MDE: Ingeniería dirigida por modelos. Otra forma de construir software

Dr. Juan Manuel Cueva Lovelle
Dra. B. Cristina Pelayo García-Bustelo

Departamento de Informática

Universidad de Oviedo

Universidad Distrital Francisco José de Caldas
Bogotá, Noviembre de 2008

MDE
Contenidos

MDE
Contenidos

MDE
Contenidos

MDE
Contenidos

Contenidos

Contenidos

Resumen

- El Desarrollo de Software Dirigido por Modelos (DSDM) es una propuesta para la construcción de software en la que se le atribuye a los modelos el papel principal de todo el proceso, frente a las propuestas tradicionales basadas en lenguajes de programación, plataformas de objetos y componentes software.

- DSDM persigue elevar el nivel de abstracción en el desarrollo de software, convirtiendo a los modelos y a las transformaciones entre ellos en los principales artefactos de todas las fases del proceso de desarrollo de software: captura y gestión de los requisitos, diseño, análisis, implementación, despliegue, configuración, mantenimiento, evolución, etc. [BBC+06]

Resumen

- El éxito de esta iniciativa ha originado la evolución de distintos paradigmas englobados dentro del contexto del DSDM, como por ejemplo:
  - la iniciativa MDA (Model Driven Architecture) de la OMG
  - las técnicas de Agile Model-Driven Development (AMDD)
  - las propuestas de Domain Specific Modeling (DSM)
  - las estrategias de Domain-Oriented Programming (DOP)
  - las Software Factories (SF)
- Cada una de ellas aborda el proceso de DSDM de diferente forma, y con distintos mecanismos.

- En particular, la iniciativa MDA cubre un amplio espectro de áreas de investigación:
  - metamodelos basados en MOF
  - perfiles UML
  - transformaciones de modelos y definición de lenguajes de transformación (QVT)
  - construcción de modelos PIM y PSM y transformaciones de PIM a PSM
  - construcción de herramientas de soporte
  - aplicación en métodos de desarrollo y en dominios específicos
- Algunos de estos aspectos están bien fundamentados y se están empezando a aplicar, otros sin embargo están todavía en fase inicial

- MDE es el acrónimo de Model Driven Engineering (Ingeniería Dirigida por Modelos) y hace referencia al uso sistemático de modelos, como los elementos principales en la ingeniería de software, durante el ciclo de vida completo del proyecto [Sch06]
- Concibe al desarrollo de software bajo la idea central de que los artefactos fundamentales son los modelos (y no los programas)
- Implica la generación automática de programas a partir de modelos. Model Driven Engineering (MDE) es una forma genérica de definir el DSDM y los MDA

Resumen

- El principal objetivo del MDE es bajar el coste y mejorar la calidad de las inversiones en software.
- Las soluciones que aportan este tipo de modelos son:
  - Separar los aspectos del dominio de los aspectos de la tecnología
  - El conocimiento queda registrado en los modelos y las transformaciones y puede ser rehusado
  - Se automatizan partes significantes del proceso
  - Implementación de componentes usables por otras partes

Contenidos

Planteamiento del problema

En la sociedad actual:

- Demanda del Software ⇑ y el coste ⇑
- Se intenta aumentar la productividad:
  - ⇑ nivel de abstracción
  - ⇑ nivel de reutilización

Planteamiento del problema

En la sociedad actual:

- Demanda del Software ⇑ y el coste ⇑
- Se intenta aumentar la productividad:
  - ⇑ nivel de abstracción
  - ⇑ nivel de reutilización

Premisa

¿Cómo aumentar la productividad en la construcción de software a través del aumento de la abstracción y el aumento del nivel de reutilización?

La iniciativa Model Driven Architecture (MDA)

Es una iniciativa del Object Management Group (OMG)

La iniciativa Model Driven Architecture (MDA)

Es una iniciativa del Object Management Group (OMG)

Planteamiento de los MDA's

El proceso de crear software debería ser dirigido por la formulación de MODELOS en lugar de por la escritura manual de código fuente.

