos, durante la operación de desplazamiento del puntero, el usuario no tiene que alcanzar un punto determinado del objeto destino, bastando con que sitúe el puntero en cualquier punto dentro del área que éste ocupa.

Al aplicar la Ley de Fitt se ha de tomar en cuenta que lo que se está midiendo es el movimiento de la mano o el dedo y no el movimiento del puntero en la pantalla. Así por ejemplo, para medir el tiempo requerido para seleccionar un menú en la pantalla del ordenador, se ha de medir la distancia entre la posición inicial y el menú, así como el tamaño de éste en términos del movimiento de la mano, es decir, la distancia que el ratón debe desplazarse y no en términos de la distancia que el puntero debe recorrer.

Las teorías que intentan explicar la Ley de Fitt se basan en la precisión limitada que posee el procesador motriz humano [Keele (1986), Newell (1991)]. Según estas teorías cuando se desplaza un puntero hacia su destino, la precisión limitada de este procesador motriz impide trazar una línea recta perfecta desde el punto original hasta el destino, por lo que es necesaria una estrecha colaboración con el procesador perceptivo, el cual proporciona –a través del sentido de la

vista– información necesaria para realizar las oportunas correcciones a la trayectoria actual del desplazamiento para adecuarla lo más posible a la línea recta.

Nótese que éstas correcciones han de realizarse tanto en la dirección seguida como en la velocidad del desplazamiento so pena de sobrepasar el objeto destino una vez alcanzado éste. Obviamente, mientras más pequeño es el objeto destino mayor será el número de correcciones ha realizar, lo que explica el aumento en el tiempo de desplazamiento previsto por la ley. En la Figura 90 se pueden apreciar las correcciones realizadas por un usuario al desplazar el puntero desde el área superior de una diálogo interactivo para pulsar sobre el botón CANCEL situado en la esquina inferior derecha. La línea azul entre la posición inicial y la posición final del puntero representa la trayectoria ideal.

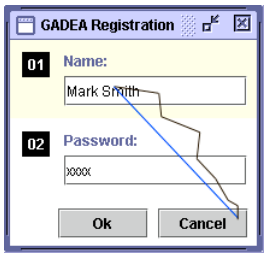


Figura 90. Trayectoria seguida por el puntero al pulsar sobre el botón *CANCEL* (consultar versión en color en el capítulo 24: *Imágenes en Color*, p. 445).

El agente *gadea.ants.MotionAccuracy* hará uso precisamente de este proceso de corrección continua para determinar la *Precisión Motriz* del usuario, examinando la trayectoria real seguida por el desplazamiento del ratón por la pantalla del ordenador al seleccionar un *widget*, comparando dicha trayectoria con la trayectoria ideal, es decir, con el camino más corto entre la posición de partida y aquella en la que se ha realizado la pulsación. Por ello, este agente subscribe el evento MouseMotion con el objeto de almacenar temporalmente las coordenadas de pantalla empleadas por el usuario en el desplazamiento del puntero en operaciones de selección de widgets. Si tras iniciar el desplazamiento del ratón, el usuario realiza una pulsación con el mismo dentro de un período de tiempo razonable, la secuencia de coordenadas obtenidas como subscriptor del evento son procesadas como válidas para el cálculo de la *Precisión Motriz*. En lugar de aplicar un costoso proceso de integración para determinar el área existente entre la trayectoria real y la trayectoria ideal (que no es otra cosa que una medida bastante fiable de la *Precisión Visual*), el agente opta por un análisis simplificado basado en el cálculo del número de correcciones realizadas por el usuario en su trayectoria (N). Para ello se crea un hilo de baja prioridad capaz de detectar en cuales de las coordenadas registradas se ha producido una modificación de la trayectoria.

Este proceso de análisis consiste en trazar una línea recta entre dos parejas de coordenadas consecutivas y hallar su pendiente mediante la función:

$$M_{i+1} = (Yi_{+1} - Y_i) \ / \ (Xi_{+1} - X_i)$$

Una vez calculadas las pendientes M_i correspondientes a todos los segmentos que forman parte de la trayectoria, la detección de un punto de corrección en la trayectoria es tan sencillo como detectar una pareja consecutiva de pendientes de valores distintos. Como conclusión de este proceso, el agente obtendrá un número N correspondiente al número de cambios de trayectoria practicados por el usuario en un desplazamiento simple.

Pulsar sobre un objeto destino puede ser analizado como la aplicación de una serie de movimiento simples en donde cada uno de estos movimientos acerca el puntero cada vez más al objetivo. Al inicio de cada movimiento, el usuario observa la posición actual del puntero y del objetivo tomando una decisión que implica variar la dirección y velocidad de la mano. En cada uno de estos movimiento simples se corrige el error introducido por el movimiento anterior mejorando la trayectoria en una proporción fija que depende de la *Precisión Motriz* a la que por simplificar denotaremos por P.

Suponiendo que X_0 representa la distancia original que existe desde la coordenada de inicio hasta aquella en la que el usuario realiza la pulsación (D), podemos decir que tras la aplicación del primer movimiento simple y sin correcciones posteriores, solo sería posible atinar a un objetivo de tamaño PX_0. En el punto X_1 el usuario puede modificar la trayectoria para intentar corregir el error introducido desde el punto X_0, consiguiendo con ello alcanzar un objeto destino de tamaño AX_1. Este proceso se aplica continuamente a lo largo de toda la trayectoria seguida, obteniendo la siguiente serie:

$X_0 = D$

$X_1 = PX_0$

...

$X_{n+1} = PX_n$

Resolviendo la serie, se obtiene que:

$X_n = P^n X_0$

O lo que es lo mismo:

$$X_n = P^n D$$

Este proceso de refinación progresiva de la trayectoria continuará hasta que el puntero se encuentre dentro del área del objeto destino, situación definida por la siguiente desigualdad:

$$X_n <= S/2$$

Dado que el proceso se detiene tan pronto como sea posible, esta desigualdad se puede convertir en una igualdad obteniendo finalmente que:

$$X_n = S/2$$
$$X_n = P^n D$$

Lo cual es equivalente a decir que:

$$S/2 = P^n D$$

Dado que el agente calcula el valor de N mediante la técnica ya descrita, todos los elementos de la ecuación anterior son conocidos salvo P, el cual se consigue aplicando logaritmos en los dos miembros de la ecuación del modo siguiente:

$$S/2 = P^N D$$

$$\ln(S/2D) = \ln P^n$$

$$\ln(S/2D) = N \ln P$$

$$(\ln(S/2D))/N = \ln P$$

Por último, y solo queda despejar P aplicando exponenciales en los dos miembros de la ecuación para obtener P.

$$e^{(\ln(S/2D))}/e^N = e^{\ln P}$$

Obteniendo que:

$$P = (S / 2D) / e^{N}$$

Este resultado permite obtener la *Precisión Motriz* de una operación de desplazamiento de puntero de forma mucho más sencilla y eficaz que mediante integrar cada segmento de la trayectoria real con respecto a su correspondiente segmento de la trayectoria ideal, reduciendo con ello el tiempo de ejecución necesario para obtener P.

El valor de este parámetro se calcula cada vez que el usuario realiza una operación de este tipo, actualizando el valor promedio del mismo en el modelo de usuario individual. Nótese que ése valor es una medida absoluta de la eficacia del usuario en el desarrollo de este tipo de tareas y por lo tanto no puede ser empleada directamente por el motor de inferencias de GADEA. Para ello, este valor absoluto es convertido en un grado de pertenencia a los conjuntos difusos definidos para *Precisión Motriz* (*Nula, Baja, Normal, Alta* y *Excelente*). Esta conversión se realiza una vez por sesión, durante el proceso de inicio de esta. De este modo, se evitan cambios bruscos en los resultados obtenidos por el proceso de adaptación realizado por DEVA, ya que estos cambios, de producirse, se realizan entres sesiones y no a lo largo de una sesión de interacción.

19.2.1. Evaluación

Naturalmente, el uso de agentes para la observación, detección y análisis de los distintos patrones de comportamiento empleados por el usuario a lo largo de su sesión de interacción tienen un precio en términos de rendimiento. De hecho, el único argumento negativo que tiene el uso masivo de agentes para la actualización de las diferentes entradas del modelo de usuario radica en una posible disminución de rendimiento, que pueda conducir a la incapacidad operativa del sistema, tanto en términos de usabilidad como en términos de funcionalidad. Es por ello que una vez desarrollado el sistema de agentes aquí descrito, se procedió a la elaboración de pruebas de rendimiento para cuantificar la posible pérdida de éste, así como determinar el número de actuaciones adecuado de los agentes por sesión de trabajo del usuario.

Para simplificar el proceso de evaluación se consideró únicamente al *gadea.ants.MotionAccuracy* descrito en el apartado 19.2 (*Especialización de las Hormigas*) por ser aquel agente hormiga de ejecución más costosa en tiempo y en recursos, dado que se trata del único agente que no obtiene sus conclusiones directamente a partir de sus observaciones sino que ha de realizar ciertos cálculos de complejidad moderada antes de poder dictaminar su conclusión. Una vez seleccionado el agente de peor rendimiento, se procedió además a simular la peor de la situaciones posibles para el uso de este agente, que no es otra que la detección de trayectorias complicadas y con un gran número de observaciones. En

concreto, se crearon de forma artificial y aleatoria trayectorias formadas por veinte tramos, todos ellos con pendientes distintas, simulando por tanto trayectorias con veinte puntos de corrección. En realidad, en las pruebas unitarias realizadas durante el desarrollo de ANTS, este parámetro se sitúa en torno a los doce segmentos para espacios visuales de unos 400 x 300 pixeles. Con el objeto de complicar aún más el escenario en términos de rendimiento, se supuso un complejidad notablemente elevada para la actividad a realizar por la hormiga, determinándose el empleo simultáneo de hasta treinta agentes de observación por operación, cuando lo habitual es el empleo de un solo agente por operación.

Tabla 40. Variables lingüísticas para la simulación de carga de trabajo.

CARGA DE TRABAJO	CICLOS DE EJECUCIÓN
Muy Alta	1.000
Alta	750
Moderadamente Alta	500
Moderadamente Baja	250
Baja	150
Muy Baja	50

Suponiendo el peor de los casos posibles para el escenario de las pruebas, es de esperar recoger la cota superior del rendimiento de las hormigas. Para adecuar este contexto a un escenario real, se realizó esta simulación suponiendo seis tipos distintos de cargas de trabajo para la aplicación, entendiendo por carga de trabajo la duración media de cada unidad de producción de los procesos internos de la aplicación, midiendo esta carga de trabajo en términos de esfuerzo de computación. Para ello se diseñó una clase *workload* encargada de simular el comportamiento de aquel proceso interno de la aplicación invocado por el proceso de usuario disparado a partir de una determinada interacción del usuario con el diálogo interactivo generado por DEVA (y por ende observado por ANTS). Naturalmente, mientras mayor es la carga de trabajo de este proceso interno, mayor será su duración y dado que éste proceso dispone de la máxima prioridad de ejecución, menor serán los recursos asignados para la ejecución de los agentes de observación. Es por ello que a mayor carga de trabajo de una aplicación y a mayor saturación de los componentes hardware del equipo en la que ésta se ejecuta, mayor será el peso de los agentes sobre el rendimiento. Como no podía ser de otra manera, los diferentes cargas de trabajo se diseñaron como las variables lingüísticas incluidas en la Tabla 40. Cada una de estas variables representa un determinado número de ciclos de ejecución en la simulación y por lo tanto a mayor la carga, mayor será el tiempo de ejecución.

La simulación se realizó en un viejo ordenador Pentium MMX a 200 Mhz. dotado de 64 Mb. de RAM bajo el sistema operativo Windows NT 4. Aunque las prestaciones de este máquina distan mucho del estándar que rige el mercado actual, la obsolescencia del equipo resulta un valor

añadido a la efectividad de ANTS cuando sus agentes se ejecutan sobre máquinas de gama baja. En la Figura 91 se puede apreciar la evolución del tiempo de ejecución total de la simulación para cada una de las cargas de trabajo de la Tabla 40 en base al número de agentes incluidos en la simulación. Naturalmente, a mayor número de agentes, mayor es el tiempo total de ejecución consumido. Análogamente, mientras mayor sea la carga de trabajo, mayor es el tiempo de ejecución requerido.

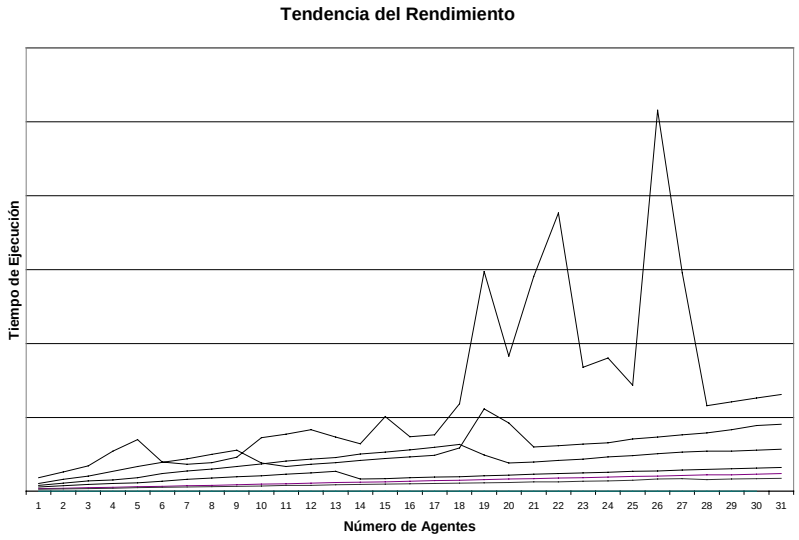


Figura 91. Evolución del tiempo de ejecución requerido por una secuencia de cero a treinta agentes de tipo *gadea.ants.MotionAccuracy* calculada en base a diferentes cargas de trabajo. De arriba abajo: *muy alta, alta, moderada alta, moderada baja, baja* y *muy baja.*

En la Figura 91 se puede estimar a simple vista la tendencia de las curvas para todas las cargas de trabajo, siendo todas estas curvas lineales. Como se puede observar, la actuación simultánea de un mayor número de agentes de observación por ciclo de computación no implica una aumento cuadrático del tiempo de ejecución sino lineal, con la consecuencia obvia de que el tiempo de ejecución extra aportado por los agentes se acopla en las curvas siguiendo un criterio acumulativo, con lo que la pérdida de rendimiento no es tan drástica como podía parecer en un principio.

De todos modos, ésta pérdida existe y es grande, especialmente cuando se dispara el número de agentes implicados en el proceso de observación. En la Figura 92 se puede observar en detalle el aumento progresivo del tiempo de ejecución global para la simulación para una carga de trabajo *moderada baja* en función del número de agentes implicados. Como se puede observar, el tiempo de ejecución empleado por la aplicación en ausencia de agentes observadores se duplica cuando el número de estos agentes llega a siete, cota máxima aconsejable para este tipo de carga de trabajo y en general para cualquier escenario.

Dado que con el empleo simultáneo de aproximadamente siete agentes de observación por cada unidad de producción se alcanza a una cota de degradación de rendimiento cercana al 100%, la cota máxima

para la pérdida global de rendimiento para una aplicación se puede situar en un 14% de promedio por agente y en función de la carga de trabajo, lo cual aconseja el uso de un máximo de uno o dos agentes de este tipo de forma simultánea. Esta cota superior disminuye a medida que aumenta la carga de trabajo de la aplicación, aumentando ligeramente para las cargas de trabajo más ligeras.

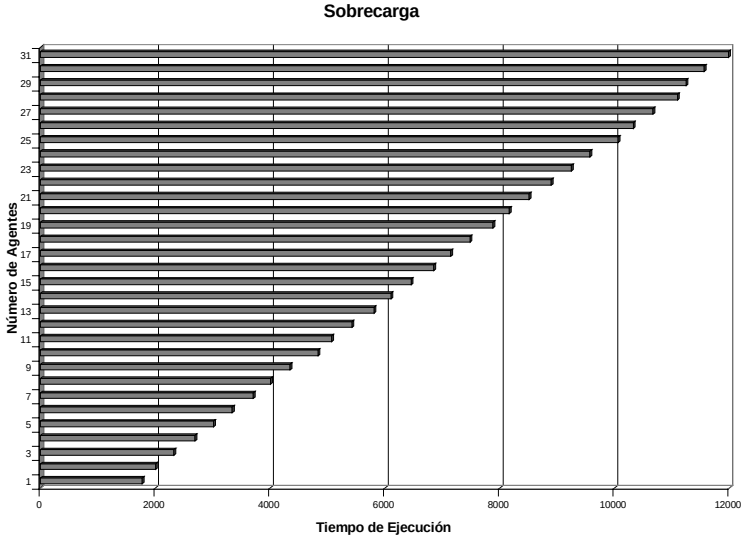


Figura 92. Sobrecarga del tiempo de ejecución en función del número de agentes de tipo *gadea.ants.MotionAccuracy* empleados en una sesión de carga de trabajo *moderada baja.*

A raíz de estos resultados y con el objeto de mejorar el rendimiento de los agentes de tipo *gadea.ants.MotionAccuracy*, es decir, de aquellos que requieren de un proceso de análisis extra de la información recolectada, se procedió a modificar su diseño para que fuesen capaces de ejecutarse en un hilo de ejecución paralelo dotado de menor prioridad de aquel en el que se ejecuta la aplicación principal. El resultado obtenido fue una considerable reducción en el peso que la actividad de este tipo de agentes tiene sobre el tiempo total de ejecución.

En la Figura 93, se puede apreciar de forma gráfica como se reduce el tiempo de ejecución global de la simulación al introducir la multitarea en el diseño de ANTS. Esta reducción puede situarse en torno al 32% para un solo agente y en el 44% para el total de siete agentes ilustrado en la Figura 93, dado que mientras mayor es el número de agentes empleados de forma simultánea, mayor es el tiempo ahorrado por la modalidad de ejecución en multitarea. De este modo, la cota máxima para la pérdida de rendimiento puede pasar de un 14% a un moderado 9,5% mediante el empleo de un sistema de agentes ejecutables en varios hilos de ejecución en paralelo. De este modo, el número de agentes a emplear de manera simultánea podría aumentar en la misma proporción dado el mismo coste.

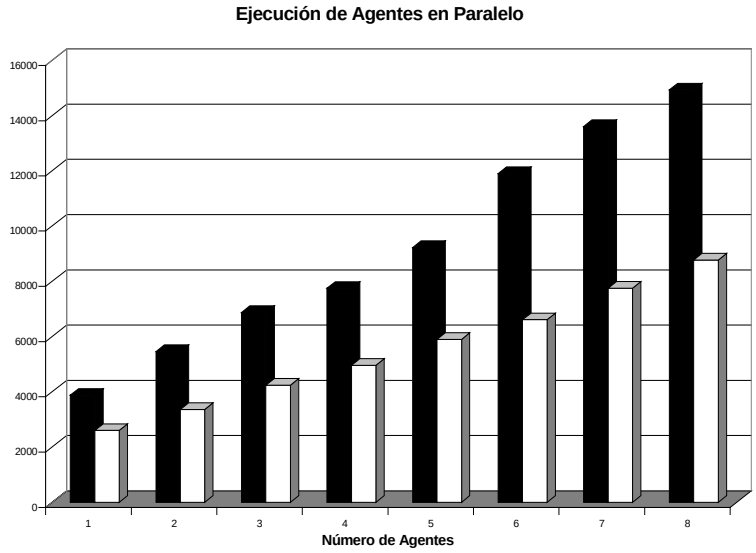


Figura 93. Disminución del tiempo de ejecución de los agentes tipo *gadea.ants.MotionAccuracy* en una sesión de carga de trabajo *moderada alta* cuando la ejecución se realiza mediante hilos en paralelo. En negro están representados el tiempo de ejecución sin hijos en paralelo. En blanco se representa el tiempo de ejecución cuando se hace uso de la multitarea.

De todos modos la tasa de actualización del modelo de usuario requerida por GADEA para alcanzar un nivel óptimo en sus procesos de adaptación no requiere en ningún caso el empleo de más de un agente de este tipo a la vez. Ni tan siquiera requiere el uso continuo de estos agentes en una secuencia consecutiva de interacciones, pues solo es requerida la captura de información de forma periódica. Esta estrategia de actualización por incrementos se realiza en base a promedios, es decir, en función de las actuaciones previas del individuo observado, detectando pequeños cambios más que modificaciones bruscas de comportamiento. De cualquier modo, la pérdida puntual de un tope máximo del 9,5% de rendimiento resulta un coste barato si se toma en cuenta los enormes beneficios obtenidos en términos de usabilidad.

20

20 Usabilidad

Podemos ejecutar un gran test, obtener montones de datos pero si los participantes en las pruebas no son representativos del grupo de usuarios de nuestro producto ¿qué hemos ganado?

George Flanagan

20.1 Calidad de la Navegación a Nivel Semántico

Tanto en la *World Wide Web* como en una aplicación multimedia, los usuarios esperan mecanismos y metáforas de navegación efectivos y fáciles de usar. Los usuarios necesitan determinar lo que hay en el entorno y como acceder a ello. Un buen sistema de navegación es de importancia crítica para el éxito de un producto, tanto o más que la utilidad que éste tiene o la potencia de su sistema de hipertexto. Desgraciadamente, el proceso de navegación no es una tarea sencilla: los usuarios se pierden y un proceso de búsqueda erróneo crea frustración [Furnas (1997)]. Sin embargo, un buen diseño del proceso de navegación es capaz de proporcionar nuevas formas de navegar, así como un mejorar el acceso a la información [Nielsen (1993a), p. 125-143].

Durante la fase de análisis a nivel semántico de la interfaz de aplicación, los diseñadores han de determinar los objetivos a alcanzar, es decir, deben determinar la cantidad mínima de información que los usuarios finales obtendrán al emplear la herramienta. La información relevante para estos objetivos es entonces recabada y organizada en categorías primarias, que proporcionarán el primer borrador de la estructura del conocimiento.

La siguiente tarea (fase de diseño) se ocupa de simplificar la estructura del conocimiento obtenida, creando pequeñas unidades por medio de una estrategia de *tipo divide y vencerás*. A nivel semántico, cada unidad representará un nodo conceptual en el modelo de navegación final. Al principio de esta fase de diseño, todo el conocimiento a transmitir se concentra en una única unidad. Esta unidad se irá dividiendo en diferentes unidades más sencillas a lo largo de todo el proceso de desarrollo. Este proceso, de carácter evidentemente recursivo, proporcionará la lista final de los nodos de la interfaz del producto para todos sus procesos de usuario. A nivel conceptual que no funcional, cada uno de estos nodo representará la unidad de conocimiento mínima de la aplicación y podrá ser considerada como una fuente autónoma de información [Nielsen (1993a), p. 101-106]. Los nodos así obtenidos son encadenados entre sí por medio de enlaces, creando una estructura en forma de grafo dirigido, conocida generalmente como modelo de navegación según la terminología empleada por algunos enfoque metodológicos para el desarrollo de artefactos cognitivos navegables como es el caso de [OOHDM Schwabe et. al. (1996)].

Al final de la fase de diseño, es frecuente considerar el empleo de mas de un mapa de navegación dependiendo de los distintos enfoques empleados para el estructurar la base de conocimiento [González 1999a]). La complejidad de este proceso de diseño aumenta más incluso si consideramos que la cantidad y tipología de los modelos de navegación obtenidos dependen no solo de la base de conocimiento, sino también del tipo de usuario destinatario de la información. La naturaleza de cierto tipo de aplicaciones, abiertos a una muy amplia variedad de usuarios distintos, como por ejemplo, los quioscos multimedia o los sitios web corporativos, implica como vimos en el capítulo 10 (*El Lenguaje*) la necesidad de un diseño que abarque las expectativas de diferentes entornos culturales

En ocasiones, un único modelo de navegación es incapaz de satisfacer las necesidades de los usuarios, de tal modo que es necesario incluir varios en una misma aplicación. Algunos enfoques metodológicos como OOHDM, permiten el diseño de distintos modelos de navegación a partir de una única base de conocimiento [Schwabe (1995)]. Este objetivo se alcanza mediante el diseño de distintos modelos orientados a las necesidades concretas de los distintos tipos de usuarios finales y depende obviamente de procesos de adaptación semánticos que exceden el marco de estudio de la presente investigación. En el ejemplo de la Figura 94, la base de conocimiento identificada durante la fase de análisis puede ser dividida en dos o más nodos distintos durante la fase de diseño. Aunque la base de

conocimiento es la misma para los dos nodos (novato y experto), la cantidad y tipo de información facilitada por cada uno es distinta. Mientras que los usuarios novatos reciben información general, los usuarios expertos requieren por lo general de una colección de datos más precisos. Sin embargo, este no tiene porque ser un modelo estático, ya que los usuarios pueden cambiar dinámicamente de un tipo a otro por medio de sistemas de evaluación.

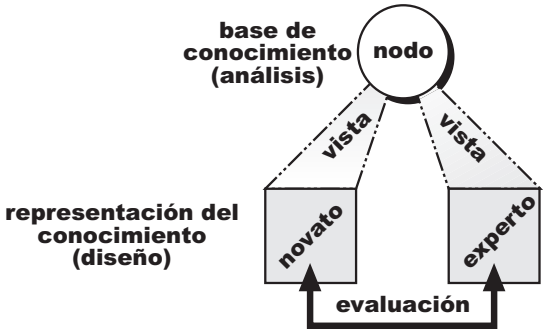


Figura 94. Adaptación semántica de los contenidos del nodo de un proceso de usuario en función de dos tipos de usuario distintos.

Dado que las estrategias de navegación se basan en consideraciones de orden topológico con respecto a la base de conocimiento [Jul et. al. (1997)], la orientación del usuario durante el proceso de navegación implica alinear su propia representación espacial con la representación mental del diseñador. La representación mental del usuario está basada en su conocimiento previo del tema, su experiencia personal y las distintas visiones que tiene de la estructura del modelo de navegación. Obviamente, una relación estrecha entre las dos representaciones mentales (la del usuario y la del diseñador) es crítica para la navegación, dado que la estructura del conocimiento va a determinar como se mueve el usuario a lo largo de los modelos de navegación correspondientes a los procesos de usuario de la aplicación. Más importante aún será el hecho de que esta relación determina también como se extrae conocimiento del producto.

20.1.1. Prueba Clásica de un Modelo de Navegación

El objetivo principal de la fase de pruebas del proceso de desarrollo de una interfaz de navegación a nivel semántico consiste en medir la calidad de los modelos de navegación definidos durante la fase de diseño. Los diseñadores necesitan saber cual de los modelos de navegación diseñados se adapta mejor a las necesidades del usuario, determinado saber que tan útil puede resultar un modelo de navegación para un determinado tipo de audiencia.

Desgraciadamente, por el momento no existe una forma sencilla de predecir como navegarán los usuarios por el producto, ya que –a nivel semántico– existen demasiados factores cognitivos involucrados. La navegación es un proceso donde las decisiones son tomadas continuamente, eligiendo entre las distintas estrategias que conducen al objetivo final. Este proceso requiere la evaluación continua del entorno

para determinar si el objetivo inicial ha sido alcanzado. Estas decisiones siguen en ocasiones un plan y en otras pueden ser respuestas al propio entorno [Jul et al (1997)]. De este modo, un pequeño cambio en la estructura de navegación puede conducir a la aplicación de una estrategia de navegación completamente distinta. Como resultado, los diseñadores han de exponer sus modelos de navegación al escrutinio de los usuarios reales.

La exposición de estos modelos de navegación suele ser realizada mediante técnicas de diseño y construcción evolutiva de prototipos, bajo la cual se desarrollan versiones preliminares del modelo de navegación en estudio para verificar su facilidad de uso [Apple (1992)]. Una vez que el prototipo está listo, éste puede ser verificado mediante pruebas de usabilidad en las que se observa y se escucha cuidadosamente a los usuarios mientras estos trabajan con el prototipo en la realización de una serie de tareas predeterminadas.

El principal objetivo de estas pruebas de usabilidad es obtener tanta información como sea posible acerca de cómo se desplaza el usuario a través del conjunto de modelos de navegación proporcionados. Naturalmente, esta información debe contrastarse con lo que los diseñadores esperaban encontrar, con el objeto de cuantificar la calidad de los diseños.

Las pruebas de usabilidad son una técnica de utilidad probada en la mejora del comportamiento de las interfaces de usuario de muchas aplicaciones y dispositivos hardware. Estimando los costes asociados con la ingeniería de usabilidad, un estudio determinó que los beneficios obtenidos pueden superar hasta en 5.000 veces los costes [Nielsen 1993)]. Las pruebas de usabilidad, son también una técnica potente a la hora de descubrir el comportamiento oculto del usuario e identificar metáforas internas de navegación se trabaja con la interfaz de algunas máquinas [Shulz et al. (1997)].

Sin embargo, la adaptación de las pruebas de usabilidad clásicas al proceso de prueba de la navegación tiene algunas desventajas que pueden convertir este proceso en una serie de tareas lentas, caras y poco precisas, desventajas que pasamos a citar a continuación.

- Las pruebas de navegación requieren de mucha precisión, así como de la observación permanente y continua de los voluntarios que participan en las mismas. Como ésta no es una tarea sencilla, cada voluntario necesita al menos un observador con el elevado coste económico que ello conlleva.

- Debido al alto coste de las pruebas, el tamaño de las muestras en los experimentos es por lo general muy pequeño, de modo que la calidad de las pruebas disminuye considerablemente.

- Durante las pruebas, se suele elegir a los voluntarios entre gente con el mismo bagaje cultural que el usuario promedio de tal

modo que es realmente difícil descubrir nuevos tipos de usuarios para una aplicación concreta.

- Durante las pruebas de usabilidad, los voluntarios son conscientes de que son observados. Esta situación añade factores externos de carácter cognitivo al proceso de pruebas, incluyendo confusión, nervios, etc. En nuestra experiencia en el diseño de este tipo de pruebas para el programa de investigación Tirsus citado en su momento, hemos comprobado como la sola presencia de una cámara de vídeo (requerida en muchas ocasiones para registrar el comportamiento del usuario), modifica perceptiblemente la actitud de los voluntarios. Estos factores afectarán directamente al comportamiento del usuario en procesos de navegación y por lo tanto afectarán también a las conclusiones extraídas de las pruebas.

- El conocimiento que el usuario dispone acerca del experimento puede influir negativamente en su comportamiento, debido principalmente a que éste sabe lo que se espera de él –o al menos puede inferirlo– por lo que puede adaptar sus propias metáforas internas de navegación en función modelo de navegación prototipo proporcionado. Este conocimiento destruye el comportamiento espontáneo, que es precisamente el factor que los diseñadores de las pruebas están intentando obtener al realizar éstas.

- Los tests de usabilidad suelen ser realizados en ordenadores que cumplen con las especificaciones técnicas necesarias para ejecutar el la aplicación en estudio. Este puede no ser el caso del ordenador que empleará el usuario final. Esta demostrado que algunos factores técnicos, como por ejemplo grandes demoras en las respuestas proporcionadas por un sistema, pueden cambiar por completo el comportamiento del usuario. Algunos autores [Johnson (1997), Johnson y Gray (1996), Thomas (1996), O'Donnell y Draper (1996) y Byrne y Picking (1997)] han puesto de manifiesto las distintas estrategias aplicadas por los usuarios cuando éstos obtienen datos desde servidores web colapsados. En estos casos, los usuarios suelen incrementar la memoria caché del explorador de Internet, suelen abrir más de una sesión de navegación, suelen grabar todo el sitio web en disco para un análisis fuera de línea, o incluso peor, algunos usuarios se dedican a seleccionar de forma compulsiva en cada enlace disponible con el objeto de mantener la máquina ocupada descargando datos de Internet, sin importarles demasiado la carga semántica de dicho enlace. En las pruebas de usabilidad clásicas, este tipo de efectos no suele ser detectado.

- Como las pruebas de usabilidad suelen tener lugar detrás de los muros de un laboratorio, los diseñadores pierden el papel que el entorno del entorno de trabajo del usuario juega en el comportamiento del mismo a la hora de navegar por los distintos nodos de un proceso de usuario.

20.1.2. Pruebas Remotas de Navegación

Si se desean evitar las desventajas comentadas, las pruebas de usabilidad deberían realizarse bajo las mismas condiciones bajo las cuales se va a utilizar el producto final, esto es, probando los prototipos en el mismo entorno de computación del usuario y sin la presencia física de los diseñadores. En esta investigación, a este tipo de pruebas la hemos bautizado como *Pruebas Remotas de Navegación* o *Remote Navigability Testing* (RNT) [González (1999b)].

En esta situación ideal, los voluntarios en las pruebas ignoran la naturaleza del experimento, incluso el papel que juegan en él. Los voluntarios se sienten libres para explorar los modelos de navegación proporcionados, ya que no se encuentran bajo presión (no hay diseñadores observándoles), de tal modo que los factores externos del tipo del nerviosismo, la confusión, etc., no afectarán a los resultados obtenidos. Como las pruebas de navegación se realizan en el entorno de trabajo del usuario, ahora es posible obtener información acerca de cómo dicho entorno afecta a la navegación. Gracias a las RNT, los cambios en el comportamiento del usuario debidos a demoras en la respuesta del sistema pueden ser detectados corregidos mediante el diseño de nuevos modelos de navegación adaptados al sistema de computación del usuario, distribuyendo adecuadamente la carga de trabajo asignada a su máquina.

Esta situación ideal permite también detectar nuevos tipos de usuario. Nótese como con las técnicas de tests de usabilidad clásicas es el usuario el que viene al laboratorio para probar el software. Con las pruebas de navegación remotas es el software el que va a la casa del usuario para ser probado. De este modo, una vez que el prototipo esta listo, puede ser distribuido de forma gratuita como una versión de demostración del producto, pudiendo ser probado por cualquier posible usuario interesado y no sólo por el mal llamado usuario medio de la aplicación.

Como se puede apreciar, el módulo ANTS es ideal para la realización de este tipo de pruebas, permitiendo así una doble funcionalidad. Por un lado, el módulo permite la actualización y adquisición automática a nivel sintáctico de la información almacenada en el modelo de usuario individual y el UAM, mientras que por otro lado, el mismo módulo puede servir como una herramienta muy eficaz para las pruebas a nivel semántico de los modelos de navegación diseñados para la interfaz de cualquier tipo de aplicación, tanto si cumple con los requisitos de compatibilidad de GADEA como si no.

Dado que en las pruebas de segundo tipo los observadores no pueden estar presentes durante las sesiones de navegación del usuario,

el módulo ANTS facilita la tarea de experimentación remota permitiendo a los investigadores concentrar sus esfuerzos en la realización de una sola tarea (la observación de la navegación) garantizando además que las pruebas de usabilidad se puedan realizar en situaciones que no se encuentran bajo el control del observador. Para ello, ANTS se diseñó con la facultad de poder de observar a los usuarios mientras estos navegan a lo largo de los modelos de navegación, registrando su comportamiento en el servidor (hormiguero) para un análisis posterior. Esta observación se realiza de modo totalmente silencioso ya que los usuarios son incapaces de detectar el modo en el que su comportamiento es observado.

Dado que se espera una carga de comunicación moderada el módulo ANTS es perfectamente escalable, siendo capaz realizar pruebas automáticas de navegación sobre varios modelos de navegación distintos de manera simultánea. Estas pruebas remotas de navegación se realizan de forma totalmente independiente con respecto a la ubicación física del dispositivo a probar, ya que el proceso de observación puede establecerse bien sobre dispositivos locales (para realizar las pruebas de usabilidad clásicas) o bien sobre dispositivos remotos (para dar soporte a las pruebas remotas). Además, como ya se ha comentado en el capítulo 19 (*Espías*), ANTS puede integrarse fácilmente con cualquier tipo de dispositivo disponible, incluyendo aplicaciones hipermedia, quioscos multimedia y sitios web.

La información recogida habitualmente durante las pruebas de navegación consiste principalmente en la obtención de los puntos de referencia, rutas y mapas mentales utilizados por los usuarios en su proceso de extracción de información de la base de conocimiento de una aplicación [Jul et al. (1997)]. Los puntos representan perceptiva y conceptualmente, los distintos nodos de un proceso de usuario. El conocimiento de las rutas representa la concepción del entorno descrito en términos de las conexiones entre los nodos o puntos de referencia. Finalmente, el conocimiento extraído de los mapas describe las relaciones entre los nodos, proporcionando información acerca de la representación mental que el usuario tiene del modelo de navegación.

Dependiendo de la información obtenida, es posible detectar el tipo de estrategia de navegación empleada por el usuario, la cual puede ser del tipo de navegación por ubicación o de navegación basada en planes predeterminados [Jul et al. (1997)]. Para la primera estrategia, el usuario navega empleando conocimiento específico del entorno y de los hitos; haciendo uso de información incompleta. Esta estrategia de tanteo, es empleada cuando el objetivo buscado parece accesible y/o cercano. La segunda estrategia hace uso del conocimiento de los mapas para obtener por adelantado un plan completo que permita alcanzar el lugar deseado. Las diferencias individuales determinan la estrategia a emplear, ya que la selección de ésta depende de factores tales como la experiencia del navegante, el conocimiento que posee acerca del modelo de navegación o sus habilidades de orientación espacial.

Para obtener los puntos de referencia, las rutas, los mapas y la estrategia de navegación empleada por los voluntarios de una prueba remota de navegación, ANTS es capaz de recoger la siguiente

información por usuario, sintetizándola en un informe destinado al grupo de evaluación de una aplicación.

- Hora de llegada a un nodo, así como la hora de salida del mismo.
- Lista de los nodos más visitados de un modelo de navegación.
- Conjunto de destinos más populares a partir de un nodo.
- Relación temporal de visitas a los nodos más populares.
- Tiempo de permanencia de un usuario en un nodo antes de moverse a otro, o lo que es lo mismo, cuánto tiempo requiere un usuario para obtener la información que busca dentro de un nodo.
- Conjunto de los nodos de información visitados durante una sesión de navegación por cada usuario, incluyendo su orden, es decir, la ruta seguida por el usuario para extraer la información de la base de conocimiento de una aplicación a través de su modelo de navegación.

20.2 Evaluación de la Técnica

Las técnicas de pruebas remotas de navegación requieren que los prototipos que contienen los modelos de navegación bajo prueba se distribuyan libremente entre los voluntarios, de tal modo que éstos puedan explorarlos usando su propio equipo informático. De este modo se garantiza que las pruebas se realizan sobre el entorno real del usuario. En función del tipo de aplicación a probar, los diseñadores deben distribuir diferentes versiones de los modelos de navegación:

- Para aquellos usuarios sin acceso a Internet, el prototipo de modelo de navegación debe ser distribuido con una copia funcional y operativa del hormiguero. En este caso, el servidor debe arrancarse automáticamente cada vez que el usuario ejecuta una copia del prototipo y detenerse cuando éste abandone la aplicación, guardando la información recolectada en disco para un análisis posterior.
- Cuando se esté probando un kiosco multimedia, el prototipo debe ser instalado en el mismo pasillo u habitación en donde se ejecutará la versión final del proyecto. Este detalle es muy

importante de cara a poder cuantificar la posible influencia del entorno de computación sobre la navegación.

- Si el modelo de navegación a ser probado se corresponde con el de un sitio web, la única acción ha ser realizada por los investigadores consiste en su instalación en el correspondiente servidor de páginas web.

Hay que destacar que para utilizar esta técnica como para emplear GADEA, los voluntarios deben ser advertidos de su participación en una prueba de usabilidad desde el mismo inicio del experimento. El único factor que deben ignorar es la técnica de observación que los investigadores van a emplear. Esta es una práctica común en los estudios de mercado en donde los investigadores observan el comportamiento de los posibles compradores por medio de cámaras de vídeo ocultas [Masson y Wellhoff (1990)]. Esta técnica es muy similar a la empleada en los tests de personalidad en donde los psicólogos están más interesados en la actitud de los voluntarios mientras resuelven el test, que en el propio resultado del mismo. Otras herramientas para pruebas de usabilidad de interfaces de usuario [Ergolith (1999)] emplean técnicas similares.

El informe recogido en el Código 36 ha sido obtenido empleando ANTS para probar el modelo de navegación del sitio web de la Historia de Asturias [Arqueoastur (2001)] como parte del programa de investigación Tirsus. Como se puede apreciar, el usuario protagonista de la sesión de navegación llegó al sitio web a las 19:35:22 horas (hora local del servidor) empleando un ordenador localizado dentro de la zona horaria ECT. El código empleado para la sesión de navegación se corresponde con la dirección IP empleada por su explorador de Internet (212.89.19.216), así como el nombre de la máquina del cliente (petren.telecable.es).

Código 36. Sesión de navegación de un usuario anónimo recogida por ANTS en un portal web.

```
DATE: October 14Th 1999 (Thursday)
TIME: 19H35' GMT+00:00

IP ADDRESS: 212.89.19.216 (petren.telecable.es)
TIME ZONE: ECT (-2)

LOGS: (All dates are local to server)

19:35:22 - 19:38:42 [00:03:20]-> User visits <Portal>
6 seconds out.
19:38:48 - 19:40:01 [00:01:13]-> User visits <History>
8 seconds out.
19:40:09 - 19:45:22 [00:05:13]-> User visits <Roman Army>
22 seconds out.
19:45:44 - 19:46:03 [00:00:19]-> User visits <Roman Cities>
19:46:03 -> Log out.

Nodes Visited: 4.
Different nodes visited: 4.

Total Time: 00:10:41 seconds. AVG: 00:02:40 seconds.
Trans Time: 00:00:36 seconds. AVG: 00:00:09 seconds.
Real Time:  00:10:05 seconds. AVG: 00:02:31 seconds.
```

Como se puede observar, el comportamiento del usuario, así como el tiempo empleado en cada nodo está perfectamente registrado. En el ejemplo, el usuario llega al nodo *Portal* a las 19:35:22 y lo abandona a las 19:38:42, de tal modo que el usuario permanece en este nodo por espacio de 3 minutos y 20 segundos. A continuación el usuario se mueve al nodo *History*, llegando a él a las 19:38:48 (la hormiga asociada al nodo *History* ha empleado 6 segundos en ser descargada desde el hormiguero y establecer una conexión con su asistente-hormiga). Por último, el informe muestra algunas estadísticas acerca del tiempo empleado por el usuario en todo el modelo de navegación. En el ejemplo, el usuario visitó 4 nodos distintos, empleando un tiempo promedio de 2 minutos y 31 segundos en cada uno de ellos.

Para mediar la potencia y flexibilidad de la técnica de pruebas de usabilidad propuesta y probar la solidez construcción actual de ANTS, se ha probado la herramienta con numerosos problemas de la vida real [González (1999), González et al (2000f) y González (2000i)] de los que ya se ha citado uno en el capítulo 3 (*La Individualidad*) de este documento y se va a proceder a citar uno más, el cual refleja con claridad la capacidad que tiene este sistema para mejorar la calidad de la interfaz de usuario de un producto software a nivel semántico. Este ejemplo trata de cómo se pueden emplear pruebas de navegación remotas para mejorar la usabilidad de web destinada a la educación, la cual ésta dotada de mecanismos diseñados para dar un soporte internacional a sus contenidos.

Como ya se ha comentado en su debido momento, parte de las técnicas de navegación empleadas por DEVA han sido probadas por medio del programa de investigación Tirsus. Este programa comprende el desarrollo de una base de conocimiento hipermedia para la enseñanza de historia antigua y prehistoria para el Museo Arqueológico de Asturias. El producto final incluirá un CD-ROM, un kiosco multimedia (que proporcionará información a los visitantes en cada una de las salas del museo) y el sitio web del propio museo [Arqueoastur (2001)].

Las estadísticas recopiladas por el propio museo desde 1997 nos dicen que éste recibe muchos visitantes provenientes de distintas regiones de España, así como de varios países europeos. Nuestro primer intento ha sido naturalmente proporcionar a los usuarios del sitio web del museo toda aquella información requerida en un formato internacional, de tal modo que los contenidos iniciales de los modelos de navegación fueron escritos en menos tres idiomas: ingles (para acceso internacional), castellano (para acceso nacional) y asturiano (para acceso local). En el primer prototipo desarrollado, la selección del idioma tenía que ser realizada desde el punto de entrada del mapa de navegación (el portal del sitio web). Sin embargo, una vez pasado este punto y a lo largo de toda sesión de navegación los usuarios tenían libertad total para cambiar el idioma de cada nodo pulsando en el selector de idioma situado en la esquina superior derecha de cada página (ver Figura 95). Este diseño flexible permitía a los usuarios

explorar la información proporcionada por cada nodo en varios idiomas.

