UN SISTEMA OPERATIVO PARA EL SISTEMA ORIENTADO A OBJETOS INTEGRAL OVIEDO3

D. Álvarez Gutiérrez[†], L. Tajes Martínez[‡], F. Álvarez García[‡], M.A. Díaz Fondón[‡], J.M. Cueva Lovelle[†], R. Izquierdo Castanedo[†]

[†]Área de Lenguajes y Sistemas Informáticos
[‡]Área de Ciencias de la Computación e Inteligencia Artificial
Departamento de Matemáticas, Universidad de Oviedo
c/ Calvo Sotelo, s/n, 33007 OVIEDO - ESPAÑA
{darioa, tajes, fag, fondon, cueva}@pinon.ccu.uniovi.es

Resumen

Este trabajo describe los principales aspectos del sistema operativo para el sistema Oviedo3. Se ha concebido como un sistema operativo basado totalmente en el paradigma de orientación a objetos y para dar soporte al mismo. Este sistema operativo se construye sobre una máquina abstracta orientada a objetos, independiente de la plataforma subyacente, que proporciona el modelo básico de objetos que se utiliza en todo el sistema. Los cuatro elementos básicos que caracterizan el sistema operativo son la seguridad, la persistencia, la distribución y la concurrencia. La seguridad se basa en el envío de mensajes utilizando capacidades como mecanismo de protección. La persistencia ortogonal completa ofrece al usuario un espacio infinito donde existen indefinidamente los objetos hasta que ya no son necesitados. La distribución permite que los objetos residan en cualquier máquina del sistema de manera totalmente transparente. La concurrencia permite explotar el paralelismo tanto a nivel interno como entre objetos diferentes.

1. Introducción

La amplia extensión de las tecnologías orientadas a objetos en el diseño de aplicaciones da paso a la necesidad de sustituir el soporte de diseño clásico por otro nuevo que facilite el uso y la implantación de dichas tecnologías. La mejor manera de lograr esto es utilizar un sistema que esté concebido usando de manera integral en todos sus elementos el mismo paradigma que las propias aplicaciones: la orientación a objetos.

El sistema integral Oviedo3[1] está diseñado con una arquitectura orientada a Objetos que proporciona un soporte adecuado para estas tecnologías. Desde las interfaces del usuario hasta la propia máquina utilizan los mismos términos que las aplicaciones orientadas a objetos, pasando por el sistema operativo, base de datos, lenguajes y compiladores, etc..

[1] Oviedo3 está actualmente en fase de desarrollo por un equipo de profesores y alumnos de la Universidad de Oviedo, encontrándose muy adelantada la primera versión operativa de la máquina abstracta que dará soporte al resto del sistema.

Los fines de Oviedo3 pretenden ser tanto docentes como investigadores, de tal forma que dé soporte a prácticas de asignaturas, elaboración de Proyectos Fin de Carrera y Tesis Doctorales.

El sistema estará constituido por todos los subsistemas necesarios para permitir el desarrollo de aplicaciones. La figura 1 refleja esta situación:

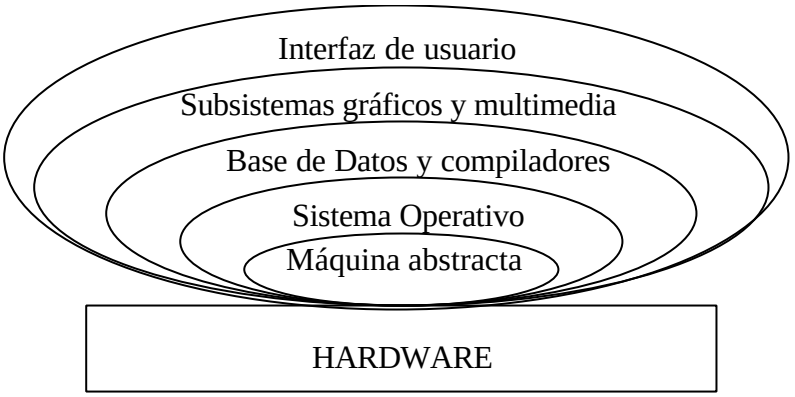

Figura 1. Componentes del sistema Oviedo3.