Con ello se conseguiría: $\Uparrow$ nivel de abstracción + $\Uparrow$ nivel de reutilización + Interoperabilidad

La iniciativa Model Driven Architecture (MDA)

Los modelos de MDA [RFW+04] representaran todos los aspectos del programa final ejecutable, por tanto deben ser:

- formales
- precisos
- semántica bien definida

Deben conseguir un nivel de abstracción que permita su realización en múltiples plataformas [MRM03]

La iniciativa Model Driven Architecture (MDA)

Los modelos de MDA [RFW+04] representaran todos los aspectos del programa final ejecutable, por tanto deben ser:

- formales
- precisos
- semántica bien definida

Deben conseguir un nivel de abstracción que permita su realización en múltiples plataformas [MRM03]

Transformación [KWB03]

El proceso de transformar una especificación software en un programa ejecutable será automático, por tanto el código fuente de las aplicaciones se debería generar a partir de los modelos en un proceso de transformación

Desarrollo ágil de software

Define nuevas metodologías para afrontar el desarrollo del software de una forma más eficiente, y por tanto menos costosa [Lar03]

Los métodos ligeros asumen el cambio como algo inevitable y están centrados en el código, que será la principal documentación del proyecto.

Los métodos ágiles suponen procesos y relaciones con el cliente y su mantenimiento, no código.

Desarrollo ágil de software

Define nuevas metodologías para afrontar el desarrollo del software de una forma más eficiente, y por tanto menos costosa [Lar03]

Los métodos ligeros asumen el cambio como algo inevitable y están centrados en el código, que será la principal documentación del proyecto.

Los métodos ágiles suponen procesos y relaciones con el cliente y su mantenimiento, no código.

Modelos ejecutables [Amb04]

Si asumimos que un modelo ejecutable es código, se pueden aplicar los principios ágiles a la construcción de modelos ejecutables

Planteamiento del problema

La propuesta MDA de OMG tiene dos importantes carencias relacionadas con el tratamiento de los modelos iniciales:

- No se aclara cómo deben manejarse los modelos ligados al negocio, lo que se conoce como modelos independientes de la computación ó CIM.
- No se describe cómo deben asociarse esos modelos CIM con los primeros modelos del software a desarrollar, es decir, con los PIM (modelos independientes de plataforma).

Las metodologías de desarrollo ágil de software tiene como punto fuerte la conexión de los aspectos asociados al negocio con los tecnológicos. Sin embargo, no se ajusta a la especificación MDA.

Hipótesis

La combinación de la especificación MDA con la idea del desarrollo ágil resultaría muy ventajosa de cara a un desarrollo de software flexible y asociado a los procesos del negocio.

Objetivo Principal

Definir una metodología para el desarrollo de software que permita la creación de software a partir del modelado de procesos del negocio de acuerdo a la especificación MDA y el desarrollo de software ágil.

Objetivo Principal

Definir una metodología para el desarrollo de software que permita la creación de software a partir del modelado de procesos del negocio de acuerdo a la especificación MDA y el desarrollo de software ágil.

Centrado en las etapas iniciales del proceso de desarrollo, aquellas relacionadas con los modelos CIM y PIM de MDA.

Objetivos Parciales

- **Objetivo Parcial 1:**
  Realizar un estudio de las áreas relacionadas con el objetivo principal, es decir, MDA, desarrollo ágil de software, patrones de diseño y estándares.

Objetivos Parciales

- **Objetivo Parcial 1:**
  Realizar un estudio de las áreas relacionadas con el objetivo principal, es decir, MDA, desarrollo ágil de software, patrones de diseño y estándares.
- **Objetivo Parcial 2:**
  Definir una metodología para el desarrollo de software ágil de acuerdo a la especificación MDA.

Objetivos Parciales

- **Objetivo Parcial 1:**
  Realizar un estudio de las áreas relacionadas con el objetivo principal, es decir, MDA, desarrollo ágil de software, patrones de diseño y estándares.
- **Objetivo Parcial 2:**
  Definir una metodología para el desarrollo de software ágil de acuerdo a la especificación MDA.
- **Objetivo Parcial 3:**
  Crear un prototipo que implemente la metodología.

Objetivos Parciales

- **Objetivo Parcial 1:**
  Realizar un estudio de las áreas relacionadas con el objetivo principal, es decir, MDA, desarrollo ágil de software, patrones de diseño y estándares.
- **Objetivo Parcial 2:**
  Definir una metodología para el desarrollo de software ágil de acuerdo a la especificación MDA.
- **Objetivo Parcial 3:**
  Crear un prototipo que implemente la metodología.
- **Objetivo Parcial 4:**
  Desarrollar una aplicación usando la metodología y utilizando el prototipo.

MDE
Contenidos

Contenidos

Software Ágil

- Métodos *ligeros* de desarrollo
- Desarrollado por Kent Beck [BEC99a]
- Proceso de desarrollo de Software es impredecible
  - Debe ser FLEXIBLE
  - Caso de Estudio: Extreme Programming

Software Ágil

- Métodos *ligeros* de desarrollo
- Desarrollado por Kent Beck [BEC99a]
- Proceso de desarrollo de Software es impredecible
  - Debe ser FLEXIBLE
  - Caso de Estudio: Extreme Programming

MDA ágiles

Código = los modelos ejecutables
Un modelo ejecutable puede ser construido, ejecutado, probado y modificado en ciclos cortos e incrementales.