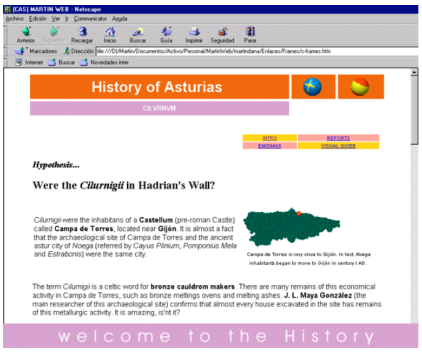


Figura 95. Ejemplo de una de las páginas web del prototipo de portal para el Museo Arqueológico de Asturias. Aunque esta página está escrita en inglés, los usuarios pueden cambiar de idioma pulsando en los botones situados en la esquina superior derecha (consultar versión en color en el capítulo 24: *Imágenes en Color*, p. 445).

Comenzamos nuestro trabajo desarrollando algunos prototipos para los mapas de navegación e instalándolos en un sitio web provisional. Después de promocionar adecuadamente el sitio web, se procedió a probar los modelos de navegación con ANTS, instalando el hormiguero en un obsoleto PC i486 con 16 Mb de RAM, ejecutándose bajo Linux y Java 1.1.6. Uno de los principales objetivos planteados al probar la navegación por el sitio web era obtener una idea clara de las preferencias de idioma de sus visitantes con el objeto evidente de conocer si había mercado suficiente como para desarrollar el producto en los idiomas propuestos. Se esperaba que el experimento contribuyese a determinar cuantos recursos se debían asignar al desarrollo de los contenidos de la web bajo soporte internacional.

Después de tres semanas de pruebas, ANTS registró 518 sesiones de navegación. En ellas se detectó el uso del castellano en 454, el asturiano en 95 sesiones y el inglés en tan solo 41 sesiones. Los resultados obtenidos podrían inducir a concentrar los esfuerzos de desarrollo en la construcción de versiones en castellano y en asturiano, dado que los dos idiomas parecen ser bastante populares entre los visitantes del portal. Obviamente, esta evidencia es falsa.

Dado que el número final de sesiones de navegación empleando cada idioma no coincide con el número final de sesiones de navegación registradas, es obvio que muchos usuarios seleccionaron más de un idioma durante sus sesiones de navegación. Después de un análisis cuidadoso de los informes generados por ANTS, se detectó la aplicación de esta estrategia de navegación en 72 sesiones de navegación (13,89%). En estas sesiones, los usuarios emplearon una estrategia de navegación similar a la incluida en el Código 37. En la sesión de navegación anterior, el usuario seleccionó el idioma castellano nada más llegar a la página *Portal*. Este idioma fue empleado también para visitar la página *Romans*. Después de un rato, el usuario decidió cambiar el idioma de la página, pasando al asturiano, empleando a continuación 36 segundos en la posible lectura de la página en el nuevo idioma. A continuación, el

usuario selecciona el castellano de nuevo el cual es empleado como idioma por defecto para el resto de la sesión de navegación (páginas *Romans* y *Bronze_Age*).

Código 37. Patrón de navegación haciendo uso de varios idiomas similar al empleado en el 13,89% de los casos registrados por ANTS al evaluar un sitio web con soporte internacional.

```
DATE: September 6Th 1999 (Tuesday)
TIME: 11H20' GMT+00:00

IP ADDRESS: 135.35.15.32 (teleline.es)
TIME ZONE: ECT (-2)

LOGS: (All dates are local to server)

11:20:15 - 11:20:52 [00:00:37]-> User visits <Portal>
8 seconds out.
11:20:52 - 11:24:02 [00:03:10]-> User visits <spanish-Romans>
7 seconds out.
11:24:02 - 11:24:38 [00:00:36]-> User visits <asturian-Romans>
2 seconds out.
11:24:38 - 11:25:49 [00:01:11]-> User visits <spanish-Romans>
5 seconds out.
11:25:49 - 11:31:54 [00:06:05]-> User visits <spanish-Bronze_Age>
11:31:54 -> Log out.

Nodes Visited: 5.
Different nodes visited: 4.

Total Time: 00:12:01 seconds. AVG: 00:03:00 seconds.
Trans Time: 00:00:22 seconds. AVG: 00:00:05 seconds.
Real Time:  00:11:39 seconds. AVG: 00:02:54 seconds.
```

La razón para esta estrategia de navegación puede basarse en la propia curiosidad del usuario. Dado que el usuario tiene la oportunidad de echar un vistazo a una página escrita en otro idioma, aprovecha la ocasión para conocer la sintaxis y gramática del mencionado idioma. Solo en dos de estas sesiones híbridas (sesiones que emplean más de un lenguaje), el usuario decidió continuar con la navegación en el segundo lenguaje seleccionado. En el resto de estas sesiones híbridas (97,23%), el usuario volvió al idioma inicial tras un breve lapso de tiempo.

Una muy alta proporción de las sesiones híbridas tiene al asturiano como segundo idioma implicado. 54 de las sesiones comenzadas en castellano emplearon más tarde el asturiano. En 12 de las sesiones iniciadas en inglés ocurrió lo mismo. Solo en 5 de las sesiones comenzadas en inglés, los usuarios cambiaron al castellano y solo en 1 de las sesiones iniciadas en castellano se cambió al inglés. En ninguna de las sesiones iniciadas en asturiano se cambió a otro idioma. Es muy posible que la popularidad del idioma asturiano en las sesiones híbridas se deba a que muy poca gente lo conoce, incluso en España, luego es normal que despierte cierta curiosidad entre los no asturiano parlantes.

Si no se toman en cuenta las 72 sesiones híbridas reportadas por ANTS, se obtiene 446 sesiones válidas, 394 en castellano (88,34%), 29 en asturiano (6,5%) y 23 en inglés (5,16%). De las 95 sesiones de navegación que empleaban el idioma asturiano, solo en 29 de ellas se hacía un uso racional del mismo. Desde un punto de vista de la administración de recursos y considerando únicamente factores

económicos, la conclusión evidente que se extrae de este experimento es que se deberían centrar los esfuerzos de desarrollo única y exclusivamente en la versión en castellano del producto, ya que tanto los idiomas inglés y asturiano son usados por una muy baja proporción de los visitantes del sitio web.

Otra importante conclusión que se puede extraer de los resultados obtenidos en este experimento es que se debería eliminar la posibilidad que tienen los usuarios de poder cambiar de idioma en cada nodo, ya que esta característica no proporciona ninguna ventaja adicional a la navegación y distrae la atención del usuario del objetivo real de la aplicación: proporcionar información acerca de la Historia. Este es otro ejemplo en donde el viejo principio de diseño *menos es más* se aplica a la perfección.

20.2.1. Ventajas Adicionales

Las técnicas de prueba de navegación remota mejoran notablemente la calidad y la productividad de la fase de pruebas de la interfaz de usuario de una aplicación a un nivel semántico. Este enfoque elimina factores externos (como la confusión o los nervios), captura información adicional (influencia del entorno de computación en la navegación), obtiene información precisa de forma automática y abarata los costes de la fase de pruebas.

Al emplear herramientas de prueba automática y remota del tipo de ANTS, los datos fluyen desde su origen (la sesión de navegación) hasta el sistema de almacenamiento en el servidor. Dado que el proceso de navegación tiene lugar en el propio entorno de computación del usuario, no hay necesidad por tanto de asignar costosos recursos de laboratorio para la fase de pruebas. Los diseñadores se liberan de la aburrida tarea de recolectar información, de modo que ahora pueden concentrar sus esfuerzos en analizar los resultados obtenidos, mejorando la calidad de sus mapas de navegación. Dado que no es necesario asignar recursos humanos para la fase de recolección de datos, el proceso de pruebas de usabilidad se abarata.

Por medio de esta técnica es factible registrar un número mayor de sesiones de navegación con el mismo número de recursos humanos. Ahora es posible incrementar el número de voluntarios que participan en las pruebas de usabilidad orientadas a la navegación, de tal modo que al aumentar el tamaño de las muestras, la calidad de las pruebas aumenta en la misma medida. En los experimentos referidos en este documento, se han analizado un total de 860 sesiones de navegación (518 en el portal web y 342 en el experimento de la barra de navegación). Estas cifras hubiesen sido completamente imposibles de alcanzar por medio de sesiones de usabilidad clásicas, dado el número limitado de recursos humanos y de hardware del que se dispuso a lo largo de todo el proceso de investigación. Sin embargo, por medio de las técnicas RNT, se ha podido afrontar la realización de experimentos de tal envergadura con la única ayuda de un PC completamente obsoleto.

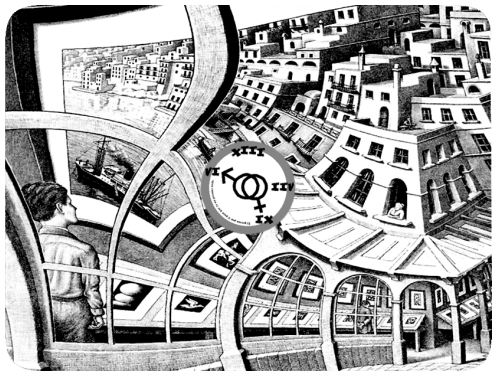
Otra ventaja importante de las pruebas de navegación remotas radica en que la fase de pruebas se puede extender a toda la vida útil del producto, ya que una vez que la versión final de éste haya sido terminada y publicada, es posible obtener información acerca del comportamiento de los usuarios reales de forma automática. De este modo la información recogida puede ser empleada para mejorar la calidad de la próxima versión del producto. Obviamente, para ello es necesario el consentimiento de los usuarios finales de la aplicación, de modo que los mecanismos de transmisión de información puedan ser activados en sus máquinas.

Es necesario destacar que las técnicas RNT no pretenden substituir a las pruebas de usabilidad clásicas dado que su intención no es otra que la de optimizar el funcionamiento dichas pruebas cuando se aplican a ciertas tareas. Las pruebas remotas son capaces de detectar acciones pero no pueden realizar ninguna inferencia a nivel semántico acerca de como realizadas éstas. Por ejemplo, con RNT resulta muy sencillo registrar el número de veces que un botón ha sido pulsado durante una sesión de navegación, pero es completamente imposible conocer las dificultades que el usuario pudo tener para hacerlo. Esta es una tarea en donde las pruebas de usabilidad clásicas parecen invencibles.

Las pruebas remotas funcionan bien cuando las diferentes características de un producto ha ser probadas pueden ser cuantificadas y las tareas ha ser realizadas por los voluntarios son simples y sencillas. Esta técnica no debería ser aplicada para medir factores subjetivos o para evaluar tareas o modos de interacción complejos.

Una ventaja adicional de realizar las pruebas de usabilidad por medio de ANTS radica en que dicha herramienta puede actuar independientemente como un tipo especial de GPS (*Global Positioning System*) para la navegación en universos electrónicos. Dado que el hormiguero conoce la posición exacta de un usuario cualquiera dentro un modelo de navegación, se puede emplear este conocimiento para ayudar al usuario a encontrar el camino deseado dentro del modelo (de modo que los usuarios puedan hacer uso de la estrategia de navegación por planes predeterminados). La versión actual de ANTS dispone de objetos GPS que pueden ser empleados por cualquier otro objeto escrito en Java para proporcionar ayuda a los usuarios en sus sesiones de navegación.

Esta característica de ANTS ha sido empleada con éxito en los prototipos desarrollados para varios sitios web, incluyendo el del Museo Arqueológico de Asturias . Para ello, se ha incluido un *banner* como parte de las herramientas de navegación de las páginas web. En el caso del sitio web del Museo Arqueológico de Asturias, el *banner* se ubica en la parte inferior del espacio visual proporcionado por el explorador (ver el mensaje *Welcome to the History* en la Figura 95), el cual permanece todo el tiempo visible. El *banner* (desarrollado como un *applet* de Java) incluye un objeto GPS (conectado al hormiguero) el cual proporciona información al *banner* acerca de la posición exacta del usuario dentro del modelo de navegación del sitio web, de tal modo que la información mostrada por el *banner* varía de acuerdo con el nodo en el que se encuentra el usuario.



Galería de Grabados © 1956 por M. C. Escher

P A R T E V I

Conclusiones

21 Un Nuevo Modelo

Modelo de desarrollo de interfaces centrado en el usuario. Relación de tareas implicadas. Evaluación del modelo GADEA con respecto al modelo tradicional: un caso de estudio. Soporte a la especificación formal de interfaces. CodeX como generador de documentación. Construcción y evaluación rápida de prototipos de modelos de interacción con GADEA.

p. 395

22 La Adaptabilidad

Diseño de las pruebas de usabilidad para GADEA. El Entruger como herramienta de recogida de información. Evaluación del modelo de señuelos y predadores sobre aplicaciones hipermedia. Evaluación de la navegación. Mensajes subliminales y transmisión de información a través de canales

21

21 Un Nuevo Modelo

Aquella teoría que no encuentre aplicación práctica en la vida es una acrobacia del pensamiento

Swami Vivekananda

21.1 Anatomía de una Interfaz

En los últimos años, el desarrollo y mantenimiento del software de aplicación supera con creces al coste del hardware en prácticamente cualquier sistema complejo. El coste del software de un sistema de este tipo se estima en cerca del 80% del total del coste del sistema [Remington (2001)]. Gran parte de la culpa de este elevado coste se debe al desarrollo de la interfaz del software, ya que se ha estimando entre 30% y el 80% del total de las líneas de código generadas se dedican a este menester. En un estudio realizado por Myers y Rosson (1992) realizado sobre 74 proyectos de software, se determinó que el 48% de las líneas de código estaban destinadas a la creación de la interfaz de comunicación con el usuario. El promedio de tiempo dedicado al desarrollo de la interfaz en función de cada fase del ciclo de vida del proyecto, determinó que un 45% para la fase de diseño, un 50% durante la fase de desarrollo y un 37% durante la fase de mantenimiento. Ante esta magnitudes, no cabe la menor duda del fuerte impacto que sobre el

presupuesto y el tiempo de desarrollo de una aplicación tiene la selección de las herramienta y la metodología empleada para la construcción de su interfaz con el usuario.

Naturalmente, si el empleo de una técnica de desarrollo concreta presenta un fuerte impacto sobre el ciclo de vida de un producto software, más lo puede tener la selección de un sistema de gestión de interfaces de usuario de la naturaleza de GADEA. Es por ello que se hace necesario una evaluación del coste que puede representar el empleo de este producto, analizando las diferencias existentes entre este sistema y aquel al que pretende suplantar, estimando las ventajas y desventajas aportadas por cada uno.

21.1.1. Evaluación del Nuevo Modelo

Para destacar estas diferencias, se ha codificado una pequeña aplicación de ejemplo mediante el modelo de desarrollo clásico –representado por el paquete AWT de Java– así como mediante el modelo planteado por GADEA, es decir, mediante inspección automática de ficheros binarios.

La aplicación en cuestión es una pequeña calculadora que sólo realiza divisiones enteras a partir de dos elementos numéricos obtenidos del usuario, mostrando el resultado cada vez que el usuario modifica uno cualquiera de los valores (ver Figura 96). Para complicar un poco las cosas, los valores permitidos tanto para el denominador como para el numerador han de estar comprendidos dentro de cierto rango especificado por programa a través de las constantes *MIN_VALUE* y *MAX_VALUE*. Si el usuario introduce un valor que no se encuentra dentro del rango propuesto, la aplicación se lo hace saber mediante un mensaje de error y le insta a que introduzca de nuevo el valor del parámetro.

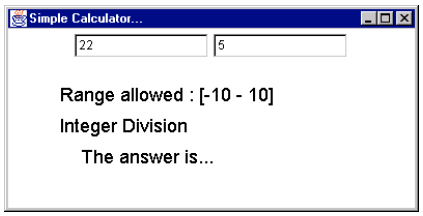


Figura 96. Interfaz de la calculadora básica empleada para evaluar el modelo de desarrollo de interfaces basado en reflexión de GADEA.

En el Código 38 se ha incluido la versión de esta calculadora haciendo uso del paradigma clásico de desarrollo propuesto por la plataforma Java [Eckel (2001)]. Tal y como se puede apreciar, hemos tenido que seleccionar de antemano el *widget* a emplear (*TextField*) para obtener del usuario los valores correspondientes al numerador y al denominador. Estos *widgets* (variables *firstNumber* y *secondNumber*) son declarados como propiedades de la clase *calculator* e iniciados en su constructor. Nótese que siguiendo este paradigma, se ha de elegir también la naturaleza perceptible de la propia clase *calculator*, la cual hereda de la clase *java.awt.Panel*, ya que la funcionalidad de un panel es

necesaria para poder situar los *widgets firstNumber* y *secondNumber* en pantalla por medio del método *add* del constructor de la clase.

Código 38. Construcción de la calculadora básica empleando el paradigma clásico de desarrollo de una interfaz de usuario.

```
import java.awt.*;
import java.awt.event.*;

/*
Simple calculator. Performs integer division.
*/

public class Calculator extends java.awt.Panel
   implements ActionListener
{
   TextField firstNumber, secondNumber= null;
   String  answer = null;

  public final static int MIN_VALUE = -10;
  public final static int MAX_VALUE = 10;

   /*******************
       CONSTRUCTOR
    *******************/

   /**************
    Calculator

   Panel with two Textfields where the user has to write two
   numbers. When the user push the <ENTER> key in any of the
   Textfields, the panel shows the result of the integer division
   between the two numbers.
   ***/

   public Calculator ()
   {
      super ();

      firstNumber = new TextField (15);
      add (firstNumber);
      firstNumber.addActionListener (this);

      secondNumber = new TextField (15);
      add (secondNumber);
      secondNumber.addActionListener (this);

     answer = "The answer is... ";
   }

  // Show the value of the <answer> variable

   public void paint (Graphics g)
   {
      Font font = new Font ("Arial", Font.PLAIN, 18);
      g.setFont (font);

      g.drawString ("Range allowed : ["+MIN_VALUE+" - "+
         MAX_VALUE+"]", 50, 70);


      g.drawString ("Integer Division ", 50, 100);
      g.drawString (answer, 70, 130);
   }

   public void calculation () throws NumberOutOfRange  {
      int firstInt = Integer.parseInt(firstNumber.getText());
      int secondInt = Integer.parseInt(secondNumber.getText());

      if ( (firstInt < MIN_VALUE) || (firstInt > MAX_VALUE) ||
         (secondInt < MIN_VALUE) || (secondInt > MAX_VALUE))
      {
         NumberOutOfRange e =
            new NumberOutOfRange ("Number out of Range");
         throw e;
      }
```

```
      answer = String.valueOf (firstInt / secondInt);
   }

   public void actionPerformed(ActionEvent e)
      throws ArithmeticException, NumberFormatException

   {
      try
      {
         calculation ();
      }
      catch (NumberFormatException p)
      {
         answer = ("Only numbers are allowed!");
      }
      catch (NumberOutOfRange n)
      {
         answer = ("Number out of range!");
      }

      repaint ();
   }

   /**********
       MAIN
    **********/

   public static void main (String args[])
   {
      Frame window = new Frame("Simple Calculator...");


      Calculator calculator = new Calculator ();

      window.add (calculator);
      window.setSize (400, 200);

      window.show ();
   }
};

public class NumberOutOfRange extends Exception
{
   public NumberOutOfRange (String msg)
   {
      super (msg);
   }
}
```

Una vez creados los *widgets* será el propio programador el encargado de seleccionar aquellos eventos que son de interés para la funcionalidad de la aplicación, así como de subscribirse a los mismos (método *addActionListener* del constructor) y de gestionarlos (método *actionPerformed*). Del mismo modo, será el programador el encargado de diseñar los métodos adecuados para la correcta visualización de la información a través del método *paint*, definiendo la fuente tipográfica a emplear, sino también su tamaño, su apariencia y la posición que ocuparán todos y cada uno de los objetos gráficos que intervienen en el proceso de comunicación.

Cuando el usuario introduce uno cualquiera de los parámetros en juego dentro de la operación de división, es la aplicación la encargada de validar esos parámetros comprobando que los valores introducidos se corresponden con valores numéricos (método *actionPerformed*) y que éstos se encuentran dentro del rango especificado por MIN_VALUE y MAX_VALUE (método *calculation*). En el caso en el que el valor introducido se encuentre fuera de rango, la aplicación genera una

excepción de control, la cual es empleada para mostrar el tipo de error producido al usuario. En caso contrario, el resultado de la división entera es calculado extrayendo los valores de los *widgets firstNumber* y *secondNumber* y convirtiéndolos a valores enteros –desde su condición original como cadenas de caracteres– para su despliegue manual a través del método *paint*.

En el Código 39 se puede ver la versión GADEA del código necesario para implementar una calculadora similar. La primera diferencia apreciable se encuentra en el tipo de datos empleados para recoger la información aportada por el usuario, ya que en esta segunda versión se emplean datos numéricos directamente en lugar de *widgets* (cadenas de tipo *gadea.datatypes.String*: *firstNumber* y *secondNumber*. Hay que recordar que será el propio sistema de gestión de interfaces quien seleccione los componentes adecuados en tiempo real teniendo en cuenta las características cognitivas, perceptivas y motrices del usuario receptor de la información.

Código 39. Construcción de la calculadora básica por medio del paradigma de desarrollo propuesto por GADEA.

```
import gadea.datatypes.*;

public class GADEACalculator
{
   protected PropertyChangeSupport changeAgent = null;

   gadea.datatypes.Integer firstNumber, secondNumber = null;
   gadea.datatypes.String answer = null;

   public final static int MIN_VALUE = -10;
   public final static int MAX_VALUE = 10;

   UIMS interface = null;

   // Constructor
   GADEACalculator (UIMS myInterface)
   {
      interface = myInterface;

      firstNumber = new gadea.datatypes.Integer (0, false);
      firstNumber.setRange (MIN_VALUE, MAX_VALUE);
      secondNumber = new gadea.datatypes.Integer (0, false);
      secondNumber.setRange (MIN_VALUE, MAX_VALUE);

      (interface.getDefaultCommunicationChannel()).registerProperty
         (firstNumber, "firstNumber", this);

      (interface.getDefaultCommunicationChannel()).registerProperty
         (secondNumber, "secondNumber", this);

      // Answer
      (interface.getDefaultCommunicationChannel()).add
         (new gadea.datatypes.String ("Range allowed :
         ["+MIN_VALUE+" - "+ MAX_VALUE+"]", true));

      (interface.getDefaultCommunicationChannel()).add
         (new gadea.datatypes.String ("Integer Division", true));

      answer = new gadea.datatypes.String ("The answer is...",
      false);

      (interface.getDefaultCommunicationChannel()).registerProperty
         (answer, "answer", this);

      // agent for events broadcasting.
      changeAgent = new PropertyChangeSupport (this);
   }
```

```
   // Invoked by GADEA when the user modifies the <fistNumber>'s
   value
   public void setfirstNumber (myValue Object)
   {
      firstNumber = (gadea.datatypes.Integer) myValue;
      recalculate();
   }

   // Invoked by GADEA when the user modifies the <secondNumber>'s
   value
   public void setsecondNumber (myValue Object)
   {
      secondNumber = (gadea.datatypes.Integer) myValue;
      recalculate ();
   }

   // Updates the value of the <answer> property. The value is
   updates in GADEA as well.
   public void recalculate ()
   {
      gadea.datatypes.String oldAnswer = answer;

      answer.setValue(String.valueOf (firstNumber.getValue() /
           secondNumber.getValue()));

      changeAgent.firePropertyChange ("answer", answer,
         oldAnswer);
   }
};
```

En el Código 39 se aprecia como el constructor de la segunda versión de la calculadora se dedica a la inicialización de las primitivas de datos empleadas y a su registro en GADEA. Nótese como ahora el programador se concentra en la naturaleza de la información con la que se desea trabajar (números y cadenas de caracteres) en lugar de tomar en consideración aspectos de la interfaz de usuario como puede ser la selección de los *widgets* necesarios para la representación visual (o sonora) de la información, así como de los eventos relacionados con estos *widgets*. Este enfoque permite que el diseño se concentre en la información a manejar en lugar de en su aspecto, lo cual facilita la definición de la naturaleza de los datos gestionados, permitiendo por ejemplo definir el rango de valores aceptados por una componente de datos básica como una característica de la propia componente, tal y como se puede apreciar en la invocación al método *setRange* de las propiedades *firstNumber* y *secondNumber* en el constructor de la aplicación.

Gracias a que la clase *GADEACalculator* cumple fielmente con la especificación de todo componente Java [Sun (1996)], los valores de los parámetros necesarios para ejecutar la operación de división pueden ser establecidos de forma externa por GADEA cada vez que dichos valores son modificados por el usuario en la interfaz. Sin embargo, a diferencia de la versión de esta calculadora especificada en el Código 38, cuando GADEA establece estos valores, ya ha realizado por su cuenta la validación de los mismos, asegurándose que se trata de valores numéricos comprendidos entre *MIN_VALUE* y *MAX_VALUE*, liberando de este modo al programador de la tediosa y peligrosa tarea de la validación continua de la información recibida por su aplicación.

Gracias a que la información recibida ha sido correctamente validada, la operación a realizar se simplifica ya que solo consiste en

aplicar el operador de división entera (método *recalculation*) actualizando el valor de la variable de salida *answer*. Una vez que éste valor es modificado, se notifica a *CodeX* del evento por medio de agente *changeAgent* para que esta modificación quede reflejada en la interfaz. Nótese que en la operación de división ya no es necesario acceder a la información contenida en los *widgets TextField* para convertir ésta en valores numéricos por medio del método *Integer.parseInt*(), sino que al tratarse ya de valores enteros, la operación de división se realiza de forma automática.

21.1.2. Tareas Implicadas

Como se puede comprobar en los ejemplos que ilustran esta páginas, el tamaño del código que implementa la versión GADEA de la calculadora básica es de tamaño similar al empleado para implementar la versión clásica, resultando en todo caso ligeramente menor. Sin embargo, el número de clases implicadas es sensiblemente menor en la versión GADEA, puesto que las clases relacionadas con los *widgets* que han de representar la información desaparecen. Otro tanto de lo mismo ocurre con las clases envoltorio intermedias (*Integer*, *String*, etc.) empleadas para realizar las oportunas conversiones entre la información proporcionada por un *widget* (generalmente un *String*) y el tipo de dato empleado para realizar los cálculos (en esta caso un *Integer*). Esta disminución en el número de clases empleadas implica naturalmente una disminución en la complejidad del código así como de los conocimientos necesarios para construir una aplicación tan simple como la presente.

Esta reducción en la complejidad queda patente también en la tipología de las operaciones efectuadas si comparamos las clases de operaciones empleadas en las dos versiones de la aplicación. Analizando en detalle el Código 38 se puede observar como las operaciones incluidas en el mismo pueden reducirse a la siguiente lista.

1. Selección de *widgets* (*constructor*).
2. Ubicación de *widgets* en une espacio visual (clase panel).
3. Manejo de eventos producidos por los *widgets* (método *actionPerformed*).
4. Verificación y validación de la entrada (método *calculation*).
5. Operación a realizar (método *calculation*).
6. Formato y gestión de la salida de datos (método *paint*).

Nótese como en salvo para la quinta operación es necesario tener un conocimiento avanzado acerca de los *widgets* a emplear (*TextFields*), tanto en sus características perceptibles (tamaño, ubicación, fuente tipográfica) como en cuanto a su funcionamiento interno (eventos generados). Nótese además que dicho conocimiento avanzado es dependiente en muchos casos de la plataforma de desarrollo en concreto. La complejidad se eleva si tomamos en cuenta además que a todo este conocimiento técnico avanzado hay que añadir el

conocimiento necesario para decidir el tipo de *widgets* a emplear, es decir, todos aquellos principios de interacción y comunicación humana aplicables en una situación dada, conocimientos que dependerán del dominio de la aplicación y del tipo de usuario concreto de la aplicación. Por otra parte, si analizamos el modelo de desarrollo propuesto para GADEA, podremos identificar las siguientes operaciones:

1. Selección y registro de estructuras de datos (*constructor*).
2. Operación a realizar (método *recalculate*).

La evidente reducción del número de tareas a realizar en la versión GADEA de la calculadora básica se debe principalmente a la eliminación de la necesidad de selección temprana de los *widgets* necesarios para la interacción el usuario por parte de los diseñadores de la aplicación, selección que como sabemos es realizada automáticamente por DEVA. El hecho de que esta decisión recaiga en GADEA implica además una gestión automática de todas las tareas relacionadas con dichos *widgets* por parte de este sistema de gestión de interfaces de usuario. Por ello no es necesario que el diseñador o programador de una aplicación ubique unos *widgets* que no gestiona y mucho menos que se interese por los eventos generados por los mismos. Evidentemente, estas tareas son parte del trabajo de GADEA.

21.1.3. Cualificación del Personal de Desarrollo

Como se puede apreciar, el enfoque aportado por GADEA al diseño rápido de aplicaciones orientadas a la gestión permite que el nivel de conocimiento requerido por el personal implicado en el desarrollo de dichas aplicaciones sea mucho menor. Nótese que con este enfoque no es necesario conocer los *widgets* soportados por una plataforma en concreto, ni tampoco las recomendaciones que la ingeniería de factores humanos a establecido para su uso, ni mucho menos el formato de la información obtenida a través de ellos.

Dado que GADEA reduce el tipo de componentes a emplear a una primitivas de datos básicas, el conocimiento requerido para su manejo es menor y la complejidad disminuye. Nótese que GADEA trabaja en términos de los tipos de datos que se incluyen en el modelo de datos de la aplicación. Datos por otra parte de fácil manejo y que resultan intuitivos para cualquier personal informático, no solo para los expertos en interacción y comunicación humana.

El empleo de tipos de datos básicos en lugar de *widgets* no solo reduce el nivel de conocimiento necesario para crear una aplicación sino que además simplifica su construcción dado que el número de tipos de datos presentes en una plataforma suele ser sensiblemente menor que el número de *widgets*.

Si a todo ello se añade el hecho de que no es necesario diseñar el nivel sintáctico de una interfaz teniendo en consideración los requisitos cognitivos, perceptivos y motrices de un tipo de usuario concreto (con la gran cantidad de recursos de diseño y construcción que ello conlleva) se puede percibir claramente como el modelo planteado por GADEA

puede llegar a reducir sensiblemente el coste de un proyecto. Dado que los productos desarrollados mediante el uso de esta estrategia pueden llegar a un número mayor de usuarios, queda de manifiesto el incremento en el rendimiento y productividad que presenta la misma en el desarrollo de aplicaciones, sobre todo en aquellas enfocadas a la explotación de sistemas comerciales.

21.2 Desarrollo Formal de Interfaces

Aunque el objetivo de esta investigación no es alcanzar una separación total entre el código y la funcionalidad de una aplicación, los resultados obtenidos han sido muy positivos, ya que esta separación se alcanza en la mayoría de las situaciones presentadas a lo largo del ciclo de interacción de un modelo WIMP. Precisamente, gracias al éxito obtenido en este proceso de separación básico es posible disminuir considerablemente el tiempo de desarrollo de una interfaz, permitiendo la construcción rápida de múltiples prototipos de la misma para prueba en las primeras fases de desarrollo, aumentando con ello la calidad y fiabilidad de los productos generados.

El diseño flexible del módulo CodeX permite además su aplicación conjunta con técnicas de especificación formal de interfaces de usuario, facilitando así la tarea del diseñador permitiendo una definición temprana de los requisitos de interacción de cualquier aplicación WIMP. El empleo correcto de GADEA en el desarrollo de interfaces a nivel semántico permite al diseñador alcanzar los objetivos impuestos para toda especificación formal de interfaces de usuario, los cuales que han sido recogidos por Rouff (1996) en la siguiente lista:

1) Documentación de los aspectos que se están desarrollando.
2) Comunicación entre las fases del ciclo de vida del producto.
3) Verificación de requisitos.
4) Mejora en la fiabilidad del producto.
5) Identificación de los aspectos de diseño.
6) Reducir la fase de pruebas.
7) Reducir el código.

Con respecto al primer objetivo, gracias al mecanismo de reflexión automática en el que se basa CodeX, es posible generar informes y demás documentación a medida acerca de los diversos procesos de usuario implementados en un producto en fase de desarrollo, bastando para ello un análisis del código binario generado. De este modo, CodeX permite alcanzar fácilmente los objetivos primero, segundo y tercero, permitiendo el estableciendo de estrategias de desarrollo que permitan aproximar la fecha de finalización del producto con respecto al plazo previsto en función del número y complejidad de los procesos de usuario disponibles y ya totalmente finalizados.

Las comprobaciones semánticas realizadas tanto sobre el dominio de las operaciones como sobre los distintos conjuntos de datos

recolectados en tiempo de ejecución, así como los registros de errores generados y recogidos por ANTS a partir de la realización de pruebas de usabilidad con usuarios reales, permite obtener valiosa información de depuración. Esta información permite alcanzar el cuarto objetivo planteado por Rouff en sus dos vertientes, tanto la del punto de vista de la ingeniería del software como la óptica de la ingeniería de factores humanos.

La necesidad de estructurar el código de un producto software con respecto a qué hacer en lugar del cómo hacerlo obliga a los diseñadores a identificar los aspectos del diseño de la interfaz de usuario en las primeras fases del ciclo de vida del producto y todo ello con total independencia del enfoque metodológico empleado.

Este modelo de desarrollo no solo es compatible con los enfoques metodológicos basados en la Teoría General de Sistemas como Métrica o Merise [MAP (1994), MAP (1994a), MAP (1991) y Gabay (1991)] o los enfoques basados en la orientación a objetos como Booch u OMT [Martin y Jesse (1996), Rumbaugh (1996), Booch (1994) y Booch (1996)], sino que al contrario, se apoya en las técnicas desarrolladas y/o empleadas por estos enfoques para facilitar el desarrollo de un producto, tal es la situación por ejemplo de los casos de usos empleados por UML [Muller (1997)], los cual se han revelado como una herramienta extraordinariamente útil en la identificación y configuración de los distintos procesos de usuario de una aplicación.

Gracias a esta estrategia de diseño centrada en el usuario [Norman (1988a)] aspectos tan delicados para un buen diseño de una interfaz, tales como los puntos de referencia, las rutas entre dos nodos de una red semántica y el modelo mental empleado por el usuario para moverse a través del modelo de navegación diseñado, alcanzan la misma relevancia que otros elementos importantes del diseño como lo son los datos y los procesos. Después de todo, los puntos de referencia equivalen grosso modo a los canales de comunicación, las rutas se pueden identificar con las operaciones disponibles en un canal de comunicación, mientras que un modelo mental genérico para la aplicación se puede obtener fácilmente a partir del diagrama de estados de un proceso de usuario, diagrama de estados que puede ser obtenido por CodeX en tiempo de ejecución para cada usuario. De este modo, el quinto objetivo se cumple también de manera satisfactoria.

El objetivo sexto se alcanza como consecuencia de la propia estructura de GADEA. El código necesario para poner a funcionar una aplicación (objetivo sexto) se reduce al mínimo ya que toda la gestión de ventanas, diálogos, etc. es realizada por los agentes inteligentes incorporados en DEVA. De este modo el código de las aplicaciones se concentra en los requisitos funcionales en lugar de dedicar espacio a los mecanismos de interacción.

21.2.1. Prototipos, Pruebas y Depuración

Debido a que los mecanismos de interacción son patrimonio exclusivo del sistema experto que los gestiona, la fase de pruebas

también se reduce, ya que el módulo ANTS realiza de pruebas de usabilidad automáticas de forma continua sobre los mecanismos de interacción empleados. Estas pruebas se llevan a cabo empleando hordas de agentes hormiga, los cuales se especializan en capturar trazas de uso. Estas trazas permiten detectar y corregir errores de forma automática en los mecanismos de interacción, así como adaptarlos a los perfiles cognitivos, perceptivos y motrices de los usuarios. Hay que destacar una vez más que éstas pruebas de usabilidad se realizan sobre sesiones reales, empleando usuarios reales, trabajando sobre las versiones finales de las aplicaciones finales y sobre máquinas reales.

Por otro lado, la complejidad de las pruebas a realizar sobre la navegación en el espacio de exploración de las aplicaciones también se reduce, gracias a que los agentes ANTS se encargan de identificar la posición del usuario dentro de la red semántica y/o diagrama de estados que compone la aplicación, obteniendo información sobre el tiempo empleado en cada uno de los nodos de la aplicación, así como del camino seguido y las rutas (procesos de usuario) empleadas. Esta característica permite desarrollar pequeños y sencillo programas de verificación capaces de detectar posibles errores semánticos en la definición de la redes, trabajando ya a un nivel de metáfora. En estos casos es posible contrastar el comportamiento real de los usuarios con el esperado, verificando que todos los parámetros sean correctos. En caso de existir divergencias importantes, los diseñadores de las pruebas disponen de información de primera mano (obtenida de sus usuarios reales) para modificar el comportamiento de su aplicación, suprimiendo o modificando el funcionamiento de las operaciones conflictivas y substituyéndolas por otras, que pasarían entonces a formar parte de la cartera de pruebas. Todo ello de forma automática.

Una de las características más ventajosas de GADEA con respecto al desarrollo de productos software es precisamente su capacidad para la creación rápida de prototipos a partir de especificaciones formales de alto nivel. La ventaja del uso de GADEA para el diseño de estos prototipos radica en que el componente humano del proceso se ve liberado del desarrollo del nivel sintáctico de la interfaz de los prototipos, ya que éste es llevado a cabo íntegramente por DEVA. De este modo, concentrando los esfuerzos de desarrollo y de prueba en el nivel semántico de los canales de comunicación y diálogos interactivos que componen una interfaz, el proceso de desarrollo se acelera y se hace fiable puesto que el número de variables en juego a la hora de realizar las pruebas de usabilidad disminuye drásticamente.

Gracias a la información recabada por los agentes hormiga de ANTS, el período de pruebas de una aplicación se puede extender a lo largo de toda la vida útil. Dado que la información recogida por éstos agentes puede ser almacenada en un modelo de usuario centralizado y clasificarse por aplicación, ahora es posible conseguir una especificación exacta y precisa de las características que definen al usuario típico de toda aplicación. Paradójicamente, esta investigación partió precisamente la negación de la existencia de un usuario típico al que ahora es posible definir.

Conclusiones

El esquema de trabajo planteado por CodeX obliga a los diseñadores a planificar cuidadosamente las características perceptibles del artefacto que están diseñando, así como su funcionamiento, especificando con gran detalle las partes que estarán accesibles al usuario en cada estado de la aplicación, así como los datos que dicha aplicación requiere para un funcionamiento correcto del sistema. Esta estrategia obliga a una planificación temprana de la interfaz desde las primeras fases del proyecto, sobre todo durante el proceso de análisis. De este modo, además de identificar los procesos y datos requeridos por el sistema, el analista tiene como recordatorio permanente la obligación de identificar los procesos de usuario, así como las *precondiciones* y *postcondiciones* asociadas. No en balde estos procesos de usuario son el verdadero motor de la aplicación, pues van a jugar de verdaderos detonantes de la ejecución del resto de los procesos, así como de controladores de la transiciones entre los distintos estados de la aplicación.

La identificación de los datos requeridos por cada proceso cada uno de los canales de comunicación y diálogos interactivos asociados permite elaborar un esquema del modelo de datos bastante aproximado al modelo final ya desde las primeras fases del diseño. Este esquema permite abordar el problema del diseño de la aplicación indistintamente desde un enfoque de datos o desde el punto de vista de los procesos, siendo perfectamente compatible con las estrategias de desarrollo dual dato-proceso de enfoques metodológicos de fuerte implantación en la industria, como es el caso de Métrica [MAP (1994)] o de Merise [Gabay (1991)].

En este tipo de procesos metodológicos, el diseñador puede empezar el análisis por el modelo de datos o por el modelo de procesos de forma indistinta. La estrategia de CodeX refuerza este planteamiento favoreciendo la identificación de procesos de usuario (procesos del más alto nivel en ambos enfoques metodológicos) o de los datos requeridos por cada uno de estos procesos, permitiendo así identificar el flujo de datos partícipe de la comunicación externa de la aplicación (flujo de datos del más alto nivel). CodeX permite al diseñador trabajar con un gran nivel de abstracción, pues su desarrollo se concentra en los niveles de metáfora y superiores, dejando los aspectos de bajo nivel (colores, tamaño y posición de los objetos, etc.) al cuidado de DEVA, quien los adaptará según las necesidades concretas y particulares del modelo cognitivo, perceptivo y motriz de cada usuario. Todo ello de forma transparente, tanto para el diseñador como para el colectivo de usuarios finales de la aplicación.

22

22 La Adaptabilidad

Nuestra opinión de las acciones humanas depende siempre del placer o del dolor que nos causan

Anatole France

22.1 Evaluación

En los capítulos correspondientes a las tres partes precedentes de la presente se ha realizado un análisis crítico de las bases teóricas en las que se fundamenta la arquitectura de adaptación propuesta, obteniendo como resultado una serie de técnicas de aplicación práctica que permiten llevar a cabo dicho proceso de adaptación de forma sistemática.

A lo largo del proceso de investigación que ha dado origen al presente documento, las técnicas desarrolladas para GADEA han sido puestas en práctica y evaluadas a través de diversos proyectos, los cuales han servido de verdaderos campos de prueba para la tecnología aquí desarrollada. La naturaleza de estos proyectos ha sido tan diversa como la propia variedad de aspectos que este sistema de gestión de interfaces de usuario emplea en su proceso de adaptación. Así, se han desarrollado aplicaciones multimedia de todo tipo y condición

(englobadas dentro del programa Tirsus), herramientas de autor adaptables para trabajo en grupo (programa CineMedia Astur), agentes inteligentes capaces de emular el comportamiento humano y/o de vulnerar la barrera de su atención mediante el empleo de mensajes subliminales (programa Inxena) o sistemas de desarrollo de pruebas de usabilidad automáticas basadas en agentes, los cuales constituyen una rama independiente del módulo ANTS del propio GADEA.

Diversa ha sido también la audiencia a la que estos proyectos están dirigidos, intentando con ello involucrar el mayor rango posible de usuarios en las pruebas y evaluación de las mencionadas técnicas con el objetivo de detectar posibles requerimientos especiales de interacción – a nivel de sintaxis para proceder después a un cuidadoso análisis de los mismos en busca de las claves que permitan adaptar los diálogos interactivos a estos requerimientos. Así, se han empleado las herramientas y aplicaciones desarrolladas para experimentar con usuarios de diversas edades (desde adolescentes hasta ancianos), diversos requisitos de interacción y percepción (diferentes grados de *Precisión Visual* y/o *Precisión Auditiva*), diversos entornos culturales y por supuesto con usuarios de ambos sexos.

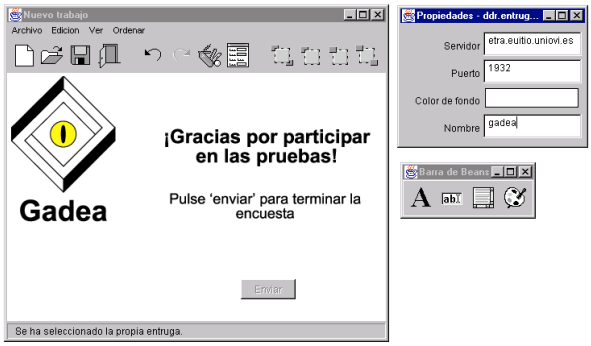


Figura 97. Diseño de una de las pruebas de usabilidad empleadas para evaluar GADEA por medio del módulo Entruger de CineMedia Astur.

Al respecto cabe destacar la gran importancia que Internet ha tenido en este estudio al poder enrolar en las pruebas de usabilidad a gran cantidad de voluntarios de diferentes edades, sexos, etc, lo cual posibilitó el estudio de las técnicas desarrolladas para amplias muestras de población asegurando una amplia variedad en los modelos cognitivos, perceptivos y motrices involucrados en el estudio. En este apartado es necesario el empleo de una amplia plataforma de captación de voluntarios basada en un popular sitio web, adaptado para el desarrollo de pruebas de usabilidad automáticas y para la realización de encuestas distribuidas en Internet, tarea en la cual fue de inestimable la experiencia y tecnología desarrollada en el marco del programa CineMedia Astur a través de su módulo de encuestas o *Entruger* [Díez (1998)].