Una de las características básicas de este sistema es que está basado en una máquina abstracta. Esto permite la portabilidad del mismo a diferentes plataformas de hardware de manera directa. Todos los elementos del sistema, desde el SO hasta las aplicaciones desarrolladas en él podrán funcionar sin modificaciones en cualquier plataforma hardware.

El núcleo central del sistema Oviedo3 es el sistema operativo. De cara al exterior, su misión es proporcionar un entorno de computación que usa únicamente el paradigma orientado a objetos. Este entorno proporciona un espacio ilimitado en el que los objetos se pueden crear permaneciendo hasta que son destruidos, pudiendo enviar mensajes a otros objetos (cuya localización física es indiferente) durante su existencia para desarrollar las tareas de manera concurrente y controlada mediante un sistema de seguridad.

Internamente, la característica fundamental de este sistema operativo, que lo distingue de los demás, es que se construye sobre una máquina abstracta orientada a objetos. Esta máquina proporciona un soporte básico para los mismos. En el modelo de objetos del sistema no existe el concepto de dirección de memoria ni de espacio de direcciones: la única entidad que existe es el objeto, con un identificador único. Además, encapsulan su propia computación (objetos activos). No se hace distinción entre objetos de usuario y del sistema.

En el siguiente apartado, se resume la arquitectura de la máquina abstracta [2], cuyo diseño definitivo estará condicionado a las necesidades de implantación de las características del sistema operativo: seguridad [3], persistencia [4], distribución [5] y concurrencia [6].

2. La máquina abstracta Carbayonia

En esta arquitectura se soporta un modelo de objetos cuyas características fundamentales son las siguientes:

- ? Abstracción e identidad de los objetos
- ? Encapsulación
- ? Herencia
- ? Polimorfismo o paso de mensajes

En la figura 2 se muestra la arquitectura de la máquina abstracta Carbayonia [2], que está compuesta por cuatro áreas: Área de clases, de referencias, de instancias y referencias del sistema.

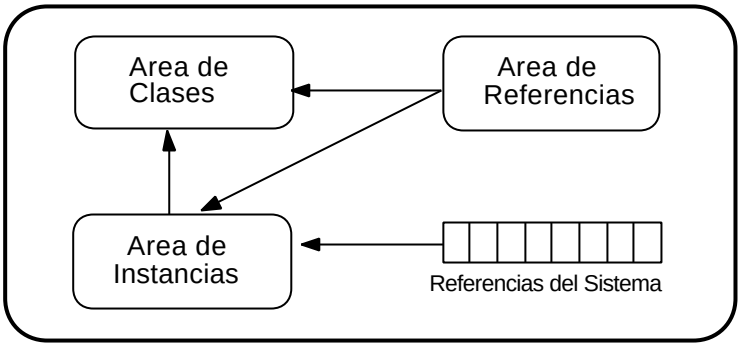

Figura 2. Arquitectura de la máquina Carbayonia.

Cada área se puede considerar como un objeto que se encarga de la gestión de sus datos.

Su principal característica es que toda acción sobre un objeto se realiza a través de una referencia al mismo.

? **Área de clases**. Guarda la descripción de cada clase.
? **Área de referencias**. Almacena las referencias a los objetos. Cada una tiene un tipo (se relaciona con el área de clases) y apunta a un objeto de ese tipo (se relaciona con el área de instancias)
? **Área de instancias**. Cuando se crea un objeto se almacena en este área y cuando se destruye se elimina de aquí. Se relaciona con el área de clases de manera que cada objeto pueda acceder a la información de la clase a la que pertenece.
? **Área de referencias del sistema**. Que tienen funciones específicas y existen de manera permanente en el sistema.