Principios del Software Ágil (1/2)

- Los métodos *ligeros* están centrados en el código
- Solución de compromiso entre la ausencia de proceso y el exceso de proceso

Diferencias entre métodos pesados y ágiles según Fowler [FOW95]

Adaptación al Cambio:Los métodos ágiles son adaptables, los métodos pesados que planifican el proceso de desarrollo sólo funciona mientras las cosas no cambian.
Aceptación a la naturaleza de las personas: Centrados en las personas no en el proceso.

Principios del Software Ágil (2/2)

Manifiesto para el Desarrollo de Software Ágil [BEC01]

Se prefiere	**frente a**
las personas y las relaciones	los procesos y herramientas
el software que funciona	la documentación
la colaboración del cliente	los contratos
responder a los cambios	seguir un plan

- Procesos y relaciones con el cliente y su mantenimiento, no código.

Métodos del Software Ágil

- Extreme Programming (XP)
- Scrum
- Adaptive Software Development (ASD)
- Crystal Clear y otras metodologías de la familia Crystal
- DSDM
- Feature Driven Development
- Lean software development

Las 12 Prácticas de Extreme Programming (XP)

1. Planificación (historias, iteracciones y tareas)
2. Versiones Cortas
3. Metáfora
4. Diseño Simple
5. Pruebas continuas
6. Factorización
7. Programación en parejas
8. Propiedad colectiva del código
9. Integración continua
10. Trabajo de 40 horas semanales
11. Cliente en el sitio
12. Estándares de codificación

Contenidos

MDE
Model Driven Architecture (MDA)

- Arquitectura formulada por el Object Management Group (OMG)
- Proporciona un enfoque para:
  - Especificar un sistema de manera independiente a la plataforma de soporte
  - Especificar plataformas
  - Elegir una plataforma particular para el sistema
  - Transformar la especificación del sistema para una plataforma en concreto.

MDE
Model Driven Architecture (MDA)

- Arquitectura formulada por el Object Management Group (OMG)
- Proporciona un enfoque para:
  - Especificar un sistema de manera independiente a la plataforma de soporte
  - Especificar plataformas
  - Elegir una plataforma particular para el sistema
  - Transformar la especificación del sistema para una plataforma en concreto.

Objetivos [PGBCLST05]

Separación arquitectónica de conceptos para:

- Portabilidad
- Interoperatibilidad
- Reusabilidad

Conceptos Básicos (1/4)

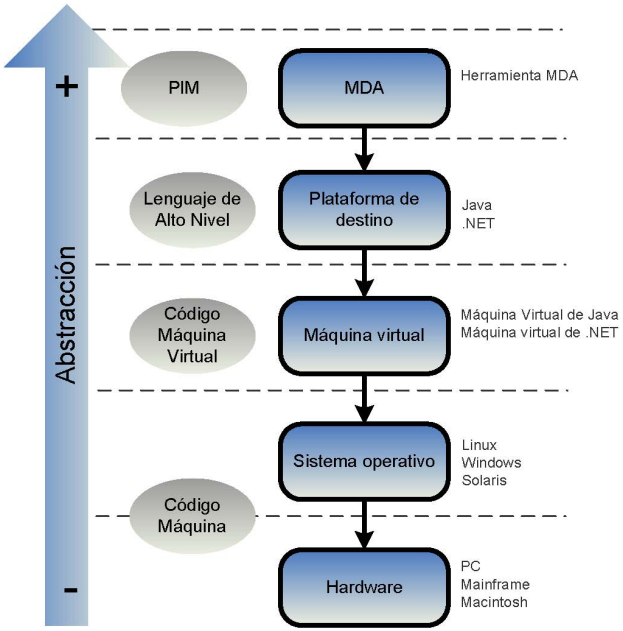
- Sistema: Los conceptos de MDA serán expresados en términos del sistema existente.
- Modelo: Descripción o especificación del sistema mediante un modelo.
- Dirigido por modelos: aporta los medios para usar los modelos directamente para el entendimiento, diseño, construcción, despliegue, operación, mantenimiento y modificación.
- Arquitectura del sistema es una especificación de las partes y los conectores del sistema y las reglas para la interacción de las partes usando conectores.
- Punto de vista es la técnica para la abstracción utilizando un conjunto seleccionado de conceptos de la arquitectura y reglas estructurales