El módulo Entruger de CineMedia Astur permite confeccionar encuestas de todo tipo mediante el uso de componentes *JavaBeans* que se ensamblan en una o más escenas (nodos) de la aplicación multimedia resultante. Esta funcionalidad permite crear un modelo de navegación para la propia encuesta, el cual puede ser aprovechado para informar al

usuario acerca de la naturaleza de la encuesta, sus objetivos y posibles resultados, además de servir de ayuda para cubrir los diversos funcionar formularios que componen dicha encuesta. De hecho, ésta ha sido la técnica empleada para informar a los participantes en las pruebas de usabilidad acerca del tipo de información que debían aportar en cada prueba en función del parámetro de GADEA evaluado.

El producto generado por Entruger consiste en una aplicación multimedia completa encapsulada dentro de un applet de Java, el cual se puede instalar fácilmente en cualquier sitio web, siendo reconocido por cualquier explorador de Internet que de soporte al sistema de eventos de Java 1.1, lo que significa que esta herramienta puede ser empleada a partir de la versión 3.1 del Netscape Communicator y de la versión 4.0 del Microsoft Explorer.

Cuando el usuario accede a un sitio web con funcionalidad Entruger, puede acceder a cualquiera de las encuestas instaladas en el mismo, navegando por los diferentes nodos que las componen, proporcionando la información requerida por medio de campos de texto, menús desplegables, botones de radio, etc. (ver Figura 97). Una vez que el usuario ha cubierto correctamente el formulario que constituye la encuesta, la información es enviada automáticamente a un servidor establecido en tiempo de edición, el cual se encarga de almacenarla, bien en una serie de ficheros o en una base de datos.

El sitio web elegido para efectuar las encuestas en gran parte de las pruebas de esta evaluación has sido uno dedicado a la Historia de Asturias [Arqueoastur (2001)], el cual recibía un promedio de 25 accesos diarios en el momento de la realización de dichas pruebas. Este sitio web contaba con una comunidad de usuarios estable, lo cual ha sido una enorme ventaja a la hora de conseguir voluntarios para las pruebas de usabilidad realizadas sobre diferentes los aspectos de las técnicas de interacción aquí descritas. Dado que los usuarios de este sitio web mantienen un lazo de unión con otros usuarios a través de una lista de correo electrónico, se empleó este medio para solicitar la colaboración de sus miembros en las pruebas de usabilidad. Como veremos a continuación, estas pruebas se han realizado tanto remotamente (por medio de ANTS y Entruger) como localmente, empleando las técnicas de prueba de usabilidad tradicionales en el Laboratorio de Tecnologías Orientadas a Objetos del Departamento de Informática de la Universidad de Oviedo.

22.1.1. Factores de Usabilidad

El núcleo del gestor de conocimiento de DEVA está basado en el modelo de objetos señuelo y objetos predador desarrollado en GADEA el cual, tal y como se ha podido apreciar en los capítulos 14, 15 y siguientes, se ha desarrollado en función de los mecanismos de selección focal empleados por la atención humana en el procesador perceptivo. En dichos capítulos se ha dejado constancia de varios experimentos realizados al respecto por investigadores de la psicología cognitiva en los que se demuestra la existencia de varios niveles de

procesamiento cognitivo de los estímulos percibidos por un humano, siendo la naturaleza de estos estímulos tanto sintáctica como semántica.

Aún así, dentro del marco de esta investigación se ha querido validar el modelo de objetos señuelo y objetos predador propuesto, evaluando cuantitativa y cualitativamente su potencia en aplicaciones basadas en multimedia e hipertexto. Para ello se han diseñado y desarrollado algunos artefactos hipermedia en los que se aplica esta técnica, midiendo su eficacia mediante pruebas de usabilidad. Como ya se ha mencionado en el capítulo (objetos señuelo), estos artefactos se han desarrollado dentro del marco del programa de investigación Tirsus, el cual emplea la Historia como base de conocimiento, en concreto la Historia de Asturias. Para ello se ha contado con la colaboración y el asesoramiento técnico de la dirección del Museo Arqueológico de Asturias, así como de varios arqueólogos y licenciados en historia que centran su actividad investigadora en la base de conocimiento citada.

La naturaleza abstracta y altamente evolutiva de la Historia, convierten la enseñanza integral de esta disciplina en un reto difícil de asumir mediante los métodos expositivos tradicionales, basados por lo general en técnicas narrativas secuenciales. La incompatibilidad manifiesta entre una base de conocimiento formada por multitud relaciones causa-efecto y una estructura narrativa lineal, en la que sólo es posible abarcar un conjunto limitado de relaciones, da lugar a una exposición parcial de los contenidos didácticos.

La existencia de sistemas de enseñanza basados en hipertexto permite la creación de productos educativos que favorecen la interconexión de conceptos y el establecimiento de unidades didácticas distribuidas [Nielsen (1993)] en un discurso narrativo adaptable al modelo cognitivo del receptor [Olsina (1998)]. De aquí parte nuestro interés en el desarrollo de artefactos centrados en esta base de conocimiento concreta.

El tipo de usuarios hacia los que está dirigido el material didáctico contenido en los proyectos del programa Tirsus comprende principalmente jóvenes estudiantes de educación básica, siguiendo el plan de estudios adaptado para la ESO en Asturias, así como adultos que establecen un primer contacto con esta base de conocimiento. En algunos proyectos, la base de conocimiento es compartida también por expertos en el tema de estudio, principalmente investigadores. Como se puede observar, el tipo de usuarios a los que están destinados estos proyectos es muy amplio y las características de sus modelos cognitivos son muy diversas. Obviamente, el modo en el que el producto será utilizado varía considerablemente de una categoría de usuario a otra, dependiendo de la variedad de inteligencias y de los diferentes niveles y formas que presenta las personas a la hora de aprender [Álvarez y Soler (1999)]. Por ejemplo, las estrategias de navegación empleadas por los usuarios novatos para explorar la base de conocimiento son distintas de las empleadas por los usuarios con algún conocimiento previo. Mientras que los primeros suelen aplicar estrategias basadas de tanteo, los segundos emplean mapas mentales para obtener por adelantado un plan completo que permita alcanzar el objetivo deseado [Jul y Furnas (1997)].

Como consecuencia de esta diversidad, todos los proyectos del programa Tirsus han sido diseñados teniendo en cuenta que los modelos de interacción a nivel semántico han de adaptarse no sólo a factores de diversidad explícitos, como por ejemplo la edad de los usuarios [Stone y Glock (1981), p. 419-426]; sino también en función aspectos puramente cognitivos como lo son la atención [Hillstrom y Yantis (1994), p. 399-411] o la percepción [Wertheimer (1958)]. Es por ello que estos proyectos han sido desarrollados empleando metodología OOHDM (*Object Oriented Hypermedia Development Model*) la cual permite diseñar distintos modelos de navegación a partir de una única base de conocimiento, separando así los contenidos (que permanecen inalterables) del acceso a los mismos, existiendo tantas vistas de dichos contenidos como categorías de usuario existan [Schwabe y Rossi (1995)]. El desarrollo modular de OOHDM en distintas fases permite además el intercambio de esquemas y patrones de diseño entre más de un proyecto [Schwabe et. al. (1996) y Rossi (1997)], lo cual proporciona una ventaja añadida de gran importancia, al poder extrapolar los resultados obtenidos de un determinado proyecto multimedia al resto de los proyectos del programa Tirsus. En el momento de escribir estas líneas, el programa Tirsus comprende los siguientes proyectos Tirsus I: *El Museo Arqueológico de Asturias* [Cordero (1999)], Tirsus II: *Astures y Romanos* [Sánchez (2000a)], Tirsus III: *El Reino de Asturias* [García Castro (2001)], Tirsus IV: *La Guerra Civil en Asturias* [García Fernández (2001)] y Tirsus V: *Los Indianos*, habiendo finalizado ya los dos primeros proyectos.

Aunque, como se verá en el capítulo 23 (*La Tecnología*) se siguen ensayando nuevas técnicas de estructuración del conocimiento para GADEA en los proyectos Tirsus III, Tirsus IV y Tirsus V, el modelo de objetos señuelo y objetos predador se ha evaluado formal y exhaustivamente sobre el proyecto Tirsus II (Astures y Romanos). Este proyecto tiene por objetivo recoger la base de conocimiento disponible acerca de la protohistoria y el inicio de la historia de Asturias, comprendiendo el intervalo de tiempo entre principios de la Edad del Hierro y las primeras invasiones bárbaras a la caída del Imperio Romano (1000 a. C. al 500 d. C. aproximadamente).

En la primeras fases del diseño de contenidos de Tirsus II, las pruebas de usabilidad a pequeña escala desarrolladas sobre un grupo muy reducido de voluntarios revelaron la existencia de determinados nodos en los que existía una saturación importante de información, saturación que afectaba incluso al sistema perceptivo del usuario. Como consecuencia, se procedió a la reestructuraron los nodos, de forma que los contenidos incluidos en los nodos originales se distribuyeron en nuevos nodos creados a partir de los primeros. Siguiendo un paradigma de desarrollo multimedia basado en la confección de múltiples prototipos, seguida de una evaluación de usabilidad, se procedió a una reestructuración continua de la información dividiendo aquellos *chunks* dotados de excesiva carga semántica en otros *chunks* de menor carga.

La información contenida en este nodo si llegó a ser manejable por los usuarios tipo de la aplicación. Este ejemplo da una idea del grado de

estructuración de la información necesario para este tipo de aplicaciones, así como la naturaleza iterativa del proceso de desarrollo.

Aun así, existen situaciones en las que el contenido de la unidad didáctica representada por un nodo no puede ser dividido ni asignado a otros nodos, aún cuando la cantidad de información desborda el espacio visual del nodo. Este problema, se resolvió distribuyendo la información de cada nodo en tres niveles, haciendo uso de sistemas de interacción multimodal combinación elementos visuales y no visuales. La jerarquía en la que se organizó la información para Tirsus II se puede ver en la Figura 98.

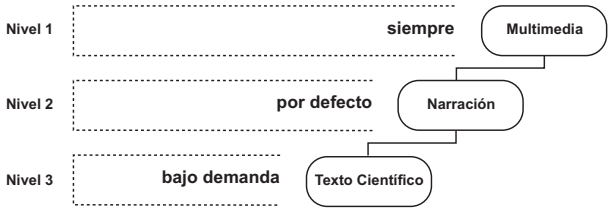


Figura 98. Jerarquía de contenidos de un nodo en Tirsus II.

En un primer nivel de información se sitúan los elementos multimedia básicos para la comprensión del objetivo didáctico del nodo (textos, imágenes, vídeos, sonidos, etc.). Este nivel pretende una rápida asimilación de los contenidos intentando basar el peso informativo en imágenes en lugar de los textos. Se trata por tanto de objetos señuelo dotados de una carga semántica global que permite al usuario asimilar grandes cantidades de información poco detallada, interesarse por un tema concreto y profundizar en el modelo de navegación mediante objetos señuelo. En la Figura 99 se puede ver un ejemplo de este nivel, en un nodo diseñado para explicar una campaña militar mediante animaciones sobre un mapa. Como se puede ver, se hace un uso muy limitado del texto.

Figura 99. Explicación de una campaña militar mediante objetos señuelo (consultar versión en color en el capítulo 24: *Imágenes en Color*, p. 445).

El segundo nivel de información está activo por defecto, aunque el usuario puede desactivarlo en cualquier momento. En este nivel se

complementa la información del primer nivel mediante un sistema de narración con voz en *off*, describiendo en mayor detalle la información visual ya recibida por el usuario. Los dos niveles se sincronizan para actuar a la vez. En el ejemplo anterior, la voz en *off* narra el resultado de las batalla de forma sincronizada con las indicaciones que aparecen en el mapa, señalando la dirección del movimiento de tropas.

Por último, el tercer nivel proporciona el mayor grado de detalle posible, proporcionando información en formato texto. El acceso a este nivel se establece bajo demanda, pulsando sobre un botón especial presente en los nodos con este servicio activando un bloc de notas en el que se incluye el texto (ver Figura 100). El texto incluido en este bloc hace las veces de objeto predador para la información transmitida en el primer nivel.

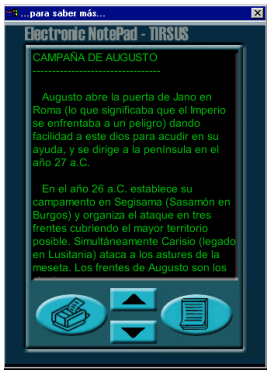


Figura 100. Objeto predador empleado para describir en detalle la evolución de la campaña militar ilustrada en la Figura 99 (consultar versión en color en el capítulo 24: *Imágenes en Color*, p. 445).

Una vez terminado el diseño y construcción del motor de navegación de la base de conocimiento, el prototipo fue sometido a pruebas de usabilidad en las que participaron trece voluntarios reclutados entre colaboradores de la Escuela Universitaria de Ingeniería Informática de Oviedo (EUITIO), la Asociación de Amigos del Museo Arqueológico de Asturias y usuarios del sitio web de la Historia de Asturias.

La primera parte de las pruebas consistió en la realización de un pequeño test por escrito en el que se hacían preguntas básicas acerca del conocimiento que el voluntario tienen del modo de operar en sistemas informáticos multimedia, así como de su relación con el tema de la base de conocimiento. De este modo se distinguieron dos clases de usuarios (a nivel semántico) en función de dos variables: grado de conocimiento sobre la informática y grado de conocimiento sobre la Historia de Asturias.

En una segunda parte, se dejó que los voluntarios se familiarizasen con el producto realizando una navegación libre a través de los distintos contextos de navegación, siempre bajo la observación continua por parte de quien subscribe. Durante este proceso de navegación libre se observó cuidadosamente las operaciones realizadas por los voluntarios, los nodos accedidos, la información obtenida, centrando el

interés de la observación en el provecho obtenido de la estructuración de contenidos en los niveles descritos.

Por último, la prueba de usabilidad concluyó con una pequeña tarea de búsqueda de información en la que el voluntario debería responder a una serie de preguntas acerca de la Historia de Asturias, cuyas respuestas había que buscarlas en la base de conocimiento de Tirsus II. Aunque para resolver estas preguntas no era estrictamente necesario acceder al tercer nivel de información (objetos predador en estado puro), si era recomendable para resolver dichas preguntas con un determinado grado de detalle.

La relación completa de las pruebas de usabilidad efectuadas sobre Tirsus II, así como copias de los test empleados, se pueden consultar en Sánchez (2000a).

Tabla 41. Clasificación de los usuarios participantes en las pruebas de usabilidad de Tirsus II en función de su grado de experiencia en informática y en Historia de Asturias, así como la proporción de usuarios por grupo que hicieron uso de los objetos predador.

	HISTORIA (ALTO)	HISTORIA (BAJO)
INFORMÁTICA (ALTO)	2(1) – 50%.	4(4) – 100%.
INFORMÁTICA (BAJO)	6(2) – 33%.	1(0) – 0%.

En la Tabla 41 se pueden ver los resultados de las pruebas de usabilidad con respecto al uso de los objetos predadores. Como se puede apreciar, de los trece participantes, dos de ellos tenían un alto conocimiento en informática y en historia, cuatro lo tenían sólo en informática, seis lo tenían en historia solamente y solamente uno era novato en temas de historia y de informática.

El valor reflejado entre paréntesis en cada celda de la Tabla 41 representa el número de voluntarios del grupo que hicieron uso de los objetos predador bien durante la fase de exploración libre, o bien durante la tercera fase en la que tenían que emplear el artefacto multimedia para responder a las respuestas planteadas. Se puede observar como los objetos predador fueron empleados por siete de los trece participantes en las pruebas (53,84%) y como los usuarios con un nivel elevado de conocimientos en informática fueron los más beneficiados en la aplicación de este modelo, pues cinco de los seis voluntarios pertenecientes a esta clasificación hicieron uso de los objetos predador (83,33%).

Terminadas las pruebas y entrevistados los usuarios acerca de su opinión sobre el prototipo, se descubrió que, salvo uno, todos aquellos voluntarios que no habían utilizado los objetos predadores desconocían su existencia al no haber percibido claramente su presencia en la interfaz del nodo correspondiente.

La clara relación existente entre el desconocimiento de la existencia de objetos predador y un nivel bajo de conocimiento en informática parece indicar que el uso de los objetos predador es una característica atribuible solo a los usuarios veteranos de una aplicación y es ese precisamente el uso que se lo otorga en GADEA. Si recurrimos una vez

más a los ejemplos que ilustran el capítulo 13 (*El Procesador Humano*) vemos como GADEA conduce al usuario novato en el aprendizaje del uso de estos objetos mediante el establecimiento de un diálogo interactivo en dos o más fases, transmitiendo los objetos señuelo en primer lugar y los objetos predador en segundo lugar. Sólo cuando el usuario ha alcanzado cierto grado de veteranía será cuando DEVA transmite todos los objetos predador en una sola fase.

A partir de los resultados de esta experiencia, podemos concluir que el empleo del modelo de objetos señuelo y objetos predador es altamente positivo en un sistema de comunicación basado en diálogos interactivos, siempre y cuando la presencia de los objetos predador sea claramente perceptible por parte usuario destinatario de la información. El grado de percepción dependerá del grado de adaptación de la interfaz a las características perceptivas del usuario, así como de la experiencia previa del mismo (grado de veteranía) con sistemas similares.

Otra interesante experiencia en la que se ha aplicado con éxito el modelo de objetos predador y objetos señuelo es el de los sistemas de ayuda basados en la transmisión de objetos señuelo de baja carga cognitiva a través de un canal de comunicación secundario, de tal modo que el usuario de la aplicación pueda seguir concentrado en la tarea primaria desarrollada en el canal de comunicación primario, mientras atiende al canal de comunicación secundario en un nivel de atención subliminal. Por medio de esta técnica, el usuario será consciente de la información transmitida a través del canal de comunicación secundario sólo cuando ésta información le sea relevante y atraviese por tanto la barrera impuesta por la atención y pase a la consciencia.

Para evaluar la utilidad de esta técnica se confeccionó un pequeño experimento aprovechando una situación coyuntural favorable presentada en una de las asignaturas en las que imparte docencia en la Universidad de Oviedo quien subscribe este documento. En la mencionada situación se requería contactar con determinados alumnos de la asignatura para que éstos aportasen al profesorado cierta documentación. Dado que los mencionados alumnos asistían regularmente a las clases de prácticas de la asignatura, se decidió notificarles el requerimiento de información durante las mismas, empleando para ello un mecanismo similar al empleado por GADEA para transmitir información de carácter secundario.

Las clases prácticas de esta asignatura consisten en la elaboración de pequeñas programas y aplicaciones a partir de instrucciones incluidas en el sitio web de la asignatura, de tal modo que los alumnos se ven obligados a consultar el mencionado sitio web al menos una vez por sesión. No obstante, los informes aportados por ANTS al verificar este sitio, nos muestran que el promedio de consultas por usuario a la página web en la que se incluye el enunciado de la práctica a realizar se sitúa en torno a los cinco accesos por sesión, es decir, un acceso cada doce minutos.

Dado este perfil de acceso constante a una página web concreta, se aprovechó ésta para incluir en ella un canal de comunicación

secundario en el que se transmite información de ayuda para la realización del trabajo práctico. Esta información es transmitida de forma constante, mediante un *banner* situado en la zona superior del espacio visual del sitio web (ver Figura 101). Este *banner* es único para todo el sitio web y está conectado a un hormiguero ANTS mediante un sistema de GPS, de tal modo que es capaz de transmitir información sensible al contexto, lo cual es la esencia más pura del modo de objetos señuelo y objetos predador. Nótese que la información de ayuda transmitida por este canal de comunicación secundario se va tornando más precisa en cuanto el usuario más se aproxime a la página web deseada, es decir, se la transmisión se inicia con objetos señuelo para las páginas web de introducción (las que se encuentran en la cima de la jerarquía del modelo de navegación) y se finaliza con la transmisión de objetos predador para las páginas destino.

Figura 101. Página principal del sitio web de la asignatura. El *banner* con información de ayuda complementaria se encuentra en la parte superior del espacio visual (en rojo) y es común a todas las páginas (consultar versión en color en el capítulo 24: *Imágenes en Color*, p. 445).

Dado que es muy difícil cuantificar el grado de utilidad de la información transmitida por medio de este sistema de ayuda subliminal (salvo por encuesta directa), además de determinar la proporción de los mensajes transmitidos que llegan realmente a su destino, se procedió a insertar información de especial relevancia para ciertos usuarios entre los mensajes de ayuda transmitidos. Esta información consistía en mensajes del tipo *Fulanito de tal, póngase en contacto con los profesores de la asignatura* y obviamente estaba dirigida a aquellos alumnos que debían aportar documentación pendiente.

Durante el experimento, se observó personalmente la reacción de los alumnos a los que los mensajes estaban dirigidos, cuantificando el tiempo transcurrido desde el inicio de sesión hasta que los mencionados alumnos percibieron el mensaje y se pusieron en contacto con su profesor de prácticas. Esta información tomada *in situ* se comparó con la información aportada por los agentes ANTS, de tal modo que también fue posible determinar la cantidad de tiempo total que el banner estuvo visible antes de que el mensaje fuese captado. Estos tiempos han sido recogidos en la Tabla 42.

Tabla 42. Tiempos de exposición (T) a los mensajes secundarios para cada uno de los usuarios del experimento, así como el momento (M) en el que la información fue captada.

ID	T. EXP. BRUTO	M. PERCEP. BRUTO	T. EXP. NETO	M. PERCEP. NETO
A	56 minutos.	6 minutos.	22 minutos (39,28%).	4 minutos (18,18%).
B	52 minutos.	No percibido.	15 minutos (28,84%).	No percibido.
C	48 minutos.	12 minutos.	16 minutos (33,33%).	11 minutos (68,75%).
D	32 minutos.	15 minutos.	25 minutos (78,12%).	6 minutos (21,42%).
E	48 minutos.	10 minutos.	28 minutos (58,33%).	6 minutos (21,4%).
F	53 minutos.	2 minutos.	32 minutos (60,37%).	2 minutos (6,25%).
G	52 minutos.	No percibido.	21 minutos (40,38%).	No percibido.
		Promedio	27 minutos. (46,62%)	6 minutos (22,22%).

De los siete usuarios a los que estaba dirigida la información transmitida, solamente dos de ellos no fueron capaces de percibir el mensaje. En todo caso, sobre esta pequeña muestra de siete usuarios y en un tiempo de transmisión bruto de unos cincuenta minutos, esta técnica de transmisión alcanzó un éxito del 71,42%. Analizando los resultados obtenidos se puede apreciar como de los veintisiete minutos de promedio netos en los que un usuario está expuesto a la transmisión de los mensajes del *banner* de ayuda, el mensaje relevante suele ser percibido en torno a los seis minutos, una vez transcurrido un 22% del tiempo de exposición. Este resultado parece indicar que el mensaje no se capta en las primeras etapas de transmisión, sino mucho más adelante en el proceso, siendo requerida una repetición constante del mensaje a través del canal de comunicación secundario para poder ser captado.

A la vista de los resultado podemos concluir que el mecanismo de transmisión subliminal de información de baja prioridad no solo es efectivo (el mensaje fue recibido en el 71,42% de los casos) sino que no afecta significativamente al rendimiento del canal de comunicación principal sobre el que no parece introducir ruido, ya que el mensaje es recibido en un lapso de tiempo considerablemente grande con respecto al inicio de la emisión.

Esta técnica de transmisión de información es tan potente que durante la realización del proceso experimental, uno de los usuarios percibió el mensaje a el dirigido a través del canal de comunicación secundario de otro usuario. A pesar de que esta persona se encontraba concentrada en la tarea principal de su sesión de trabajo (la realización de la práctica), detectó de forma inconsciente la presencia de su nombre en el *banner* mostrado en la pantalla del ordenador de un usuario situado tres metros delante de ella. A partir de entonces, esperó de forma consciente a que el mensaje apareciese de nuevo (en el ordenador vecino) antes de contactar con su profesor. Sirva esta anécdota para ilustrar la potencia del proceso de captación subliminal de estímulos.

22.2 Calidad de la Adaptación

Para evaluar la calidad del proceso de adaptación realizado sobre los distintos aspectos de un canal de comunicación, es decir, la adecuación de dicho proceso de adaptación a las necesidades del usuario, se han realizado unas pequeñas de pruebas de usabilidad en condiciones controladas, así como una serie de pruebas de usabilidad remotas en las que es el propio voluntario participante las pruebas quien responde de manera voluntaria a una serie de preguntas acerca de la calidad de la adaptación tras evaluar distintos aspectos de la interfaz.

Las pruebas controladas se realizaron sobre una pequeña muestra de cinco usuarios en donde existía una gran variabilidad en cuanto a los parámetros empleados por GADEA para realizar la adaptación de los canales de comunicación. Se trataba pues de usuarios con valores dispares para los parámetros *Precisión Visual*, *Edad* y *Sexo* (ver Tabla 43). El único parámetro que los voluntarios mantenían en común es su escasa experiencia en el uso de herramientas informáticas y sobre todo de dispositivos de señalización mecánica (el ratón principalmente), requisito que como se verá a continuación era necesario para evaluar la eficiencia con la que los agentes hormiga eran capaces de determinar la eficiencia –valga la redundancia– con la que los usuarios manejaban dichos dispositivos, parámetro que, como ya se ha señalado en su oportuno momento es empleado por GADEA para establecer la ubicación de los denominados *widgets* activos.

Tabla 43. Valor de los parámetros de adaptación de los cinco voluntarios participantes en las pruebas. Los valores de los parámetros *Precisión Visual* y *Edad* se recogen en base al grado de pertenencia de los mismos a los conjuntos difusos incluidos en el capitulo 11 (*El Application Explorer*).

ID	Precisión Visual	Edad	Sexo
A	Bajo	Anciano	Mujer
B	Normal Bajo	Adulto Maduro	Mujer
C	Normal Alto	Adulto Maduro	Hombre
D	Excelente	Joven Adulto	Hombre
E	Alto	Adulto	Mujer

Los factores evaluados en esta parte de las pruebas se correspondieron con aspectos puramente visuales de la aplicación, generando de antemano una serie de posibles diálogos interactivos con cinco variaciones cromáticas importantes en los colores empleados para los fondos y para las formas. A partir de estas cinco variaciones cromáticas, se generaron versiones adaptadas de los diálogos para distintos tipos de usuarios y situaciones, definiéndose un total de siete plantillas etiquetadas como I, II, III, IV, V, VI y VII (ver Tabla 44). Con la excepción de las plantillas I y VII, el resto de las plantillas fueron diseñadas por GADEA a partir de los parámetros de adaptación de cada uno de los usuarios implicados en las pruebas, creando así representaciones abstractas de las interfaces. A continuación, éstas plantillas fueron cargadas en el Entruger y mostradas a los usuarios una

a una, con el objetivo de que estos ordenasen las plantillas presentadas de acuerdo con la calidad de ajuste a su sistema perceptivo. Aquellas versiones que se percibiesen mejor (de acuerdo con el *punto de vista* de cada usuario) debían incluirse primero en la lista. En la Tabla 44, se recogen plantillas generadas para cada una de las variaciones cromáticas.

Tabla 44. Especificación de los valores de adaptación para las plantillas generadas por GADEA en las pruebas de adaptabilidad.

ID	DEFINICIÓN DE PLANTILLA	USUARIOS DESTINO
I	Original	Sin adaptación
II	Edad: Anciano, Contraste Visual: Máximo, Sexo: Mujer	A
III	Edad: Adulto Maduro, Contraste Visual: Normal, Sexo: Mujer	B, C
IV	Edad: Adulto Maduro, Contraste Visual: Medio Bajo, Sexo: Hombre	C, B
V	Edad: Joven Adulto Maduro, Contraste Visual: Nulo, Sexo: Hombre	D, E
VI	Edad: Joven Adulto, Contraste Visual: Medio Bajo. Sexo: Mujer	E, D
VII	Obscuridad	Cualquiera

Las siete plantillas fueron mostradas simultáneamente a cada uno de los usuarios para cada una de las cinco variaciones cromáticas definidas. Para ello fueron diseñadas cinco secciones en la encuesta registrada en el Entruger, una para cada una de las variaciones cromáticas. En cada una de las secciones se incluyeron las siete plantillas referidas. De este modo, el usuario percibía no sólo la plantilla confeccionada expresamente para él, sino también el resto de las plantillas –diseñadas para otro tipo de usuarios.

El objetivo evidente de este enfoque era el introducir cierta dosis de ruido en el proceso de selección de la plantilla más relevante por parte de cada usuario. Dado que todas las plantillas mostradas participan por igual en el proceso selectivo (el voluntario debía tomar en cuenta todas las plantillas percibidas a la hora de confeccionar la respuesta) si el proceso de adaptación no es el correcto, es posible que el usuario seleccione como plantilla favorita alguna que no ha sido diseñada para él en exclusiva.

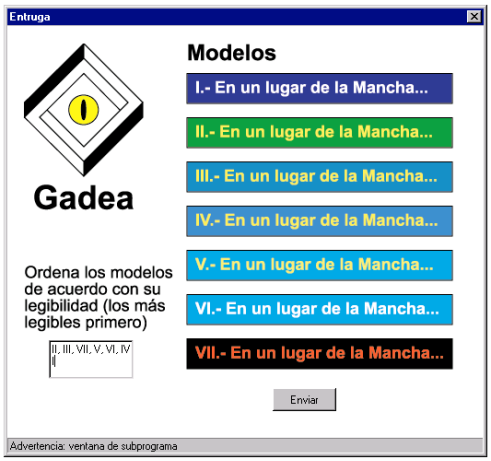


Figura 102. Plantilla de selección cargada en el Entruger y visualizada desde un navegador de Internet por medio de un applet de Java (consultar versión en color en el capítulo 24: *Imágenes en Color*, p. 445).

En líneas generales, aunque no se detecta un nivel de adaptación total, se puede decir que resultados de estas pruebas fueron satisfactorios, dado que la plantilla seleccionada en primer lugar por los usuarios solió coincidir con aquella diseñada por GADEA para usuarios con características idénticas a los que éstos poseían. Los resultados globales indican que en veintiuna ocasiones de las veinticinco posibles (84,5%) la elección realizada por el usuario coincidió con la elección efectuada por el agente interactivo DEVA (ver Tabla 45).

Tabla 45. Secuencia de respuestas proporcionadas por los voluntarios en las pruebas de adaptabilidad.

ID	V1	V2	V3	V4	V5	ADAPTACIÓN
A	II, IV, VII, III, VI, V, I	II, IV, VII, III, VI, V, I	VII II, IV, III, V, VI, I	II, VII, IV, III, VI, V, I	VII, II, III, IV, VI, V, I	3/5 (60%)
B	III, II, IV, VII, V, VI, I	III, IV, II, VII, V, VI, I	III, IV, VII, VI, II, V, I	III, VII, II, IV, V, VI, I	III, VII, IV, II, VI, V, I	5/5 (100%)
C	IV, III, II, VII, V, VI, I	IV, V, VI, III,II, VII, I	IV, III, II, VII, V, VI, I	IV, VII, III, II, VI, I, V	IV, III, II, VII, V, VI, I	5/5 (100%)
D	V, VI, II, IV, III, VII, I	IV, VII, III, V, II, VI, I,	V, IV, VI, VII, III, II, I	V, III, IV, VI, II, VII, I	V, VII, VI, IV, II, I, III	4/5 (80%)
E	VII, VI, II, V, IV, III, I	VI, V, IV, VII, III, II, I	VI, VII, IV, V, II, III, I	VI, III, IV, VII, V, I, II	VI, VII, V, III, IV, II, I	4/5 (80%)
					Valor medio	(21/25) 84%

Los mejores resultados fueron obtenidos en el caso de usuarios con el parámetro *Edad* igual a *Adulto Maduro* (sujetos B y C) de forma independiente de los otros dos parámetros en juego (*Percepción Visual* y *Sexo*). El grado de satisfacción en la adaptabilidad conseguido por los sujetos B y C fue 100% ya que ambos seleccionaron la plantilla diseñada por DEVA como primera elección en todos los casos. Los resultados con éxito intermedio se cosecharon para los usuarios más jóvenes en donde se alcanzaron grados de satisfacción del proceso de adaptación del 80%. Los peores resultados se obtuvieron con el sujeto A, el cual sólo alcanzó un grado de satisfacción del 60% en las cinco variaciones cromáticas. Es

necesario destacar que en las dos ocasiones en las que éste sujeto no seleccionó como preferente aquella plantilla adaptada por DEVA (II), seleccionó sin embargo la plantilla VII, aquella adaptada para situaciones especiales de obscuridad, dejando la plantilla II como segunda elección. Una posible motivo para seleccionar la plantilla II (la adaptada para este sujeto en concreto) y la plantilla VII (la adaptada para situaciones extremas de obscuridad) podría ser el valor Anciano del parámetro Edad, dado que las personas de edad avanzada tienden a ver mejor en situaciones de obscuridad. Obviamente, en este sentido será necesario repetir experimentos de esta naturaleza con muestras más amplias para este tipo de usuarios concreto antes de aventurar conclusiones definitivas.

Analizando las respuestas proporcionadas por los voluntarios (recogidas en la Tabla 45), se puede comprobar como la plantilla I, es decir, el diálogo interactivo tal cual, sin adaptar, es seleccionado por todos los usuarios en último lugar salvo en tres de los veinticinco casos planteados, lo cual representa un 88% de los casos. En el 12% de los casos restantes, la plantilla I se encuentra entre las tres últimas preferencias del sujeto (de siete posibles). Estos valores no solo demuestran que el proceso de adaptación planteado es adecuado sino que además es necesario, ya que en la prueba descrita, el usuario rara vez selecciona la interfaz estándar (I) creada por el diseñador de la aplicación.

Por el momento es imposible determinar si la desviación existente entre los resultados esperados y los resultados obtenidos se debe la incapacidad del sistema para adaptarse completamente a las encespedas cognitivas, perceptivas y motrices de sus usuarios a un desajuste en los parámetros internos del sistema, principalmente en la definición de los valores frontera de los distintos conjuntos difusos empleados, aunque dado el elevado índice de adaptación obtenido, se sospecha lo primero.

En este sentido se plantea incorporar la tecnología Entruger en combinación con el módulo ANTS para poder realizar pruebas de usabilidad remotas automáticas de los mecanismos de adaptación de GADEA y poder así ajustar al máximo los valores de los parámetros empleados. De este modo, los propios agentes ANTS serían capaces de modificar la base de conocimiento del motor de inferencias de DEVA, adaptando las fronteras de los conjuntos difusos empleados por varias de las variables lingüísticas que participan en el proceso de adaptación, disminuyendo así el margen de error en dicho proceso.

En estas pruebas de usabilidad realizadas con pequeñas muestras, ha quedado patente que todos los usuarios muestras una clara predilección por los diálogos interactivos adaptados con relación a los diálogos no adaptados. Además, en una elevada proporción de los casos (84%) los usuarios seleccionan el diálogo interactivo expresamente adaptado por DEVA para ellos como el diálogo idóneo. Dada la inexistencia de un sistema de gestión de interfaces de usuario auto-adaptables tan completo como el presente, ha sido totalmente imposible realizar una evaluación integral con otros sistemas para poder cuantificar de algún modo el nivel de calidad de la adaptación realizada por GADEA con

respecto a dichos sistemas. Obviamente, los resultados obtenidos no demuestran en ningún caso que el proceso de adaptación realizado por GADEA sea perfecto, ni tampoco que sea el más adecuado para un usuario concreto, pero si demuestra sin embargo que dicho proceso de adaptación es efectivo, potente, novedoso y sobretodo demuestra ser cualitativa y cuantitativamente superior a otros procesos de adaptación sobre la sintaxis de una interfaz.

23

23 La Tecnología

No hay nada que avance más rápido que un científico con dinero

Juan José Guadiola

23.1 Factor de Adaptación

La calidad del proceso de adaptación alcanzado por una interfaz generada a medida puede evaluarse en función de los distintos aspectos de la interfaz que son susceptibles de ser adaptados, así como de la efectividad del proceso de adaptación, es decir, en que medida la adaptación del canal de comunicación se adecua a las necesidades manifestadas explícita o implícitamente por los diferentes grupos de usuarios del mismo.

En cuanto al primer parámetro de evaluación, se ha podido comprobar a lo largo de este documento como el número de elementos adaptados por GADEA en un canal de comunicación supera ampliamente al número de los elementos adaptados de forma individual por otros sistemas inteligentes (ver capítulo 2: *Interfaces Inteligentes*). También se ha podido constatar como la calidad del proceso de adaptación es superior al realizarse todo el proceso de

adaptación en función de un modelo cognitivo único, basado en el modelo de procesador humano (capitulo 13: *El Procesador Perceptivo*) y en la TUC (capítulo 16: *Un Modelo de Evaluación*). Este modelo, de fiabilidad fuertemente contrastada, no solo por medio de la presente investigación sino también a través de la labor realizada en el campo de la psicología cognitiva por varios investigadores citados oportunamente en esta obra, permiten una solución integral de los problemas de adaptación planteados en distintos puntos de la interfaz, de tal modo que el proceso de adaptación sea coherente en todos los componentes del sistema de comunicación.

La arquitectura diseñada en base a este modelo cognitivo permite capturar la información relevante en el lugar y momento adecuado para responder más tarde a los requerimientos implícitos o explícitos de adaptación teniendo en cuenta factores tales como la edad del usuario, su sexo, su habilidad motriz, la capacidad de su memoria episódica o el estado de su visión entre otros factores. Por otro lado la flexibilidad del modelo permite adaptar prácticamente cualquier aspecto de la interfaz en los niveles cognitivo, perceptivo y motriz. Tal y como se ha podido reconocer a lo largo de toda esta obra, GADEA es capaz de gestionar y adaptar los siguientes elementos de una interfaz:

- **Widgets Inteligentes Auto-adaptables**. Los componentes de interacción y despliegue de información de DEVA, estratifican la información proveniente de CodeX en varias fases de comunicación en función de la capacidad cognitiva del usuario, así como de su grado de veteranía en el empleo del *widget* y del contexto en el que éste se desenvuelve.

  Componentes del tipo de grupos de *checkboxes*, botones de radio, botones mutuamente excluyentes, menús, etc. distribuyen la carga de información a transmitir en varias secuencias para no sobrepasar la capacidad del procesador perceptivo del usuario en cuestión. Este proceso de adaptación evoluciona a lo largo de una sesión de trabajo a medida que la capacidad del procesador disminuye debido al cansancio y a la fatiga.

  A medida que el usuario estructura el modelo mental equivalente en su memoria a largo plazo por medio del uso continuo de determinados diálogos interactivos y/o procesos de usuario, su desenvolvimiento en el intercambio de información mejorará, evento detectado por los agentes ANTS y comunicado a los *widgets* implicados para reducir así el número de fases de comunicación empleadas.

- **Interfaz Multicanal Auto-adaptable.** La transmisión de información en GADEA no se restringe al empleo de técnicas multimodales sino que el conocimiento es estratificado en varias capas en función de su importancia y de se relevancia para el usuario, así como de la prioridad asignada al mismo por parte de

los diseñadores del diálogo interactivo a través del cual dicha información es transmitida.

En el proceso dinámico de adaptación de los canales de comunicación empleados, DEVA desvía la información poco relevante a través de canales de comunicación secundarios, los cuales emplean técnicas de transmisión de mensajes subliminales para no afectar al desarrollo de las tareas que se están ejecutando en el canal de comunicación principal.

- **Adaptación Automática de Contenidos**. GADEA no solo reconoce de forma automática los procesos de usuario, diálogos interactivos, canales de comunicación y otros definidos por los diseñadores de una aplicación sino que adapta de forma automática la carga semántica de estos elementos por medio *del Language Manager* de CodeX. Esta adaptación se aplica además sobre aspectos muy concretos del nivel sintáctico de la interfaz, incluyendo por ejemplo la representación iconográfica de ciertos conceptos, las descripciones de ayuda rápida, las notaciones empleadas para usuarios con discapacidades visuales o las combinaciones vocales empleadas en los códigos de transmisión diseñados para usuarios con discapacidades auditivas entre otros.

- **Gestión Automática de Tareas de Vigilancia**. Este sistema de gestión de interfaces facilita que los usuarios se concentren su atención en las tareas a realizar en el canal de comunicación principal mientras que tareas de duración extrema se ejecutan en procesos de aplicación y en procesos de usuario en paralelo por medio de hilos de baja prioridad.

  Cuando uno de estos procesos finaliza su ejecución o debe notificar algún resultado al usuario, DEVA gestiona de forma automática el proceso de captura de la atención del usuario mediante las técnicas descritas en su momento, eliminando con ello la sobrecarga y fatiga del procesador perceptivo del operador al suprimir tareas de atención continua en toda interfaz generada.

- **Individualización de *Widgets*.** Adaptación de la disposición de elementos en un diálogo interactivo para facilitar la percepción individual de determinados componentes o grupos de componentes en función del nivel de prioridad asignado a los mismos por los diseñadores del diálogo. Para ello se adapta automáticamente el contraste existente entre dos objetos vecinos en el canal de comunicación activo. Esta variación de contraste se consigue mediante la evaluación y variación de los elementos cromáticos de los componentes (caso del canal de comunicación visual), su separación o agrupación con otros componentes, su disposición espacial, etc.

Conclusiones

- **Estratificación de Contenidos (señuelos y predadores)**. Optimización del ancho de banda real efectivo de un canal de comunicación mediante el control automático de la información a transmitir en función de la relevancia que la misma tiene para el usuario. Para ello se estratifican los contenidos en capas dependiendo de la prioridad asignada a los mismos por los diseñadores de la aplicación y detectada directamente por CodeX en el código binario de la misma.

  Los objetos prioritarios (señuelos) son transmitidos primero para copar el ancho de banda efectivo con información útil pero de escasa carga semántica, para substituirlos en posteriores fases de la transmisión por menor objetos pero dotados de una mayor carga semántica (predadores. El ancho de banda efectivo se calcula dinámicamente en función del ancho de banda real del canal de comunicación, de la capacidad del procesador perceptivo del usuario en el momento en el que se efectúa la transmisión y de su experiencia en el empleo del diálogo interactivo.

  El mecanismo de transmisión en varias fases y la consecuente estructuración de la información en capas semánticamente relacionadas entre sí se aproxima al modelo de asociación de *chunks* empleado tanto por la memoria semántica como por la memoria episódica para la creación de las claves de búsqueda.

- **Selección de *Widgets* Adaptada a un Usuario Concreto**. Basándose en el modelo GOMS, el agente interactivo DEVA descompone los pasos necesarios para la ejecución de una determinada tarea (dentro del marco de un modelo de interacción concreto) en aquello operadores necesarios para efectuar la mismas, seleccionando aquellos que mejor se aproximen a las características cognitivas, perceptivas y motrices de un usuario concreto del sistema de gestión de interfaces. Esta funcionalidad permite entre otras cosas seleccionar dinámicamente el modelo de interacción a emplear, así como los *widgets* que se usarán en el mismo.

  Este enfoque esta orientado a aumentar la productividad global del sistema ya que facilita una rápida ejecución de las tareas previstas en el mismo por parte del usuario, pero se puede adaptar fácilmente a un modelo que premie la flexibilidad y la adaptación al modelo de interacción favorito del usuario, ya que las funciones de evaluación empleadas por DEVA para realizar la selección de componentes, se pueden ponderar en función del historial de actuación de un determinado usuario con el sistema GADEA.

  Este planteamiento permite además adaptar fácilmente el proceso de evaluación y selección de componentes para usuarios con valores extremos para sus parámetros cognitivos, perceptivos y motrices extremos, ya que basta con asignar valores extremos o imposibles a los correspondientes valores de los argumentos que participan en las funciones de evaluación o desactivar el proceso de ponderación, de tal modo que en el proceso de selección sólo sean

favorecidas aquellas técnicas de interacción que puedan aplicarse correctamente al usuario en cuestión.

- **Adaptación de Canales de Comunicación a la Diversidad**. El conocimiento que el sistema de gestión de interfaces tiene del usuario gracias a la existencia de un modelo de usuario que se actualiza constantemente, permite adaptar diversos parámetros de todo canal de comunicación en función de diversas características del usuario, como la edad, el sexo, la precisión auditiva, visual, etc. Los parámetros de adaptación pueden ser genéricos para un colectivo o únicos para un individuo concreto, en función de determinadas características, todo lo cual permite un diseño dinámico ad hoc de los diálogos interactivos empleados por DEVA.