Definición del lenguaje de la máquina abstracta

El lenguaje máquina de la arquitectura se basa en la utilización exclusiva de objetos. Es un lenguaje orientado a objetos puro de bajo nivel que será el lenguaje intermedio de la máquina abstracta. Permite declarar clases, definir métodos y manejar excepciones. Presenta una serie de clases primitivas definidas de forma permanente en el área de clases.

3. Seguridad para el SOOO de Oviedo3

En un SOOO, la entidad a proteger es el objeto, más concretamente se trata de controlar quién tiene acceso a las operaciones del objeto. Es decir qué objetos pueden acceder a qué operaciones de un objeto determinado.

Protección

El sistema Oviedo3, se propone integrar la gestión de la protección a bajo nivel, dentro de la máquina abstracta, lo que mejoraría considerablemente el rendimiento con respecto a una gestión a nivel de sistema operativo, sobre todo teniendo en cuenta que el número de invocaciones a operaciones es muy grande.

Las dos ideas básicas propuestas en Oviedo3 para la protección de objetos son las siguientes:

? **Utilización de capacidades como referencias a los objetos** [8]. La utilización de capacidades en un sistema operativo orientado a objetos puede integrarse de forma sencilla en el modelo, haciendo que el mecanismo de protección no altere el modelo de objetos que adopte el sistema. La capacidad puede enviarse dentro del propio mensaje de petición de la operación, por tanto no supone apenas adición de carga. No se han elegido otros mecanismos de protección como listas de control de acceso, o mecanismos mixtos, debido a que esto implica una alteración no deseada en el modelo de objetos, al introducir información de protección dentro del estado de un objeto, y delegar en él la comprobación de la protección. Sin embargo, el uso de capacidades conlleva un inconveniente diferente que es necesario solucionar: la revocación de capacidades.
? **Integración del mecanismo de protección en el mecanismo de envío de mensajes del sistema operativo**, haciendo que el mensaje únicamente llegue a su destino si el mecanismo de protección comprueba que la capacidad es correcta (figura 3). De esta manera, conceptualmente, el mecanismo de protección mediante capacidades se realizaría como parte integral del mecanismo de envío de mensajes entre objetos (invocación de operaciones). Por tanto, la seguridad se implanta de manera uniforme ya desde el corazón del sistema operativo.

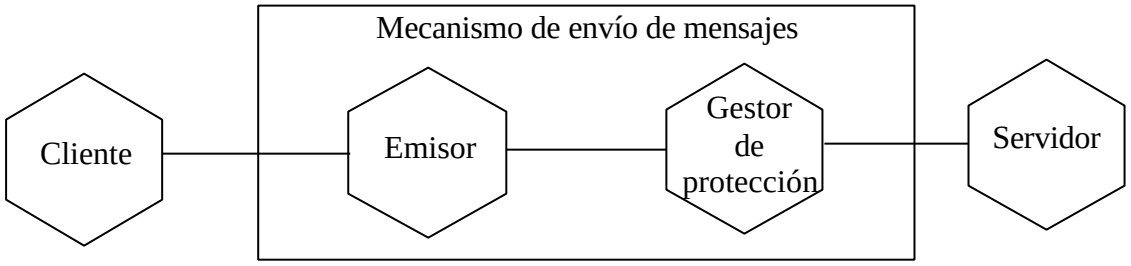

Figura 3. Integración de la protección en el mecanismo de envío de mensajes.

Implantación

Oviedo3 dispone de la máquina orientada a objetos Carbayonia, que está basada en la existencia de objetos (con un identificador único) y en el uso de referencias a objetos. Estas referencias almacenan los identificadores únicos de objetos.

1- Adición de atributos en las referencias

Para implantar la protección se propone ampliar la arquitectura de la máquina para que las referencias pasen a ser capacidades (figura 4). Es decir, ampliar el contenido de una referencia con la información de protección.

Figura 4. Adición de información de protección en las referencias.