Conceptos Básicos (2/4)

- Vista: representación del sistema desde la perspectiva del punto de vista elegido
- Plataforma: hace referencia a los detalles tecnológicos y de ingeniería que no son relevantes de cara a la funcionalidad esencial del sistema
- Aplicación: funcionalidad que se está diseñando
- Independencia de Plataforma
- El modelo de computación independiente (CIM): no enseña la estructura de los sistemas, el principal usuario son los conocedores del dominio. Es un puente entre los expertos del dominio y sus requisitos, y los expertos en el diseño y construcción de artefactos que juntos satisfacen los requisitos del dominio.

Conceptos Básicos (3/4)

- El modelo de plataforma independiente (PIM): vista del sistema centrada en la operación del mismo que esconde los detalles necesarios para una determinada plataforma, puede ser utilizado con varias plataformas diferentes de forma similar. Una técnica muy común es a través de una máquina virtual

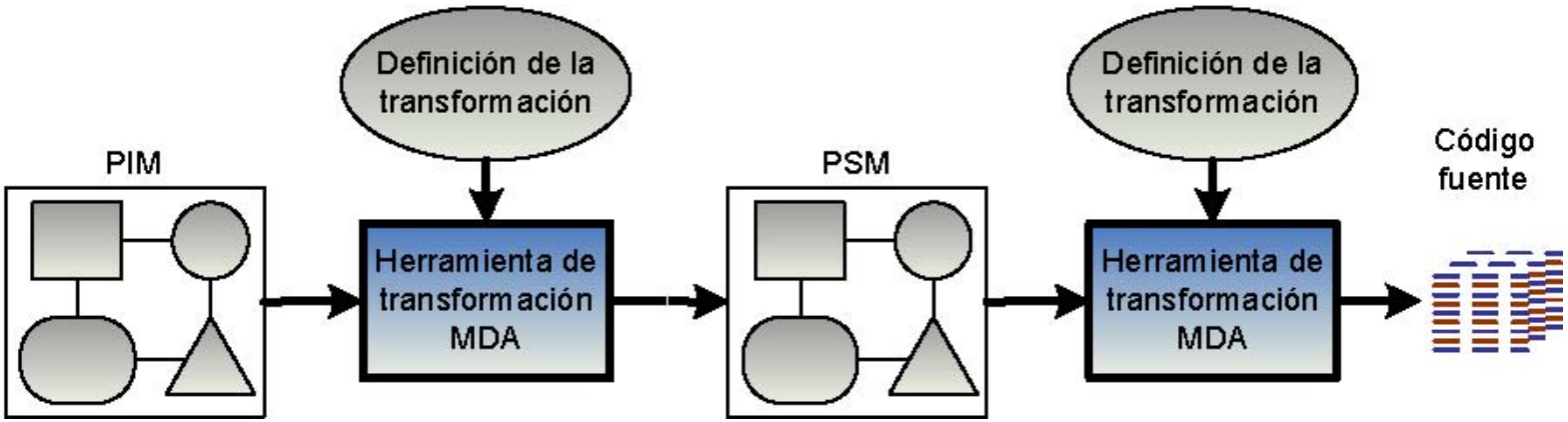


Conceptos Básicos (4/4)

- El modelo de plataforma especifica (PSM): combina las especificaciones del PIM con los detalles que especifica cómo utiliza el sistema un tipo particular de plataforma
- Transformación del modelo: proceso de convertir un modelo en otro del mismo sistema.
- Servicios difundidos: servicios disponibles en una amplia gama de plataformas.
- Implementación: especificación, que suministra toda la información necesaria para la construcción del sistema.

Transformaciones de modelos (1/3)

- Las transformaciones son un punto clave de MDA



Transformaciones de modelos (2/3)

- Transformaciones de PIM a PSM:
  - Implementaciones del estándar QVT (Query/View/Transformation) de OMG:
    - Borland Together Architect
    - SmartQVT
    - MOMENT
  - VIsual Automated model TRAnsformation (VIATRA 2)
  - ATLAS Transformation Language (ATL)
  - Motor de Transformaciones de BOA 2
  - Model-To-Model Transformation (M2M)

Transformaciones de modelos (3/3)

- Transformaciones de PSM a Código:
  - Sistemas de Generación de Código:
    - *Code munging*
    - Expansor de código en línea *(inline-code expander)*
    - Generación de código mezclado *(mixed-code generation)*
  - Motores de Plantillas