  El empleo de un motor de lógica difusa para la toma de decisiones por parte de éste agente interactivo permite eliminar la ambigüedad del lenguaje natural en el que están escritas la reglas de adaptación a la vez que permite efectuar dicho proceso de adaptación de un modo gradual, sin brusquedades dado que el paso de un estado del proceso de adaptación al siguiente se realiza a través de variables difusas en lugar de emplear variables discretas. De este modo, el paso de un nivel de adaptación a otro se difumina en una gran serie de pasos intermedios en lugar de efectuar un único cambio brusco.

  No obstante, la mayoría de los valores de los parámetros implicados en los procesos de adaptación de una interfaz (configuración de los objetos principalmente) se realizan una vez por sesión, permaneciendo constantes a lo largo de la misma. Esta práctica no solo garantiza un nivel de eficiencia óptimo sino que erradica la existencia de posibles problemas de usabilidad dentro del marco de una misma sesión.

- **Distribución Espacial y Temporal de Widgets Adaptable**. Por medio del empleo de metáforas de orientación fuertemente contrastadas a través de diversos procesos experimentales, el agente interactivo DEVA es capaz de distribuir los diferentes *widgets* que componen un diálogo interactivo en función del tipo de usuario activo, tanto en base a parámetros genéricos como su entorno cultural o en función de parámetros específicos como por ejemplo su lateralidad.

  En función del grado del tipo de comunicación soportada por un *widget*, éstos son clasificados en objetos activos (comunicación dúplex) o pasivos (comunicación *half* dúplex). Mientras que la ubicación de los segundos depende exclusivamente de aspectos relacionados con la percepción de contenidos (semántica), para la ubicación de los segundos se tienen en cuenta además aspectos relacionados con el sistema motriz del usuario, dado que éste debe acceder a estos objetos mediante dispositivos mecánicos (ratón, joystick, señalización directa, etc.) la lograr una interacción plena.

- **Adaptación Automática de Patrones Cromáticos**. En base al patrón cromático especificado por los diseñadores de la aplicación en función de condicionantes semánticos el agente interactivo DEVA es capaz de generar nuevas gamas cromáticas (respetando el patrón original) para generar valores que permitan conseguir una individualización efectiva de los componentes desde el punto de vista de la percepción. Esta individualización emplea técnicas novedosas de adaptación de los valores de brillo en los colores, valores empleados para conducir la atención consciente del usuario hacia determinadas áreas de un canal de comunicación visual.

  Por otro lado, DEVA adapta los patrones cromáticos resultantes en función del estado de determinadas características perceptivas de los usuarios, tales como la edad, la precisión visual, etc. Para ello, se modifica la proporción de la cantidad de azul presente en determinados colores garantizando una correcta percepción tanto en los colores empleados para las formas (colores cálidos principalmente) como en los colores usados en los fondos (colores fríos en su mayoría), todo ello naturalmente de forma automática.

  Este proceso de adaptación puede afectar a otras características de un canal de comunicación visual, como por ejemplo la configuración espacial de determinados objetos con el fin de garantizar las mejores condiciones posibles de percepción en base a los colores elegidos, así como el máximo grado de contraste visual posible.

  El modelo de adaptación cromático empleado por GADEA prevé el empleo futuro de dispositivos de medición de las condiciones de luz del entorno de trabajo del usuario, con el objeto de incluir este parámetro en el proceso de adaptación. De este modo, la adaptación se realizaría teniendo en cuenta no sólo las características individuales del usuario, sino también el entorno empleado por este para trabajar con la aplicación.

Gracias precisamente a la flexibilidad del modelo cognitivo en el que se basa la arquitectura de GADEA, es posible añadir nuevos factores al conjunto de elementos rastreados por el módulo ANTS, así como nuevas reglas y estructuras de conocimiento al módulo DEVA sin que la estructura del sistema de gestión de interfaces se resienta. Dado que tanto las operaciones de captura de información como la posterior adaptación de los canales de comunicación en base a la información capturada es realizada por un pequeño conjunto de agentes software, es posible añadir funcionalidad al sistema por medio de nuevos agentes o modificando el comportamiento de los ya existentes.

La separación entre funcionalidad e interfaz conseguida por medio del módulo CodeX permite la posibilidad de refinar el comportamiento de los agentes interactivos y de captura de información de forma evolutiva y constante a lo largo del tiempo sin que con ello se afecte en lo más mínimo a la funcionalidad de las aplicaciones que se ejecutan

bajo el entorno GADEA. De este modo, las aplicaciones que emplean este sistema pueden beneficiarse de una mejora en la adaptabilidad de sus interfaces a medida que se incorporan nuevos agentes al sistema sin que por ello sea necesario realizar ninguna modificación en el código de dichas aplicaciones.

23.2 Ámbito de Aplicación de GADEA

A lo largo de los capítulos que componen esta obra se han ido planteando las distintas características que la arquitectura GADEA ofrece para la adaptación automática de distintos tipos de canales de comunicación, en distintas situaciones y sobre distintos tipos de diálogos interactivos. Por motivos de economía, el proceso de investigación que ha dado lugar a esta obra se ha centrado fundamentalmente sobre el modelo de interacción WIMP empleado principalmente por la ingeniería del software en el desarrollo de aplicaciones netamente orientadas al comercio a gran escala. Es por ello patente que el espacio de trabajo natural para los prototipos de GADEA desarrollados hasta el momento se centre en este sector, tanto en su vertiente clásica como en su versión Internet.

El alto grado de adaptabilidad de los diálogos interactivos diseñados estáticamente por la ingeniería del software permite una amplia difusión de los mismos en cualquier contexto cultural, dado que se ha tomado especial cuidado en que tanto la base de conocimiento de DEVA, como las reglas empleadas por su motor de lógica difusa, sean independientes de procesos cognitivos superiores. De este modo, el proceso de adaptación de los diálogos interactivos se realiza a un nivel puramente sintáctico, siendo de interés para prácticamente cualquier tipo de usuario. En todo caso, la flexible arquitectura de GADEA permite incorporar nuevas características de adaptabilidad para usuarios previamente no contemplados de forma fácil y eficaz.

Este enfoque de evolución constante permite incrementar la eficacia y potencia del proceso de auto-adaptación de cualquier aplicación desarrollada con GADEA por el mero hecho de actualizar la versión de su sistema de gestión de interfaces de usuario sin que sea por ello necesario tocar ni una sola línea del código de la aplicación. El beneficio potencial que proporciona la implantación de un sistema de este tipo como UIMS en un sistema operativo es desde luego enorme ya que el poder de adaptabilidad de la interfaz de usuario de todas las aplicaciones desarrolladas para dicha plataforma se vería incrementado con cada nueva versión de GADEA. Es precisamente el papel de sistema de gestión de interfaces de usuario el marco de actuación natural de GADEA ya que inicialmente fue pensado como UIMS del sistema operativo Oviedo-3, un sistema operativo desarrollado integralmente con tecnologías orientadas a objetos [Martínez et al (2000), Martínez et al (1998) y Tajes et al (1998)].

El empleo de GADEA como UIMS de un sistema operativo proporcionaría sin duda notables ventajas a ambos sistemas. Mientras que el segundo obtendría un mecanismo de adaptabilidad eficaz y

sencillo para sus aplicaciones, el primero se beneficiaría de la codificación de parte de sus rutinas de bajo nivel en capas cercanas al núcleo del sistema operativo con lo que se ganaría en eficiencia, especialmente en el uso de los agentes recolectores de información y en la capa de bajo nivel del motor de inferencias. La conversión de buena parte del código de GADEA (escrito para la máquina virtual de Java) en código nativo para una plataforma específica redundaría sin duda en una mejora de la velocidad de procesamiento de este UIMS, dejando un margen más amplio de recursos de ejecución para los procesos privados de las aplicaciones que hacen uso de GADEA. Otra importante ventaja relacionada con el uso de GADEA como UIMS de un sistema operativo es el rápido adiestramiento y captación de nuevos usuarios para el sistema. Como resulta obvio, para aquellos usuarios que conocen el modo de operación de GADEA de otras aplicaciones les resultará más sencillo trabajar con la interfaz de aplicaciones novedosas para su esquema mental, haciendo posible el reciclaje de *chunks* de conocimiento de estos usuarios.

Dada la adopción del modelo de interacción WIMP por parte de la versión actual de los prototipos de GADEA, son las aplicaciones de gestión las principales beneficiarias del poder de adaptación proporcionado por este sistema operativo, puesto que la mayoría de los patrones de diseño del modelo de navegación de este tipo de aplicaciones se asemejan en gran medida al patrón *ping-pong* seguido en la construcción de formularios [Green y Jacob (1991 y Rouff (1996)]. De acuerdo con este patrón el usuario imparte órdenes o solicita un determinado tipo de información al sistema, el cual confecciona la respuesta y se la proporciona al usuario. En ocasiones, para poder confeccionar la respuesta, es posible que la aplicación requiera información por parte del usuario, inquiriendo a éste directamente por ella. En la metáfora que a nombre a este patrón, la pelota (el control de la ejecución) pasa de un lado a otro de la mesa en función de la lógica definida por los requerimientos del sistema. Este es precisamente el mismo esquema adoptado por CodeX a la hora de gestionar el flujo de control de la funcionalidad de una aplicación y asignar cuotas de trabajo, distribuyendo el control de forma alternativa entre el usuario y la aplicación de forma automática y transparente tanto para el usuario como para el programador de la aplicación. Nótese que en GADEA, esta distribución del control de la interacción puede ser tanto síncrona como asíncrona, puesto que el sistema proporciona soporte a la ejecución paralela de determinados procesos de usuario y de no usuario cuando éste actor dispone del control de la interacción.

Existen multitud de aplicaciones de ofimática y de gestión en donde éste patrón es empleado para definir el modelo de interacción principal. El caso de los formularios de inscripción en una actividad, la obtención y consulta de informes, la configuración de equipos y programas, la ejecución de cálculos avanzados o el propio esquema de diseño de una arquitectura cliente servidor son paradigmas clásicos de este modelo de interacción en el que GADEA puede dar excelentes resultados.

Otro campo de aplicación que puede beneficiarse enormemente de las aplicaciones construidas con tecnología GADEA es del los

desarrollos multimedia basados en hipertexto. Cuando se describió el módulo ANTS (a partir del capítulo 19: *Espías*) pudimos analizar que uno de los mayores problemas del desarrollo de toda aplicación hipermedia radica en una concepción genérica de su modelo de navegación, en especial en la detección en tiempo de diseño de aquellos aspectos sintácticos de una interfaz que pueden servir de puntos de referencia y/o de rutas en los modelos mentales construidos por los usuarios cuando hacen uso de una de estas herramientas. Es precisamente en este aspecto en donde destaca una de las características más versátiles y flexibles de GADEA, ya que este modelo es capaz de obtener de forma automática ingentes cantidades de información individual para cada usuario acerca de sus preferencias de navegación, información que es luego utilizada para seleccionar el tipo y cantidad de información a ser desplegada en un momento dado, adaptando el dialogo interactivo en el que ésta se distribuye en función del historial de navegación de cada usuario.

Empleando tecnología GADEA como motor de navegación en aplicaciones multimedia, éstas pueden beneficiarse de una adaptación rápida y automática de los modelos de navegación teniendo en cuenta parámetros como la experiencia del usuario, su fatiga, los puntos de referencia empleados, etc. permitiendo descomponer la cantidad de información a distribuir en *chunks* del tamaño adecuado para el usuario dado, agrupándolos posteriormente en chunks de menor tamaño físico (a nivel sintáctico) pero de mayor carga semántica a medida que el usuario gana soltura con el empleo de la aplicación. Esta tecnología permite una aproximación gradual a los contenidos por parte de un usuario inexperto tanto en el uso de las técnicas de navegación en mundos virtuales como en la naturaleza de los contenidos manejados por la aplicación multimedia. Herramientas educativas, portales web y quioscos interactivos son algunas de las aplicaciones que pueden beneficiarse del nivel de adaptación semántico proporcionado por este sistema de gestión de interfaces de usuario.

23.2.1. Ventajas

A un nivel sintáctico, el elevado grado de adaptación proporcionado por GADEA permite obtener un máximo rendimiento aplicando esta tecnología en productos software dirigidos a usuarios con diversos grados de capacidades y/o discapacidades perceptivas o motrices, permitiendo el diseño de canales de comunicación libres de contexto. De este modo los diseñadores de la aplicación pueden concentrar sus esfuerzos en el análisis de los modelos de interacción a emplear desde un punto de vista puramente semántico, definiendo de antemano los distintos entornos culturales a los que va dirigida una aplicación, pero sin realizar ninguna consideración de tipo sintáctico ya que el agente interactivo DEVA garantiza un acceso universal a la información contenida en los mencionados modelos de interacción.

Por medio de pequeñas modificaciones en el modelo de desarrollo de una aplicación, es posible incorporar tecnología GADEA en el desarrollo de aplicaciones auto-adaptables de uso genérico para su

empleo por cualquier tipo de usuario. El coste de esta pequeña adaptación en el modelo desarrollo resulta despreciable si se tiene en consideración los beneficios económicos y sociales que representan la incorporación al conjunto de usuarios de estas aplicaciones a todas aquellas personas que debido a algún grado de discapacidad perceptiva o motriz carecían de los requisitos necesarios para emplear una interfaz diseñada para un usuario *estándar*.

Al respecto destacan multitud de aplicaciones de gestión y enseñanza concebidas para su empleo por usuarios con un elevado nivel cognitivo, perceptivo y motriz. Estas aplicaciones, del tipo de gestores de bases de datos, exploradores de Internet, lectores de correo, etc., se encuentran vedadas completamente para usuarios con cierto tipo de discapacidades ya que las cuantiosas y costosas modificaciones necesarias para adaptar sus canales de comunicación a este tipo de usuarios concreto, hacen impensable este proceso de adaptación en términos económicos. El modelo propuesto sin embargo, permite diseñar este tipo de aplicaciones realizando una abstracción total con respecto al usuario destino de las mismas, permitiendo una adaptación dinámica y automática de sus canales de comunicación, como contrapartida a los procesos de adaptación estáticos y manuales de la estrategia clásica, reduciendo a cero los costes de este proceso. Esta reducción elocuente del coste de adaptación y la generación automática de canales de comunicación para cualquier tipo de usuario permite desarrollar verdaderas aplicaciones todo usuario con beneficios sociales y económicos evidentes.

En el plano social, el empleo de tecnología GADEA en las aplicaciones de gestión empleadas por grandes corporaciones, tanto públicas como privadas, permitirá incorporar al mercado de trabajo a un elevado número de usuarios que sufren de algún tipo de discapacidad. Como muestra ejemplar podemos citar el caso de Asturias en donde según datos de la Consejería de Asuntos Sociales del Principado se encuentran censados más de 110.000 discapacitados (un 10,2% de la población) los cuales se podrían beneficiar de inmediato de esta tecnología [Rodríguez (2000)]. Estos usuarios podrán ahora desarrollar su actividad profesional usando las mismas herramientas informáticas empleadas por compañeros no discapacitados, pudiendo competir con ellos en igualdad de condiciones, al menos en lo que se refiere el uso de estos dispositivos. De forma similar, la capacidad que tiene GADEA para adaptar la representación sintáctica de todo canal de comunicación a determinados requisitos determinados por la *Edad* de un usuario, permitiría alargar considerablemente el período de actividad de este tipo de usuarios en el sector de la informática, mejorando así su calidad de vida.

En la vertiente económica, la apertura de las interfaces de usuario de multitud de aplicaciones comerciales a un extenso rango de nuevos usuarios por medio de la tecnología GADEA supone un incremento notable del número de clientes potenciales para estas aplicaciones y por lo tanto de su posible beneficio económico. Obviamente esta apertura del mercado no sólo esta restringida a la compraventa de software sino también a las implicaciones que esta industria tiene sobre otros sectores

de la economía mundial, en especial el de las ventas de productos y conocimientos a través de Internet por medio de herramientas de comercio electrónico.

El diseño de este tipo de herramientas mediante el modelo de desarrollo propuesto por GADEA no solo permite la incorporación de un inmenso mercado de nuevos clientes a la red global sino que además este diseño es sencillo y extremadamente barato incluso con los prototipos actuales de esta tecnología gracias precisamente a la especialización de estos prototipos en el modelo de interacción WIMP empleado en las transacciones económicas a través de Internet. En este sentido destaca especialmente la aplicación de la tecnología GADEA en el desarrollo de herramientas para la exploración de Internet dotadas de extensiones para el comercio electrónico, proceso que representará una de las posibles líneas de investigación futuras para esta tecnología (ver apartado 23.3: *Líneas de Investigación Futuras*).

La construcción de un explorador de Internet auto-adaptable permitiría un acceso a la red confortable para todos aquellos usuarios que sufren de algún tipo de discapacidad física o que debido a su elevada *Edad* este acceso resultaría penoso en cualquier otra circunstancia. Como paradigma del beneficio aportado por esta tecnología al campo de la comunicación e interacción humana, la posibilidad abierta para que personas ciegas o con serias discapacidades visuales puedan acceder a la extensa red de contenidos que Internet representa. Mediante la combinación entre un explorador dotado de tecnología GADEA y una definición de contenidos de sintaxis pura para Internet basado en el lenguaje ACML definido en este documento es posible adaptar la representación del contenido de cualquier página web a las características cognitivas, perceptivas y motrices de un usuario de Internet, incluso en situaciones en las que dicho usuario sea ciego.

Tal y como se ha comentado en su momento (ACML, definición) el lenguaje ACML no solo permite especificar el contenido libre de sintaxis de cualquier diálogo interactivo, sino que permite además la definición de estructuras de datos básicas (enteros, cadenas de caracteres, hiperenlaces, etc.) capaces de mostrar o de recoger cualquier tipo de información al o desde el usuario. Se puede apreciar claramente el potencia que tiene éste lenguaje para servir de base a la definición de un lenguaje para la definición de páginas web auto-adaptables con soporte pleno para comercio electrónico. Nótese como, al igual que ocurre con cualquier aplicación de gestión diseñada para GADEA, las estructuras de datos básicas soportadas por ACML permitirían enviar y recoger información hacia o desde el usuario a través de una página web auto-adaptable, pudiendo establecer un puente directo entre dicha página web y una base de datos remota, mediante la simple definición de un proceso de usuario en la mencionada página.

Tal y como se puede apreciar, el diseño de modelos de interacción para herramientas de comercio electrónico de interfaz auto-adaptables es tan sencillo y barato como en el diseño de modelos de interacción para cualquier otro tipo de aplicaciones GADEA. El potencial que tiene este campo de aplicación es inmenso ya no solo permite una mejora

notable en la calidad del proceso de transmisión de información entre los agentes comerciales de una aplicación de este tipo (base de datos, motores de búsqueda, etc.) y sus usuarios (gracias al proceso de adaptación automático de los procesos de usuario) sino que permite aplicar este proceso sobre otros usuarios a los que era imposible acceder, como por ejemplo personas con serias discapacidades visuales.

Ahora, Mediante el diseño correcto de los procesos de interacción de alto nivel de una aplicación de (semánticos) y mediante el explorador de Internet apropiado, es posible extender el ámbito de aplicación del comercio electrónico a un extenso abanico de nuevos usuarios.

23.3 Líneas de Investigación Futuras

Los diversos prototipos desarrollados para la evaluación del sistema de gestión de interfaces de usuario descrito en esta obra implementan gran parte de la tecnología cognitiva desarrollada teóricamente por disciplinas como la ergonomía, la ingeniería de factores humanos, la interacción y comunicación humana o la psicología cognitiva entre otras. Sin embargo, debido a la restricción temporal que impone la limitación en los recursos disponibles para esta investigación, tanto en tiempo como en dinero, ha sido imposible integrar en GADEA todos los aspectos del modelo cognitivo disponibles en este momento por la ciencia, tarea que se pretende llevar a cabo con paciencia y tesón durante los próximos años por medio de diversas líneas de investigación, algunas de las cuales ya se encuentran en marcha en el marco del Área de Interacción y Comunicación Humana (HCI) del Laboratorio de Tecnologías Orientadas a Objetos (OOTLab) del Departamento de Informática de la Universidad de Oviedo, que pasaremos a describir a continuación.

23.3.1. Precisión de los Sensores

Esta claro que gran parte del alto grado de adaptabilidad alcanzado por GADEA se debe a la calidad de la información recogida por los agentes hormiga ANTS. A pesar de gran parte de esta información se recoge implícitamente mediante la observación constante del usuario, y por lo tanto su captura se realiza de forma transparente para el mismo, en la versión actual de GADEA aún existen algunos parámetros que han de obtenerse del usuario de forma explícita aún cuando –al menos sobre el papel– estos parámetros pueden obtenerse de forma implícita.

Entre estos parámetros se encuentran la Precisión Visual, la Precisión Auditiva y la Lateralidad del usuario. Actualmente, dentro del programa de investigación Inxena, descrito en su momento dentro del marco de en esta obra, estamos experimentando con técnicas para capturar esta información de forma implícita. En el caso de los dos primeros parámetros, comentamos al ver el *Application Explorer* (ver capítulo 11: *El Application Explorer*) como la utilización de sencillos test del tipo de los empleados en oftalmología para la graduación visual [Saraux y Bias (1972) y Domingo et al. (1988)] en donde el usuario

experimenta de forma individual con una serie de parámetros sobre un tablero de control en el que es posible alinear la *weltanshaung* del usuario con la empleada por DEVA.

Con respecto al factor Lateralidad, actualmente nos encontramos estudiando un método de observación especial basado en la dirección y sentido de los desplazamientos del cursor por la pantalla ante determinadas situaciones y que podría permitir determinar si un usuario es zurdo o diestro de una forma automática y con un grado de error muy bajo.

Esta técnica, aún en fases tempranas de experimentación consiste en determinar el ángulo formado por la trayectoria seguida por el cursor durante el lapso de tiempo inmediatamente anterior e inmediatamente posterior a una pulsación del ratón sobre un *widget* activo, en función de la posición relativa de este *widget* dentro de un canal de comunicación visual. Aún cuando los resultados de este proceso experimental no son concluyentes, se ha detectado que la trayectoria seguida por el cursor en un proceso de pulsación se asemeja al símbolo < cuando el usuario es diestro y el *widget* pulsado se encuentra situado en la zona izquierda del espacio visual, siguiendo una forma similar al símbolo > cuando el usuario es zurdo y el *widget* se encuentra a su derecha.

Si el soporte económico para este proceso de investigación fuese el ideal, el empleo de sistemas hardware EMR (*Eye Movement Recorder*) permitiría incrementar notablemente la precisión con la que actúan los agentes hormiga al poder analizar la refracción en la retina del usuario de ráfagas de rayos infrarrojos que simularían la disposición de los objetos en un canal visual de tal modo que podría detectarse aquellas zonas a las que un usuario en concreto presta más atención con el beneficio evidente de mejorar la calidad de la adaptación al emplear dichas zonas para desplegar aquellos *widgets* a los que el diseñador ha proporcionado los mayores niveles de prioridad.

Fantaseando en la quimera de una situación ideal, se podría solucionar la asignatura pendiente planteada en el capítulo 18 (*Adaptación Cromática*) al poder emplear sensores capaces de medir las condiciones de luminosidad del recinto empleado por el usuario para llevar a cabo su sesión de trabajo, pudiendo entonces modificar plenamente el algoritmo de adaptación cromática planteado mediante un equilibrio adecuado entre estas condiciones de luz y la capacidad perceptiva del usuario.

23.3.2. Pruebas de Usabilidad Automáticas

Cuando se planteó el diseño de ANTS en el capítulo 19 (*Espías*), se pudo apreciar el potencial de este módulo para la realización de pruebas automáticas de usabilidad remotas de cualquier tipo de interfaces, especialmente en aquellas en las que el factor de navegación repercuta especialmente sobre la calidad de las mismas.

Aunque como ya se ha visto, el módulo ANTS ha sido diseñado y desarrollado para poder funcionar por separado al margen de GADEA

y sobre cualquier tipo de herramienta basada en hipertexto (aplicaciones hipermedia, portales web, quioscos multimedia, etc.), la información recabada por ANTS puede resultar un tanto críptica para un usuario no avezado en el tema. Es por ello que se encuentra en marcha un proyecto para dotar al módulo de una interfaz cuya semántica se encuentre adaptada al entorno cultural de los diseñadores de este es tipo de productos, pudiendo realizar análisis automáticos sobre la información recogida, mostrando el resultado de estos análisis de forma condensada pero eficaz.

Al respecto, el proyecto contempla la representación del mapa mental del modelo de navegación para todos y cada uno de los usuarios de una aplicación, sobre el cual pueden representarse fácilmente todos aquellos puntos de referencia calientes, es decir, aquellos puntos de referencia empleados asiduamente por el usuario. Esta información es de probada utilidad para encontrar los temidos cuellos de botella de un modelo de navegación o los aún más peligrosos nodos sumidero a los que el usuario accede pero que es incapaz de abandonar.

Superponiendo los mapas mentales de todos los usuarios sobre el modelo de navegación de la aplicación es posible reconstruir el mapa mental global de todos estos usuarios, el cual permite definir las preferencias de los mismos y modificar el modelo de navegación subyacente con el objeto de mejorar la calidad del mismo en aquellos entornos en los que el nivel semántico de los canales de comunicación de la interfaz no son dinámicamente adaptables.

23.3.3. Refinamiento de los Parámetros de Adaptación

La inexistencia en el momento de escribir estas líneas de una *killer aplication* para GADEA, es decir, una aplicación popular para este sistema que pueda ser empleada por un gran número de usuarios ha hecho imposible la realización de pruebas de usabilidad de la tecnología con grandes masas de usuarios. Esta limitación ha impedido afinar al nivel requerido y deseado parte de los valores asignados a los parámetros de adaptación. Es por ello que, tal y como se indicó en el capítulo 22 (*La Adaptabilidad*), el nivel de adaptación proporcionado por GADEA, aunque necesario no garantiza la perfección.

Con los valores proporcionados por cientos de usuarios el afinado de los valores asignados a los parámetros de adaptación conducirá a resultados más precisos. Nótese que la obtención de estos valores será automática una vez que el sistema se encuentre en marcha a gran escala, gracias naturalmente a la labor silenciosa y automática de los agentes hormiga.

A este respecto, una interesante tarea pendiente para esta investigación es la de modificar los valores de ponderación de ciertas funciones de evaluación y algunas reglas del motor de inferencias para cambiar el objetivo del proceso de adaptación en función del sistema de comunicación en el que se integra una aplicación concreta, pudiendo pasar de un objetivo basado en productividad a un objetivo basado en el ocio, todo ello de forma dinámica, controlada directamente por el

diseñador de la aplicación mediante órdenes al UIMS a través de la clase homónima de CodeX.

23.3.4. Comercio Electrónico Universal

La *killer application* comentada en el apartado 23.2.1 bien podría ser el explorador de Internet basado en tecnología GADEA en el que se trabaja actualmente. Inicialmente, esta previsto que este explorador disponga de una funcionalidad básica formada por un puñado de procesos de usuario que permitan a éste obtener páginas web siguiendo un formato de navegación hipermedia habitual en otros exploradores pero con la salvedad de que la representación final de dichas páginas web pueda ser adaptada dinámicamente a las características cognitivas, perceptivas y motrices del usuario del explorador.

Aunque sencilla, ésta herramienta permitiría el acceso a Internet a usuarios con diversos grados de discapacidad visual, e incluso ciegos, sin que para ello sea necesario realizar adaptaciones ad hoc del aspecto una página web. Para ello, los diseñadores de contenidos deberán escribir sus páginas en una versión especial del lenguaje ACML, definiendo tan solo los contenidos de éstas con pocas o ninguna alusión a su representación visual.

Bajo este enfoque, el proceso de diseño de una página web se asemejaría mucho a la elaboración inicial de contenidos para un libro o para un artefacto hipermedia según el modelo de desarrollo de enfoques metodológicos como OOHDM [Schwabe y Rossi (1995)]. De la siguiente fase, la edición de contenidos y su disposición en el canal de comunicación activo, se encargará obviamente DEVA. De este modo, una misma página web se *visualizará* de forma distinta dependiendo del usuario al que ésta va dirigida. Así, en función de la capacidad de la memoria a corto plazo del usuario y de su grado de experiencia con la página web requerida la cantidad de información a ser desplegada será distinta. Un razonamiento similar se aplica a la configuración de distintos aspectos de la sintaxis de la interfaz tales como la intensidad de los colores empleados, el contraste entre objetos vecinos, el tamaño de los mismos, etc. En este punto es necesario destacar la capacidad de la arquitectura propuesta para gestionar y crear procesos de usuario de forma dinámica ya que para este explorador universal, las sesiones de navegación son tratadas como procesos de usuario (uno por cada página web transferida) de tal modo que el módulo ANTS puede tomar buena nota de las operaciones realizadas por el usuario dentro del marco de acción de cada una de estas sesiones, con el objeto de determinar su veteranía con respecto a la información manipulada en cada sitio web.

Nótese que para expresarnos correctamente, el término *visualizar* debería ser substituido en este contexto por el término percibir, ya que no se garantiza que la página vaya a ser visualizada por el usuario destino. Así por ejemplo, si el usuario que ha requerido la página web presenta una *Precisión Visual* igual a *Ciego*, los contenidos de dicha página serán organizados de tal forma que faciliten su lectura en voz

alta y por lo tanto éstos nunca llegarán a ser visualizados por este tipo de usuario concreto.

Aparte de las sesiones de navegación, el explorador de Internet que se pretende construir debería satisfacer los requisitos funcionales de cualquier otro tipo de proceso de usuario, en especial aquellos derivados del comercio electrónico. Dado que los diseñadores de contenidos de un portal web codifican estos en una versión especial ACML, es por ello factible diseñar diálogos interactivos por medio de este lenguaje, diálogos que serán mostrados al usuario por el propio explorador. Con un mecanismo adecuado para la descarga de las clases que implementan la funcionalidad de os procesos de usuario desde el servidor central en el que se aloja el portal (o desde cualquier otro), sería sencillo poder enlazar la información recogida del usuario en tareas de comercio electrónico y establecer canales de comunicación segura desde la máquina del usuario hacia cualquier servidor, almacenado entonces la información proporcionada por el usuario en una base de datos remota. Nótese que gracias al filtro impuesto por la pareja CodeX – DEVA, no es necesario validar la información proporcionada por el usuario, ya que GADEA garantiza que ésta será siempre correcta, al menos a nivel sintáctico. Si a esto añadimos la verificación semántica que el diseñador del proceso puede imponer en el código que lo implementa, vemos que todo el proceso de verificación y control –o al menos una parte muy significativa– se realiza del lado del cliente, liberando al servidor de esta sobrecarga.

23.3.5. Generador de Interfaces Automático

La característica que presenta CodeX para la separación de la funcionalidad de un aplicación de su interfaz, aunque muy básica es lo suficientemente potente como para permitir un elevado nivel de automatización en el proceso de desarrollo de modelos de interacción. Una muy interesante ampliación de esta arquitectura consiste en el empleo de Codex para generar de forma automática el esqueleto funcional de una aplicación, incluyendo en éste los métodos necesarios para implementar procesos de usuario, gestión de eventos, adaptación semántica de contenidos y validación de información a niveles semánticos superiores a los proporcionados por DEVA. Este esqueleto podría ser generado a partir de unas especificación formal de los procesos de usuario a implementar, así como de los datos requeridos por estos procesos.

Esta nueva funcionalidad para el sistema de gestión de interfaces de usuario posiblemente requerirá de un refinamiento progresivo del modelo de eventos de CodeX para ampliar el alcance de la separación entre funcionalidad e interfaz, así como para facilitar las operaciones de mantenimiento de los procesos de usuario generados mediante esta guisa. De este modo, uno de los objetivos de esta línea de investigación concreta será la de ampliar el campo de acción de CodeX para cubrir aspectos del tipo de la definición del intercambio de información en modelos de interacción no WIMP.

Dado el interés de la comunidad científica internacional [Paterno y Palenque (1996)] por conseguir una estandarización en los modelos de especificación formal de interfaces del tipo de la conseguida en la especificación formal de otros aspectos del diseño de un sistema informático como ha ocurrido con UML, la incorporación del planteamiento de CodeX al estándar y viceversa podría significar un espaldarazo definitivo a esta tecnología a la vez que permitiría detectar posibles problemas o deficiencias en patrones de diseño específicos para modelos de interacción en los que CodeX puede intervenir.

23.3.6. Navegación Multidimensional

En el capítulo 15 (*El Procesador Perceptivo*) se pudo ver como el gestor de conocimiento de DEVA, encargado de clasificar y de mostrar al usuario la información proveniente de Codex emplea como estrategia de comunicación el modelo de objetos señuelo y objetos predador descrito en su momento, descomponiendo el flujo lineal de información en varias fases en función de las características cognitivas, perceptivas y motrices del usuario. Aunque potente, esta estrategia no es la única que se puede aplicar a la hora de generar un modelo de navegación eficiente. De hecho, recientemente hemos descubierto una estrategia de diseño alternativa que puede complementar a la primera para cierto tipo de tareas. A esta estrategia la hemos bautizado con el nombre de navegación multidimensional y está siendo estudiada dentro del programa de investigación Tirsus referido en su momento en este documento [García et al (2001)].

Los proyectos Tirsus III (*El Reino de Asturias*), Tirsus IV (*La Guerra Civil en Asturias*) y Tirsus V (*Los Indianos*), todos ellos relacionados con la enseñanza de la Historia han sido y están siendo el campo de pruebas para este tipo de navegación, la cual tiene unas raíces profundamente afincadas en la enseñanza de esta base de conocimiento concreta. Existen acontecimientos históricos que se caracterizan por estar formados una serie de eventos que suceden en paralelo en un período de tiempo relativamente corto. A este tipo de acontecimientos pertenecen las guerras, revoluciones, crisis políticas, etc. La comprensión adecuada de los sucesos desarrollados durante este tipo de acontecimientos requiere el procesamiento simultáneo de todos los eventos que forman el acontecimiento [García et al (2000a)]. Si se tiene en cuenta que el acontecimiento en cuestión dura por lo general muy poco tiempo (unas horas, unos días o a lo sumo unos pocos meses), el proceso de enseñanza de este tipo de conceptos se complica.

En el caso de una batalla por ejemplo, en cada instante de tiempo se producen simultáneamente multitud de eventos, los cuales definirán a la larga el resultado de la misma. Mientras que un ejército ataca por un por un sector, puede estar siendo atacado por otro. Mientras se bombardea unas posiciones, se puede estar siendo bombardeado en otras. La suma de todos estos eventos y sobre todo, las relaciones entre los mismos, van a determinar el resultado de la batalla.

El alto grado de complejidad del sistema resultante hace muy difícil efectuar una valoración en conjunto de los puntos clave del

acontecimiento mediante el empleo de estructuras narrativas secuenciales del tipo de las manejadas por DEVA. La simple enumeración secuencial de los eventos producidos (perdiendo información sobre sus relaciones) puede llegar a saturar rápidamente la memoria a corto plazo de los usuarios, superando el valor del parámetro μ_{MCP}. Hay que tener en cuenta además que la cantidad de información concentrada en cada evento es considerablemente alta. Volviendo al ejemplo de la batalla, estos datos pueden estar integrados por la identificación de cada unidad militar, su tipo, armamento, mandos, número de efectivos, relación de bajas, posición geográfica, dirección de avance o retirada y unidades contra las que se enfrenta en un determinado momento.

Como se puede observar, la cantidad de información a manejar es muy elevada. La narrativa secuencial no sólo supera con creces el límite de la memoria a corto plazo sino que además dificulta el establecimiento de categorías o de niveles de información que permitan una rápida asimilación de conceptos en la memoria a largo plazo.

El proyecto Tirsus IV, plantea una alternativa a la narración secuencial empleando modelos de navegación en varias dimensiones, que permiten al usuario extraer información de la base de conocimiento de una forma global permitiendo el desplazamiento temporal espacial y estructural por el acontecimiento histórico, facilitando así un análisis global del mismo [García et al (2000)]. Para realizar este prototipo, se ha empleado como base de conocimiento La Guerra Civil en Asturias, recogiendo los acontecimientos bélicos acontecidos entre julio de 1936 y noviembre de 1937. Esta aplicación, distribuida en formato CD-ROM, permite una navegación en varias dimensiones por los frentes de toda Asturias, incluyendo los choques bélicos más importantes de la campaña así como del estado de las unidades militares de ambos bandos.

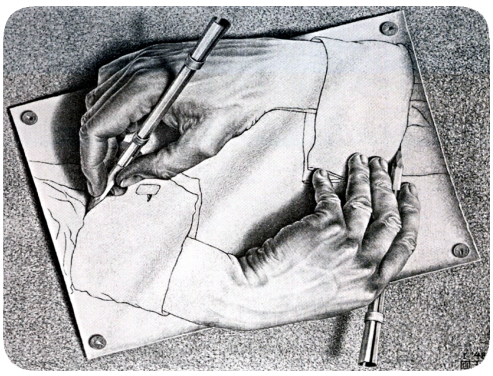
Para conseguir un modelos de navegación en varias dimensiones, se ha procedido a estructurar la base de conocimiento existente, la cual procede principalmente de relatos y testimonios de supervivientes, cronologías de la guerra y archivos fotográficos y de vídeo. Dado que la información se ha obtenido directamente de fuentes orales o de obras escritas, ésta tiene un marcado carácter secuencial. El primer paso ha sido convertir dicha estructura secuencial en un diseño conceptual basado en unidades de información autónomas, lo cual facilita la creación de contextos de navegación en varias dimensiones. En cada uno de los escenarios bélicos planteados mediante contextos de navegación, el usuario puede realizar comparaciones entre los bandos enfrentados, obteniendo informes de cada una de las unidades en combate, así como su posición geográfica en un momento cronológico concreto. La navegación se puede realizar por tanto en tres dimensiones: estructural, temporal y espacial permitinedo obtener datos acerca de las unidades de combate en un momento y espacio concreto.

Como se puede observar, con este diseño es posible navegar por más de una dimensión de forma simultánea, empleando diferentes niveles de detalle. El usuario no está limitado a la realización desplazamientos

temporales (desde el inicio hasta final de una campaña por ejemplo) pues también puede realizar desplazamientos espaciales. En función de las necesidades de información, el usuario puede realizar un zoom sobre la base de conocimiento para analizar con mayor detalle cada uno de los eventos que conforman el acontecimiento histórico.

La técnica de la navegación multidimensional no está limitada a su aplicación sobre bases de conocimiento basadas en acontecimientos de corta duración (como la Guerra Civil Española) sino también en acontecimientos de larga duración como es el caso del Reino de Asturias (Tirsus III) o aspectos tan variopintos como la vida de un personaje (Tirsus V) o la historia de un edificio, tecnología, grupo social o simplemente de una idea y es aquí donde se encuentra la gran ventaja de esta técnica pues ésta se puede aplicar directamente en el modelado de los perfiles de aprendizaje de un usuario sobre cualquier concepto, pudiendo evolucionar en el tiempo, en el espacio o en la estructura de su base de conocimiento.

Lo que comenzó siendo una estrategia de diseño para casos muy concretos, en el momento de escribir estas líneas, se ha convertido en todo un patrón de diseño aplicable a situaciones muy diversas y que posee claras implicaciones en la mejora de los modelos de navegación destinados a productos multimedia educativos y al diseño de fuentes de información en sistemas de gestión de conocimiento. Por todo ello es previsible que un futuro cercano se pueda incorporar la técnica de la navegación multidimensional a GADEA permitiendo al diseñador de una aplicación decidir que estrategia de navegación es aplicable en función de la estructuración de sus contenidos.



Manos Dibujando © 1948 por M. C. Escher

P A R T E V I I

Apéndices y Referencia

24 Imágenes en Color

Cuadernillo con la versión en color de todas aquellas imágenes en blanco y negro incluidas en los capítulos previos de este documento, para cuya correcta interpretación resulta imprescindible el color. Cada imagen conserva la numeración original y se encuentra clasificada de acuerdo con el capítulo al que pertenece.

p. 445

25 Reseñas Bibliográficas

Lista ordenada alfabéticamente y por año de producción de todas las publicaciones a las que se hacen referencia en este documento. Sigue la norma UNE 50-112-04 de AENOR con algunas pequeñas modificaciones para poder incluir referencias a productos software.

p. 463

26 Glosario

Catálogo detallado de los térmicos y acrónimos empleados en este documento, explicando el significado concreto dado en esta obra, el cual puede no coincidir con el empleado en otros ámbitos de la ciencia, sobre todo en términos compartidos con otras disciplinas.

p. 489

27 Índice

Relación (ordenada alfabéticamente) de acrónimos, conceptos, investigadores y tecnologías de las que se hace uso y referencia en este documento.

p. 517

24

24 Imágenes en Color

El arte no reproduce lo visible, sino que hace visible lo que no siempre lo es

Paul Klee

24.1 Cuadernillo en Color

El cuerpo central de este documento ha sido diseñado para ser impreso con la mayor calidad posible por medio de la tecnología aportada por la impresión láser. Debido a consideraciones de tipo económico y teniendo en cuenta sobre todo que más del 95% de las páginas de este documento carecen de motivos, figuras, textos o tablas que requieran de impresión en color, se ha decido imprimir dicho cuerpo central por medio de impresoras láser en blanco y negro, evitando el empleo de costosas impresoras láser en color.

Sin embargo, como se ha podido apreciar a lo largo de la lectura de este documento, existen algunos conceptos cuyo planteamiento y posterior desarrollo se hace muy difícil, cuando no imposible, sin hacer uso de representaciones en color. Obviamente, en una obra de este tipo no se puede renunciar de ningún modo al empleo de la mencionada policromía, recogiéndose dichas representaciones de forma

exclusivamente en este capítulo, el cual es el único de la obra impreso a todo color. De este modo, se ha podido restringir el elevado coste de un proceso de impresión en color a un solo capítulo, sin renunciar al empleo de policromía.

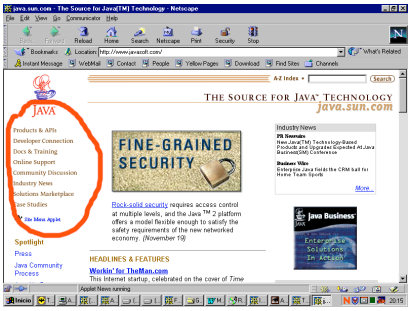
Cuando a lo largo de los capítulos de esta obra se ha necesitado hacer uso de representaciones en color, éstas se han incluido en blanco y negro en lugar que le correspondería por imperativos de la secuencia narrativa pero con una referencia a este capítulo, en el que la imagen se incluye de nuevo, pero esta vez en su versión en color.

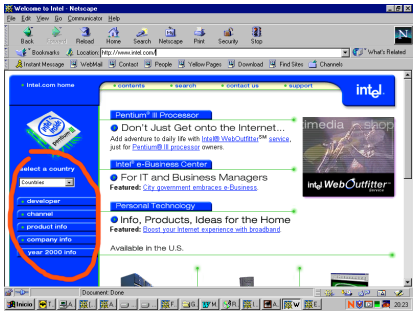
En este capítulo se incluyen las versiones en color de las imágenes en blanco y negro empleadas en los capítulos precedentes, ordenadas de acuerdo con el capítulo en el que aparecieron e incluyendo exactamente el mismo pie de figura, conservando incluso la numeración original. De este modo, este capítulo actúa como un catálogo de todas las imágenes en color incluidas a lo largo de la obra.

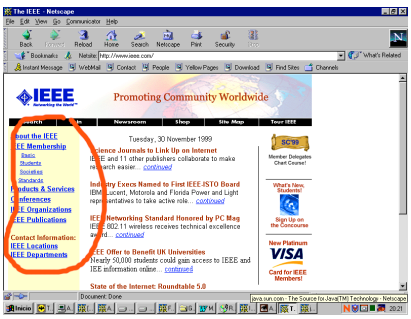
Es necesario destacar que el modelo de color con el que éstas imágenes han sido creadas (modelo RGB o modelo HSB) no coincide con el modelo empleado para su impresión (modelo CMYK) por lo que la apariencia de las mismas puede no coincidir con su apariencia en la pantalla del ordenador, por lo que éstas imágenes pretenden ser tan solo una representación de las originales.