2- Adición y modificación de operaciones con referencias

Además de las operaciones actuales con referencias: crear y eliminar los objetos a los que apuntan, deben añadirse operaciones adicionales que tengan en cuenta la nueva categoría de las

referencias como capacidades. En principio pueden proponerse las siguientes, algunas cambian su semántica, otras son nuevas: Crear, Eliminar, Copiar, Crear Capacidad Restringida.

3- Modificación del mecanismo de envío de mensajes

El mecanismo de envío de mensajes de la máquina se basa en tomar una referencia, localizar el objeto e invocar la operación deseada. Ahora, tras localizar el objeto mediante el identificador del mismo, se debe examinar la información de protección y si se tienen permisos para ello, invocar la operación deseada. En caso que de no se tengan los permisos necesarios, se eleva una excepción.

Ventajas de este diseño

Protección automática de las capacidades

Las capacidades están segregadas, ya que al estar incorporadas como parte de las referencias que usa la máquina abstracta, no pueden ser manipuladas por el usuario más que con las operaciones de la máquina que están definidas al efecto. La máquina garantiza que no pueden modificarse arbitrariamente. Al asegurar la imposibilidad de falsificación o modificación no deseada de las capacidades por "hardware" evitamos el coste adicional que suponen las técnicas de capacidades dispersas.

Sencillez y facilidad de uso

Por otro lado, se pueden utilizar directamente dentro de estructuras del usuario ya que constituyen precisamente la referencia que utilizan los usuarios en sus estructuras para acceder a los objetos de manera normal. Al permitir la utilización como referencias normales en estructuras de usuario, se facilita el uso del sistema al utilizar un único modo de trabajo con los objetos, sin la dualidad de uso que supone el tener que acceder de una manera especial al área segregada donde están las capacidades.

4. Persistencia para el SOOO de Oviedo3

Se elabora a continuación una propuesta preliminar para la integración de la persistencia en el sistema operativo orientado a objetos de Oviedo3.

Idea básica

En esencia, la idea básica para la integración de la persistencia ortogonal completa [8] en el sistema operativo de Oviedo3 es proporcionar la abstracción de un espacio de objetos potencialmente infinito en el que coexisten simultánea e indefinidamente todos los objetos del sistema, hasta que ya no son necesitados. Podría considerarse como una memoria virtual persistente para objetos. El usuario simplemente trabaja con los objetos en ese espacio virtual.

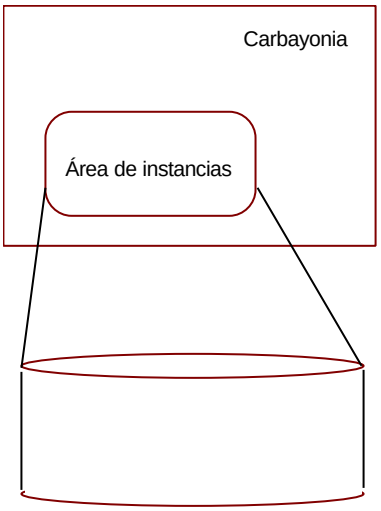

Figura 5. Área de instancias virtual

El sistema de persistencia deberá proporcionar con los recursos existentes en el entorno Oviedo3 (fundamentalmente la máquina abstracta orientada a objetos Carbayonia) la abstracción anterior. Se puede implementar la persistencia como una extensión del área de instancias (figura 5), es decir lograr la ilusión de un área de instancias virtual (memoria virtual), utilizando para ello un almacenamiento secundario para hacer persistir los objetos, guardándolos

cuando no estén (o no quepan) en el área de instancias. Todo ello de manera totalmente transparente para el resto del sistema.