Implementaciones de MDAs

- En el mercado existen múltiples herramientas que implementan total o parcialmente los conceptos de MDA
- Herramientas analizadas:
  - AndroMDA (version 3 y papers de la version 4)
  - OpenArchitectureWare (oAW)
  - Software Factories de Microsoft
  - Borland Together Architect Edition

Contenidos

Conceptos

- Los Patrones de Diseño [GHJV95] son la base para la búsqueda de soluciones a problemas comunes en el desarrollo de software
- Los patrones de diseño software describen soluciones simples y elegantes a una serie de problemas concretos que tienen lugar en el diseño orientado a objetos
- Cuatro elementos esenciales:
  1. El nombre del patrón permite describir un problema de diseño así como sus soluciones y consecuencias
  2. El problema describe cuándo aplicar el patrón
  3. La solución describe los elementos que constituyen el diseño, sus relaciones, responsabilidades y colaboraciones
  4. Las consecuencias son los resultados de aplicar el patrón, así como sus ventajas e inconvenientes

Clasificación

- Patrones de Creación: tienen un propósito de creación de objetos
- Patrones Estructurales: tratan con la composición de clases u objetos
- Patrones de Comportamiento: caracterizan el modo en que las clases y objetos interactúan y se reparten la responsabilidad

Contenidos

MDE
Unified Modeling Language (UML)

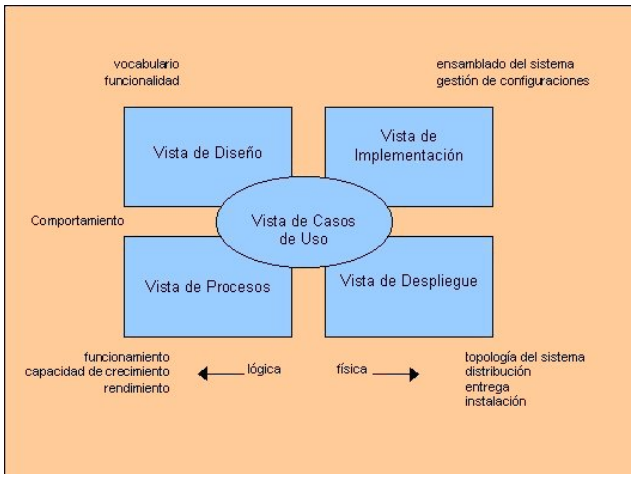
- El lenguaje unificado de modelado es una especificación del Object Management Group (OMG)
- UML es un lenguaje estándar para escribir planos de software
- UML es un lenguaje
  - para visualizar
  - para especificar
  - para construir
  - para documentar

Modelo conceptual de UML

- Elementos principales:
  - los bloques básicos de construcción de UML
  - las reglas que dictan cómo se pueden combinar estos bloques físicos
  - mecanismos comunes que se aplican a través de UML

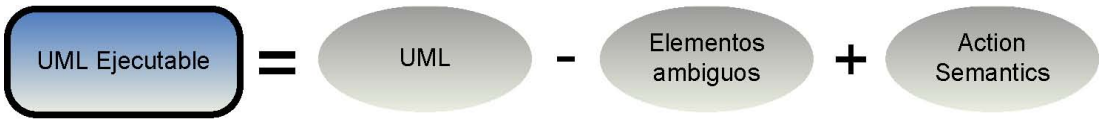
Arquitectura

- Vista de Casos de Uso
- Vista de Diseño
- Vista de Procesos
- Vista de Implementacion
- Vista de Despliegue



UML ejecutable

- Es un perfil de UML que define una semántica de ejecución para un subconjunto de UML [Mel04]
- Toma un subconjunto del lenguaje UML sin construcciones semánticamente ambiguas y les añade la semántica de ejecución a través de Action Semantics
- Se obtiene una propuesta computacionalmente completa



Object Constraint Language (OCL)

- OCL es un lenguaje para la descripción precisa de restricciones que se aplican a los modelos gráficos UML
- Desarrollado por IBM y adoptado en octubre de 2003 por el grupo OMG como parte de UML 2.0

Características del Lenguaje

- Lenguaje de expresión puro: garantiza que una expresión OCL no tendrá efectos colaterales, no puede cambiar nada en el modelo
- Lenguaje de modelado no es un lenguaje de programación: no es posible escribir lógica de programa o flujo de control en OCL
- Lenguaje Formal: todos los constructores tienen un significado formalmente definido, por ser OCL parte de la especificación UML