24.2 Las Imágenes

24.2.1. Capítulo 3 (La Individualidad)







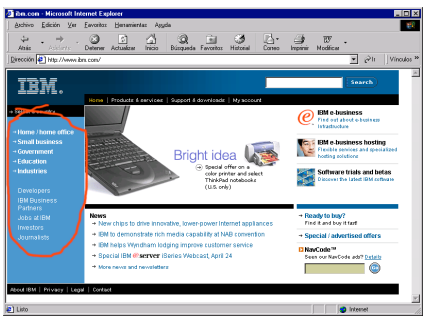


Figura 4. Ejemplo de portales de Internet en donde la barra de navegación (marcada por un círculo rojo) se encuentra situada a la izquierda. De izquierda a derecha y de arriba abajo se puede ver el portal de JavaSoft (http://www.javasoft.com), el portal de Intel (http://www.intel.com) , el portal de IEEE (http://www.ieee.com) y el portal de IBM (http://www.ibm.com), (consultar versión en color en el capítulo 24: *Imágenes en Color*, p. 445).

24.2.2. Capítulo 6 (El Modelo de Interacción)

Figura 13. Posible distribución de los componentes de la Tabla 2 en un diálogo interactivo sin tomar en cuenta las relaciones semánticas entre los mismos. Resulta obvio que este esquema es desastroso en términos de usabilidad

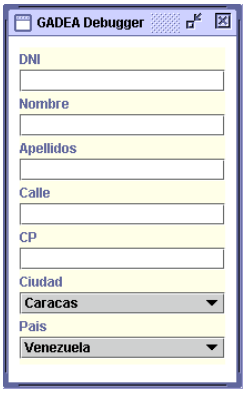


Figura 14. Distribución de los componentes de la Tabla 2 en función de las relaciones semánticas de precedencia y de cohesión que han de existir los mismos

24.2.3. Capítulo 8 (El Diálogo)

Figura 19. Posible construcción del canal de comunicación y diálogo interactivo asociado del Código 11 (consultar versión en color en el capítulo 24: *Imágenes en Color*, p. 445).

24.2.4. Capítulo 9 (La Semántica)

Figura 24. Diálogo interactivo del Código 11. Nótese la numeración y color de fondo distinto para cada *chunk* (consultar versión en color en el capítulo 24: *Imágenes en Color*, p. 445).

24.2.5. Capítulo 11 (El Application Explorer)

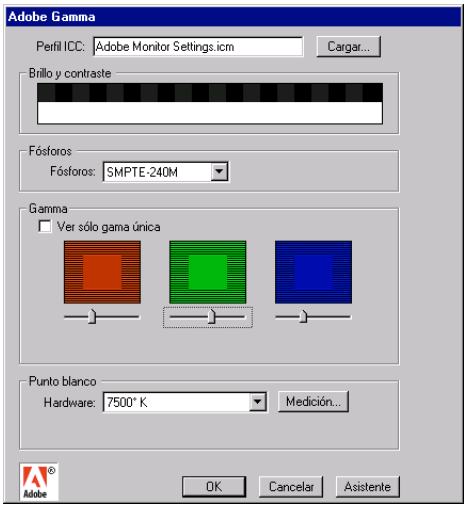


Figura 34. Calibración de un monitor con la herramienta *Gamma* de Adobe Photoshop 5.0. (consultar *versión en color en* el capítulo 24: Imágenes en Color, p. 445).

24.2.6. Capítulo 13 (El Procesador Humano)

Figura 37. Posible diálogo generado a partir del documento ACML del Código 24. La cantidad de *chunks* en el grupo de botones de radio (25) supera ampliamente la capacidad de la MCP de cualquier sujeto

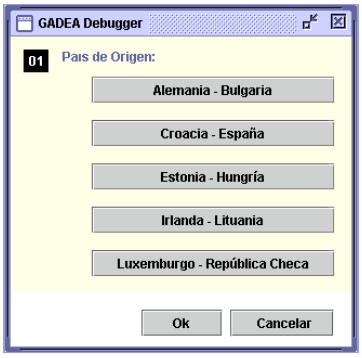


Figura 38. Diálogo adaptado para un usuario con MCP igual a 5 elementos, partiendo del diálogo interactivo incluido en el documento ACML de la Figura 37

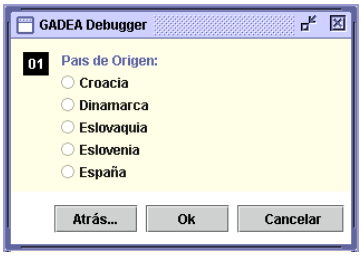


Figura 39. Aspecto del *widget* de la Figura 38 al seleccionarse el segundo botón por arriba. Gracias al mecanismo de adaptación, el número de chunks *empleado* sigue sin superar la capacidad de la MCP del sujeto

24.2.7. Capítulo 14 (La Atención)

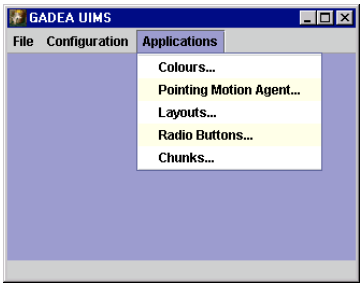


Figura 44. Incremento de contraste visual entre elementos de un menú DEVA para facilitar la búsqueda e identificación de dichos elementos (consultar versión en color en el capítulo 24: *Imágenes en Color*, p. 445).

24.2.8. Capítulo 15 (El Procesador Perceptivo)

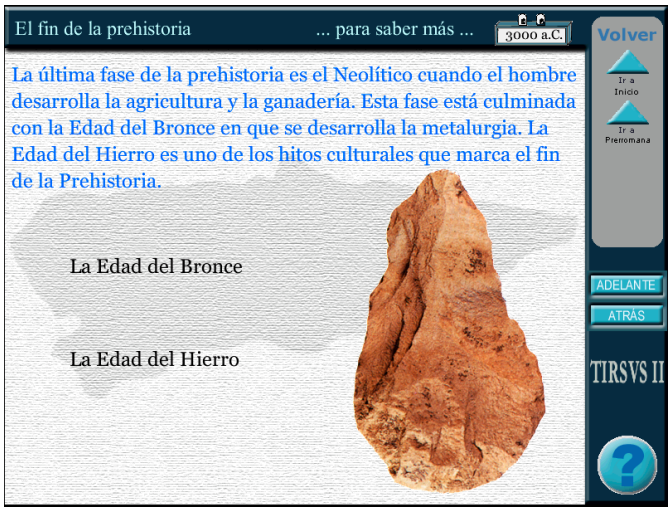


Figura 49. La información visual y la información sonora poco detalladas actúan como objetos señuelo en Tirsus II (consultar versión en color en el capítulo 24: *Imágenes en Color*, p. 445).

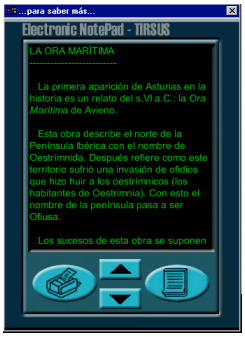


Figura 50. Objeto predador en Tirsus II encapsulado en un canal de comunicación auxiliar (consultar versión en color en el capítulo 24: *Imágenes en Color*, p. 445).

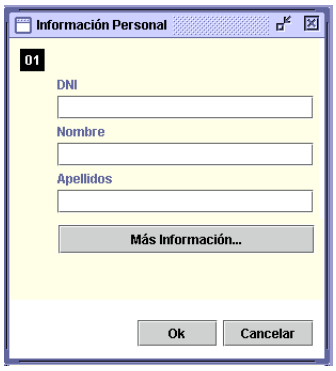


Figura 53. Diálogo interactivo del proceso de usuario *infoPersonal* diseñado para un usuario novato y para una capacidad de transmisión de cinco objetos (consultar versión en color en el capítulo 24: *Imágenes en Color*, p. 445).

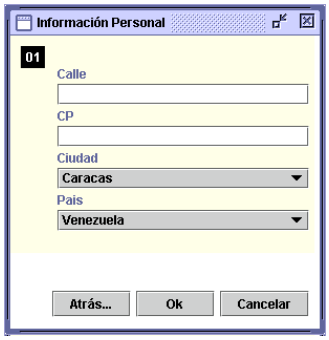


Figura 54. Segunda fase del proceso de comunicación abierto al invocar al proceso de usuario *infoPersonal* (consultar versión en color en el capítulo 24: *Imágenes en Color*, p. 445).

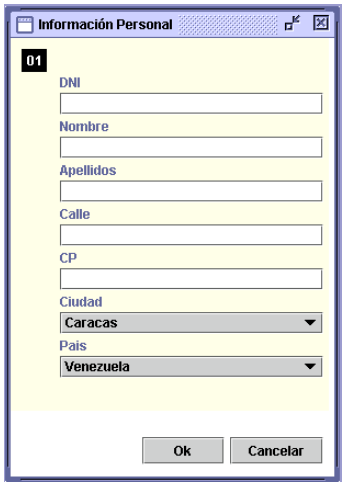


Figura 55. Construcción del diálogo interactivo correspondiente al documento ACML incluido en el Código 26, diseñado dinámicamente para usuarios avanzados (consultar versión en color en el capítulo 24: *Imágenes en Color*, p. 445).

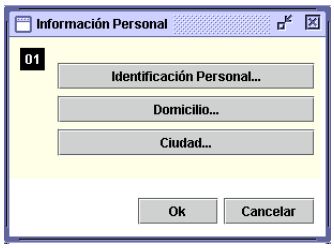


Figura 56. Diálogo interactivo creado a partir del documento ACML del Código 28 para un usuario novato (consultar versión en color en el capítulo 24: *Imágenes en Color*, p. 445).

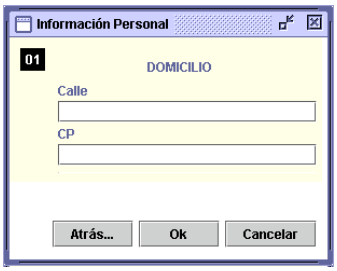


Figura 57. Representación de los objetos predador correspondientes al objeto señuelo *domicilio* de la Figura 56 (consultar versión en color en el capítulo 24: *Imágenes en Color*, p. 445).

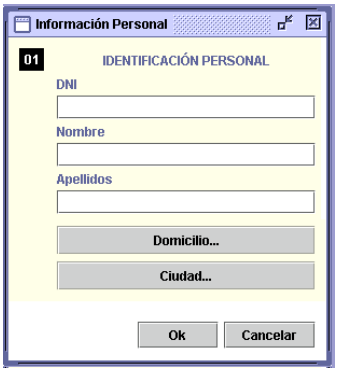


Figura 58. Diálogo interactivo de la Figura 56 adaptado para un usuario con un grado moderado de veteranía (consultar versión en color en el capítulo 24: *Imágenes en Color*, p. 445).

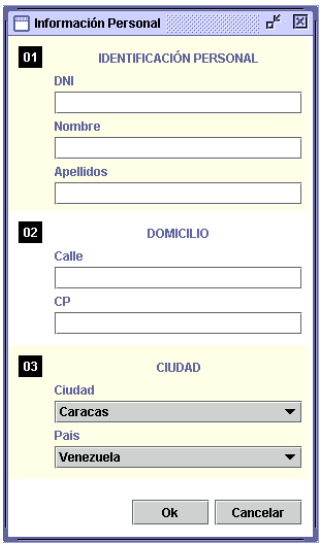


Figura 59. Versión para usuarios veteranos del diálogo interactivo del documento ACML incluido en el Código 28. Nótese como el color de fondo aumenta la cohesión perceptiva de la información contenida en cada *chunk* (consultar versión en color en el capítulo 24: *Imágenes en Color*, p. 445).

24.2.9. Capítulo 16 (Un Modelo de Evaluación)

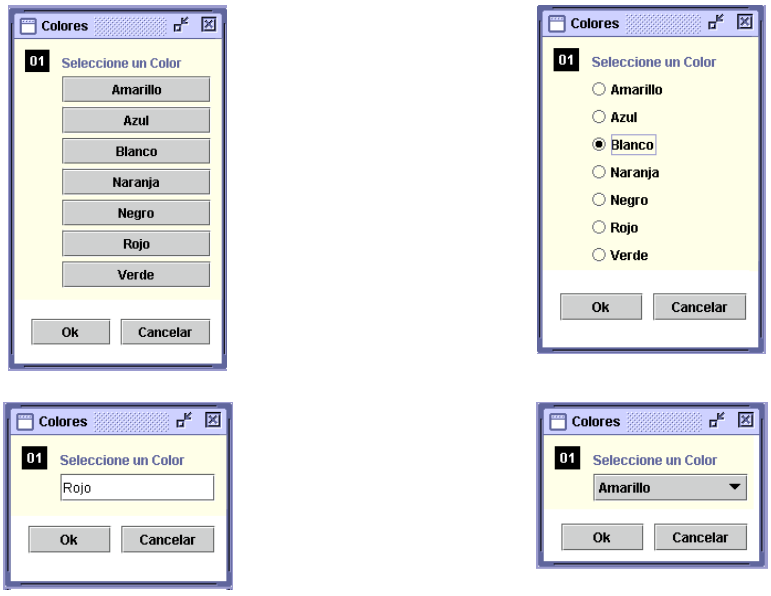


Figura 60. Representación de los cuatro posibles *widgets* candidatos a satisfacer los requisitos de interacción del segundo objeto incluido en el documento ACML del Código 30 (consultar versión en color en el capítulo 24: *Imágenes en Color*, p. 445).

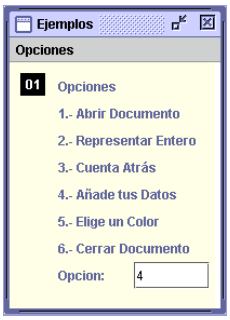


Figura 61. Modo de interacción CLI clásico empleado para satisfacer la transmisión de información determinada por los parámetros del la Tabla 25 (consultar versión en color en el capítulo 24: *Imágenes en Color*, p. 445).

Figura 62. Menú GUI empleado para mostrar las opciones de la Tabla 25. Este menú se puede emplear en combinación con un menú CLI tal y como se puede apreciar en la figura (consultar versión en color en el capítulo 24: *Imágenes en Color*, p. 445).

24.2.10. Capítulo 17 (La Diversidad Humana)

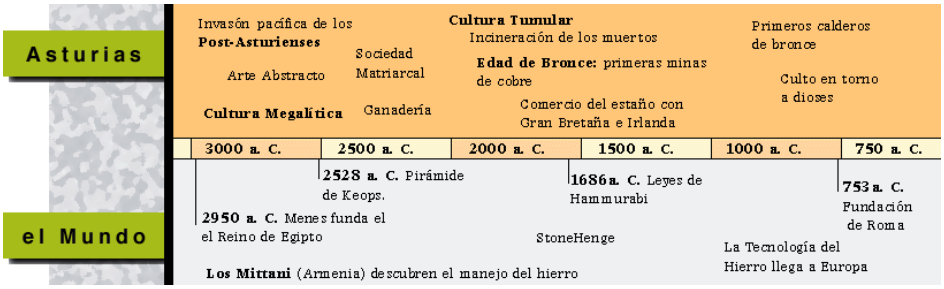


Figura 67. Representación cronológica de la Historia de Asturias. En este caso, a la izquierda se representa el pasado y a la derecha el futuro. [Fuente: ARQUEOASTUR (2001)] (consultar versión en color en el capítulo 24: *Imágenes en Color*, p. 445).

24.2.11. Capítulo 18 (Adaptación Cromática)

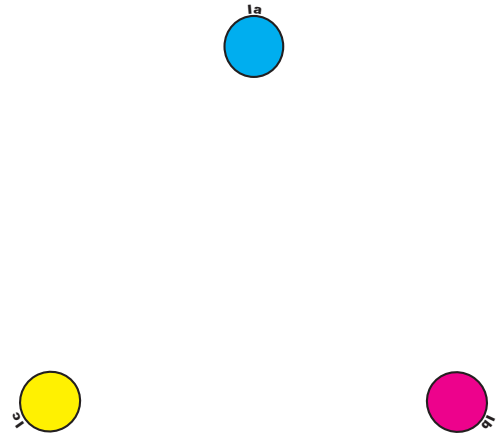


Figura 70. Colores primarios empleados en el modelo de color de Ostwald. Estos colores son el cian (Ia), el magenta (Ib) y el amarillo (Ic). (consultar versión en color en el capítulo 24: *Imágenes en Color*, p. 445).

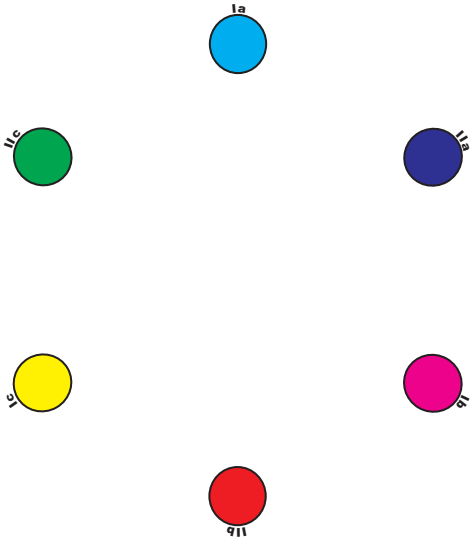


Figura 71. De la mezcla de los tres colores primarios se obtienen tres colores secundarios, a saber, el azul violáceo (IIa), el rojo naranja (Iib) y el verde (Iic). (consultar versión en color en el capítulo 24: *Imágenes en Color*, p. 445).

Figura 72. Representación del espectro cromático de la luz o arco iris formado por los tres colores primarios y los tres colores secundarios (consultar versión en color en el capítulo 24: *Imágenes en Color*, p. 445).

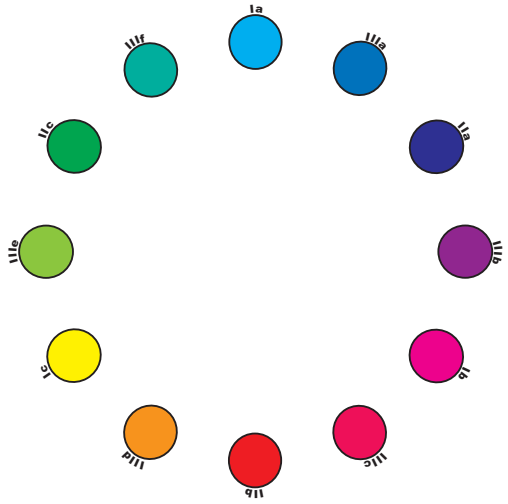


Figura 73. Circulo cromático básico con tres colores primarios, tres colores secundarios y seis colores terciarios (consultar versión en color en el capítulo 24: *Imágenes en Color*, p. 445).

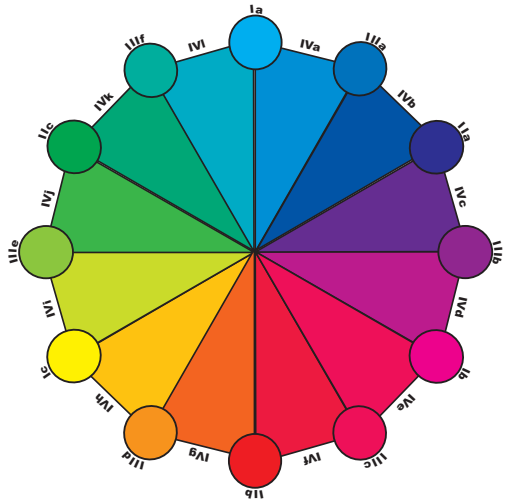


Figura 74. Círculo cromático de Ostwald formado por un total de veinticuatro colores (consultar versión en color en el capítulo 24: *Imágenes en Color*, p. 445).

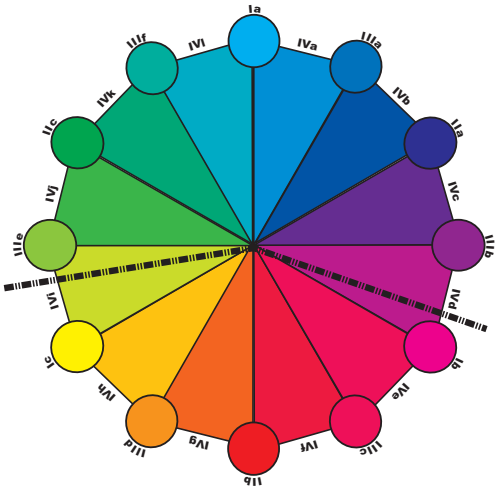


Figura 75. Regiones cálidas y frías del círculo cromático. Todos los colores por encima de la línea divisoria (tonos verdes y azules) se consideran fríos. Los que se encuentran por debajo (tonos amarillos y naranjas) se perciben como colores cálidos (consultar versión en color en el capítulo 24: *Imágenes en Color*, p. 445).

Figura 76. Proceso de adaptación cromática típica para un usuario octogenario y con *Precisión Visual* baja (Eliminación de azul: 40% y *Contraste Visual* del 80%). En la primera fase, se retira el 40% del azul del color de fondo (azul) y del color de la forma (blanco) seleccionado por la aplicación, obteniendo una variante fría del primero (verde) y una variante cálida del segundo (amarillo). A continuación se aumenta el contraste, consiguiendo un color de fondo aún más frío. En la tercera fase se aumenta el tamaño de la fuente al ser el fondo demasiado obscuro y por último se muestran tres variantes de los colores obtenidos en la tercera fase. (consultar versión en color en el capítulo 24: *Imágenes en Color*, p. 445).

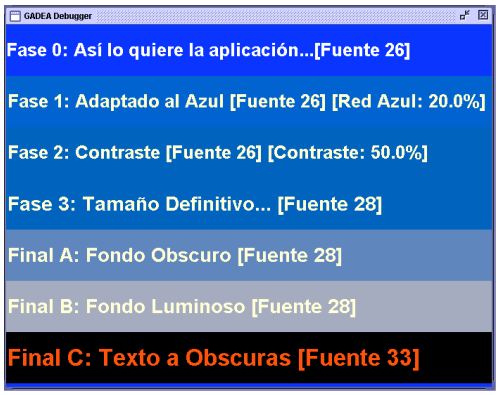


Figura 77. Adaptación visual del ejemplo de la Figura 76 cuando el usuario es más joven y por lo tanto la cantidad de azul a retirar es menor (20%). Nótese como tras el proceso de eliminación de parte del azul en la primera fase de adaptación, los resultados no son tan drásticos como en la Figura 76, ya que el color de fondo sigue siendo azul (en un tono más pálido) y no verde (consultar versión en color en el capítulo 24: *Imágenes en Color*, p. 445).

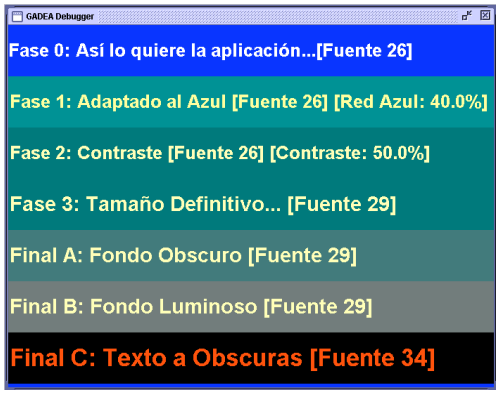


Figura 78. Adaptación cromática de los colores elegidos en la Figura 76 (blanco para la forma y azul para el fondo) para un usuario dotado de una *Precisión Visual* mayor. En este caso el *Contraste Visual* exigido es de tan solo el 50%. Nótese como el color verde claro de fondo obtenido en la primera fase del proceso de adaptación (al retirar un 40% de azul) es convertido en un verde más obscuro para aumentar el contraste con respecto al color de la forma (amarillo). (consultar versión en color en el capítulo 24: *Imágenes en Color*, p. 445).

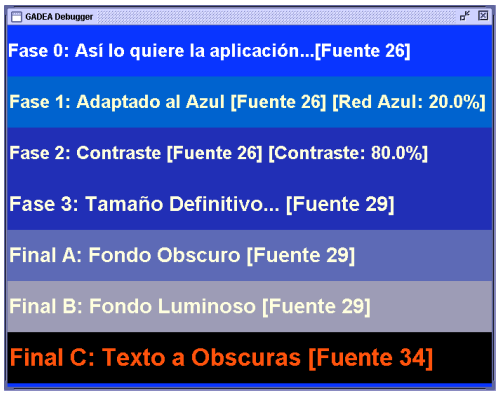


Figura 79. Aumento de contraste para el ejemplo iniciado en la Figura 76, suponiendo ahora que los colores van a ser empleados por un usuario joven (cantidad de azul a retirar igual al 20%) pero con *una Precisión Visual*

baja, ya que requiere un *Contraste Visual* del 80%. Aunque el color obtenido tras la primera fase sigue siendo un tono azul, éste es obscurecido durante esta fase de aumento de contraste (consultar versión en color en el capítulo 24: *Imágenes en Color*, p. 445).

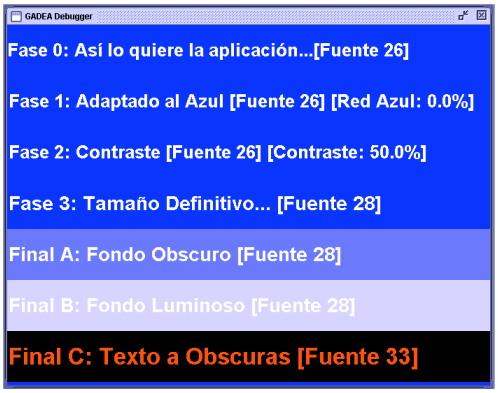


Figura 80. Adaptación cromática del ejemplo de la Figura 76 para un usuario muy joven (en donde no es preciso retirar cantidad alguna de azul) y dotado de una visión relativamente buena ya que solo requiere de un contraste visual del 50%. A pesar de ello, se puede observar como es necesario aumentar el tamaño físico de la fuente tipográfica en la tercera fase, pasando de los 26 puntos originales a 28 puntos. Esto se debe a que el color de fondo (azul) es demasiado obscuro (consultar versión en color en el capítulo 24: *Imágenes en Color*, p. 445).

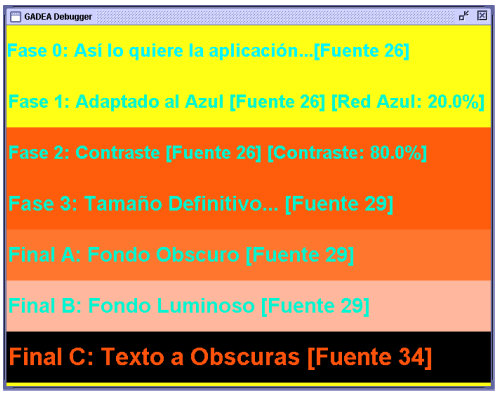


Figura 83. Proceso de adaptación cromática de DEVA ante una situación difícil. El azul claro original de la forma presenta muy poco contraste con respecto al amarillo del fondo. Tras eliminar un 20% de azul y convertir el azul original en un verde claro, el proceso de aumento de contraste cambia el amarillo de fondo por un rojo, el cual es finalmente convertido en un naranja claro y un rosa para su uso como fondo de los *chunks* (final A y final B). (consultar versión en color en el capítulo 24: *Imágenes en Color*, p. 445).

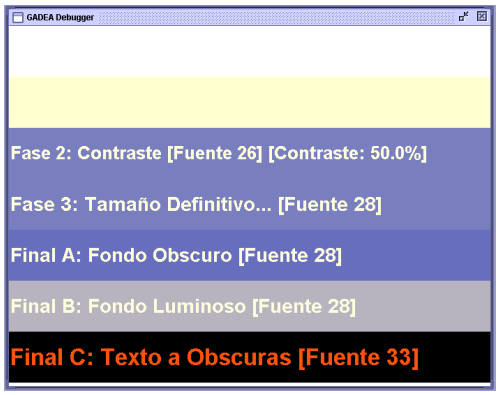


Figura 84. Situación imposible ya que el programador pretende emplear un mismo color para el fondo y para la forma (el blanco). DEVA transformará el blanco del fondo en amarillo y más tarde en azul (durante el proceso de aumento de contraste). El color de la forma pasará de blanco a un tono de amarillo pálido, garantizando la legibilidad (*final A* y *final B*). Durante el proceso de ajuste también se aumenta el tamaño de la fuente, la cual pasa de 26 a 28 puntos (consultar versión en color en el capítulo 24: *Imágenes en Color*, p. 445).

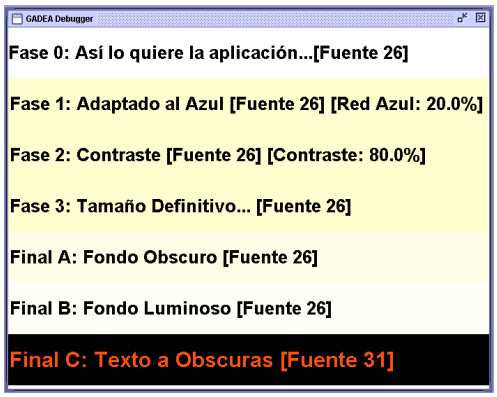


Figura 85. El color blanco para el fondo y el negro para las formas son los colores por defecto de DEVA. En esta ilustración se observa el proceso de adaptación de estos colores para un usuario relativamente joven (eliminación de azul del 20%) y de *Precisión Visual* baja (*Contraste Visual* requerido: 80%). El resultado (*Final A* y *Final B*) ha sido el empleado para configurar los ejemplos de diálogos interactivos incluidos en prácticamente todo este documento (consultar versión en color en el capítulo 24: *Imágenes en Color*, p. 445).

24.2.12. Capítulo 19 (Espías)

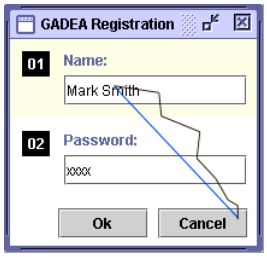


Figura 90. Trayectoria seguida por el puntero al pulsar sobre el botón *CANCEL* (consultar versión en color en el capítulo 24: *Imágenes en Color*, p. 445).

24.2.13. Capítulo 20 (Usabilidad)

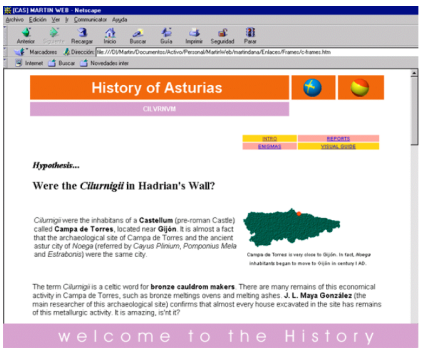


Figura 95. Ejemplo de una de las páginas web del prototipo de portal para el Museo Arqueológico de Asturias. Aunque esta página está escrita en inglés, los usuarios pueden cambiar de idioma pulsando en los botones situados en la esquina superior derecha (consultar versión en color en el capítulo 24: *Imágenes en Color*, p. 445).

24.2.14. Capítulo 22(La Adaptabilidad)

Figura 99. Explicación de una campaña militar mediante objetos señuelo (consultar versión en color en el capítulo 24: *Imágenes en Color*, p. 445).

Figura 100. Objeto predador empleado para describir en detalle la evolución de la campaña militar ilustrada en la Figura 99 (consultar versión en color en el capítulo 24: *Imágenes en Color*, p. 445).

Figura 101. Página principal del sitio web de la asignatura. El *banner* con información de ayuda complementaria se encuentra en la parte superior del espacio visual (en rojo) y es común a todas las páginas (consultar versión en color en el capítulo 24: *Imágenes en Color*, p. 445).

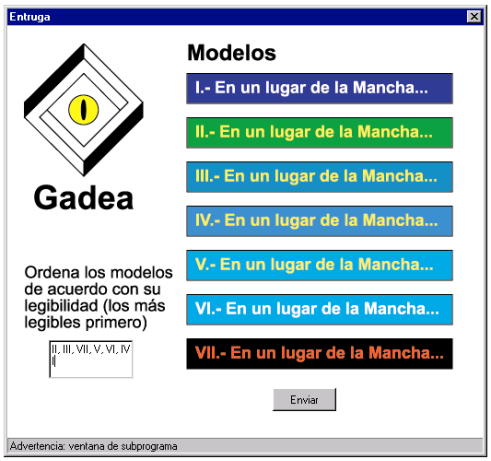


Figura 102. Plantilla de selección cargada en el Entruger y visualizada desde un navegador de Internet por medio de un applet de Java (consultar versión en color en el capítulo 24: *Imágenes en Color*, p. 445).

25

25 Reseñas Bibliográficas

El recuerdo que deja un libro a veces es más importante que el libro en sí

Adolfo Bioy Casares

25.1 Acerca del Formato

En este capítulo se recogen las referencias bibliográficas de toda la obra ordenadas alfabéticamente por el primer apellido del autor principal y año de publicación (las publicaciones más recientes primero). En el caso en el que se haga referencia a más de una publicación de un autor principal en un mismo año, estas publicaciones se identifican con letras situadas entre corchetes y colocadas a la izquierda de la publicación. La primera publicación del autor en un año no lleva letra.

Las referencias a estas publicaciones dentro del texto principal se realizan indicando el apellido del autor y el año de publicación seguido de la letra, si ello procede. Así, la primera publicación de un hipotético autor *Pérez* en el año 2000 se citaría en el texto como [Pérez (2000)], la segunda se citará como [Pérez (2000a)], la tercera como [Pérez (2000b)] y así sucesivamente . Dicha referencia aparecería como:

Pérez (2000) Primera Publicación…

[*a*] Pérez (2000) Segunda Publicación...

[*b*] Pérez (2000) Segunda Publicación...

El formato de las reseñas esta basado en la norma UNE 50-112-04 que describe el contenido, forma y estructura de las referencias bibliográficas, la cual ha sido editada por AENOR. Las referencias incluidas en este capítulo siguen fielmente esta norma para la mayoría de los casos con la excepción de pequeñas adaptaciones aplicadas para completar la información contenida en el formato estándar, incorporando información acerca de lugares adicionales en donde se pueden encontrar los documentos citados.

Esta adaptación consiste en añadir la referencia entre corchetes al final de la reseña. Esta referencia puede referirse a un servidor de Internet desde donde se puede descargar el documento o los códigos de identificación interna empleados por diversas bibliotecas nacionales o extranjeras, como es el caso del Catálogo de la Biblioteca del Congreso Estadounidense (LCC), la biblioteca de la Universidad de Oviedo (Uniovi) o la Biblioteca Nacional sita en Madrid (BN).

Otra pequeña licencia tomada sobre la norma UNE 50-112-04 ha sido la codificación de productos software en estas reseñas. Esta se hace indicando el nombre del producto, el intervalo de producción de la producción reseñada y si es posible, la dirección en Internet desde donde se ha descargado la versión del producto que se ha evaluado. Para un rápida identificación de los productos software, éstas referencias se señalan con el símbolo de la arroba (@).

25.2 Reseñas

A

ABASCAL, Julio; GARAY, Nestor; GARDEAZABAL, Luis; (2000) *Sistemas de Interacción Persona-Computador para Usuarios con Discapacidad*. Actas de las *I Jornadas de Interacción Persona-Ordenador Interacción 2000*. Granada, 19 y 20 de junio de 2000.

ACM SIGCHI; (1992) *The UIMS Tool Developers Workshop: A metamodel for the Runtime Architecture of an Interactive System*. ACM SIGCHI Bulletin, 24.

@ ADOBE (1986-2001) *Photoshop*. [http://www.adobe.com/products/photoshop/main.html].

@ ALTOVA (2000- 2001) *XML Spy*. [http://www.xmlspy.com].

ÁLVAREZ, Luis; SOLER, Enrique (Cords) (1999) *Enseñar Para Aprender*. Madrid: Editorial CSS. ISBN 84-8316-231-8.

ÁLVAREZ BLANCO, Roberto; (1999) *ANTS II: Control de Navegación en Artefactos Hipermedia*. Proyecto de Fin de Carrera dirigido por GONZÁLEZ RODRÍGUEZ, Martín. Escuela Universitaria de Ingeniería Informática de Oviedo. Universidad de Oviedo.

ÁLVAREZ FERRANDO, Juan; (1994) *Interfaces de usuario: Principios y Criterios de Psicología Perceptiva*. Universidad de Oviedo.

ÁLVAREZ GUTIERREZ, Darío; TAJES MARTINEZ, Lourdes; (1997) *An Object-Oriented Abstract Machine as the Substrate for an Object-Oriented Operating System*. Actas de la *11th. European Conference on Object-Oriented Programming* (workshop). Jyväskylä, Finlandia.

ANDRIOLE, S. J.; (1986) *Graphic-Equivalence, Graphic Explanations, and Embedded Process Modelling for Enhanced Process Modelling for Enhanced User-Systems Interactions. IEEE Transactions on Systems, Man and Cybernetics*, 16.

APARICIO FERRERAS, Roberto; (1999) *Sistema de Publicidad Inteligente*. Proyecto de Fin de Carrera dirigido por GONZÁLEZ RODRÍGUEZ, Martín. Escuela Universitaria de Ingeniería Informática de Oviedo. Universidad de Oviedo.

APPLE COMPUTER INC; (1992*): Macintosh Human Interface Guidelines*. Adison-Wesley Publishing Company. ISBN 0-201-62216-562216.

[*a*] APPLE COMPUTER INC; (1992*): Inside Macintosh:: Macintosh Toolbox Essentials*. Adison-Wesley Publishing Company. ISBN 0-201-63243-863243.

[*b*] APPLE COMPUTER INC; (1992*): Inside Macintosh:: More Macintosh Toolbox*. Adison-Wesley Publishing Company. ISBN 0-201-63299-3.

APPLE COMPUTER INC; (1987*): Multifinder, Guía del Usuario. Apple Macintosh Multifinder*.

@ ARQUEOASTUR; (2001) *Sitio Web de la Historia de Asturias*. GONZÁLEZ RODRIGUEZ, Martín (Ed.) [http://members.xoom.com/arqueoastur].

B

BAEKER, R. M.; GRUDIN, J.; BUXTON, W. A. S.; GREENBERG, S.; (EDS) (1995) *Readings in Human-Computer Interaction: Toward the Year 2000*. Morgan Kaufmann Publishing: San Francisco.

BALLESTEROS JIMENEZ, Soledad (1994)*; Cognitive Approaches to Human Perception*. Editorial Lea.

[*a*] BALLESTEROS JIMENEZ, Soledad (1994) *Cognitive Approaches to Human Perception*. Editorial Hillsdale: Nueva Jersey . ISBN: 0805810439. [Uniovi: 694907 (Amicus)].

BALSAS, J. R.; DÍAZ, M. C.; MONTEJO A.; MARTÍNEZ F.; GARCÍA M.; UREÑA, L. A.; (2000) *Arquitectura para Agentes de Interfaz Inteligentes: El Ordenador Sugerente*. Actas de las *I Jornadas de Interacción Persona-Ordenador Interacción 2000*. Granada, 19 y 20 de junio de 2000.

BAUER, Ben; (1997) *The Stroop Effect (What does it tell us about attention and memory?)* Psychology 207, Invierno. [http://www.cgl.uwaterloo.ca/~bgbauer/chapters/stroop.html]

BERKAN, Riza, C.; (1997) *Fuzzy Systems Design Principles. Building Fuzzy IF-THEN Rule Bases*. IEEE Press, 1997.

BERTALANFFY (Von), Ludwing; (1968) *General System Theory*. Nueva York: Wiley and Sons.

BICKMORE, Timothy; COOK, Linda, K; CHURCHILL, Elizabeth F; SULLIVAN, Joseph W.; (1998) *Animated Autonomous Personal Representaatives*. Agents 98.

BICKMORE, Timothy; COOK, Linda, K; CHURCHILL, Elizabeth F; SULLIVAN, Joseph W.; (1998) *Animated Autonomous Personal Representaatives*. Agents 98.

BLUMENTHAL, P. J.; (1991) *¿Por qué nos falla la Memoria?*. Muy Interesante. No. 120, Mayo. Madrid.

BODART, F; VANDERDONCKT, J.; (1996) (Eds.) Actas de *Desing, Specification and Verification of Interactive Systems'96*. Namur. Viena: Springer Verlag.

BORENSTEIN, Nathaniel S.; (1991*) Programming as if People Mattered: Friendly Programs, Software Engineering and Other Noble Delusions*. Princenton, New Jersey: Princenton University Press. ISBN 0-691-03763-9.

BOOCH, G.; (1994) *Object-Oriented Analysis and Design with Applications*. 2ª. Edición. Addison –Wesley.

BOOCH, G.; (1996) *Object Solutions*. Addison-Wesley, 1996.

BREHM, J. W.; SELF, E. A.; (1989) *The Intensity of Motivation*. Annual Review of Psychology, 40, 109-131.

BROADBENT, D. E.; (1971) *Decision and Stress*. Academic Press: Londres. ISBN: 0121355500. [Uniovi: 96407 (Amicus)].

BROADBENT, D. E.; (1958) *Perception and Communication*. Pergamon: Londres.

BUMBULIS, P.; ALENCAR, P. S. C.; COWAN, D. D.; LUCENA, C. F. P.; (1995) *A Framework for protyping and Mechanically Verifying User Interfaces*. Actas de AMAST'95.

BYRNE, Anthony; PICKING, Richard; (1997) *Is time out to be the big issue?* Artículo presentado en el *Time and the Web Workshop*, Staffordshire University, 19 de junio de 1997. [http://www.soc.staffs.ac.uk/seminars/web97].

C

CADRECHA, Miguel A.; HERNÁNDEZ, Jesús; LUEGO, Miguel A.; REIBELO, Juan; NEIRA, R. Teófilo; (1999) *Enseñar para Aprender: Procesos Estratégicos*. ÁLVAREZ, Luis; SOLER, Enrique (Eds.). Editorial SCS. ISBN 84-8316-231-8.

CARD, Stuart K.; MORAN Thomas P.; NEWELL, Allen; (1983) *The Psychology of Human-Computer Interaction*. Lawrence erlbaum Associates.

CAREY, Susan; (1985) *Conceptual Change in Childhood*. The MIT Press: Massachusets. ISBN262031108. [Uniovi: 72156 (Amicus)].

CARNEIRO-COFFIN, L. M. F.; COWAN, D.; LUCENA, C. F. P.; SMITH, D; (1995) *An Experience using JASMINUM. Formalization Helping the Desing of User Interfaces*. TAYLOR, R.; COUTAX, J. (Eds.) Actas del *Workshop on Software Engineering and Human-Computer Intraction: Joint Research Issues*. Lecture Notes in Computer Science. Viena: Springer-Verlag.

CASTELLS, Pablo; SZEKELY, Pedro A; SALCHER Ewald; (1997) *Declarative Models of Presentation*. Intelligent User Interfaces.

CHAPANIS, Alphonse; (1965) *Ingeniería Hombre Máquina*. México: Compañía Editorial Continental [Uniovi: V 6 CHAP].

CHECKLAND, Peter; (1981) *Systems Thinking, Systems Practice*. Chinchester: John Wiley [Uniovi: CJS.BC EO-0142].

CHECKLAND, Peter; (1995) *Metodología de los Sistemas Suaves de Acción*. 1ª Edición. LIMUSA. ISBN 968-18-4912-4.

CHERNICOFF, Stephen; (1985) *Macintosh Revealed: Unlocking the Toolbox*. Hayden Book Company. ISBN 0-8104-6551-5 [LCC QA76.8M3C48].

[*a*] CHERNICOFF, Stephen; (1985) *Macintosh Revealed: Programming with the Toolbox*. Hayden Book Company. ISBN 0-8104-6561-2 [LCC QA76.8M3C48].

CHUNG WONG, Gary Yat; WAI CHUN, Andy Hon; (2000) *A Software Framework for the Implementation of Fuzzy Logic Systems*. Actas de *PA Java 2000: The Second International Conference on The Practical Application of Java*. Manchester, del 12 al 14 de Abril de 2000. ISBN 1-902426-09-6.

COAD, Peter; YOURDON, Edward; (1991) *Object-Oriented Analysis*. Englewood Cliffs: Yourdon Press. ISBN 0136299814. [Uniovi: BTE.L U681B COAD].

CORDERO NOVAL, Patricia; (1999) *Tirsus I (Prehistoria Asturiana): Planteamiento Hipermedia de la Enseñanza de Historia*. Proyecto de Fin de Carrera dirigido por GONZÁLEZ RODRÍGUEZ, Martín. Escuela Universitaria de Ingeniería Informática de Oviedo. Universidad de Oviedo.