Las características propuestas para el sistema de persistencia son las siguientes:

? **Persistencia ortogonal completa**. Absolutamente todos los objetos creados en el sistema son persistentes por definición. Esto incluye además a los objetos del sistema operativo, que deben tener la misma categoría que los objetos de usuario. No es necesario almacenar y recuperar explícitamente los objetos, el sistema lo realiza de manera transparente. Sólo existe un único conjunto de operaciones para manipular los objetos.
? **Estabilidad y resiliencia**. El espacio virtual persistente de objetos debe proporcionar las propiedades de estabilidad y resiliencia. Es decir, el sistema debe poder reanudar el funcionamiento con seguridad después de una caída inesperada, como por ejemplo un fallo de alimentación.
? **Encapsulamiento de la computación**. Se propone un modelo de objetos que encapsule además, como parte de su estado, la computación. Puede hacerse que la máquina soporte directamente este modelo de objetos. Esto no dificulta la persistencia, ya que la computación estará contenida totalmente dentro del objeto.
? **Identificador uniforme único**. Se utiliza un único identificador para referenciar los objetos, independientemente de su localización física. Este identificador es usado tanto por el usuario como por el sistema internamente para referenciar el objeto en memoria principal y en almacenamiento persistente.

Ventajas

Además de las ventajas generales de los sistemas de persistencia, en concreto se pueden destacar las siguientes ventajas al utilizar un esquema como el que se ha propuesto:

? **Sistema y entorno continuo**. Al ser la persistencia ortogonal completa, con estabilidad y resiliencia, se obtiene un entorno continuo, al persistir los objetos del usuario. Además, al ser también objetos normales los objetos del propio SO, son a su vez persistentes. Se logra un sistema continuo. Se mantiene totalmente el estado del sistema entre interrupciones del mismo, tanto debidas al usuario (desconexión) como inesperadas (fallo de alimentación).
? **Memoria virtual distribuida**. Se crea fácilmente una verdadera memoria virtual distribuida de objetos al utilizar un identificador unívoco de objetos. Simplemente basta con utilizar un mecanismo que haga que este identificador sea diferente para todos los objetos, independientemente de la máquina en que se crearon. Se utilizará este identificador único en conjunción con el mecanismo de distribución para localizar en todo el sistema distribuido un objeto dado.
? **Eficiencia interna**. En principio debe esperarse una buena eficiencia interna al utilizar un identificador uniforme y no necesitar de la sobrecarga del mecanismo de conversión de punteros (*pointer swizzling*), en el paso entre memoria principal y almacenamiento persistente. Por otro lado al usarse como identificador el propio identificador *hardware* de la máquina, la eficiencia debe ser mayor al no introducir el paso intermedio debido a un identificador *software*.
? **Abstracción única de memoria**. Al eliminar la dualidad de la abstracción entre memoria de largo plazo y de corto plazo, y sustituirlas por una única abstracción de espacio virtual persistente de objetos se facilita la labor de los programadores. Abstracciones como fichero ya no son

necesarias (aunque podrían implementarse como objetos normales, si se desea). Simplemente se trabaja con un único paradigma, que es el de la orientación a objetos.

? **Interfaces más intuitivos**. Al poder existir un sistema y un entorno continuo, se puede diseñar un interface de usuario más intuitivo. Por ejemplo, interfaces del tipo "escritorio" se comportarían de una manera más intuitiva y cercana a su funcionamiento en el mundo real. Cuando se abandona el escritorio (desconexión), su estado permanece exactamente igual cuando se regresa a él (conexión), debido a esta persistencia. No requiere la existencia de conceptos poco intuitivos como fichero, configuraciones de arranque, etc.

5. Distribución para el SOOO de Oviedo3

Evidentemente Oviedo3 ha de ser un sistema operativo *distribuido* [9]. Oviedo3 se ejecuta sobre una arquitectura distribuida formada por una serie de máquinas Carbayonia que están unidas por una red de comunicaciones (ver figura 6). Todas las máquinas Carbayonia son iguales, lo cual no quiere decir que el hardware subyacente tenga que serlo.