Utilización de OCL

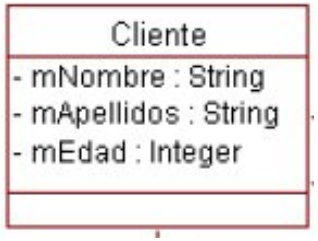
- Como lenguaje de consulta
- Dentro del modelo de clase para expresar invariantes sobre clases y tipos
- Para especificar tipos invariantes para Estereotipos
- Para describir pre y postcondiciones sobre Operaciones y Métodos
- Para describir controles
- Para especificar objetivos para mensajes y acciones
- Para especificar restricciones sobre operaciones
- Para especificar reglas de derivación de atributos para una expresión sobre el modelo UML

Elementos de OCL

- Expresiones, tipos y valores en OCL
- Tipos definidos por el usuario
- Tipos Collection

Ejemplo de utilización OCL para invariantes

- Tenemos un diagrama de clases con un clase Cliente de Banco
- Restricción: La edad de cualquier cliente es siempre mayor o igual a cero.



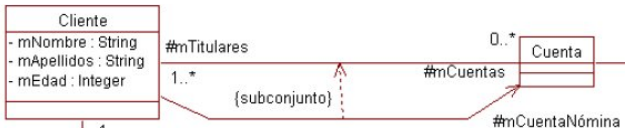
Restricción de Invariante

context Cliente inv:

mEdad>=0

Ejemplo de utilización OCL para pre y postcondiciones

- Extracto del diagrama de clases con dos clases Cliente y Cuentas de Banco
- Restricción: Retirada de efectivo si el saldo de la cuenta es mayor que el importe



Restricción de pre y postcondicion

context Cuenta :: retirar(importe : real) : real
pre : importe>0 and getSaldo()>=importe
post : result=importe

Contenidos

MDE
Meta-Object Facility (MOF)

- El Meta-Object Facility es un estándar del Object Management Group (OMG)
- MOF se utiliza como Metamodelo para definir UML

MDE
Meta-Object Facility (MOF)

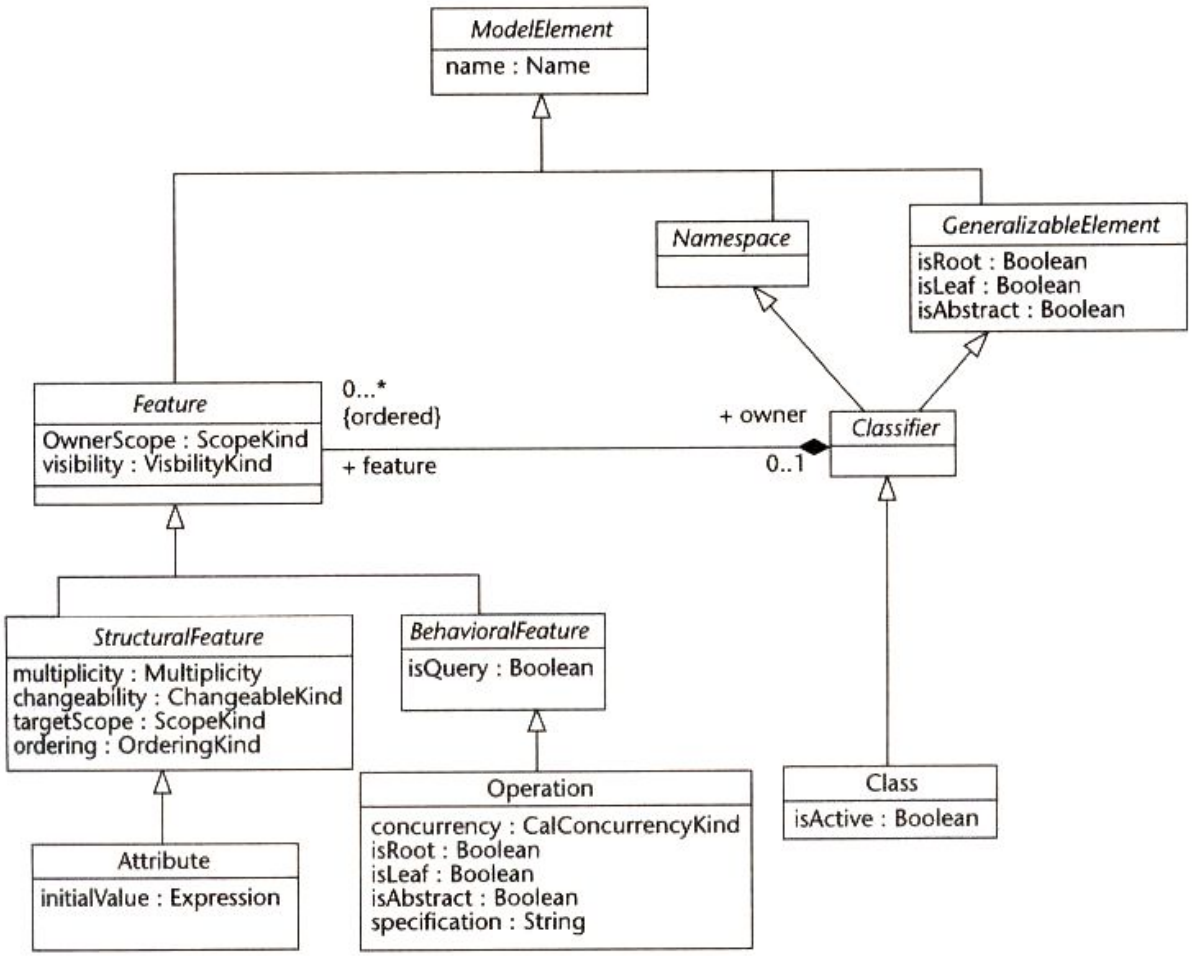
- El Meta-Object Facility es un estándar del Object Management Group (OMG)
- MOF se utiliza como Metamodelo para definir UML