COREN, Stanley; PORAC, Clare; WARD, Lawrence M.; *Sensation and Perception*. Academic Press: NuevaYork. ISBN 012188550X. [Uniovi: 203034 (Amicus)].

COREN, Stanley; GIRGUS, Joan Stern; (1978) *Steing is Deceiving : the Psychology of Visual Illusions*. Editorial Hillsdale: Nueva Jersey. ISBN 0470265221. [Uniovi: 96498].

COUTAZ, J.; PATERNO; F., FACONTI, G.; NIGAY; (1993) A *Comparison of Approaches for Specifying MultiModal Interactive Systems*. Actas del *ERCIM Workshop on MultiModal Human Interaction*. Nancy, del 3 al 5 de noviembre.

CUEVA LOVELLE, Juan Manuel; (1994*) Conceptos Básicos de traductores, Compiladores e Interpretes*. Universidad de Oviedo. Cuadernos Didácticos, Departamento de Matemáticas. Oviedo.

D

D'AMORE, Marta; (2000) *Las TI al Servicio del Proceso de la Mejora Continua*. Actas de las *V Jornadas Sobre Calidad del Software*. 6 y 7 de julio de 2000 San Sebastián. EOI: Madrid.

DANEMAN, M.; MERIKLE, P. M. (1996). *Working memory and language comprehension: A meta-analysis*. Psychonomic Bulletin & Review, 3. [http://watarts.uwaterloo.ca/~pmerikle/].

DECAROLIS, B.; DEROSIS, F. (1994) *Modelling Adaptive Interaction of Opade by Petri Nets*. ACM SIGCHI Bulletin. Vol. 36. No. 2.

DECAROLIS, B.; DEROSIS, F. (1993) *Specifying User Adapted Man Machine Dialogues by Extended Petri Nets*. Actas *del Workshop on CSCW. 14th International Conference on Application and Theory of Petri Nets*. Chicago.

DE LUCENA, Fabio Nogueira.; LIESENBERG, Hans, Kurt Edmund; (1995) *X-CHART-based Complex Dialogue Development*. [http://www.ic.unicamp.br/ic-tr-ftp/ALL/Titles.html].

[*a*] DE LUCENA, Fabio Nogueira.; LIESENBERG, Hans, Kurt Edmund; (1995) *Facilitando o Desenvolvimento de controle complexo usando X-CHART*. [http://www.ic.unicamp.br/ic-tr-ftp/ALL/Titles.html].

DE SOUSA, C. S.; (1996) *The Semiotic Engineering of Concreteness and Abstractness: From User Interface Languages to End User Programming Languages*. Actas del *Dagstuhl Seminar on Informatics and Semiotics*.

DE SOUSA, C. S.; (1993) *The Semiotic Engineering of User Interface Languages*. International Journal of Man-Machine Studies No. 39, p. 753.

DEUTSCH, Diana, DEUTSCH, J. Anthony; (1975) *Short-term Memory*. Academic Press: Nueva York. ISBN 122133501. [Uniovi: 96450 (Amicus)].

DI CARO, G. DORIGO, M.; (1997) *AntNet: A Mobile Agents Approach to Adaptive Routing*. Tech. Rep. IFIDIA/97-12. Université Libre de Bruxelles, Bélgica.

DÍEZ REDONDO, Diego; (1998) *Sistema de Encuestas Distribuido para Internet*. Proyecto de Fin de Carrera dirigido por GONZÁLEZ RODRÍGUEZ, Martín. Escuela Universitaria de Ingeniería Informática de Oviedo. Universidad de Oviedo.

DOMINGO, Andrés; CLEMENT CASADO, Francisco; ANDRÉS DOMINGO, M. L. (1987) *Oftalmología*. Luzán 5: Madrid. ISBN 8486152372, [Uniovi: CDU 617.7].

DORIGO, M.; MANIEZZO, V.; COLORNI, A.; (1996) *The Ant System: Optimization by a Colony of Cooperating Agents*. *IEEE Transactions on Systems, Man and Cybernetics-PartB.*

DUNCAN, Luce R. (1986) *Response Times: Their Role in Inferring Elementary Mental Organization*. Oxford Psychology Series. Oxford Unversity Press: Nueva York. ISBN: 0195036425. [Uniovi: 88668 (Amicus)].

DUNCAN, J.; (1981) *Directing Attention in the Visual Field*. *Perception and Psychophysics*. 30.

DUNCAN, J.; (1980) *The Locus of Interference in the Perception of Simultaneous Stimuli*. *Psychological Review*. 87.

DUNCAN, J.; (1977) *Response Selection Rules in Spatial Choice Reaction Tasks*. Editorial Hillsdale: Nueva Jersey.

DUNFEE, W.; MCGEHE, J.; RAUF, R.; SHIPP, K; (1988*) Designing SAA Applications and User Interfaces*. IBM Systems Journal, 27.

E

ECKEL, Bruce; (2000) *Thinking in Java*. Segunda Edición. ISBN 0-13-027363-5.

[@] ECO PRODUCTIONS (2001) *Color System*. Ostwald: *Farbfibel*. [http:/ / www.colorsystem.com / projekte / engl / 32oste.htm].

ECO, Umberto; (1976) *Tratado General de Semiótica*. Editorial Perspectiva.

ELIADE, Mircea (1983) *Imágenes y Símbolos*. Editorial Taurus: Madrid.

ERGOLIGHT (2001); *Usability software: GUI and Lab tester for Windows* [http:/ / www.ergolight-sw-com].

F

FALGUERAS, J.; GUEVARA, A.; (2000) *Metodología y Herramientas para la Evaluación Automática de la Usabilidad*. Actas de las *I Jornadas de Interacción Persona-Ordenador Interacción 2000*. Granada, 19 y 20 de junio de 2000.

FARADAY, Pete; (1999) Visually Critiquing Web Pages. Actas *del Fifth Eurographics Workshop on Multimedia*, Milán, Italia, del 7 al 8 de Septiembre de 1999. CORREIA Nuno; CHAMBEL, Teresa; DAVENPORT Glorianna; (Eds.) *Multimedia'99*. Viena: Springer-Verlag ISBN 3-211-83437-0.

FARADAY, Pete; SUTCLIFFE, Alistair G.; (1998) *Making Contact Points Between Text and Images.* ACM Multimedia.

FARADAY, Pete; SUTCLIFFE, Alistair G.; (1997) *Multimedia: Design for the Moment*. ACM Multimedia.

FEINER, Steven; (1991) An Architecture for Knowledge-Based Graphical Interfaces. *Intelligent User Interfaces*. SULLIVAN, Joseph W.; TYLER, Sherman W; (Eds.) ACM Press. [LCC: QTA76.9H85A73].

FERNÁNDEZ, Antonio; (1985) *Nada es Como Parece. Muy Interesante*. Año 4, No. 44. Editora Cinco: Bogotá.

FOLEY, James; KIM, Won Chun; KOVACEVIC, Srdjan; MURRAY, Kevin (1991) UIDE: *An Intelligent User Interface Design Environment*. *Intelligent User Interfaces*. SULLIVAN, Joseph W.; TYLER, Sherman W; (Eds.) ACM Press. [LCC: QTA76.9H85A73].

FRANK, Martin R.; SUKAVIRIYA, Piyawadee Noi; FOLEY, James D.; (1995) *Inference Bear: Designing Interactive Interfaces Through Before and After Snapshots*. Symposium on Designing Interactive Systems.

FROVA, Andrea (1999). *El Engaño de los Sentidos. Newton*. Ediservicios M-2000: Madrid.

FURNAS, G. W. (1997) *Effective View-Navigation. Human Factors in Computing Systems CHI97 Conference Proceedings*, New York, NY: ACM Press.

G

GABAY, Joseph; (1991) *Aprender y Practicar Merise*. Ed. Masson.

GALLEGO-SCHMID, Marcos (2000); *Multi-Agent based Genetic Routing Algorithms: AntNet Variations Applied in a Java Telecommunication Network Simulator*. Actas del *Second International Conference on The Practical Application of Java (PA Java'2000)*, Manchester, Inglaterra, del 10 al 14 de Abril de 2000, . Londres: *The Practical Application Company*. ISBN 1-902426-09-6.

GAMMA, Erich; (1996*) Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley. Reading, Massachusetts. ISBN 0201633612. [Uniovi: CFM K68A-0534].

GARCÍA ALCÁZAR, Enrique; MONZÓN, Antonio; (2000) *Ventajas de la utilización de un Marco Metodológico de Ingeniería de requisitos para la Implantación de un Sistema de Calidad*. Actas de las *V Jornadas Sobre Calidad del Software*. 6 y 7 de julio de 2000 San Sebastián. EOI: Madrid.

GARCÍA CASTRO, Santiago; (2001) *Tirsus III (Reino de Asturias): Aprendizaje mediante Técnicas de Información Convergente*. Proyecto de Fin de Carrera dirigido por GONZÁLEZ RODRÍGUEZ, Martín. Escuela Universitaria de Ingeniería Informática de Oviedo. Universidad de Oviedo.

GARCÍA FERNÁNDEZ, Ana María; (2001) *Tirsus IV (Guerra Civil en Asturias): Navegación Multidimensional por Escenarios Historiográficos*. Proyecto de Fin de Carrera dirigido por GONZÁLEZ RODRÍGUEZ, Martín. Escuela Universitaria de Ingeniería Informática de Oviedo. Universidad de Oviedo.

GARCÍA FERNÁNDEZ, Ana María; GONZÁLEZ RODRÍGUEZ, Martín; PAULE RUÍZ, María del Puerto; RAMÓN PÉREZ, Juan; (2001) *The Multidimensional of Hypermedia Devices based on Historical Knowledge Bases*. Actas de *EuroMedia 2001: Sixth Annual Scientific Conference on Web Technology, New Media, Communications and Telematics Theory, Methods, Tools and Applications*, Valencia, del 18 al 20 de Abril de 2000. ISBN 1-56555-217-2.

GARCÍA FERNÁNDEZ, Ana María; GONZÁLEZ RODRÍGUEZ, Martín; SÁNCHEZ PADIAL, Jesús Antonio; REDONDO LÓPEZ, José Manuel; PAULE RUÍZ, María del Puerto; RAMÓN PÉREZ, Juan; (2000) *Navegación Multidimensional en Acontecimientos Históricos. Tirsus IV: una Visión Dinámica de la Historia*. Actas del *II Simposio Internacional de Informática Educativa (SIIE 2000)*. Puertollano, Ciudad Real, del 15 al 17 de Noviembre de 2000. ORTEGA, Manuel; BRAVO, José;(Eds.*) Informática y Educación para una Sociedad Conectada*. ISBN 84-607-1341-5.

[*a*] GARCÍA FERNÁNDEZ, Ana María; GONZÁLEZ RODRÍGUEZ, Martín; REDONDO LÓPEZ, José Manuel; SÁNCHEZ PADIAL, Antonio Jesús; (2000) *Tirsus IV: Navegación Multidimensional en Aplicaciones Hipermedia sobre Acontecimientos Históricos.* Actas de *Interacción 2000*, Granada, España, del 19 al 20 de junio de 2000. ISBN.

GASH, Manuel; (1991). *Modelismo y Dioramas. Miscelánea Modelística.* Ediciones Génesis SA. Madrid.

GEA, M.; MARTÍN, D.; (2000) *Descripción de la Expresividad de Agentes Inteligentes mediante Alhambra.* Actas de las *I Jornadas de Interacción Persona-Ordenador Interacción 2000*. Granada, 19 y 20 de junio de 2000.

GONZÁLEZ RODRÍGUEZ, Martín; CUEVA LOVELLE, Juan Manuel; (2001) *El Modelo Cognitivo en la Explotación de Sistemas de Comunicación*. Publicación pendiente como *Cuadernos de Investigación en Ingeniería Informática.* Oviedo: Editorial Servitec.

[*a*] GONZÁLEZ RODRÍGUEZ, Martín; LÓPEZ PÉREZ, Benjamin, PAULE RUÍZ, María del Puerto; CUEVA LOVELLE, Juan Manuel; (2001) *GADEA: A General Framework for the Development of User Interfaces Adapted to Human Diversity*. Publicación pendiente en actas de *EDMEDIA 2001: World Conference on Educational Multimedia, Hypermedia & Telecommunications*, 25 de Junio Tampere, Finlandia.

[*b*] GONZÁLEZ RODRÍGUEZ, Martín; PAULE RUÍZ, María del Puerto; RAMÓN PÉREZ, Juan; DEL MORAL PEREZ, Esther (2001) *Automatic Low-Level Interactive Discourses for Multimodal User Interfaces.* Actas de *EuroMedia 2001: Sixth Annual Scientific Conference on Web Technology, New Media, Communications and Telematics Theory, Methods, Tools and Applications*, Valencia, del 18 al 20 de Abril de 2000. ISBN 1-56555-217-2.

GONZÁLEZ RODRÍGUEZ, Martín; REDONDO LÓPEZ, José Manuel; SÁNCHEZ PADIAL, Jesús Antonio; FERNÁNDEZ GARCÍA, Ana María; RAMÓN PÉREZ, Juan; PAULE RUÍZ, María del Puerto; (2000) *CineMedia Astur: Herramienta Avanzada para el Desarrollo de Software Educativo*. Actas del *II Simposio Internacional de Informática Educativa (SIIE 2000)*. Puertollano, Ciudad Real, del 15 al 17 de Noviembre de 2000. ORTEGA, Manuel; BRAVO, José; (Eds.) *Informática y Educación para una Sociedad Conectada*. ISBN 84-607-1341-5.

[*a*] GONZÁLEZ RODRÍGUEZ, Martín; ÁLVAREZ GUTIERREZ, Darío; (2000) *Análisis Remoto de Procesos de Exploración en Internet*. Actas del *Simposio Español de Informática Distribuida (SEID 2000)*. Orense, del 25 al 27 de septiembre de 2000. BARRO, Senen; BUSTA, José María; CORCHADO, Juan Manuel; CUESTA, Pedro (Eds.) Orense: ISBN 84-8158-163-1.

[*b*] GONZÁLEZ RODRÍGUEZ, Martín; REDONDO LÓPEZ, José Manuel; SÁNCHEZ PADIAL, Jesús Antonio; FERNÁNDEZ GARCÍA, Ana María; (2000) *CineMedia Astur: Uso de Técnicas Avanzadas de Desarrollo de Software para el Aprendizaje por Ordenador*. Actas del *III Congreso Internacional sobre Comunicación Tecnología y Educación: Redes, Multimedia y Diseños Virtuales*, Oviedo, del 12 al 15 de Septiembre de 2000. PÉREZ PÉREZ, Ramón; PASCUAL SEVILLANO Ángeles; GARCÍA RODRÍGUEZ, Marta; MORAL PÉREZ, Esther; ÁLVAREZ GARCÍA, Concepción; SIERRA ARIZMENDIARRIETA, Beatriz; (Eds.) *Redes, Multimedia y Diseños Virtuales*. Oviedo: Universidad de Oviedo ISBN 607-1118-8

[*c*] GONZÁLEZ RODRÍGUEZ, Martín; SÁNCHEZ PADIAL, Jesús Antonio; FERNÁNDEZ GARCÍA, Ana María; REDONDO LÓPEZ, José Manuel; (2000) *Modelos Contextuales Multidimensionales en la Enseñanza Hipermedia de la Historia*. Actas del *III Congreso Internacional sobre Comunicación Tecnología y Educación: Redes, Multimedia y Diseños Virtuales*, Oviedo, del 12 al 15 de Septiembre de 2000. PÉREZ PÉREZ, Ramón; PASCUAL SEVILLANO Ángeles; GARCÍA RODRÍGUEZ, Marta; MORAL PÉREZ, Esther; ÁLVAREZ GARCÍA, Concepción; SIERRA ARIZMENDIARRIETA, Beatriz; (Eds.) *Redes, Multimedia y Diseños Virtuales*. Oviedo: Universidad de Oviedo ISBN 607-1118-8.

[*d*] GONZÁLEZ RODRÍGUEZ, Martín; ÁLVAREZ GUTIERREZ, Darío; (2000) *Data Gathering Agents for Remote Navigability Testing*. Actas de *SCI'2000 (Systemics, Cybernetics and Informatics)*, Orlando, EUA. 23 al 26 de julio de 2000. ISBN 980-07-6693-6.

[*e*] GONZÁLEZ RODRÍGUEZ, Martín; VIDAU NAVARRO, Águeda; (2000) *Verificación Remota de Modelos de Navegación en Hipermedia.* Actas de las *V Jornadas sobre Calidad del Software*, San Sebastián, España, del 6 al 7 de Julio de 2000. Madrid: EOI: Escuela de Organización Industrial.

[*f*] GONZÁLEZ RODRÍGUEZ, Martín; VIDAU NAVARRO, Águeda; CUEVA LOVELLE, Juan Manuel; (2000) *Tests de Usabilidad Remotos en Interfaces de Navegación para Universos Hipermedia*. Actas del X *Congreso Español de Informática Gráfica (CEIG 2000)*, Castellón, España, del 28 al 30 de Junio de 2000. Arinyo Joan, Robert; Navazo Álvaro, Isabel; Quirós Bauset, Ricardo; (Eds). *Collecció Treballs d'Informatica i Tecnologia*, Num 3. ISBN 84-8021-314-0.

[*g*] GONZÁLEZ RODRÍGUEZ, Martín; LABRA GAYO, Emilio; CUEVA LOVELLE, Juan Manuel (2000) *Web Navigability Testing with Remote Agents*. Actas del *Second ICSE Workshop on Web Engineering*, Limerick, Irlanda, del 4 al 5 de Junio de 2000. Springer-Verlag: Viena.

[*h*] GONZÁLEZ RODRÍGUEZ, Martín; ÁLVAREZ GUTIERREZ, Darío; (2000) *Automatic Navigability Testing for Hypermedia based on Data Gathering Agents*. Actas de *EuroMedia 2000: Fifth Annual Scienfific Conference on Web Technology, New Media, Communications and Telematics Theory, Methods, Tools and Applications*, Amberes, Bélgica, del 8 al 11 de Mayo de 2000. ISBN 1-56555-202-4.

[*i*] GONZÁLEZ RODRÍGUEZ, Martín; (2000*) Remote Navigation Testing for Hypermedia with Java*. Actas del *Second International Conference on The Practical Application of Java (PA Java'2000),* Manchester, Inglaterra, del 10 al 14 de Abril de 2000, . Londres: *The Practical Application Company*. ISBN 1-902426-09-6.

[*j*] GONZÁLEZ RODRÍGUEZ, Martín; (2000) *ANTS: An Automatic Navigability Testing Tool for Hypermedia*. Actas *del Fifth Eurographics Workshop on Multimedia*, Milán, Italia, del 7 al 8 de Septiembre de 1999. CORREIA Nuno; CHAMBEL, Teresa; DAVENPORT Glorianna; (Eds.) *Multimedia'99*. Viena: Springer-Verlag ISBN 3-211-83437-0.

GONZÁLEZ RODRÍGUEZ, Martín; (1999) *Automatic Usability Testing for Hypermedia Navigational Graphs*. Actas del *International Software Quality Week Europe: Lessons Learned*, Bruselas, Bélgica, del 1 al 5 de Noviembre de 1999. Software Research Institute.

[*a*] GONZÁLEZ RODRÍGUEZ, Martín; CUEVA LOVELLE, Juan Manuel; (1999) *Un Hypermedia Design Process (HDP) para JEDI-Leia*. Cuadernos de Investigación en Ingeniería Informática No. 2. Oviedo: Editorial Servitec. ISBN 84-699-0051-X.

GONZÁLEZ RODRIGUEZ, Martín (1995). *El Diseño Gráfico Global*. [http: / / members.nbci.com / martinzone / Diseno / index.html].

GORDON, Druce; (1993) *Designing Covers*. *MacFormat*. Issue 14. Mayo. Future Publishing Ltd.: Bath, Inglaterra.

GOTTSCHALDT, Kurt; (1938) *Gestalt factors and Repetition*. W. Ellis (Ed.). *A source book of Gestalt psychology* (pp. 89-94). Routledge & Kegan Paul: Londres. Reproducido en: [http:/ / psychclassics.yorku.ca].

GRAND, Mark (1999) *Patterns in Java*. Wiley Computer Publishing.

GREEN, M.; JAKOB, R. J. K.; (1991*) Software Architectures and Meaphors for Non-WIMP User Interfaces. Computer Graphics*. Vol. 25. No. 3. Julio.

GREEN, M.; (1985) *The University of Alberta UIMS*. Actas *de SIG-GRAPH'85. Computer Graphics*, 19 (3).

[*a*] GREEN, M.; (1985) *Report on Dialogue Specification Tools. User Interface Management Systems*. PFAFF, G. E. (Ed.) Springer Verlag: Berlin.

GREENE, Judith; (1986) *Language Understanding, a Cognitive Approach*. Open University, Milton Keynes. ISBN: 0335153267. [Uniovi: FP.BC X15U-0029].

H

HALL, Arthur. D.; (1980) *Ingeniería de Sistemas*. México: Compañía Editorial Continental.

HERSHENSON, M.; (1998) *Visual Space Perception*. ISBN 0-262-58167-1.

HEFLEY, W. E.; MURRAY, D.; (1993*) Intelligent User Interfaces*. GRAY, W. D.; HEFLEY, W. E.; MURRAY, D.; (Eds). Actas del *International Workshop on Intelligent User Interfaces*, Orlando, FL. ACM Press,

HEINZ, Hartmann (1969); *Ensayos sobre la Psicología del Yo*. Fondo de Cultura Económica: México. [Uniovi: 102501 (Amicus)].

HILGARD, Ernest R.; (1978) *Condicionamiento y Aprendizaje*. Editorial Trillas: México. [Uniovi: 76909 (Amicus)].

HILLSTROM, A. P.; YANTIS, S. (1994*) Visual Motion and Attentional Capture. Perception and Pcychophysics*, 55 (4).

HITLER, Adolfo; (1925) *Mi Lucha*. NSDAP, Nuremberg. [Uniovi: NC.G CG92 Hitler].

@ HOWARD, Chris; CRONJE, Carmi; (2000-2001) *XML Writer*. Wattle Software. [http:/ / XMLWriter.net].

I

@ IBM (2001) *IBM ViaVoice Developer's Corner*. [http://www-4.ibm.com/software/speech/dev].

@ IBM (1994-2001) *Visual Age for Java; Visual Age for SmallTalk*. [http://www.alphaworks.com].

IBM; (1989) *Systems Application Architecture, Commoun User access Advanced Interface Design Guide*. IBM Corporation Publications.

ITTELSON, William H; (1960) *Visual Space Perception*. Springer Publishing Company, Inc. New York.

IVENS, Kathy; (1996) *Windows NT Workstation Professional Reference*. New Riders, Indianapolis. ISBN: 1562056921. [Uniovi: CFMI KP68A-357].

J

JACOBSON, Ivar; CHRISTERSON, M.; JONSSON, P.; OVERGAARD, G.; (1992) *Object-Oriented Software Engineering, A Use Case Driven Approach*, Addison-Wesley.

JOHN, David; BOUCOUVALAS, Anthony; (2000) *The Effect of Cognitive Sytle on User Performance with Audio*. Actas de *Euromedia 2000, Fifth Annual Scientific Conference on Web Technology, New Media, Communications and Telematics Theory, Methods, Tools and Applications*. Amberes, Bélgica, del 8 al 10 de mayo de 2000.

JOHNSON, Mark; (1999) *XML for the Absolute Beginner: A Guided Tour from HTML to Processing XML with Java*. Java World. Abril 1999. [http://www.ootlab.uniovi.es – HCI].

JOHNSON, Chris; (1997) *What's the Web Worth? The Impact of Retrieval Delays on the Value of Distributed Information*. Artículo presentado en el *Time and the Web Workshop*, Staffordshire University, 19 de junio de 1997. [http://www.soc.staffs.ac.uk/seminars/web97].

JOHNSON, Chris; GRAY, Philip; (1996). *Temporal Aspects of Usability: Assessing the Impact of Time on User Interface Design*. SIGCHI Bulletin Vol. 28, No. 2, Abril, ACM Press.

JUL, Sussane; FURNAS, George W.; (1997*). Navigation in Electronic Worlds: a CHI'97 Workshop*. ACM SIGCHI Bulletin Vol. 29, No. 4, October 1997, ACM Press.

K

KANTOROWITZ, Eliezer; SUDARSKY, Oded; (1989) *The Adaptable User Interface*. CADCM. ACM Press.

KARN S., Keith; PERRY, Thomas J.; KROLCZYK, Marc. J.; (1997) *Testing for Power Usability: a CHI 97 Workshop*. ACM SIGCHI Bulletin Vol . 29, No. 4, October 1997. [http://www.acm.org/sigchi/bulletin/1997.4/karn.htm].

KEELE, S, W. (1986) *Motor Control. Handbook of Perception and Human Performance*. Wiley: Nueva York.

KLIR, G. J.; YUAN, B.; (1995) *Fuzy Sets and Fuzzy Logic*. Prentice Hall.

KLIR, G. J.; (1972) *Trends in General Systems Theory*. *Interscience*. Nueva York: Wiley and Sons.

KRASNER, G. E.; POPE, S. T.; (1981) *A Cookbook for Using the Model-View-Controller User Interface Paradigm in Smalltalk-80*, ParcPlace Systems, Smalltalk Manual, Palo Alto.

KROENING, Mary; (2000) *Future Thinking 2000: AI and the Net. PC AI: Where Intelligent Technology Meets the Real World*. Marzo – Abril [http://www.pcai.com/pcai].

L

LAKOFF, G.; JOHNSON, M.; (1980*). Metaphors We Live By*. Chicago: University of Chicago Press.

LEMAY, Laura; (1996) *Teach Yourself Java in 21 Days*. Sams Net: Indiana. ISBN 1575210304 [Uniovi: CFMI KP68A-359].

LESTER, Martyn, MORRISH, John; (1994*) A Spot of Colour. MacFormat*. Issue 14. Julio. Future Publishing Ltd.: Bath, Inglaterra.

LI, Xie; XING, Du; JUN, Chen; YUHUA, Zheng; ZHONGXIU, Sun; (1995) *An Introduction to Intelligent Operating System KZ2. Operating Systems Review* 29(1).

LI, Xie; XING, Du; ZHONGXIU, Sun; (1992) *An Open Model for Devloping Knowledge Based User Interfaces*. Actas del *12th World Computer Congress*. Madrid. Septiembre. Vol. III.

LINDZEY, Garndner; HALL, Calvin S.; THOMPSON, Richard F.; (1985) *Psicología*. Editorial Omega: Barcelona.

LIUNGMAN, Carl G. (1992) *Dictionary of Symbols*. ABC Clio, Santa Barbara, California. ISBN: 0874366100. [Uniovi: BC.R CR003.62(03) LIUN].

LIUNGMAN, Carl G. (1972) *El Mito de la Inteligencia*. Martinez Roca: Barcelona. [Uniovi: FP.BC X15P-0074]

LOFTUS, Elizabeth F. (1979) *Eyewitness Testimony*. Harvard University Press, Cambridge, Massachusetts. ISBN: 0674287754 [Uniovi: FP.BC X15X-0014].

LONEWOLF Systems Incorporated (2001) *The ALPHA User Interface Management System* [http://www.lonewolf.com/]

LUCAS, A.; VALERO, P. M.; GARCÍA-ROS, R.; PÉREZ, M.; (2000) *Evaluación de un simulador para la estiba y desestiba portuaria.* Actas de las *I Jornadas de Interacción Persona-Ordenador Interacción 2000*. Granada, 19 y 20 de junio de 2000.

LYNCH, J.; HORTON, S.; (2001) *Yale Centre for Advanced Media* (WWW Style Manual). http://info.med.yale.edu/caim/manual/ages/editorial_style.htm.

M

MAGUIRE, Steve (1994) *Código sin Errores*. Editorial McGraw-Hill. ISBN: 84-481-1800-6.

MANRUBIA, Pablo; GONZÁLEZ RODRÍGUEZ, Martín; (2000*) Java, Internet and Artificial Intelligence*. Actas del *Second International Conference on The Practical Application of Java (PA Java'2000),* Manchester, Inglaterra, del 10 al 14 de Abril de 2000. Londres: *The Practical Application Company*. ISBN 1-902426-09-6.

MANRUBIA DÍEZ, Pablo; (1999) *Principio de Inteligencia Artificial: Un Juego de Ajedrez para Internet*. Proyecto de Fin de Carrera dirigido por GONZÁLEZ RODRÍGUEZ, Martín. Escuela Universitaria de Ingeniería Informática de Oviedo. Universidad de Oviedo.

MAP; (1994) *Metodología Métrica Versión 2: Guía de Técnicas*. Madrid: Ministerio para las Administraciones Públicas.

[*a*] MAP; (1994) *Introducción a la Metodología Métrica Versión 2*. Madrid: Ministerio para las Administraciones Públicas.

MAP; (1991) *Plan General de Garantía de Calidad Aplicable al Desarrollo de Equipos Lógicos*. Madrid: Ministerio para las Administraciones Públicas. ISBN 84-7088-596-0.

MARTI, P.; (1996) *Task-centred Design: Turning Task Modelling into Design*. ACM SIGCHI Bulletin Vol. 28, No. 3.

MARTIN, Lockheed; JESSE, Michael; (1996) *Succeeding with the Booch and OMT Methods: A practical Approach*. Menlo Park, California: Addison-Wesley. ISBN 0805322795. [Uniovi: CFMI K68A-0585].

MARTÍNEZ PRIETO, A. Belén; ÁLVAREZ GARCÍA, Fernando, ORTÍN SOLER, Francisco; CUEVA LOVELLE, Juan Manuel (2000); *Un SGBDOO soportado por el Sistema Operativo Distribuido del Sistema Integral Oviedo3*. Actas del *Simposio Español de Informática Distribuida (SEID 2000)*. Orense, del 25 al 27 de septiembre de 2000. BARRO, Senen; BUSTA, José María; CORCHADO, Juan Manuel; CUESTA, Pedro (Eds.) Orense: ISBN 84-8158-163-1.

MARTÍNEZ PRIETO, A. Belén; ÁLVAREZ GUTIERREZ, Darío, CUEVA LOVELLE, Juan Manuel, ORTÍN SOLER, Francisco; PÉREZ DÍAZ, Jesús Arturo (1998); *Incorporating an Object-Oriented DBMS into an Integral Object-Oriented System*. Actas de *SCI'2000 (Systemics, Cybernetics and Informatics)*, Orlando, EUA.

MASSON, J. E.; WELLHOFF, A.; (1990*) El Merchandising: Rentabilidad y Gestión del Punto de Venta*. Deusto: Madrid. ISBN 8423405117.

MATTHIES, Leslie H.; (1979) *Recursos Humanos en el el Diseño de Sistemas Administrativos*. ISBN: 0 471 57697-2. Limusa: México.

MCKAY, Scott; (1991) *CLIM: The Common Lisp Interface Manager*. CACM.

MCKAY, Scott; YORK, William; (1993) *Common Lisp Interface Manager Release 2.0 Specification*. Octubre. [Http://www.mikemac.com/mike/clim/cover.html].

MCLEAN, R. S.; GREGG, L. W.; (1967) *Effects of Induced Chunking on Temporal Aspects of Serial Recitation. Journal of Experimental Psychology*. 74.

MERIKLE, P. M. (2000). *Subliminal Perception*. KAZDIN, A. E. (Ed.), *Encyclopedia of Psychology*. Oxford University Press: Nueva York. [http://watarts.uwaterloo.ca/~pmerikle/].

MICROSOFT, INC. (2001) *The Windows Interface Guidelines for Software Design*. http://www.microsoft.com/win32dev/uiguide/default.htm.

MILLER, George A.; (1956) *The Magical Number Seven, Plus or Minus Two: Some Limits on Our Capacity for Processing Information. The Psychological Review*, Vol. 63. [http://www.well.com/user/smailin/miller.html].

MÖLLER, Ralf; (1996) *Knowledge-Based Dialog Structuring for Graphics Interaction*. ECAI.

MONTGOMERY, Geoffrey (1991) *Cómo y por qué ven los ojos*. Conocer. No. 102. Julio. Ediciones Tiempo: Madrid.

MORAY, N.; (1959) *Attention in Dichotic Listenning: Affective Cues in the Influence of Instructions. Quarterly Journal of Experimental Psychology*. 11.

MULLER, Pierre-Alain; (1997) *Modelado de Objetos con UML*. Barcelona: Ediciones Gestión 2000. ISBN 84-8088-226-3.

MYERS, Brad; (1998) *User Interface Software. Human Computer Interaction Institute*. [http://www.cs.cmu.edu/~bam/uicourse/1998spring].

MYERS, Brad; HOLLAND, Jim; CRUZ, Isabel; (1996) *Strategic Directions in Human Computer Interaction*. ACM Computing Surveys. Diciembre.

MYERS, Brad.; ROSSON, M.; (1992*) Survey of User Interface Programming*. Actas de *SIGCHI'92: Human Factors in Computing Systems*. Monterrey, California. Mayo.

N

NEAL G. Jeanette; SHAPIRO, Stuart C. *Intelligent Multi-Media Interface Technology*. *Intelligent User Interfaces*. SULLIVAN, Joseph W.; TYLER, Sherman W; (Eds.) ACM Press. [LCC: QTA76.9H85A73].

NEISSER, Ulric; (1981) *Procesos Cognitivos y Realidad: Principios e Implicaciones de la Psicología Cognitiva*. Editorial Marova: Madrid. ISBN 8426904351. [Uniovi: 55507 (Amicus)].

NEWELL, Allen; (1991) *Unified Theories of Cognition*, London: Harvard University Press. ISBN 0-674-92101-1.

NEWELL, A.; ROSENBLOOM P.; (1981) *Mechanisms of Skill Acquisition and the Law of Practice*. ANDERSON, J. R. (Ed.) Cognitive Skills and their Acquisition. Erlabaum: Hillsdale.

NICHOLLS, Leon Eric; (1994) *Generic User Interface package for Windows. Tesis Magistral*. Facultad de Ciencias. Universidad de Port Elizabeth. Dirigido por el Dr. WARREN, P. R. [http://www.cs.upe.ac.za/staff/csalen/guipw/contens.htm].

NICKERSON, R. S; (1972) *Binary Reaction Time: A review of some studies of Human Information*. *Procession Capabilities*. *Psychonomic Monograph Supplements*.

NIELSEN, J.; (2001) *Alertbox*. [http://www.useit.com/alertbox].

NIELSEN, Jakob; (1993) *Hypertext and Hypermedia*. Cambridge: Academic Press. ISBN 0-12-5118410-7.

[a] NIELSEN Jakob; (1993) *A Mathematical Model of the Finding of Usability Problems*. Actas del *INTERCHI'93: Human Factors in Computing Systems*, Amsterdam, Holanda, Abril.

NORMAN, Donald; DRAPER, S.; (1986) *User Centered System Design*. Lawrence Erlbaum.

NORMAN, Donald A.; (1988) *El procesamiento de la Información en el Hombre: Memoria y Atención*. Paidós: Buenos aires. ISBN 968853093-X.

[*a*] NORMAN, Donald A.; (1988) *The Psychology of Everyday Things*. Basic Books: Nueva York. ISBN 0465067093. [Uniovi: CDU-159.9].

NORMAN, Donald A.; (1970) *Models of Human Memory*. Academic Press: Nueva York. ISBN 0125213506.

O

O'DONNEL, Paddy; DRAPER, Stephen W.; (1996). *Temporal Aspects of Usability: How Machine Delays Change User Strategies*. SIGCHI Bulletin Vol. 28, No. 2, Abril 1996, ACM Press.

OLSINA, Luis Antonio; (1998) *Cognitive criteria in the development of Hypermedia Applications*. Actas de las *Conferencias EUITIO*, Universidad de Oviedo. [http://www15.uniovi.es].

OLIVEIRA PRATES, Raquel; DE SOUSA, Sieckenius Clarisse, BICHARRA GARCIA, Ana Cristina; (1997*) A Semiotic Framework for Multi-User Interfaces*. ACM SIGCHI Bulletin, Vol. 29, No. 2. [http://www.acm.org/sigchi/bulletin/1997/prates.htm].

ONCE: Organización Nacional de Ciegos Españoles (2001) [http://www.once.es].

ORTÍN SOLER, Francisco; ÁLVAREZ GUTIERREZ, Darío; CUEVA LOVELLE, Juan Manuel; (2000) *A Flexible Integral Computing System based on a Structurally-Reflective Abstract Machine*. Actas del *SEID 2000*, Orense, Spain, 25th, 26th and 27th September 2000. ISBN 84-8158-163-1.

P

PALENQUE, P; BASTIDE, R.; (1995) *Actas de Design, Specification and Verification of Interactive Systems*. Toulouse. Viena: Springer Verlag.

PALENQUE, P; BASTIDE, P.; (1990) *Petri Nets with Objects for the Design, Validation and Prototyping of User-driven Interfaces.* Actas de INTERACT'90, North-Holland, p. 625-631.

PATERNO, Fabio; PALENQUE, P; (1996) *Formal Methods in Computer Human Interaction: Comparison, Benefits, Open Questions. A CHI 96 Workshop.* ACM SIGCHI Vol. 28. No. 4. [http://www.acm.org/sigchi/bulletin/1996.4/paterno.htm].

PAULE RUÍZ, María del Puerto; GONZÁLEZ RODRÍGUEZ, Martín; PÉREZ, PÉREZ, Ramón; (2000) *Tutor Inteligente para Datación de Piezas Arqueológicas de Cerámica.* Actas del *III Congreso Internacional sobre Comunicación Tecnología y Educación: Redes, Multimedia y Diseños Virtuales*, Oviedo, del 12 al 15 de Septiembre de 2000. PÉREZ PÉREZ, Ramón; PASCUAL SEVILLANO Ángeles; GARCÍA RODRÍGUEZ, Marta; MORAL PÉREZ, Esther; ÁLVAREZ GARCÍA, Concepción; SIERRA ARIZMENDIARRIETA, Beatriz; (Eds.) *Redes, Multimedia y Diseños Virtuales*. Oviedo: Universidad de Oviedo ISBN 607-1118-8.

PETERSON, Lloyd R.; (1983) *Aprendizaje*. Editorial Trillas: México. ISBN 9682413168. [Uniovi: 75503 (Amicus)].

PIERCE, C. S.; (1992) *Collected Papers (1931 – 1958)*. Cambridge. MA: Harvard University Press.

R

REDMOND-PYLE, D. ; MOORE, A.; (1995) *Graphic User Interface Design and Evaluation*. Prentice Hall: Londres.

REDONDO LÓPEZ, José Manuel; GONZÁLEZ RODRÍGUEZ, Martín; LÓPEZ PÉREZ, Benjamin, PÉREZ PÉREZ, Juan Ramón; (2001) *Cinemedia Astur: A Scalable Hypermedia Authoring Tool for Educational Field*. Publicación pendiente en actas de *EDMEDIA 2001: World Conference on Educational Multimedia, Hypermedia & Telecommunications*, 25 de Junio Tampere, Finlandia.

REDONDO LÓPEZ, José Manuel; (2000) *CineMedia Astur: Herramienta de Autor para el Desarrollo de Productos Hipermedia.* Proyecto de Fin de Carrera dirigido por GONZÁLEZ RODRÍGUEZ, Martín. Escuela Universitaria de Ingeniería Informática de Oviedo. Universidad de Oviedo.

[*a*] REDONDO LÓPEZ, José Manuel; SÁNCHEZ PADIAL, Antonio Jesús; FERNÁNDEZ GARCÍA, Ana; GONZÁLEZ RODRÍGUEZ, Martín; (2000) *CineMedia Astur: Herramienta para la Generación de Títulos Hipermedia de Interfaz Flexible*. Actas *de Interacción 2000*, Granada, España, del 19 al 20 de junio de 2000.

REMINGTON, Robert J.; (2001) *Computer-Human Interface Software Development Survey*. CHI Rapid Prototyping and Usability Laboratory. Lockheed Martin Missiles and Space. [http://www.targetsoft-ware.com/ieeetut3.htm].

REYNOLDS, Carson; (1998) *As We May Communicate*. ACM SIGCHI Bulletin Vol. 30, No 3. Julio. ACM Press.

REYNOLDS, Carson; (1997) *A Crititcal Examination of Separable User Interface Management Systems: Constructs for Individualization*. ACM SIGCHI Bulletin. Vol 29, No. 3, Julio.

RIDING R.; (1996*) Learning Styles and Technology-Based Training*. Sheffield Department for Education and Employment.

RIDING R.; (1994) *Personal Style Awareness and Personal Development*. Birmingham, Learning & Training Technology.

RIDING, R.; (1991) *Cognitive Styles Analysis Users' Manual*. Learning & Training Technology.

RIPOTA, Peter; (1991) *Al Rojo Vivo. Muy Interesante*. No. 124. Septiembre. Madrid.

ROAST, Chris; (1997) *HCI and Requirements Engineering: Specifying Cognitive Interface Requirements*. ACM SIGCHI Bulletin Vol. 29. No. 1. Enero. [http://www.acm.org/sigchi/bulletin/1997.1/roast.htm].

ROCK, Irvin; (1985) *The Logic of Perception*. The MIT Press: Massachusetts. ISBN 0262680459 0262181096 [Uniovi: 100999 (Amicus)].

RODRÍGUEZ, Pilar; (2000) *El Principado tiene censados más de 110.000 Discapacitados. Asturias se suma al día Internacional de las personas con discapacidad*. En *La Nueva España* de Oviedo, Jueves 30 de noviembre.

ROSS, Alan O.; (1997) *Terapia de la Conducta Infantil: Principios, Procedimientos y Bases Teóricas*. Limusa: Mexico. ISBN: 9681818253. [Uniovi: FP.BC X615B-100]

ROSSI, Gustavo; SCHWABE, Daniel; GARRIDO Alejandra; (1997) *Design Reuse in Hypermedia Applications Development. Hypertext 97*. ACM.

ROUFF, C; (1995) *Formal Specification of User Interfaces*. ACM SIGCHI Bulletin. Vol. 28. No. 3. [http://www.acm.org/ sigchi/ bulletin/ 1996.3/ rouff.htm].

ROUFF, C; HOROWITZ, E.; (1991) A System for Specifying and Rapidly Prototyping User Interfaces. *Taking Design Seriously: Exploring Techniques Useful in HCI Design*. Editor: John Karat. Academic Press.

RUIZ, Ana (1991); *En la Diferencia está el Gusto. Muy Especial Sexualidad*. Otoño. G+J: Madrid.

RUIZ VARGAS, José María; BOTELLA, Juan; (1982*) Atención y Capacidad de Procesamiento de la Información*. DELCLAUX, Isidoro; SEOANE, Julio; (Eds*). Psicología Cognitiva y Procesamiento de la Información: Teoría, Investigación y Aplicaciones*. Ediciones Pirámide: Madrid.

RUMBAUGH, James; (1996) *OMT Insights: Perspectives on Modeling from the Journal of Object Oriented Programming*. Nueva York: Sigs books. ISBN 0138469652. [Uniovi: CFMI K68A-0524].

RUSSELL, Ackoff; (1978) *Arte de Resolver Problemas*. 1ª Edición. LIMUSA. ISBN 968-18-1294-8. [Uniovi: CJS.BC EO-0058].

S

SACKS, Oliver; (1996) *Historias de la Ciencia y del Olvido*. Editorial Siruela: Madrid. ISBN: 84-78443282. [Uniovi: 917950].

SALTHOUSE, T.; (1986) *Perceptual, Cognitive and Motoric Aspects of Transcription Typing. Psychological Bulletin*.

SÁNCHEZ PADIAL, Jesús Antonio; FERNÁNDEZ GARCÍA, Ana María; GONZÁLEZ RODRÍGUEZ, Martín; REDONDO LÓPEZ, José Manuel; PAULE RUÍZ, María del Puerto; RAMÓN PÉREZ, Juan; (2000) *Enseñanza de la Historia con Hipermedia Proceso de desarrollo de Tirsus II*. Actas *del II Simposio Internacional de Informática Educativa (SIIE 2000).* Puertollano, Ciudad Real, del 15 al 17 de Noviembre de 2000. ORTEGA, Manuel; BRAVO, José;(Eds.) *Informática y Educación para una Sociedad Conectada*. ISBN 84-607-1341-5.