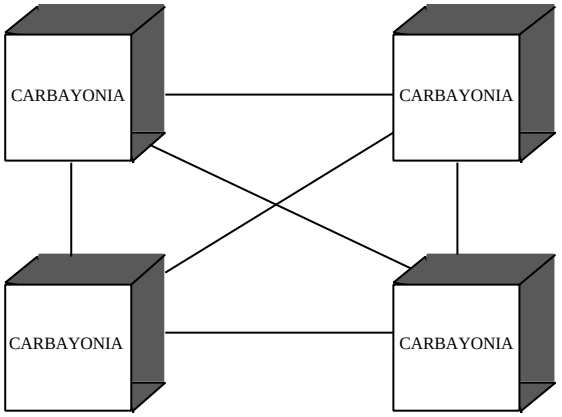

Figura 6. Arquitectura distribuida del sistema Oviedo3

Localización de objetos

Todos los objetos de Oviedo3 tendrán un identificador único. En el momento en que se cree un objeto, la máquina Carbayonia en la que residirá inicialmente el objeto habrá de encargarse de asignarle un identificador nuevo que utilizará durante toda su existencia. Una vez que un identificador ha sido asignado a un objeto, no podrá utilizarse más.

El esquema de localización de objetos de Oviedo3 es el de un servicio de denominación compuesto por múltiples servidores de nombres (identificadores, en nuestro caso), el cual va a permitir la movilidad de los objetos. Un servidor de nombres no va a ser otra cosa que un objeto especializado en mantener correspondencias entre identificadores de objetos y su localización dentro del sistema.

Una vez asignado un identificador a un objeto, deberá ser registrado en el servicio de denominación, de manera que quede constancia de su existencia y su localización inicial. Siempre que un objeto se mueva de una máquina a otra, el servicio de denominación habrá de ser advertido del cambio. Finalmente, si un objeto deja de existir en un momento dado, habrá que eliminar del servicio de denominación toda información relativa a su localización.

El servicio de denominación puede utilizar técnicas de replicación de las localizaciones de los objetos para aumentar la fiabilidad y la escalabilidad.

La utilización de identificadores únicos para los objetos del sistema va a permitir que se implementen políticas de nombrado de más alto nivel.

Comunicación entre objetos

Los objetos de Oviedo3 se comunicarán por paso de mensaje. Una vez localizado el objeto invocado, se construirá un mensaje con toda la información necesaria para que se pueda realizar la invocación: el conjunto de parámetros y la información de seguridad del objeto cliente. Finalizada la operación, los resultados se empaquetarán en un nuevo mensaje que se enviará de vuelta al objeto cliente.

Movilidad

Los objetos de Oviedo3 tienen la posibilidad de ver variada su localización. La política de migración de objetos estará basada en:

? Carga del procesador.
? Interacción con objetos remotos.

Esto quiere decir que en el caso de que se detecte una alta carga en un procesador o bien un alto grado de interactuación entre objetos locales y remotos, se lanzará un proceso de decisión en el que se estudiará qué objetos son susceptibles de ser movidos y con qué destino.

Para cada uno de los objetos a mover y su destino, se informará al servidor de nombres de que el objeto va a ser movido y se enviará a la máquina Carbayonia elegida. La información que se mueve es el estado encapsulado, que define totalmente el objeto. Este incluye también el estado de la computación dentro del objeto (que es autocontenido).

6. Concurrencia para el SOOO de Oviedo3

En nuestro sistema, donde las acciones son llevadas a cabo por objetos que pueden comportarse como servidores ante la petición de otros que asumen el papel de clientes, se intenta alcanzar la máxima concurrencia [10] garantizando siempre la corrección de las operaciones que se efectúen y con ello la consistencia en el estado del objeto. El sistema operativo debe establecer un control de la misma en cada uno de los siguientes niveles:

? entre objetos
? entre distintos métodos dentro de un objeto
? entre distintas instancias del mismo método de un objeto.

En el primer nivel, debemos permitir la coexistencia del máximo número de servidores y clientes posible según las necesidades de trabajo del momento.