Metamodelo

Es un modelo de un modelo.
Un modelo es una abstracción y un metamodelo es una abstracción de mayor nivel, cuyo propósito es definir las propiedades del modelo en sí mismo.

MDE
Meta-Object Facility (MOF)

Metamodelo de UML



Metanivel MOF en el ámbito de los MDA

Metanivel	Descripción
M3	MOF
M2	Metamodelos
M1	Modelos
M0	Objetos y Datos

Ejemplos:

- MOF: Clase MOF, Atributo MOF,...
- Metamodelos: Clase UML, Atributo UML,...
- Modelos: Clase Cuenta en un banco, clase Cliente,...
- Objetos y Datos: Cliente Cris Pelayo, Cuenta 111118,...

Implementaciones de MOF

- Metadata Repository (MDR) de la plataforma de herramientas de desarrollo NetBeans
- Eclipse Modeling Framework (EMF) es un framework de modelado y una utilidad de generación de código

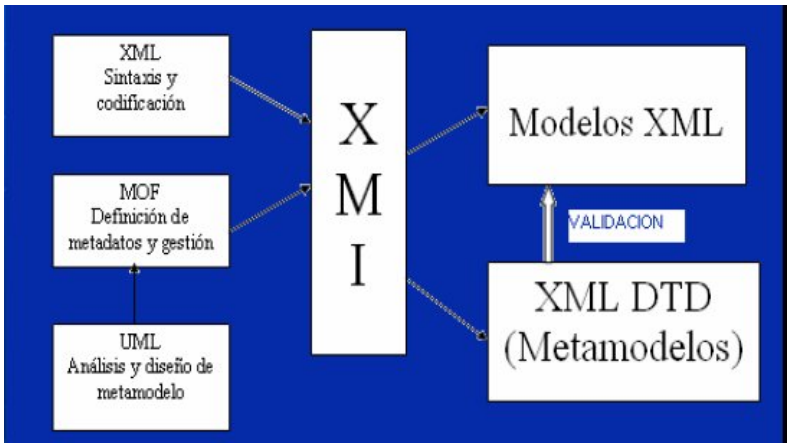
Contenidos

XML Metadata Interchange (XMI)

- XMI es un estándar del Object Management Group (OMG) para el intercambio de información vía XML
- Se utiliza como formato de intercambio de modelos UML, es decir, se puede utilizar para la serialización de metamodelos.
- Desde la perspectiva de modelado de OMG, los datos se dividieron:
  - Modelos abstractos: representan la información semántica, son instancias de lenguaje de modelado basado en MOF
  - Modelos concretos: representan diagramas visuales, para estos se utiliza el XMI