[*a*] SANCHEZ PADIAL, Antonio Jesús; (2000) *Tirsus II: Aplicación Hipermedia sobre la Historia Antigua de Asturias*. Proyecto de Fin de Carrera dirigido por GONZÁLEZ RODRÍGUEZ, Martín. Escuela Universitaria de Ingeniería Informática de Oviedo. Universidad de Oviedo.

[*b*] SÁNCHEZ PADIAL, Antonio Jesús; FERNÁNDEZ GARCÍA, Ana; GONZÁLEZ RODRÍGUEZ, Martín; REDONDO LÓPEZ, José Manuel; (2000) *Tirsus II: Aplicación de Hipermedia para la Enseñanza de la Historia*. Actas de *Interacción 2000*, Granada, España, del 19 al 20 de junio de 2000.

SARAUX, H.; BIAIS, B.; (1972) *Manual de Oftalmología*. Toray-Masson: Barcelona [Uniovi: CDU-617.7].

SAVIDIS, Anthony; STEPHANIDIS, Constantine; (1995) *Developing Dual User Interfaces for Integrating Blind and Sighted Users: the HOMER UIMS*. Actas de *ACM CHI'95*. ACM Press. [http://www1.acm.org/sigchi/chi95/proceedings/papers/sa_bdy.htm].

SCHNEIDER, W.; SHIFFRIN, R. M.; (1977) *Controlled and Automatic Human Information Processing: I. Detection, Search and Attention*. Pychological Review. 84.

SCHNEIDERMAN, B.; (1987) *Designing the user Interface*. Addison-Wesley : Reading, MA:

SCHWABE, Daniel; ROSSI, Gustavo; BARBOSA, SIMONE D. J.; BARBOSA; (1998) *Using Design Patterns in Educational Multimedia Applications*. Actas de ED-MEDIA'98. Canada.

SCHWABE, Daniel; ROSSI, Gustavo; SIMONE D. J.; BARBOSA; (1998) *Using Design Patterns in Educational Multimedia Application*. ED-MEDIA'98.

SCHWABE, Daniel; ROSSI, Gustavo; (1995) *Abstraction, Composition and Lay-Out Definition Mechanism in OOHDM*. Actas del *ACM Workshop on effective Abstractions in Multimedia*. San Francisco, California. [http://www.cs.tufts.edu/~isabel/schwabe/MainPage.html].

SCHWABE, Daniel; ROSSI, Gustavo; SIMONE D. J.; BARBOSA; (1996) *Systematic Hypermedia Applications Design with OOHDM*. Actas de *Hypertext'96*, ACM Press. [http://wwwx.cs.unc.edu/barman/HT96/P52/section1.html].

SCHWABE, Daniel; (1995) *Summary of OOHDM*. [http://www.cs.tufts.edu/~isabel/schwabe].

SHAPIRO, Stuart C.; (2000) *An Introduction to SNePS 3*. ICCS 2000.

SEIBEL, R.; (1963) *Discrimination Reaction Time for a 1023-Alternative Task. Journal of Experimental Psychology*, 66.

SHULZ, E.; VAN ALPHEN M.; RASNAKE W.; (1997). *Discovering User-generated metaphors through usability Testing*. Actas del *Second International Conference on Cognitive Technology*. Aizu, Japón.

SPERLING, Walter; (1969) *Juegos de test: ¿dónde radica mi fuerza?*. Editorial Vimala: Barcelona. [Uniovi: 767407 (Amicus)].

STERNBERG, Robert J.; (1986) *Las Capacidades Humanas*. Barcelona: Editorial Labor.

STERNBERG, Saul; (1975). *Memory Scanning: New Findigs and Current Controversies. Quarterly Journal of Experimental Psychology*.

STERNBERG, Saul; (1969). *The discovery of processing Stages: Extensions of Donder's method*. In W. G. Koster (ed.*), Attention and Performance*. Amsterdam.

STONE, D.; GLOCK, M. D.; (1981) *How do Young Adults Read Directions with and without Pictures? Journal of Educational Psychology*, vol 73, No. 3.

STROOP, J. Ridley (1935) *Studies of Interference in Serial Verbal Reactions*. Journal of Experimental Psychology, vol 18.

SUN MICROSYSTEMS; (1996) *The JavaBeans 1.0 API Specification*. [http://www.sun.com/beans].

@ SYMANTEC CORPORATION (1989-1995) *Visual Architect for C++. Vers. 8.0*. [http://www.symantec.com].

SZEKELY, P; LUO, P.; NECHES, R.; (1993) *Beyond Interface Builders: Model-Based Interface Tools*. USC/Information Sciences Institute.

SZEKELY, P; LUO, P.; NECHES, R.; (1992) *Facilitating the Exploration of Interface Design Alternatives: The HUMANOID Model of Interface Design*. Actas de *ACM CHI'92, The National Conference on Compunter-Human Interaction*. Mayo.

T

TAJES MARTÍNEZ, Lourdes, ÁLVAREZ GARCÍA, Fernando ÁLVAREZ GUTIERREZ, Darío, DÍAZ FONDÓN, María de los Angeles; (1998) *A Computational Model for a Distributed Object-Oriented Operating System Based on a Reflective Abstract Machine*. Actas de *12th European Conference on Object-Oriented Programming* (ECOOP).

@ TEK-TOOLS INCORPORATED (1995-2001) *Kawa: Simple yet Powerful Vers. 3.5*. [http://www.tek-tools.com].

THOMAS, Richard C.; (1996). *Temporal Aspects of Usability: Long-term Variation in User Actions*. SIGCHI Bulletin Vol. 28, No. 2, Abril 1996, ACM Press.

TOMLINSON, Petr; (1984) *Psicología Educativa*. Editorial Pirámide: Madrid. ISBN 8436802543.

TRANSTOOLS; (1998) JEDI: *Java Enabled Database over Internet*. ESPRIT Project (EP24231). [http://www.transtools.com/jedi].

TREISMAN, A. M.; (1960) *Contextual Cues in Selective Listening. Quarterly Journal of Experimental Psychology*, 12.

TYLER, Sherman; SCHLOSSBERG, Jon; (1992) *Interface Support for Comet: A Knowledge-based Software Reuse Environment*. ACM CHI 92.

TYLER, Sherman W.; SCHLOSSBERG, Jon L.; GARGAN, Robert A.; COOK, Linda K.; SULLIVAN, Joseph W.; (1991) *An Intelligent Interface Architecture for Adaptive Interaction. Intelligent User Interfaces*. SULLIVAN, Joseph W.; TYLER, Sherman W; (Eds.) ACM Press. [LCC: QTA76.9H85A73].

V

VELARDE LOMBRAÑA, Julián; (1994) *Filosofía del Conocimiento y Sistemas Expertos*. *El Basilisco*, 2ª. Época, No. 16. Oviedo.

VELARDE LOMBRAÑA, Julián; (1992) *Análisis Gnoseológico de la Teoría de los Sistemas Difusos (II). El Basilisco*, 2ª. Época, No. 11. Oviedo.

[*a*] VELARDE LOMBRAÑA, Julián; (1991) *Análisis Gnoseológico de la Teoría de los Sistemas Difusos (I). El Basilisco*, 2ª. Época, No. 10. Oviedo.

[*b*] VELARDE LOMBRAÑA, Julián; (1991) *Perspectivas en Ingeniería del Conocimiento. El Basilisco*, 2ª. Época, No. 8. Oviedo.

VELARDE LOMBRAÑA, Julián; (1991) *Gnoseología de los Sistemas Difusos*. Servicio de Publicaciones de la Universidad de Oviedo. ISBN: 84-7468-508-7.

VOS, Hans J.; (2000) *General Systems Theory as a Framework for Modeling the Development of Educational Courseware*. Actas de *EuroMedia 2000: Fifth Annual Scienfific Conference on Web Technology, New Media, Communications and Telematics Theory, Methods, Tools and Applications*. Amberes, Bélgica, del 8 al 11 de Mayo de 2000. ISBN 1-56555-202-4.

VURPILLOT, Éliane; (1985) *El mundo visual del Niño*. GOMEZ BELLARD, Francisco (Traductor *de Le Mode Visuel du Jeune Enfant*). Siglo Veintiuno: Madrid. ISBN 8432305065. [Uniovi: CDU-159.922.7].

W

WAHLSTER, Wolfang; (1991) *User and Discourse Models for Multimodal Communication*. SULLIVAN, Joseph W.; TYLER, W. Sherman (Eds.) *Intelligent User Interfaces*. Frontier Series. ACM Press.

WEBER, Joe; (1997) *Using Java 1.1*. Indianapolis: Editorial Que. ISBN: 0-7897-1094-3, [LCC: 96-72213], [Uniovi: KP-68ª-369].

WEINER, Irving B.; ELKIND, David; (1983) *Desarrollo Normal y Anormal del Preescolar*. Editorial Paidós: Barcelona. ISBN 8475092225. [Uniovi 37251 (Amicus)].

WERTHEIMER, M.; (1958) *Principles of Perceptual Organization*. Van Norstrand.

WHITELAW, M.; WECKERT, J.; (1997*) The Humanness of Object-Oriented Programming*. Actas del *Second International Conference on Cognitive Technology*. Aizu, Japón.

X

XING, Du; LI, Xie; (1992) *Open Model Methodology: A new Approach to the Development of Knowledge based User Interfaces. Software Engineering Journal*. 7 (2).

Y

YEO, Alvin; (1996) *World-Wide CHI: Cultural User Interfaces, A Silver Lining in Cultural Diversity*. ACM SGCHI Bulletin Vol. 28, No. 3. Julio 1996. [http://www.acm.org/sigchi/bulletin/1996/international.html].

Z

ZADEH, L. A.; (1975) *The Concept of a Linguistic Variable and its Application to Approximate Reasoning. Information Sciences*, 8.

ZADEH, Lofti, A.; (1965) *Fuzzy Sets*. Information and Control 8.

ZHOU, Michelle X.; FEINER, Steven; (1997) *Top-Down Hierarchical Planning of Coherent Visual Discourse. Intelligent User Interfaces.*

ZHOU, Michelle X.; FEINER, Steven; (1998) *Visual Task Characterization for Automated Visual Discourse Synthesis. CHI 1998.*

ZIMMERMANN, H. J.; (1996) *Fuzzy Set Theory and Its Applications*. Kluwer Academic Publishers: Boston.

26

26 Glosario y Acrónimos

La diferencia entre la palabra adecuada y la casi correcta es la misma que entre el rayo y la luciérnaga

Mark Twain

26.1 Glosario y Acrónimos

A

Acceso Secuencial
Método de acceso a un dato que prevé la exploración de todos los que le preceden. Esta es la estrategia seguida por el componente humano en tareas de exploración en espacios visuales y en su memoria a corto plazo (MCP).

Acceso Serie
Método de acceso a un dato que permite realizar directamente la función. Este modo de acceso es factible para un componente hardware diseñado para lograr el mayor nivel de eficiencia en la búsqueda. También puede ser empleado por los componentes humanos de un sistema al realizar tareas de recuperación de datos de la memoria a largo plazo (MLP) o en procesos de asociación mental de datos (*chunking*).

ACML (*Adaptive Contents Markup Language*)
Lenguaje empleado por GADEA para la definición de la semántica de una interfaz de usuario capaz de ser adaptada a nivel sintáctico. En este lenguaje, los contenidos de un artefacto hipermedia se definen libres de

contexto mediante una serie de primitivas de datos basadas en componentes. En GADEA, el lenguaje ACML sirve de puente entre los módulos CodeX y DEVA.

Adaptador
Mecanismo general para conectar entre sí dos o más componentes. Generalmente es un interfaz que permite la conexión entre dispositivos con conectores diferentes.

Adaptador de Comunicación
Módulo de DEVA encargado de la gestión y reparto del ancho de banda de un canal de comunicación a los distintos objetos a transmitir, asignado espacios en función de la prioridad asignada al objeto. Este módulo se encarga además de la configuración y adaptación de los parámetros globales del canal, en función del modelo cognitivo, perceptivo y motriz del usuario, así como del estado del ambiente operativo en el que se ejecuta la aplicación.

Agentes Interactivos
Componentes software capaces de examinar el conjunto de acciones básicas realizadas por el usuario sobre una interfaz, para inferir a partir de éstas los objetivos planteados por el usuario. Los objetivos así inferidos pueden ser empleados para modificar la apariencia o el comportamiento de la interfaz, detectar y advertir errores potenciales o ejecutar determinadas acciones para el usuario.

Agregado de Datos
Conjunto de elementos bajo un único nombre colectivo.

Alerta
Advertencia o informe acerca de un posible error producido. Por lo general se produce mediante un cuadro de alerta (ver *Cuadro de Alerta*), un sonido o una combinación de ambos.

Alfabeto
Conjunto de símbolos reconocidos en un lenguaje.

AMTG (*Apple `s Macintosh Technical Group*)
Organismo dependiente de Apple Computer Inc. en el que las compañías que desarrollan software para ordenadores Macintosh y Power Macintosh han de registrar sus productos.

Anidado
Operación que permite englobar una estructura en otra.

ANTS (Automatic Navigability Testing System)
Módulo de GADEA (ver *GADEA*) encargado de mantener actualizado los modelos de usuario particulares de cualquier usuario. Este sistema, basado en agentes especializados en la captura de información, analiza los patrones de comportamiento empleados por los usuarios durante su interacción con los diálogos interactivos creados por DEVA (ver *DEVA*) almacenando los resultados en el modelo de usuario respectivo.

APEX y GRID
Ejemplos de interfaces generadas dinámicamente por medio de agentes inteligentes basados en una arquitectura diseñada por Feiner la cual está basada en el Modelo Seeheim. Mientras que APEX es una aplicación para la generación de imágenes en 3D, GRID es una aplicación multimedia en la que se pueden incluir textos e imágenes en diálogos interactivos.

APDA (*Apple Programmer's and Developer Association*)
Organización dependiente de Apple Computer Inc. cuyo objetivo es la distribución de información técnica acerca de los productos Apple a los programadores y diseñadores. APDA es una gran fuente de notas técnicas, utilidades de programación, libros de referencia e información acerca de diferentes productos y herramientas para el desarrollo.

Aplicación
Programa o conjunto de programas escritos para la solución de un problema específico.

Applet
Clase de Java que permite que una aplicación pueda ser ejecutada como un objeto interactivo perteneciente a una página web. Los applets son reconocidos por la mayoría de

exploradores de Internet, incluyendo el Microsoft Explorer o el Netscape Communicator. Por razones de seguridad, la funcionalidad de los applets se encuentra restringida a menos que éstos hayan sido firmados y registrados por una autoridad certificadora.

Application Explorer
Aplicación de entrada en GADEA encargada de registrar nuevos usuarios en el sistema, permitir el acceso al mismo a los usuarios ya registrados y de transmitir al usuario el conjunto las aplicaciones registradas en GADEA para que éste seleccione aquella o aquellas que va a emplear durante una sesión de trabajo.

Application Record
Registro de configuración de GADEA (ver *GADEA*) en donde se incluye la dirección en donde se encuentran las clases de cada aplicación, así como el nombre de su clase principal y el punto de entrada a la respectiva base de conocimiento en función del entorno cultural del usuario tipo de la aplicación.

Application Manager
Módulo de CodeX en GADEA encargado de la gestión de las aplicaciones registradas en este sistema de gestión de interfaces de usuario. Este módulo es el responsable de arrancar las aplicaciones a petición del usuario, así como de establecer mecanismos de sincronización entre la interfaz de las distintas aplicaciones en ejecución.

Árbol
Estructura lógica ramificada que permite partiendo de un punto cualquiera, llegar al origen (el recorrido es unívoco).

Árbol AVL
También conocido como árbol simplemente equilibrado o árbol Adelson-Velski-Landis. Se caracteriza por ser una estructura de datos en forma de árbol binario en donde el número de nodos de una rama de cada subárbol solo puede superar en una unidad al número de nodos de la otra rama del subárbol. Esta estructura de datos garantiza una complejidad de búsqueda del orden de $O(Log_2N)$.

Arco, Modelo de
Ver *Modelo de Arco.*

Arrastrar
Acción de colocar el puntero en un objeto, pulsar el botón del ratón y desplazar dicho dispositivo mientras se mantiene pulsado el botón, situándolo en una nueva posición en la que se libera la pulsación.

Atajo de Teclado
Combinación de teclas que invocan a un comando desde un teclado. Por lo general, éstas combinaciones están formadas por una tecla de modificación y por la tecla correspondiente a una letra.

Atención
Concentración de las energías psíquicas de un individuo en uno –o varios– objetos.

Atención Focal
Atención empleada para captar –de forma consciente– determinadas características de los estímulos.

Atención Múltiple
Modalidad de atención en la que varios estímulos pueden ser percibidos de forma simultánea. Dependiendo del estímulo a recibido, el nivel de atención puede ser consciente o inconsciente.

Atención Subliminal
Modo de atención que permite la percepción de determinados estímulos sin necesidad de un proceso de atención consciente, es decir, los estímulos son percibidos *sin la intención* de ser percibidos.

Atención Visual
Parte de la atención encargada de la percepción, procesamiento y filtrado temprano de estímulos visuales.

Atributo
Indica una propiedad característica de uno o de más elementos.

AUI (*Adaptive User Interface*)
Sistema de gestión de interfaces de usuario desarrollado a partir de

diagramas de transición de estados que modelan el comportamiento de una aplicación desde la perspectiva del usuario. La principal virtud de este sistema radica en su capacidad para proporcionar varios modelos de interacción al usuario, el cual los puede emplear de forma simultánea. El usuario dispone de la capacidad de cambiar de un modelo a otro incluso dentro del ámbito de ejecución de un mismo proceso de usuario.

AWT (*Abstract Window Toolkit*)
Paquete de clases estándar que permite emplear ventanas, botones, menús, listas desplegables y demás *widgets* en el diseño de estilos de interacción WIMP (ver *WIMP*) en Java.

B

Background (fondo)
Área de la pantalla que circunda una figura o carácter.

Banner (Anuncio)
Sistema de transmisión de información empleado en campañas publicitarias en algunos sitios web de Internet. En los banners se emplea un mismo espacio visual para transmitir diversas imágenes en una secuencia cíclica.

Barra de Menús
Banda horizontal en la parte superior de la pantalla del ordenador o de una ventana, la cual contiene los nombres de los menús.

Barra de *Scroll*
Banda rectangular que se sitúa al costado de una ventana. Al pulsar o arrastrar el cursor sobre ella se puede desplazar el documento que ocupa la ventana.

Botón
Imagen que semeja a botón físico. Se emplea en xuadros de diálogo (ver *Cuadros de Diálogo*) en donde el usuario puede pulsar sobre él con el ratón para designar, confirmar o cancelar una acción.

Botón de Cancelar
Botón empleado en multitud de cuadros de diálogo (ver *Cuadros de Diálogo*). Al pulsar en él se cierra el cuadro de diálogo y la aplicación vuelve al mismo estado en el que se encontraba antes de que el *Cuadro de Diálogo* apareciese.

Botón de Radio
Widget estándar en muchas plataformas en el que se muestra una opción que puede tener solo dos estados: activado o desactivado. Cuando este *widget* forma parte de un grupo, sólo uno de los botones podrá estar activado.

Botón por Defecto
En un cuadro de alerta (ver *Cuadro de Alerta*) o cuadro de diálogo (ver *Cuadro de Diálogo*) se refiere al botón que se activa si el usuario presiona la tecla *Return* o Enter (ver *Tecla Return* y *Tecla Enter*).

Brillo
Medida de la cantidad de negro en un color (mientras menos cantidad de negro exista, más brillante será el color).

Buffer de Teclado
Zona de memoria en la cual el ordenador almacena las pulsaciones de teclado (realizadas a una velocidad mayor de la que el ordenador las puede procesar) para un procesamiento posterior.

Bug
Error en un programa. De este término se derivan: *debug*, que indica la operación de búsqueda de los errores y *debugger*, que indica el programa particular utilizado (ver *Debugger*) para detectar un error.

Búsqueda
Proceso de examen para la extracción de un elemento de una lista, si satisface determinadas condiciones.

C

CallBack
Llamada a un procedimiento realizada por un *widget* como respuesta a una acción del usuario.

Caja de Texto
Lugar o lugares dentro de un cuadro de diálogo en el que se puede introducir información escrita.

Campo
Representación de un dato, la cual está separada de otros campos por espacios en blanco, tabuladores u otros delimitadores específicos. También se aplica a una categoría particular de información en una base de datos.

Campo de Entrada de Texto
Area de forma rectangular situada en un cuadro de diálogo, dentro de la cual el usuario puede introducir texto.

Canal de Comunicación
Medio cualquiera a través del cual pueden transmitirse señales.

Capa (*Layer*)
Subsistema dentro de una jerarquía, el cual permite facilitar la creación de vistas fraccionadas del sistema más fáciles de comprender.

Carácter
Cualquier símbolo utilizado como parte de una colección de datos o para su control y organización. No coincide necesariamente con un carácter del alfabeto, puesto que para el ordenador se consideran caracteres también los códigos particulares utilizados para la gestión de los periféricos o como controles en los intercambios con otros sistemas.

Carpeta
ver *Directorio*.

Casos de Uso (*Use Cases*)
Técnica formalizada por Ivar Jacobson en la que se describen, bajo la forma de acciones y reacciones, el comportamiento de un sistema desde el punto de vista de un usuario, permitiendo definir los límites del sistema y las relaciones entre el sistema y el entorno.

CATNEOW
Metodología para la gerencia bancaria desarrollada a mediados de los ochenta en Venezuela. Está basada en los siguientes agentes que dan nombre a la metodología: *Clients* (clientes), *Actors* (actores), *Transformations* (transformaciones del problema), *Nature* (naturaleza del problema), *Environment* (ambiente) *Owners* (dueños del problema) y *Weltanshaung* (visión del mundo).

CHARADE (*Combining Human Assessment and Reasoning Aids for Decision making in environmental Emergencies*)
Sistema para la coordinación de equipos de salvamento en catástrofes medioambientales. El desarrollo de la interfaz de este sistema está basado en un paradigma de diseño centrado en tareas. Las tareas identificadas durante la fase de análisis son recogidas en un diagrama en forma de grafo, el cual incluye las relaciones conceptuales entre las diferentes tareas (procesos de usuario), así como los objetos sobre los que éstas actúan. CHARADE incorpora además un conjunto de técnicas para la conversión directa del diagrama de tareas en una interfaz de usuario capaz de proporcionar un soporte efectivo a la ejecución de dichas tareas.

Chat
Herramienta que permite establecer conversaciones habladas o escritas entre dos o más usuarios a través de una red de ordenadores.

CheckBox
Control que representa el estado de una propiedad, indicando que ésta se encuentra activada (valor cierto) o desactivada (valor falso). Pulsando en el *CheckBox* o en el texto que le sirve de etiqueta se invierte su valor.

CHORIS (*Computer-Human Object-Oriented Reasoning Interface System*)
UIMS para la creación de interfaces inteligentes en el que dominio de la aplicación controla y gestiona los módulos de inferencia a través de bases de conocimiento en las que se incluyen los modelos de usuario. Este sistema es capaz de realizar adaptaciones semánticas seleccionando los procesos de usuarios disponibles en un momento dado en función de estereotipo al que pertenece el usuario y de su historial de uso.

Chunk
Unidad mínima de información que maneja la memoria a largo plazo (ver *MLP* y *Chunking*).

Chunking
Mecanismo de asociación empleado principalmente por la memoria a largo plazo (ver *MLP*) para estructurar los conceptos en ella almacenados y poder recuperarlos más fácilmente. Se parte de trozos mínimos de información o *chunks* que se agrupan para crear otros *chunks* mayores, formando una estructura arbórea (ver *Árbol*). Se puede considerar a la memoria a corto plazo como un único *chunk*, contenedor a su vez de otros *chunks*.

Cinemedia Astur
Programa de investigación dirigido por Martín González Rodríguez, el cual está centrado en el desarrollo de una herramienta de autor para la creación de títulos multimedia basada en componentes y en tecnología cognitiva. La herramienta dispone de un lenguaje de especificación propio para los procesos de usuario, *precondiciones* y *postcondiciones* así como de capacidad para integración ANTS en la realización de pruebas de usabilidad automáticas para los productos multimedia generados.

Círculo Cromático
Tabla de colores en forma de círculo diseñada por Ostwald en los años veinte en la que, partiendo de tres colores primarios, se van mezclando éstos y sus productos de forma sistemática para obtener una extensa gama de colores.

Clave (*Password*)
Palabra única o conjunto de caracteres empleados para garantizar la seguridad en el acceso a determinados datos y procesos.

CLI (*Command Line Interface*)
Interfaz en el que el usuario da órdenes al sistema operativo escribiendo líneas de texto mediante teclado, en las que incluye comandos.

CodeX (*Code Xplorer*)
Módulo del sistema GADEA (ver *GADEA*) encargado de inspeccionar de forma automática el código binario de las aplicaciones compatibles con este sistema, en busca de procesos de usuario, *precondiciones* y *postcondiciones*. Este módulo es empleado por el programador para definir canales de comunicación y diálogos interactivos en los que es posible intercambiar información con el usuario. CodeX también realiza la adaptación de la carga semántica de las aplicaciones por medio de su *Language Manager*.

Código
Una representación de datos o de programas que puede ser comprendida o no por el ordenador. De este término se deriva el uso de la expresión codificación para indicar la operación de escritura de un programa, es decir , la traducción de las funciones a realizar en forma comprensible para el ordenador.

Cognitive Styles Analysis (CSA)
Batería de pruebas encaminadas a a clasificar a un individuo dentro de un determinado estilo cognitivo. Las pruebas consisten en tests en los que el usuario debe responder a una serie de preguntas en un lapso de tiempo muy corto. El test CSA clásico clasifica al individuo dentro de cuatro grandes grupos perceptivos en función de la capacidad verbal e imaginativa del observador y en base a su capacidad para realizar un análisis global o en detalle de todo objeto percibido.

Colores Activos
Se refiere a los veinticuatro colores del círculo cromático los cuales se encuentran libres de impurezas al haber sido obtenidos mediante combinaciones limpias. Estos colores poseen gran luminosidad, lo que los hace ideales para captar la atención.

Colores Cálidos
Gama de colores en cuya composición predominan los tonos rojos y naranja. Estos colores se asocian con objetos o conceptos de la naturaleza que transmiten una sensación de calor, tales como el fuego o el sol.

Colores Cuaternarios
Gama de doce colores obtenida al mezclar un color terciario con el color primario y secundario adyacente en el círculo cromático.

Colores Complementarios
Ver *Colores Secundarios*.

Colores Fríos
Grupo de colores en los que predomina la componente azul. Estos colores transmiten sensación de frío al estar relacionado su tono con el que poseen objetos de naturaleza fría, tales como la nieve o el hielo.

Colores Neutros
Colores sucios y grisáceos obtenidos al mezclar colores situados en diferentes sectores no adyacentes del círculo cromático.

Colores Primarios
Colores puros empleados para la base de la mezcla de cualquier otro color y a que a su vez no pueden ser obtenidos por mezcla. Los colores primarios son el amarillo, el cian y el magenta.

Colores Puros
Ver *Colores Activos*.

Colores Secundarios
Colores obtenidos por mezcla directa de los colores primarios en parejas. Los colores secundarios son el azul violáceo, el rojo naranja y el verde.

Colores Terciarios
Gama de colores resultante de mezclar los colores primarios con los colores secundarios adyacentes a los primeros en el círculo cromático (ver *Círculo Cromático*).

Comando
Orden emitida para utilizar una determinada función del sistema operativo, aplicación o lenguaje empleado.

Instrucción proporcionada por el usuario que obliga a un dispositivo a realizar una acción. Un comando puede ser seleccionado mediante voz, mediante un ratón o escrito mediante un teclado.

Compilador
Programa que puede traducir otros programas escritos en lenguaje simbólico a una forma comprensible para el procesador. El compilador –a diferencia del intérprete– realiza la traducción de todo el programa y genera una forma intermedia, llamada código objeto, que todavía no puede ser ejecutada.

Compresión
Método que permite la reducción de la longitud de un grupo de datos – generalmente ficheros – utilizando adecuados algoritmos de transformación. Operación apta para reducir el espacio ocupado por los datos, por ejemplo, eliminando los espacios vacíos.

Comunicación
Operación de transmisión y recepción de datos.

Comunicación Multimodal
Estrategia de comunicación en la que se emplean varios canales de comunicación de forma simultánea estableciéndose en ellos diferentes estilos de interacción con el objeto de cubrir, de forma amplia, los requisitos de interacción de múltiples usuarios.

Conductismo
Teoría psicológica que trata de explicar el comportamiento analizando las respuestas musculares y glandulares de los organismos con relación a ciertos cambios ambientales.

Contraste Cromático
Diferencia de intensidad en las componentes cromáticas del color de una forma y del color del fondo que la rodea.

Control
En la jerga de sistemas Macintosh o Windows, se refiere a un objeto en una ventana con el cual el usuario (por medio del ratón) puede causar una acción instantánea con resultados visibles o cambiar la configuración de una acción futura. Los controles carecen de sentido para usuarios con problemas de visión por lo que en la jerga de GADEA se hace referencia a los mismos en términos de *widgets*.

Control de Diálogos
Componente del Modelo Seeheim para el diseño de UIMS encargado de gestionar la sintaxis de una interfaz de usuario.

Conversacional
Método de interacción en el que el componente no humano propone al usuario una serie de datos esperando y validando las respuestas.

Convertidor Absoluto-Relativo
Módulo de un sistema de lógica difusa que transforma el valor absoluto de una variable en un valor difuso en función de los conjuntos difusos sobre los que trabaje la mencionada variable.

Convertidor Relativo-Absoluto
Módulo encargado en un sistema de lógica difusa de transformar el valor difuso de una variable lingüística en un valor absoluto. Esta transformación se hace en función del grado de pertenencia que posee el valor absoluto de la variable con respecto a cada uno de los conjuntos difusos con los que trabaja la variable.

CORBA (*Common Object Request Broker Architecture*)
Arquitectura diseñada para permitir la invocación de métodos de un objeto remoto como si se tratase de un objeto local. A diferencia de la arquitectura RMI de Java, CORBA está diseñado para poder ser empleado en cualquier tipo de lenguaje de programación.

Cortafuegos (*Firewall*)
Técnica empleada en algunas redes corporativas en las que todo el acceso a otras redes se realiza a través de un único nodo. En este nodo se suelen establecer mecanismos de control tanto de la información saliente, como de la información recibida.

CUBRICON (CUBRC Intelligent CONversational)
Sistema inteligente para la designación de objetivos y planificación de tráfico aéreo (CUBRC) de la fuerza aérea estadounidense. Dispone de una interfaz multimodal en la que el usuario imparte órdenes por medio de una combinación de voz, texto y señalización de objetivos mediante punteros. El sistema regula la cantidad de información a mostrar determinando el widget adecuado para ello en función de consideraciones de orden exclusivamente funcional, subestimando las características perceptivas de los usuarios.

Cuadro de Alerta
Ventana que aparece en la pantalla para advertir al usuario de una acción indebida o para notificarle un error.

Cuadro de Diálogo
Ventana que aparece en la pantalla para solicitar información del usuario o para reportar que la aplicación está esperando la finalización de un determinado proceso.

Cuadro de Diálogo Modal
Cuadro de diálogo que sólo permite al usuario trabajar con dicho diálogo. Todas las opciones de la aplicación quedan fuera del alcance del usuario y éste solo podrá abandonar dicho cuadro pulsando en alguno de sus botones.

Cuadro de *Scroll*
Pequeño rectángulo situado dentro de una barra de *scroll* entre las dos flechas. La posición relativa del cuadro de *scroll* dentro de la barra indica la posición de la parte visible del documento con respecto a su longitud total.

Cursor
Pequeña figura en la pantalla que permite seguir el movimiento del ratón o mostrar donde se efectuará la próxima acción del usuario. El cursor permite además transmitir información acerca del tipo de acción permitida mediante variaciones en la figura representada.

Curva de Yerkes-Dobson
Función que permite determinar del rendimiento de la atención humana de un individuo en función del número de los estímulos recibidos.

D

Data Fork
Uno de los dos componentes en los que se divide un fichero bajo el sistema operativo *Mac OS*. En esta parte del fichero se almacenan los datos a los que se accede a través del *File Manager*.

Debugger (Depurador)
Los *debuggers* son herramientas que permiten encontrar y corregir los errores de los programas. Estas herramientas suelen ir ligadas a los

compiladores de forma que el programador pueda comprobar y visualizar la correcta ejecución de un programa.

El *debugger* normalmente permite examinar instante por instante los contenidos de memoria y el estado de los datos, conduciendo a la identificación de los errores.

Defecto, por
Ver *Por Defecto.*

Desactivado
Conceptos no accesibles de momento por el usuario de una aplicación pero que son parte de ella. Aunque el usuario los percibe, éstos conceptos no pueden ser accedidos por hasta que el estado de ejecución de la aplicación no sea el adecuado. Suelen ser representados en gris en determinados sistemas de gestión de interfaces de usuario.

Destacar
Acción de hacer que un objeto aumente su contraste visual, generalmente cuando éste es seleccionado. Usualmente, este efecto se logra invirtiendo sus áreas blancas por negras u obscureciendo sus colores.

DEVA (Discourse Expert Valuator for Adaptation)
Módulo de GADEA (ver *GADEA*) que emula el comportamiento de un experto capaz de adaptar su discurso en función de la audiencia, empleando para ello un motor de lógica difusa. Dado un requerimiento de información basado en primitivas básicas de datos y *chunks* (ver *Chunk*), DEVA evalúa los *widgets* que pueden satisfacer dicho requerimiento en función de las características del usuario seleccionando los más apropiados en función de parámetros tales como la productividad o la facilidad de uso. Los *widgets* seleccionados son configurados y distribuidos por el canal de comunicación activo en función también de las características del usuario.

Diagrama de Transición de Estados
Sistema de notación que permite representar los estados de un autómata, así como los eventos que generan un cambio de estado. En ICH son empleados para representar los estados internos de una aplicación, así como las acciones que generan un cambio de estado.

Dial
Widget que representa gráficamente los rangos de valores que un usuario puede seleccionar, editar o visualizar.

Diálogos interactivos
Estrategia de comunicación en la que el papel de emisor y de receptor puede ser interpretado indistintamente por cualquiera de los participantes en el proceso de comunicación. Para ello es necesario el empleo de canales de comunicación dúplex.

Dialogue Component
Componente del Modelo de Arco encargado de establecer la secuencia adecuada para las tareas básicas que componen un proceso de usuario (ver *Modelo de Arco*).

Directorio ó Carpeta
Archivador de documentos, aplicaciones u otros directorios o carpetas. Los directorios permiten organizar la información en el modo deseado por el usuario, siguiendo una estructura jerárquica.

Diseñador de Diálogos
Entidad de DEVA encargada de la aplicación de las reglas de lógica difusa de este módulo en base a los hechos recogidos por ANTS para la configuración de las características de todo *widget* a incluir en un diálogo interactivo.

Doble Pulsación (*Doble Click*)
Doble pulsación del botón del ratón en rápida sucesión sin mover el cursor. La doble pulsación es interpretada como un solo comando distinguible en términos de interacción de una pulsación aislada.

Documento
Fichero que el usuario puede crear, abrir e imprimir.

DOF (*Degree of Fullfillment*)
Ver *Grado de Satisfacción*.

Domain-Adaptor Component
Componente del Modelo de Arco responsable de arrancar los procesos de usuario iniciados por la aplicación y de traducir las peticiones de información del usuario en invocaciones correctas a los procesos de usuario asociados (ver *Modelo de Arco).*

Domain-Specific Component
Componente del Modelo de Arco encargado de separar la funcionalidad de una aplicación de su interfaz (ver *Modelo de Arco*).

Dominio
Conjunto de datos del mismo tipo ligados por una relación.

DRC (*Design Rules Checking*)
Procedimiento de verificación de datos para evidenciar algunos tipos de error preestablecidos.

Duplex
Modo de interacción en los dos sentidos del canal de comunicación de forma simultánea. También se combina con el término *full duplex* para distinguir de la modalidad *half duplex* que no prevé la simultaneidad.

E

Early Selection Theory
Ver *Teoría de la Selección Temprana.*

Ensamblador (*Assembler*)
Programa del sistema utilizado para traducir a código máquina instrucciones escritas en lenguaje ensamblador.

Entorno Compartido (*Collaborative*)
Ambiente de trabajo compartido o aplicación facilita la comunicación y trabajo entre un grupo de personas.

Entruger
Herramienta para el desarrollo y distribución de encuestas por Internet. Basado en una arquitectura flexible que permite incorporar cualquier tipo de componente *JavaBean* en una encuesta, la herramienta gestiona la comunicación entre *applets* remotos (que contienen la encuesta) y un servidor central conectado a una base de datos en la que se recopila la información enviada por los participantes en la encuesta.

Ergonomía
Ciencia del diseño de ambientes de trabajo, la cual permite una interacción eficiente y segura entre los humanos y sus productos.

Escena
Se define como todo aquel conjunto de acontecimientos representados en un mismo ambiente o escenario. Se aplica a los componentes de un nodo en el modelo de navegación de una aplicación multimedia. En GADEA se emplea este término para identificar a los nodos del diagrama de estados que representa el armazón semántico de una aplicación. Es en las escenas donde se presentan los puntos de captura de datos en el proceso de interacción con el usuario.

Escuela de Graz
Ver *Escuela Dual.*

Escuela de la Forma
Ver *Gestalt*.

Escuela Dual
Corriente de pensamiento que propone que las percepciones se realizan a través de dos partes claramente diferencias: la sensación y la forma. Mientras que la sensación es producto de un estímulo exterior, la forma es considerada como un producto mental formado al combinar las sensaciones y las experiencias personales previas del sujeto receptor del estímulo.

Escritorio
Metáfora empleada por la interfaz de acceso de algunos sistemas operativos en donde el fondo de la pantalla simula una mesa de escritorio en la que se ubican los objetos de su equivalente del mundo real, tales como documentos, herramientas de trabajo o la papelera.

Espacio en blanco
Carácter cuya representación escrita es literalmente un espacio en blanco.

Esquema Canónico
Modelado de un conjunto de datos que representa la estructura de los propios datos independientemente tanto del uso que se deberá hacer de ellos como del hardware o del software particulares que lo emplean.

Estado Estándar
Tamaño y ubicación inicial de una ventana. Este estado es determinado por la aplicación.

Exploración
Operación con la que se examina de forma secuencial una determinada zona.

Extensión
Software que añade algún tipo de característica a una aplicación o sistema operativo.

F

File Creator (Macintosh OS)
Conjunto de cuatro caracteres añadidos a un fichero –de forma transparente al usuario –en ambientes Macintosh. El *File Creator* es utilizado para indicar el código de la aplicación que creó el fichero.

***File Type* (Macintosh OS)**
Conjunto de cuatro caracteres que se añaden a un fichero –de forma transparente al usuario– en ambientes Macintosh para indicar su tipo.

***Finder* (Macintosh OS)**
Aplicación que se encarga de gestionar la metáfora del escritorio en las plataformas Macintosh y de arrancar otros programas a petición del usuario.

Firewall
(ver *Cortafuegos*).

Fitt, Ley de
Ley de la psicología experimental que permite predecir el tiempo requerido por un individuo para desplazar un puntero sobre un objeto en base al tamaño del objeto y a la distancia que existe desde el mismo a la posición original del puntero. El tiempo de desplazamiento es proporcional a la distancia pero inversamente proporcional al tamaño del objeto destino.

Flechas de Control
Widget que consiste en dos pequeñas flechas sobre una misma dirección pero señalando sentidos distintos. Suelen ser empleadas para aumentar o disminuir un valor al pulsar sobre ellas con el cursor.

Flechas de *Scroll*
Flechas situadas en cada extremo de una barra de *scroll*. Al pulsar en una de estas flechas, se mueve el documento una línea. Si se mantiene presionada la flecha, el documento se desplazará de manera continua.

Folding
Técnica de conversión de datos a la forma deseada partiendo de otra diferente. Por ejemplo, la transformación de las letras minúsculas a mayúsculas.

Foreground
Area de pantalla ocupada por una imagen o carácter.

Framework
Ver *Marco Conceptual*.

Frecuencia
Número de veces por segundo en que una señal (periódica) asume el mismo valor; es el inverso de período.

Full-Duplex
Método de transmisión (ver *Simplex*) en el que las información se desplaza en dos sentidos al mismo tiempo.

Fuente (*Font*)
Juego de caracteres de una determinada dimensión y con características gráficas propias. En la tipografía tradicional, las fuentes estaban restringidas a un tamaño particular y estilo.

Foco
Punto de máximo interés visual de una imagen. La lectura d una imagen empieza por el foco y su ubicación depende del tipo de imagen y del entorno cultural en el que ésta ha sido diseñada.

Fuga
Zona de mínimo interés de una imagen. Suele ser la última zona explorada durante un proceso de búsqueda visual.

Función
Procedimiento que, de acuerdo con un algoritmo, restituye un valor dependiente de otros valores (parámetros).

Funcionalismo
Doctrina que considera los procesos psíquicos como funciones de adaptación al ambiente.

FuzzySys
Sistema de lógica difusa orientado a objetos para Java desarrollado por Gary Yat Chung Wong y Andi Hon Wai Chun para la Universidad de Hong Kong.

G

GADEA
Sistema de gestión de interfaces de usuario auto-adaptables. Está basado en tres módulos independientes entre sí que se encargan de separar la funcionalidad de una aplicación de su interfaz mediante operaciones automáticas de inspección de código basadas en reflexión (CodeX), adaptar el modelo de interacción y la distribución de la información en función de las características cognitivas, perceptivas y motrices de los usuarios (módulo DEVA) y obtener dichas características mediante una inspección constante de los patrones de comportamiento de los usuarios empleando agentes (módulo ANTS).

Gestalt
Escuela alemana fundada por Wertheimer, para el estudio del proceso de percepción humana. Esta escuela establece que la percepción de la realidad se realiza a través de formas o estructuras organizadas. Para los seguidores de esta escuela, el *todo* percibido es la suma de sus partes más las relaciones entre los elementos constitutivos de dicho *todo*.

Gestor de Accesos
Módulo encargado de garantizar un acceso correcto a la información confidencial almacenada en un modelo de usuario. Este módulo se encarga de verificar la identidad de quien accede, así como de codificar y de proteger los datos almacenados en dicho modelo.

Gestor de Conocimiento
Módulo del componente DEVA encargado de determinar la cantidad de información a transmitido en todo momento sobre el canal de comunicación activo. Para ello, este gestor descompone el diálogo interactivo aportado por CodeX, inspeccionándolo en busca de *chunks*, comprobando que su número no supere la capacidad del procesador humano del usuario activo.

Global Positioning System (GPS)
Metáfora empleada en los modelos de navegación de hipertexto para referirse al funcionamiento de las herramientas que permiten determinar en tiempo real la posición de un usuario dentro de dichos modelos. Esta característica es empleada para proporcionar al usuario información sensible al contexto.

Grado de Pertenencia
Probabilidad de que el valor absoluto de una variable pertenezca a un determinado conjunto difuso.

Grado de Satisfacción (*Degree of Fullfillment*)
Medida que indica en que medida un predicado satisface o cumple la lógica establecida en el antecedente de una regla en un sistema de lógica difusa.

Gramática
Ente formal para especificar, de una manera finita, el conjunto de cadenas de símbolos que constituyen un lenguaje.

GRID
Ver *APEX y GRID*.

Grupo de Atributos Mutuamente Excluyente
Grupo de atributos del cual sólo uno puede tener un valor a la vez. En un

cuadro de diálogo suelen estar representados por botones de radio.

Guardar
Proceso de almacenamiento de información al ser ésta transferida de la memoria principal a la secundaria.

GUI (*Graphic User Interface*)
Familia de interfaces de usuario en el que el usuario da órdenes al sistema operativo u aplicación mediante dispositivos de posición en dos dimensiones (ratón, pantallas sensibles, etc.) y éste obtiene respuesta mediante representaciones iconográficas que por regla general emulan objetos y metáforas del mundo real.

Guiado por Eventos
Programa que responde a la interacción del usuario en tiempo real verificando repetidamente los posibles eventos enviados por rutinas de interrupción. Un programa guiado por eventos permanece inactivo hasta que detecta un evento.

GUIDE (*Graphic User Interface Design Environment*)
Herramienta para el desarrollo de interfaces de usuario en el UIMS AUI (ver *AUI*).