No estableceremos diferencias en los dos niveles posteriores y trataremos de la misma forma las peticiones de activación de métodos dentro de un objeto, ya sean el mismo o distintos métodos.

La propuesta a continuación descrita se basa en un **modelo con un número variable de hilos activos dentro del objeto**. Este es el modelo más flexible ya que dentro de cada objeto los hilos son totalmente dinámicos y se activan en la invocación de un trabajo al objeto servidor por parte de un cliente (figura 7). Pero también el que presenta mayores problemas como mantener la consistencia del estado del objeto, sobrecarga del sistema, deadlocks, etc.

Definición de los métodos

Esta propuesta se fundamenta en la idea de que, normalmente, en la definición de un objeto se especifican dos tipos de métodos:

- los exclusivos, que modifican el estado del objeto y deben ser ejecutados sin interferencias
- los concurrentes, que pueden convivir sin problemas con otros métodos activos en el objeto ya que únicamente acceden al estado del objeto sin modificarlo.

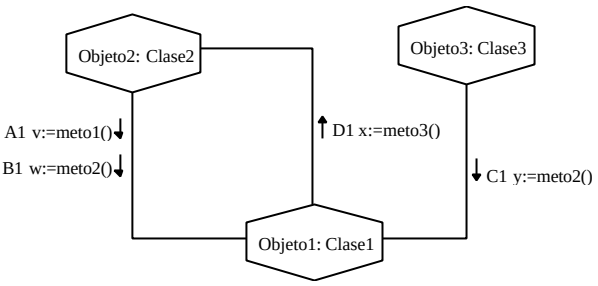

Figura 7. Llamadas concurrentes a un objeto

Invocación de métodos del objeto

La entrada al objeto se realizará siempre mediante un paso de mensaje. Las invocaciones de métodos son **síncronas**. Este hecho puede utilizarse además en la sincronización de objetos.

Se pueden dar dos tipos de mensajes: una **petición de trabajo** o una **respuesta**. En el primer caso, lo serviremos o almacenaremos dependiendo del estado del objeto. En el segundo caso, puede servirse siempre inmediatamente.

Activación de métodos en un objeto

La activación de métodos dentro de un objeto viene claramente condicionada por el tipo de método al que nos referimos.

- Si el método es exclusivo, la activación de un hilo se demorará hasta que no haya ningún hilo activo dentro del objeto. Es decir, funcionan como monitores.
- En el caso de un método concurrente, la restricción se refiere únicamente a la existencia de hilos activos ejecutando métodos exclusivos.

La activación de los hilos se hará previa evaluación de las **guardas** o **precondiciones** asociadas al método que establecen bajo qué circunstancias se permite su ejecución.

- Si la guarda se evalúa a cierto, se activará un hilo que sirva la petición.
- Si es falsa, la invocación permanecerá a la espera hasta que se den las condiciones necesarias.

Para ello, cada objeto debe registrar qué hilos están pendientes de finalizar y reúnen las condiciones necesarias para ello.

Desactivación de métodos en un objeto

Hay dos posibilidades para que un hilo que está activo ejecutando un método de un objeto deje de estarlo:

- Finalización del hilo: debe enviarse respuesta al hilo que invocó al método
- Bloqueo en una llamada, lo que lo llevaría al estado inactivo y registra el bloqueo.

En cualquier caso, se estudia la situación de todos aquellos hilos que están inactivos.

En la figura 8 se muestra, con la utilización de un diagrama de transición de estados, el comportamiento de un objeto ante la invocación de un método del mismo.

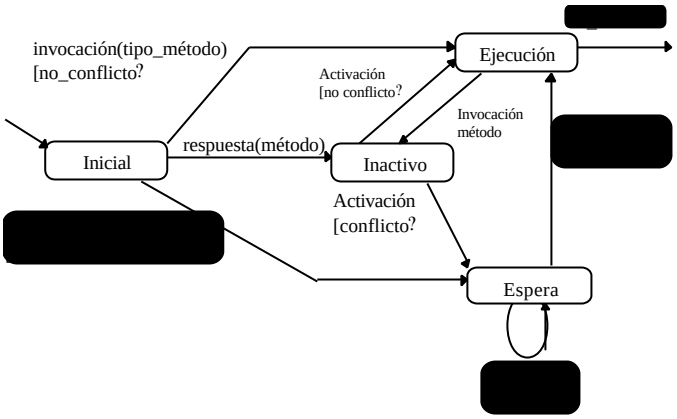

Figura 8. Comportamiento de un objeto ante la llegada de un mensaje

7. Conclusiones

El paradigma de la orientación a objetos no ha sido explotado hasta sus últimas consecuencias, especialmente en el campo de los sistemas operativos. Oviedo3 es un proyecto de investigación en la Universidad de Oviedo que pretende desarrollar un sistema integral de objetos, basado en una máquina abstracta orientada a objetos y un sistema operativo. En este proyecto el único paradigma que se utiliza en todos los apartados del mismo es la orientación a objetos.

Se han resaltado como más relevantes los aspectos de seguridad, persistencia, distribución y concurrencia a la hora de caracterizar el sistema operativo, siempre dentro del marco general del sistema Oviedo3. Las decisiones de diseño expuestas para cada uno de estos apartados permitirán ofrecer al usuario un entorno de computación totalmente basado en la filosofía de la orientación a objetos más flexible, coherente, intuitivo y fácil de utilizar, eliminando muchos de los problemas de los sistemas operativos que existen en la actualidad.

Actualmente se encuentra prácticamente finalizada la primera versión de la máquina abstracta (desarrollada sobre Windows NT) y se está comenzando el diseño detallado de los componentes del sistema operativo, para pasar a su implantación sobre la máquina abstracta.

8. Referencias

[1] Cueva Lovelle, J.M., “El sistema integral Orientado a Objetos: Oviedo3”, II Jornadas sobre Tecnologías Orientadas a Objetos, Oviedo, Marzo 1996.

[2] Izquierdo Castanedo, R., “La máquina abstracta Orientada a Objetos Carbayonia y su lenguaje Carbayón”, II Jornadas sobre Tecnologías Orientadas a Objetos, Oviedo, Marzo 1996.

[3] Díaz Fondón, M.A., “Seguridad y protección en un sistema operativo Orientado a Objetos”, II Jornadas sobre Tecnologías Orientadas a Objetos, Oviedo, Marzo 1996.

[4] Álvarez Gutiérrez, D., “Persistencia en un sistema operativo Orientado a Objetos”, II Jornadas sobre Tecnologías Orientadas a Objetos, Oviedo, Marzo 1996.

[5] Álvarez García, F., “Distribución en sistemas operativos Orientados a Objetos”, II Jornadas sobre Tecnologías Orientadas a Objetos, Oviedo, Marzo 1996.

[6] Tajes Martínez, L., “Introducción de la concurrencia en sistemas operativos Orientados a Objetos”, II Jornadas sobre Tecnologías Orientadas a Objetos, Oviedo, Marzo 1996.

[7? Dennis, J. B., and E. C. Van Horn, “Programming Semantics for Multiprogrammed Computations”, CACM (V9, N3, 1966).

[8] Dearle, A., J. Rosenberg, F. Henskens, F. Vaughan y K. Maciunas, "An Examination of Operating System Support for Persistent Object Systems", Proc. of the 25th Hawaii International Conference on System Sciences, Hawaii, U.S.A., pag. 779-789, 1992.

[9] Chin, R.S. y Chanson, S.T.. “Distributed Object-Based Programming Systems”, ACM Computing Surveys, Vol. 23, No. 1, Marzo 1991.

[10] Papathomas, M., “Concurrency Issues in Object-Oriented Programming Languages”, en Object-Oriented Development Technical Report, Centre Universitaire d´Informatique, University of Geneva, ed. Tsichritzis, D., 1989.