GUIPW(*Graphic User Interface Package for Windows*)
UIMS que basa su mecanismo de separación entre interfaz y funcionalidad en los métodos virtuales de la programación orientada a objetos. La estrategia planteada por este sistema consiste en asignar objetos interactivos (pertenecientes al modelo conceptual de la aplicación) a una colección de elementos de la interfaz que se pueden reutilizar y que implementan el nivel léxico, sintáctico y semántico de la misma. Cuando es necesario asignar carga semántica a uno de estos elementos, el método correspondiente al aspecto funcional a adaptar, debe ser redefinido en un nuevo objeto que hereda del elemento original.

H

Habilitado (*Enabled*)
Estado de un proceso que permite interrupciones.

Half Duplex
Método de transmisión en los dos sentidos alternativamente (ver *Duplex*).

Hard Copy
Copia sobre papel de una salida.

HCI (*Human Computer-Interaction*)
Término empleado para referirse a todas aquellos estudios que tienen como objetivo favorecer una comunicación natural entre el humano y el ordenador. Estos estudios provienen de muy diversas disciplinas entre las que se incluyen la ergonomía, la psicología, la pedagogía y la informática.

Hick, Ley de
Función de predicción que permite estimar el tiempo necesario para tomar decisiones sencillas. Ante una decisión en la que todas las posibles opciones presentan la misma probabilidad de ser elegidas, la estimación dependerá de una función logarítmica (en base dos) que recibe como argumento el número de opciones disponibles.

HTTP (*HyperText Transport Protocol*)
Protocolo de capa de aplicación para la transmisión en Internet de documentos basados en hipertexto. En este protocolo los enlaces entre nodos pueden ser locales (ubicación relativa a la ubicación del nodo de partida) o remotos (ubicación absoluta mediante una URL).

HUMANOID
Ver *UIDE*.

I

ICH
Ver *Interacción y Comunicación Humana*).

Icono
Representación gráfica de un objeto, concepto o mensaje, el cual es utilizado en interfaces GUI para representar ficheros y procesos.

IDE (*Integrated Development Enviroment*)
Programa de aplicación que permite reunir bajo una única metáfora de interacción un conjunto de herramientas para el desarrollo de programas. Estas herramientas suelen estar compuestas por editores de código, compiladores e intérpretes, *debuggers* y documentación.

Ideograma
Símbolo que representa una idea o concepto.

Ingeniería de Factores Humanos
También conocida como ingeniería humana, esta disciplina está relacionada con la forma de diseñar máquinas, operaciones y medios de trabajo de tal forma que se tomen en cuenta las capacidades y limitaciones humanas.

Integridad
Indica la ausencia de errores en un conjunto de datos.

Interactivo
Agente que interacciona con el usuario. Se considera que una tarea es interactiva si el desarrollo de la misma se efectúa con la participación activa del usuario.

Interacción y Comunicación Humana (ICH)
Ciencia que estudia los distintos mecanismos que tienen los humanos para comunicarse ideas, bien de manera directa a través de medios diseñados para ello (la voz, el lenguaje corporal, la escritura, etc.), o bien a través de objetos de la vida cotidiana no necesariamente diseñados ni fabricados para comunicarse (un coche, un martillo, un bolígrafo, etc.). Este término engloba al sajón HCI (ver *HCI*).

Interaction-Toolkit Component
Componente del Modelo de Arco encargado de realizar la interacción directa con el usuario, así como de implementar el nivel sintáctico y léxico de la interfaz (ver *Modelo de Arco*).

Interfaz
Protocolo de comunicación que adapta diversas señales de un dispositivo origen a las exigencias particulares los dispositivos receptores de dichas señales. Por ejemplo, en las transmisiones entre diferentes ordenadores o entre un ordenador y los periféricos, las señales deben adaptarse tanto antes de la transmisión como después de la recepción.

Interfaz de Aplicación
Módulo del Modelo Seeheim responsable de la interacción entre un UIMS y la aplicación. En este módulo se incluyen todos los mecanismos de separación entre interfaz y funcionalidad implementados por el sistema de gestión de interfaces de usuario.

Interfaz de Usuario
Reglas y convenciones mediante las cuales un sistema software se comunica con el humano que lo opera.

Internet
Red de ordenadores conectados a nivel mundial.

Interprete
Programas que ejecutan las instrucciones que encuentran en un código fuente. En muchos casos coexiste en memoria el programa fuente y el interprete en tiempo de ejecución.

Introspección (*Introspection*)
Capacidad que tiene un objeto para obtener información sobre si mismo.

Inxena
Programa de investigación dirigido por Martín González Rodríguez, el cual tiene por objetivo analizar los distintos factores que integran el modelo cognitivo humano. La información recabada se emplea para crear agentes interactivos que emulan el comportamiento humano y sirven de soporte a usuarios en la realización de tareas complejas.

IP (*Internet Protocol*)
Protocolo de comunicación no fiable y no basado en conexión situado en la capa de red de un sistema de comunicación. Este protocolo no intercambia información de control antes de transmitir datos a un sistema remoto. Los paquetes de datos son enviados al destinatario suponiendo que éste los tratará de forma adecuada. La tarea de verificar que la información contenida en los paquetes de datos es correcta será responsabilidad de la capa superior (de transporte) normalmente implementada mediante un protocolo llamado TCP (*Transmission Control Protocol*). IP define un sistema de direcciones universal único compuesto por un número de 32 bits. Dado un paquete IP, la información puede ser dirigida a su destinatario en base a la dirección IP establecida en la cabecera del paquete.

J

JASMINUM (*Joining ADV and State Machines in a Notation for User Interface Modeling*)
Formalismo visual para el diseño de sistemas interactivos. Se trata de una máquina de estados dirigida por eventos la cual permite especificar formalmente requisitos para la interfaces de usuario. JASMINUM proporciona soporte para un diseño centrado en el usuario en el que la apariencia del diseño es similar a la apariencia de la interfaz final.

JavaBeans
Modelo de componentes de Java. Permite especificar el comportamiento y apariencia de un componente, así como facilitar la comunicación entre componentes.

JDK (*Java Developer Kit*)
Conjunto de herramientas para el desarrollo de aplicaciones Java proporcionadas gratuitamente por Sun Microsystems. Entre estas herramientas se incluyen compiladores, *debuggers*, interpretes de Java, generadores automáticos de documentación, así como programas de soporte a tecnologías Java como por ejemplo RMI (ver *RMI*).

Jerarquía
Estructura de elementos subdividida en diferentes niveles, con distintos grados de importancia o prioridad.

Joystick
Dispositivo de interacción constituido por una palanca con traductores de posición que permite con sus desplazamientos controlar la posición del cursor o de otro elemento de vídeo.

JSP (*Java Servlet Pages*)
Versión especial de los *servlets* de Java (ver *Servlets*) en los que el código de los mismos se incluye dentro del código en el que están escritas páginas web estáticas, substituyendo aquellas zonas de la página en las que se va a incluir contenido dinámico. La primera vez que un usuario visita la página que contienen código JSP, el servidor web ordena la compilación de dicho código, generando un *servlet* el cual será empleado de ahora en adelante para proveer el contenido dinámico del código JSP.

K

Kawa
Entorno integrado para el desarrollo de aplicaciones Java desarrollado por *Tek-Tools*. Su diseño modular permite emplearlo con cualquier versión del JDK (ver *JDK*).

KLIDE (*Knowledge-based Layered Interface Development Environment*)
UIMS que permite realizar una separación efectiva entre funcionalidad e interfaz mediante un análisis léxico, sintáctico y semántico del conocimiento intercambiado con sus aplicaciones. El modelo está basado en cinco capas por las que transcurre la información. Las acciones básicas realizadas por el usuario con los dispositivos de entrada (ratón y teclado) son agrupadas en conceptos y éstos en secuencias de conceptos, las cuales son puestas en el contexto del dominio de la aplicación para determinar al intención final del usuario ya sí poder ayudarle a ejecutar su tarea.

L

Language Manager
Componente de CodeX (ver *CodeX*) encargado de adaptar el significado de los conceptos manejados por los diseñadores de una aplicación a nivel de metáfora en aquella representación que más se aproxime más al modelo mental empleado por el usuario para referirse al concepto. Para ello se usa un árbol semántico basado en una jerarquía de culturas.

Late Selection Theory
Ver Teoría de la Selección Tardía.

Layout
Dibujo que representa una entidad física normalmente a escala.

Lenguaje
Notación formal para describir conjuntos de datos, algoritmos o funciones. En términos formales, un lenguaje es un conjunto de cadenas de símbolos de un alfabeto determinado.

Léxico
Conjunto de todas las palabras y símbolos que componen un lenguaje. En los lenguajes de programación el léxico lo constituyen todos los elementos individuales del lenguaje, denominados *tokens.*

Ley Exponencial de la Práctica
Ley de la psicología experimental que determina el tiempo necesario para ejecutar una tarea por parte de un individuo a partir del tiempo empleado por éste la primera vez que dicha tarea fue ejecutada y del número de veces que la ha realizada.

Línea (de comunicación)
Ver *Canal de Comunicación*.

Localización (*Localization*)
Proceso de adaptación del software a una región particular, idioma o cultura. Este proceso incluye la adaptación del formato de las fechas, indicaciones horarias, formato de los números, fuentes, etc.

Lógica Difusa
Marco conceptual desarrollado por Lofti A. Zadeh a finales de los años sesenta para la aplicación de las reglas de la lógica en problemas en los que los grados de verdad de un predicado no son booleanos, es decir, en aquellos en donde existen varios grados de verdad.

M

Macintosh
Marca registrada por Apple Computer para su serie de ordenadores personales. El primer Macintosh apareció en el mercado en 1983. La novedosa concepción de su GUI, desconocido hasta el momento en el mundo comercial representó todo un hito en la historia de la ICH (ver *ICH*).

Mapa de Pixeles
Conjunto de valores que representan las posiciones y estados del conjunto de pixeles que forman una imagen.

Marco Conceptual (*Framework*)
Estructura mental empleada por un grupo humano para referirse un sistema, la cual sirve para el intercambio de información entre todos los miembros de dicho grupo. Esta estructura permite modelar un sistema, modificarlo y realizar predicciones sobre el efecto que puedan tener determinados cambios en su estructura.

Máscara
Conjunto de caracteres utilizado para controlar la adquisición o la eliminación de algunos caracteres que pertenecen a todo el conjunto.

MA (Memoria Auditiva)
Ver *Memoria Auditiva.*

MCP (Memoria a Corto Plazo)
Término empleado en psicología para referirse a la memoria efímera empleada por los humanos para almacenar datos en procesos cognitivos de corta duración, generalmente asociados con aquellos procesos en los que se requiere una atención consciente. En esta memoria se pueden almacenar entre 5 y 9 elementos (*chunks*) aunque cada uno de los cuales

puede agrupar a su vez a una serie de elementos.

Memoria Asociativa
Memoria accedida en base al contenido y no por medio de la dirección. El método permite un notable aumento de la velocidad de búsqueda en base a los atributos del campo.

Memoria Auditiva
Memoria incluida dentro de la memoria a corto plazo (ver *MCP*) encargada del almacenamiento de objetos percibidos a través del canal de comunicación auditivo. Su capacidad de almacenamiento se mide en letras en lugar de en *chunks* y se sitúa en torno a los cinco elementos.

Memoria a Corto Plazo(MCP)
Ver *MCP*.

Memoria Episódica
Ver *Memoria a Corto Plazo*.

Memoria a Largo Plazo(MLP)
Ver *MLP*.

Memoria Semántica
Ver *Memoria a Largo Plazo*.

Memoria de Trabajo
Ver *Memoria a Corto Plazo*.

Memoria Visual
Tipo de memoria empleada para el almacenamiento de objetos percibidos de forma visual. Esta memoria forma parte a su vez de la memoria a corto plazo (ver *MCP*), su capacidad de almacenamiento se mide en letras y se sitúa en torno a las diecisiete letras.

Menú
Lista de las funciones previstas en un programa, con posibilidad de selección de la deseada. En la metáfora del escritorio, los menús aparecen cuando el usuario apunta y presiona en el título de un menú en la barra de menús. Arrastrando el cursor sobre el menú y liberando el botón cuando el comando deseado es resaltado provoca la invocación del comando.

Menú Archivo
Menú que contiene comandos u operaciones que afectan a todos los documentos de una aplicación. Los procesos de usuario incluidos en este menú suelen ser del tipo de *Abrir*, *Guardar*, *Imprimir* y *Salir*.

Menú Contextual
Abstracción del concepto de menú desplegable que permite situar éste sobre cualquier *widget* que represente a un objeto visual. En este menú se suele desplegar el conjunto de procesos de usuario disponibles para el objeto en un momento dado de la ejecución de la aplicación.

Menú Emergente
Menú no ubicado en la barra de menús, sino que aparece cuando el usuario pulsa el botón del ratón en un punto particular de la pantalla.

Menú Fichero
Ver *Menú Archivo*.

Menú Jerárquico
Tipología de menú en el cual uno o más de sus ítems puede contener a su vez otro menú.

Merise
Enfoque metodológico que integra formalismos de datos y de procesos mediante una aproximación sistemática de la solución de un problema apoyada por la TGS (ver *Teoría General de Sistemas*). Esta metodología es la empleada por las administraciones públicas del Estado Francés y se basa en tres niveles: nivel conceptual, nivel de organización y nivel operacional.

Metodología
Método de trabajo que facilita la estructuración del pensamiento y la selección y enlace de técnicas para resolver problemas.

Metodología Blanda
Metodología diseñada para su aplicación en sistemas en los cuales los objetivos son de difícil definición, la toma de decisiones es incierta, las medidas de eficiencia son cualitativas y la acción humana es irracional. En los sistemas blandos, el humano aparece como principal protagonista.

Métrica
Enfoque metodológico basado en un desarrollo progresivo de modelos de

datos y de procesos hasta obtener un esquema conceptual del sistema a modelar, para luego dividirlo en un modelo externo (usuario) y en un modelo interno (sistema). Esta metodología es la empleada por las administraciones públicas del Estado Español.

MPH
Ver *Modelo de Procesador Humano*.

Modelo de Arco
Patrón base para el diseño de una arquitectura de UIMS flexible y sensible a futuros cambios en la tecnología. Este modelo está basado en tres componentes dispuestos sobre un arco, los cuales están conectados entre sí por dos adaptadores. En un extremo del arco se sitúa la aplicación y en el otro el usuario. El componente más cercano a la aplicación se encarga de realizar la separación entre funcionalidad e interfaz, mientras que el componente más cercano al usuario se encarga de la interacción directa con el mismo. El componente central se encarga de establecer la secuencia adecuada para todas y cada una de las tareas previstas en el dominio del problema.

Modelo de Arco para Interfaces Inteligentes
Evolución del modelo de arco que incorpora un motor de inferencias que permite la generación de diálogos interactivos adaptados a las características de los usuarios de los UIMS construidos bajo esta perspectiva. El resto de los componentes del Modelo de Arco original (ver *Modelo de Arco*) se encargan de suministrar la información de partida necesitada por este componente para establecer sus inferencias.

Modelo de Capacidad
Modelo que explica la limitación existente en la cantidad de información que un individuo puede procesar presuponiendo que el límite no está impuesto en su procesador perceptivo (el cual puede aceptar cualquier cantidad de información) sino en la capacidad limitada del procesador cognitivo para ejecutar varias tareas en paralelo.

Modelo de Cuello de Botella
Modelo empleado para explicar la limitación en la capacidad de procesamiento de información humana en el que se presupone un límite a la cantidad de información que se puede percibir, considerando que toda información que no supere dicho límite puede ser procesada.

Modelo de Datos
Representación detallada del conjunto de datos manejados por una aplicación así como su tipo y sus características básicas tales como *volatidad*, tamaño, etc.

Modelo de Interacción
Conjunto de datos que un usuario debe obtener o proporcionar a un sistema, así como los procesos necesarios para realizar este intercambio de información en un escenario dado.

Modelo de Procesador Humano (MPH)
Modelo diseñado por Card, Moran y Newell circa 1980 para explicar la cognición humana. El modelo está formado por tres procesadores (procesador cognitivo, procesador perceptivo y procesador motriz) y por cuatro memorias (memoria a largo plazo, memoria a corto plazo, memoria visual y memoria auditiva). Este modelo puede considerarse como la evolución natural del Sistema P (ver *Sistema P*) y tiene su continuidad en la teoría unificada de la cognición (ver *TUC*).

Modelo de Procesos
Representación de todos los procesos y acciones que una aplicación debe realizar para representar el sistema real que emula. Dependiendo del enfoque metodológico empleado para realizar modelar el sistema, existen varias formas de representar esta relación.

Modelo de Usuario
Repositorio en el que se almacena información acerca de las características específicas de un determinado usuario. Compilación de información que describe al usuario concreto de una aplicación. Esta información es empleada entre otras cosas para determinar como deben ser representados los datos

proporcionados por la aplicación, que modelo de interacción es el adecuado y que tipo de sistema de ayuda debe ser usado. En GADEA, estas características pueden ser de tipo cognitivo (capacidad de las memorias, tiempos de reacción, etc.), perceptivo (agudeza visual, auditiva, etc.) o motriz.

Modelo de Usuario por Aplicación
Base de datos en la que se almacena información relativa a las características de manejo de una aplicación determinada por parte de un usuario concreto. Esta información incluye el tiempo requerido por el usuario para satisfacer los requisitos de interacción de cada proceso de usuario o diálogo interactivo de la aplicación, así como el número de veces que éstos han sido empleados. Esta información, almacenada y gestionada por CodeX es empleada para determinar el grado de veteranía de un usuario con respecto al uso de determinadas funciones de una aplicación concreta.

Modelo Seeheim
Modelo para el diseño de sistemas de gestión de interfaces de usuario (UIMS) presentado en un workshop sobre este tipo de sistemas en la localidad alemana de Seeheim. Desde entonces, este diseño ha representado un hito para la construcción de UIMS puesto que incluye la estructura básica que debería tener todo sistema con capacidad de adaptación y separación entre funcionalidad e interfaz.

Motor de Inferencias
Módulo de un sistema de lógica encargado de la toma de decisiones en función del valor de determinadas variables y de un conjunto de reglas que hacen uso de las mencionadas variables.

MLP (Memoria a Largo Plazo)
Abstracción empleada en la psicología cognitiva para referirse a aquella parte de la memoria en la que se almacenan datos por un período de tiempo largo, que puede ir desde algunos minutos a toda la vida.

Modelo Aditivo
Modelo de color basado en la mezcla aditiva de tonos básicos de color para la generación de colores adicionales.

Multitarea
Posibilidad de que varias aplicaciones puedan ejecutarse al mismo tiempo en un ordenador personal. Para ello se asignan porciones de la memoria principal a cada aplicación, la cual tendrá entonces un espacio de direcciones propio no accesible por las aplicaciones de su entorno.

MV (Memoria Visual)
Ver *Memoria Visual*.

N

Nodo
Punto terminal de la conexión de dos o más *subgrafos*. Se identifica comúnmente con un punto de referencia en el modelo mental de navegación del usuario

Notación
Alineado de las cifras que componen un número para obtenerlo en la forma requerida por las reglas aritméticas.

O

Objetos Predador
En el ámbito de GADEA, éstos objetos representan pequeños fragmentos o detalles de un concepto global. Dadas las características específicas de estos componentes, éstos son transmitidos como complemento a la información global contenida en un objeto señuelo (ver *Objetos Señuelo*).

Objetos Señuelo.
Componentes empleados por GADEA para representar objetos dotados de gran carga semántica pero que, gracias a que ocupan muy poco espacio físico del ancho de banda de un canal de comunicación, pueden transmitirse en grandes cantidades. Estos objetos están diseñados de acuerdo con la teoría de los modelos de atención bifásicos y tienen por objeto una rápida captura de la atención del usuario.

OOHDM (*Object Oriented Hypermedia Development Model*)
Metodología de desarrollo de artefactos hipermedia orientada a objetos. Partiendo de una base de

conocimiento única sobre el dominio del problema, se pueden crear varias vistas de dicha base en función de los requisitos del usuario y mediante el diseño de contextos de navegación

OMT (*Object Modelling Technique*)
Enfoque metodológico basado en el paradigma de la orientación a objetos en el que los objetos se articulan alrededor de las nociones de clase y de asociación.

Oviedo-3
Prototipo de un sistema operativo desarrollado integralmente con tecnología basada en la orientación a objetos. Este sistema operativo se fundamenta en una máquina abstracta dotada de reflexión estructural.

P

Paleta
Nombre dado a un menú o ventana plegable. La paleta permanece visible en la pantalla para que el usuario pueda trabajar con ella sin necesidad de extenderla cada vez que lo necesite (lo que sí ocurre con los menús convencionales). Una paleta puede ser parte de una ventana y por lo general proporciona herramientas o algún tipo de objetos a elegir, como por ejemplo colores o patrones de relleno.

Patrón
Esquema de diseño que tiende a repetirse de modo invariable en distintas aplicaciones que comportan requisitos comunes o similares.

PC (*Personal Computer*)
Denominación de la primera gama de ordenadores personales de IBM. Por extensión se denomina también con este nombre a los ordenadores compatibles con los equivalentes IBM. El primer IBM-PC vio la luz en 1981.

PD (*Public Domain*)
Software de distribución gratuita que puede ser utilizado, copiado y repartido sin coste alguno.

Pegar
Colocar el contenido del portapapeles en un punto de inserción (ver *Portapapeles*).

Persona
En términos de interacción y comunicación humana, este término se refiere al individuo de la especie humana que ha alcanzado un grado de civilización suficiente como para integrarse de forma activa en una sociedad. Nótese por tanto que no todo humano puede considerarse como persona. En algunas corrientes filosóficas, aquellos humanos que no han alcanzado el grado de persona o que, debido a un comportamiento antisocial han perdido dicho grado, son conocidos por el apelativo de personas-cero.

PERSONA
Propuesta de arquitectura para UIMS adaptables basada en modelos de usuario portátiles (persona) y agentes sastre encargados de determinar la apariencia de los componentes visuales de una interfaz en función de los datos almacenados en los modelos de usuario y en base a preguntas directas al usuario.

Pixel
Abreviación del término anglosajón *picture element* (elemento de una imagen) que representa el punto más pequeño que se puede dibujar en una pantalla. También se emplea para referirse al lugar en la memoria de vídeo que se corresponde con un punto en la pantalla.

Poka-Yoka
Término japonés que literalmente significa *a prueba de errores*. Esta filosofía se emplea en los sistemas de producción JIT (*Just in Time* o *En el Momento Preciso*) y están orientados a fabricar piezas y componentes cuya instalación y manejo sea a prueba de errores con el objetivo evidente de garantizar la calidad de los productos así fabricados.

Por Defecto
Valor, acción o estado que un sistema informático asume a menos que el usuario le proporcione instrucciones explícitas para realizar lo contrario.

Portapapeles (*Clipboard*)
Área de memoria para mantener los datos adquiridos recientemente mediante una operación de cortado o copiado. Los datos contenidos en el portapapeles se pueden pegar en documentos.

Postcondición (*Postcondition*)
Estado en el queda una aplicación u objeto tras la ejecución de una operación determinada.

Precondición (*Precondition*)
Conjunto de condiciones que han de estar presentes para que un proceso pueda ser ejecutado.

Presentación
Módulo del Modelo Seeheim encargado de la gestión de representación de aspectos léxicos de una interfaz de usuario.

Presentation Component
Componente del Modelo de Arco responsable configurar los widgets empleados por el Dialogue Component en la creación de sus diálogos interactivos (ver *Modelo de Arco*).

Principio de la Incertidumbre
Ver *Hick, Ley de*.

Procesador Cognitivo
En términos del modelo de procesador humano (ver *MPH*) se refiere a aquel componente encargado de la toma de decisiones de alto nivel cognitivo mediante la aplicación de estrategias de deducción y de razonamiento.

Procesador Motriz
Módulo del modelo de procesador humano (ver *MPH*) encargado de controlar las salidas del sistema. Este módulo posee un control directo de los músculos del cuerpo a los que ordena movimientos. Dado que la precisión de este módulo es muy baja, requiere de un control constante por parte del procesador cognitivo (*ver Procesador Cognitivo*).

Procesador Perceptivo
Módulo encargado de la adquisición de información (visual, olfativa, táctil, auditiva o gustativa) en el marco del modelo de procesador humano (ver *MPH*).

Procesamiento Automático
Modo de procesamiento de la información percibida en la que no es necesaria una participación activa por parte del procesador cognitivo. Se trata de las tareas que pueden realizarse de forma inconsciente. Esta estrategia se emplea para tareas realizadas con anterioridad y en las que se ha logrado un alto grado de práctica.

Procesamiento Controlado
Estrategia de procesamiento de información proveniente del procesador perceptivo en el que es necesaria una participación activa del procesador cognitivo. Esta estrategia es empleada cuando la tarea a realizar es inusual y por lo tanto novedosa para el individuo.

Programa de Aplicación
Ver *Aplicación*.

Programación Orientada a Objetos
Modelo de programación en la que se identifican y definen clases de objetos (tipos de datos u objetos del dominio del problema con funcionalidad propia) que encierran una determinada funcionalidad a través de la cual los objetos o instancias concretas de una clase dan respuesta a las solicitudes (mensajes) que les envían otros objetos.

Programación Visual
Mecanismo de diseño mediante el cual es posible visualizar mediante prototipos la ubicación espacial exacta de los *widgets* necesarios para satisfacer determinados requisitos de interacción, así como su configuración visual.

Protocolo
Conjunto de reglas que controlan el intercambio de mensajes entre dos agentes a través de un canal de comunicación.

Prototipo
Versión de prueba de una interfaz diseñada de forma rápida y barata y cuyo objetivo se concentra en descubrir patrones de comportamiento de usuarios tipo, detectando así requisitos de interacción que pueden pasar desapercibidos mediante el empleo de otras técnicas formales de análisis. Los prototipos suelen carecer de

funcionalidad y se van refinando poco a poco al ir incorporando los requisitos descubiertos en las sesiones de prueba precedentes.

Proxy
Aplicación empleada con filtro en algunas redes corporativas que tiene la facultad de canalizar las peticiones de información de los nodos de la red a la que sirve. La centralización de las peticiones de datos en una sola aplicación permite filtrar la información que sale o llega a una red, estableciendo controles de seguridad, así como diferentes niveles de memoria caché en la que se depositan los resultados de las peticiones más frecuentes realizadas desde los nodos de red controlados por el proxy.

Pruebas de Usabilidad
Batería de tests y pruebas de utilización a las que son sometidos un conjunto de voluntarios que se supone se asemejan a los usuarios finales de una aplicación. Los voluntarios han de realizar una serie de operaciones típicas con la aplicación bajo prueba, mientras los investigadores observan su comportamiento en busca de fallos en la concepción del modelo de interacción de la aplicación.

Pruebas Remotas de Navegación
Conjunto de técnicas y herramientas aplicables a la prueba de los mecanismos de navegación a nivel semántico de la interfaz de una aplicación. Estas pruebas son realizadas en la máquina del usuario y sin la presencia de observadores humanos. Para ello, es necesario el empleo de agentes software inteligentes capaces de observar las acciones realizadas por el usuario y de transmitir estas observaciones a un servidor central para un análisis posterior.

Punto
Unidad de medida en donde 12 puntos equivalen a 1 pica. Dado que 6 picas son una pulgada, un punto equivale aproximadamente a 1/72 pulgadas. La resolución de una imagen suele medirse en puntos por pulgada.

Punto de Inserción
Posición en donde será insertado el texto en los *widgets* de entrada de datos. Esta posición suele ser señalada por medio de una línea vertical que parpadea.

R

Rango
Límites entre los cuales está comprendido el valor de un determinado elemento.

Ratón
Pequeño dispositivo de entrada con forma de caja – mas bien de ratón – que al desplazarse sobre una superficie plana emite señales destinadas a mover un puntero o cursor en la pantalla del ordenador.

Recurso Compartido
Recurso del tipo de un documento, aplicación o medio de almacenamiento que está siendo utilizado de forma simultánea por un grupo de usuarios en una red de ordenadores.

Red
Conjunto formado por una serie ordenadores conectados físicamente entre sí y del software necesario para regular los términos de la conexión. Una red permite a sus usuarios compartir datos y material hardware.

Redes Coloreadas de Petri
Ampliación del uso práctico de las redes de Petri en ICH para representar el flujo de datos entre los distintos estados de una aplicación, permitiendo alcanzar una visión más completa del modelo de interacción al poder representarse el submodelo de procesos y el submodelo de datos a la vez.

Redes de Petri
Sistema de notación que permite representar una red de conceptos relacionados entre si. En ICH tienen un uso específico para representar el estado interno de una aplicación en términos de interacción, en donde los conceptos representados son los estados de la misma y sus transiciones.

Reflexión (*Reflection*)
Técnica mediante la cual es posible inspeccionar objetos y clases en tiempo de ejecución obteniendo un listado

pormenorizado de sus constructores, métodos y propiedades.

Resource Fork (Macintosh)
Uno de los dos componentes en los que se divide un fichero bajo el sistema operativo *Mac OS*. En esta parte del fichero se almacenan los recursos a los que se accede a través del *Resource Manager*.

RGB
Modelo de color basado en la adición de tres componentes básicas: rojo (*Red*), verde (*Green*) y Azul (*Blue*).

Remote Navigability Testing (RNT)
Ver *Pruebas Remotas de Navegación*.

RMI (*Remote Method Invocation*)
Mecanismo incluido en la plataforma Java que permite invocar métodos en objetos que residen en aplicaciones localizadas en ordenadores distintos de aquel en el que se realiza la invocación. Los objetos así creados funcionan de manera idéntica a los objetos creados localmente, encargándose RMI de hacer transparente el proceso de comunicación necesario para la invocación de métodos en estos objetos remotos. Para cada objeto remoto es necesario crear dos clases auxiliares (stub y skel) encargadas de la comunicación.

Ruido
Señal de origen indeterminado que se superpone a la señal útil generando parásitos.

S

Salida
Información transferida de un ordenador a algún tipo de dispositivo externo, como por ejemplo un monitor, un módem o una impresora.

Salthouse, Regularidades de
Conjunto de veintinueve patrones de comportamiento estándar publicados por Salthouse en 1986 tras el estudio de tareas básicas de tecleado en un máquina de escribir. Estas regularidades comprenden aspectos como la variación de la velocidad de tecleado con respecto a la velocidad de lectura, la variación de la velocidad de tecleado en base al significado de las palabras a teclear, etc.

Saturación
Medida de la cantidad de blanco contenida en un color. Mientras menos blanco posea el color, más saturado estará.

Scroll
Movimiento horizontal o vertical de la imagen en la pantalla de vídeo para dejar sitio a nuevos datos. Desplazamiento de un documento o directorio en una ventana de tal modo que sea posible ver una nueva parte del documento mientras se oculta otra.

Secuencia
Constituye toda aquella serie de acontecimientos que cohabitan consecutivamente dentro de una unidad narrativa.

Selección
Serie de caracteres o posición de un carácter donde tendrá lugar la próxima operación de edición. Los caracteres seleccionados suelen resaltarse.

Selección Discontinua
Selección que consiste en una serie de objetos que no son adyacentes entre sí.

Selección por Tecleado
Habilidad para seleccionar un ítem de una lista tecleando los caracteres con los que empiece el nombre de dicho ítem.

Seleccionar
Designar donde tendrá lugar la siguiente acción. Para seleccionar con el ratón es necesario pulsar sobre un icono o realizar una operación de arrastre sobre los objetos que representan la información seleccionada.

Semántica
Conjunto de reglas que especifican el significado de cualquier sentencia sintácticamente correcta y escrita en un determinado lenguaje.

Señal
Signo que sólo puede transmitir un significado.

Sensible al Contexto
Característica mediante la cual una interfaz percibe la situación en la que se produce un evento. Por ejemplo, cuando un programa de aplicación presenta información de ayuda relativa a la tarea que se encuentra realizando el usuario, en lugar de presentar una lista general de comandos.

Servlets
Marco de trabajo para el desarrollo de aplicaciones cliente-servidor en Java. Los *servlets* están integrados por varios paquetes de clases que proveen a estos objetos de funcionalidad diversa. Su aplicación más extendida es la codificación de servidores de páginas web dinámicos, integrándose en la mayoría de los servidores web disponibles en el mercado.

SGIU
Sistema de gestión de interfaces de usuario. Ver *UIMS*.

Shareware
Software distribuido bajo el régimen de probar antes de comprar. Puede ser distribuido gratuitamente pero ha de pagarse una prima al autor por su utilización.

Simplex
Método de transmisión de datos que permite una sola vía de comunicación.

Símbolos
Signo que, a diferencia de las señales, puede sugerir más de un significado

Sintaxis
Conjunto de reglas formales que especifican la composición de las unidades de comunicación de un lenguaje.

Sistema Operativo
Gestor y administrador de recursos hardware que sirve además de interfaz entre los usuarios y dichos recursos.

Sistema P
Analogía empleada por Broadbent circa 1955 para comparar el modelo cognitivo humano con la arquitectura interna de los primeros ordenadores. Este sistema puede considerarse como el origen de las teorías generales de la cognición y será ampliado en los años ochenta por medio del modelo de procesador humano (ver *MPH*).

Sistemas de Conversación
Ver *Chat*.

SOCCER
Metodología para la gerencia desarrollada por Checkland basada en los siguientes agentes que dan nombre a la metodología: *Solutions* (soluciones), *Owners* (dueños del problema), *Culture* (cultura de los analistas), *Customs* (costumbres de los analistas), *Expentancies* (expectativas o a donde aspira a llegar quien soluciona el problema) y *Resources* (recursos).

Sockets
Interfaz empleada en varios lenguajes de programación que permite comunicar una aplicación con otra sin necesidad de trabajar directamente con las rutinas de bajo nivel del protocolo TCP/IP.

Sonido de Alerta
Advertencia sonora empleada para notificar al usuario de la existencia de una situación inusual o potencialmente peligrosa para la integridad de los datos.

Speech Manager
Componente de la plataforma Java que permite emular la voz humana por parte de un ordenador. Este componente está estructurado en módulos correspondientes a distintas entonaciones y distintos idiomas y ha de seguir las especificaciones de la API definida por Sun Microsystems. Actualmente existen varios desarrollos distintos a partir de esta API.

SUIMS (Simple User Interface Management System)
Módulo de UIDE (ver *UIDE*) encargado de la generación y evaluación de alternativas para todos los procesos de usuario identificados durante la fase de análisis.

Swing
Paquete de clases de Java no estándar definido *por Sun Microsystems para* ampliar las capacidades gráficas estándar de Java y su AWT (ver *AWT*). Este paquete convierte en componentes (*JavaBeans*) la mayor parte de las clases

de la AWT y proporciona un mayor grado de integración entre plataformas, definiendo diferentes vistas de una misma interfaz genérica.

T

Tecla de Comando
Tecla que cuando se presiona mientras se pulsa otra tecla da como resultado que se ejecute un comando (ver *Comando*).

Tecla de Espaciado
Tecla larga y sin etiqueta situada en la parte inferior de un teclado empleada para generar el espacio en blanco.

Teclas de Retardo
Opción de algunos interfaces con acceso universal que permite insertar un retardo que define el lapso de tiempo en el cual dos pulsaciones de teclas consecutivas se consideran como parte de una misma combinación de teclas.

Tecla *Return*
Tecla que ocasiona que el cursor o el punto de inserción se mueva al comienzo de la siguiente línea. También suelen, emplearse para confirmar un comando o para cerrar un cuadro de dialogo o cuadro de alerta.

Tecla *Intro*
Esta tecla notifica a la aplicación que el usuario acaba de introducir información en un área particular de un documento. El usuario puede emplear esta tecla para activar la acción por defecto de un cuadro de diálogo o de un cuadro de alerta.

Tecla Mayúsculas
Tecla que cuando se mantiene pulsada causa que toda letra tecleada a continuación sea representada en mayúsculas. Esta tecla también puede cambiar el comportamiento de acciones invocadas con el ratón.

Teclas Escape
Teclas que permiten al usuario salir de un entorno de interacción rápidamente. En algunas aplicación, esta tecla permite detener una operación en progreso. Esta teclas se suele emplear también para cancelar un cuadro de diálogo o un cuadro de alerta.

Teclas de Modificación
Teclas que cambian el comportamiento de una acción cuando son presionadas en el mismo momento en el que se invoca la acción.

Teclas de Opción
Teclas de modificación que proporcionan un significado o acción distinto a otra tecla o a una acción del ratón.

Teclas de Ratón
Mecanismo de acceso universal en el que los usuarios emplean un teclado numérico para controlar el puntero del ratón.

Teoría de la Selección Múltiple
Teoría para explicar el procesamiento preventivo de señales físicas y señales semánticas por parte de la atención. En lugar de asignar prioridad de procesamiento a los estímulos en función de la naturaleza de la señal recibida, esta teoría propone que la asignación se realiza tanto en función de las características de la señal como en función de la situación.

Teoría de la Selección Tardía (*Late Selection Theory*)
Teoría que propone un procesamiento anticipado de las señales semánticas recibidas por un sujeto, dándoles prioridad sobre el procesamiento de las señales físicas.

Teoría de la Selección Temprana (*Early Selection Theory)*
Teoría propuesta por Broadbent para explicar el procesamiento de estímulos, en la que seda mayor prioridad al procesamiento de las señales físicas que al procesamiento de las señales semánticas

Teoría General de Sistemas (TGS)
Teoría desarrollada inicialmente por Von Bertalanffy en 1968 y que plantea formalmente el estudio y modelado de sistemas basados en una serie de componentes y sus relaciones.

Teoría Unificada de la Cognición (TUC)
Teoría planteada por Newell en 1991 a partir del trabajo de Card, Moran y el suyo propio en la que se estructura el modelo cognitivo humano en varios niveles de abstracción, los cuales trabajan a distintas velocidades y con distintos sistemas de adquisición y procesamiento de datos, entre los que se puede citar el sistema motriz y el perceptivo.

THINK C
Dialecto del C++ totalmente compatible con ese lenguaje diseñado por *THINK Technologies*.

Tirsus
Programa de investigación dirigido por Martín González Rodríguez, el cual está centrado en el análisis de distintos modelos de navegación para bases de conocimiento centradas en la Historia. La naturaleza dinámica e inestable de la Historia la hacen ideal como base de conocimiento experimental ya que es posible generar múltiples vistas de un mismo acontecimiento histórico, todas ellas provistas de modelos de navegación diferentes.

Token
(1) Abreviatura en inglés para referirse a una cadena de caracteres. (2) Secuencia de caracteres delimitada de tal forma que pueda ser reconocida por un compilador.

***Toolbox* (Macintosh)**
Conjunto de tipos abstractos de datos empleados por la plataforma Macintosh para la definición de los *widgets* que forman parte de su estilo de interacción WIMP (ver *WIMP*).

Toolkits
Bibliotecas de procedimientos, rutinas y programas diseñados especialmente para crear y ejecutar los diversos objetos constitutivos de una interfaz de usuario, tales como ventanas, botones, campos de texto, menús, etc.

Track-Ball
Dispositivo de entrada constituido por una esfera giratoria la cual obtiene el desplazamiento correspondiente al cursor.

Traductor
Programa que procesa un texto fuente y genera un texto objeto. El texto fuente está escrito en un código fuente que transforma el traductor en un código objeto.

TUC
Ver *Teoría Unificada de la Cognición*.

U

UIDE (*User Interface Design Environment*)
Sistema basado en el conocimiento experto para asistir al humano en tareas de diseño, evaluación y construcción de interfaces. Este sistema proporciona herramientas de alto nivel de abstracción para el diseño conceptual, facilitando el proceso de diseño interactivo de la especificación, generación y evaluación de interfaces. Este sistema fue desarrollado por Foley, Kim, Kovacevic y Murray y fue denominado posteriormente como sistema HUMANOID (ver *HUMANOID*).

UIDL (*User Interface Description Language*)
Lenguaje de descripción de interfaces de usuario empleado normalmente en la capa de más alto nivel de un UIMS. Sirve para especificar los requisitos de una interfaz pero rara vez para implementarla.

UIMS (*User Interface Management System*)
Entorno de programación para el desarrollo de interfaces de usuario que oculta la complejidad inherente al uso de toolkits (ver Toolkits), facilitando así la tarea de desarrollo y reduciendo con ello el tiempo de programación. Estos entornos ayudan a mantener la consistencia de la interfaz entre aplicaciones, facilitando además el mantenimiento y modificación de las interfaces.

UML(*Unified Modeling Language*)
Lenguaje de modelado de objetos sucesor de las notaciones de los métodos de Booch, OMT (ver *OMT*) y *OOSE (Object Oriented Software Engineering*) a las que integra. Pretende ser una notación estándar para el

modelado de las aplicaciones construidas mediante objetos.

URL (*Uniform Resource Locator*)
Patrón universal para la identificación de equipos a nivel de aplicación en una red de ordenadores. Este patrón da soporte a numerosos protocolos de comunicación (como *http, ftp, telnet* y *gopher*) y es lo suficientemente flexible como para garantizar la incorporación de nuevos protocolos.

User Application Model
Ver *Modelo de Usuario por Aplicación*.

Usuario
Persona que opera un artefacto. Dicho artefacto puede ser de naturaleza mecánica o software.

Utilidad
Tipo especial de software que sirve de ayuda para gestionar determinado tipo de aplicaciones.

V

Valor
Información que puede ser almacenada en una variable, como por ejemplo un número o una cadena de caracteres.

Variabilidad
Grado de imposibilidad de predicción de un parámetro.

Vector de Datos
Combinación secuencial de objetos de información, objetos de interacción o combinaciones de ellos, conteniendo texto o imágenes a través de los cuales puede navegar el usuario por medio del uso del teclado o del ratón.

Ventana
Objeto de la metáfora del escritorio que sirve para representar información acerca de otro objeto abstracto, como por ejemplo un documento o un mensaje.

Visual Age
Línea de entornos de desarrollo integrados (ver *IDE*) basados en el paradigma de la programación visual construidos por de IBM para dar soporte a las plataformas Java y *SmallTalk*.

W

Weltanschaung
Visión del mundo.

Widget
Objeto visual que permite al usuario actuar sobre una interfaz. Ejemplos de *widget* son los botones, las ventanas, los cuadros de diálogo, las etiquetas, los menús, etc.

WIMP (Windows, Icon, Menu and Pointer)
Estilo de interacción basado en el uso de ventanas en donde se realiza el intercambio de información entre el usuario y la aplicación. Esta información suele estar representada por iconos y el conjunto de acciones disponibles se presentan en menús a los que se accede por medio de un puntero desplazable por la pantalla del ordenador.

***Wizard* (Mago)**
Patrón de diseño de mecanismos de interacción que consiste en una sucesión continua de diálogos interactivos de baja carga cognitiva. Esta técnica está indicada para su empleo en la ejecución de procesos de usuario largos en los que se guía al usuario instándole a tomar decisiones a lo largo de dichos procesos.

Workspace
Metáfora de interacción para IDE (ver *IDE*) en la cual el código se almacena libremente si estar ligado a un proyecto de desarrollo en concreto.

X

XML (*Xtended Markup Language*)
Metalenguaje para la descripción de otros lenguajes basados en etiquetas. Permite dotar de estructura lógica a los contenidos de un documento de texto basado en etiquetas, de tal modo que éstos contenidos puedan ser tratados como datos. Esta característica lo hace ideal para el intercambio de información entre aplicaciones.

X-CHART
UIMS desarrollado en la Universidad Estatal de Campiñas (Brasil), basado en la definición de modelos de interacción por medio diagramas de transición de estados. Este UIMS permite un intercambio continuo y fluido de información entre la interfaz del usuario y la aplicación, por medio de la creación de clientes de interfaz asociados a objetos incluidos en las aplicaciones. En esta estrategia, los clientes atienden a las peticiones de información del usuario, transformando dichas peticiones en llamadas a funciones de los objetos interactivos.

Z

Zoom
Capacidad de presentar parte de una colección de datos en una escala ampliada, tanto a un nivel físico como a un nivel semántico.

27

27 Índice Analítico

Aunque solo existiera una verdad única, no se podría pintar cien cuadros sobre el mismo tema

Pablo Picasso

27.1 Índice

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

R

S

T

U

V

W

X

Y

Z

MARTIN ASTUR

TRANS AN XXIX

CIVITAS OVETO

FECIT KAL

MCMLXIII HISP

V S L M