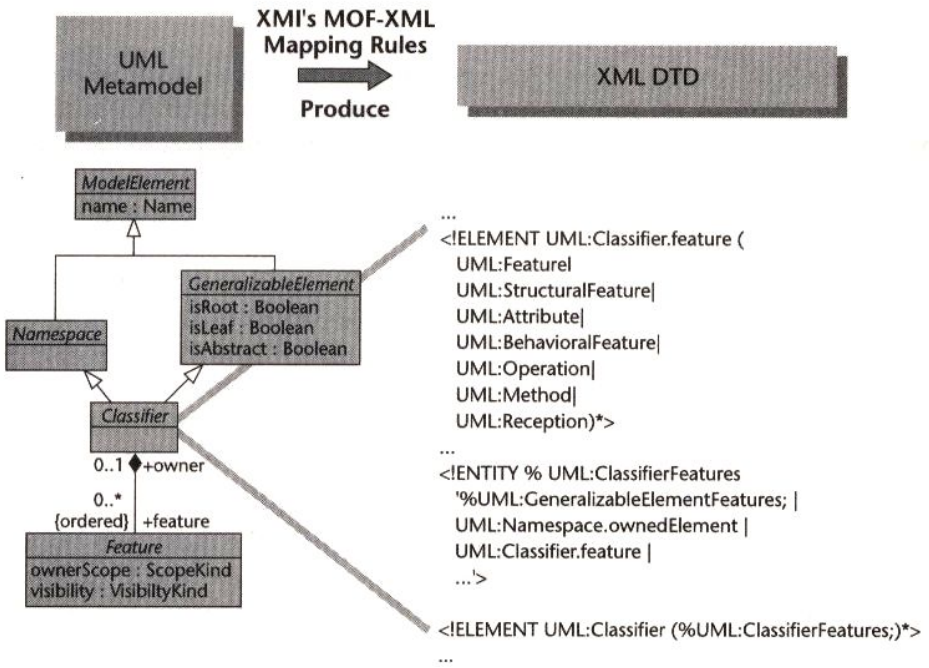
MDE
XML Metadata Interchange (XMI)

- Facilita el intercambio de los metadatos entre las herramientas de modelado UML y los repositorios de metadatos MOF
- XMI hace la correspondencia (mapping) de MOF a XML para permitir el intercambio entre herramientas de modelado
- XMI integra estándares



DTD, XML Schemas y Documentos XML

- Producción de DTD desde un modelo de objetos.
- Producción de XML Schemas desde un modelo de objetos.
- Producción de documentos XML desde un modelo de objetos.



Contenidos

Definiciones de Persistencia

- Es el tiempo en el que los datos existen y son utilizables [AMP86] , por tanto es el mantenimiento de los valores de los datos durante todo su tiempo de vida.
- Principio secundario del modelo de objetos [Boo94] : cualidad de un objeto de mantener su identidad y relaciones con otros objetos con independencia del sistema o proceso que lo creó

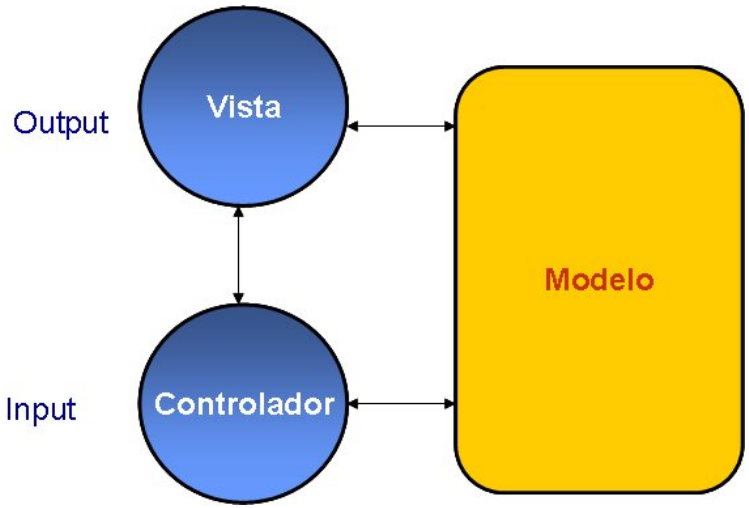
Desarrollo de Aplicaciones Persistentes

- El modelo de datos dominante es el modelo relacional representado por el lenguaje SQL
- Utilización de SQL desde un lenguaje de programación cómo Java
  - de forma directa: JDBC o SQLJ
  - de forma indirecta: sistemas de traducción objeto-relacional como JDO o Hibernate o un framework de persistencia específico (Enterprise Java Beans o Spring)

Contenidos

MDE
Generación automática de Interfaces

- Generación de interfaces de usuario a partir de modelos conceptuales.
- Dos conceptos fundamentales en la creación de interfaces: la usabilidad y la accesibilidad
- El Modelo Vista Controlador (MVC)



El Modelo Vista Controlador (MVC)

- Patrón de arquitectura de software inventado por Trygve Reenskaug e introducido en el entorno de desarrollo del SmallTalk 80
- Tres elementos:
  - **Modelo**: representación del dominio de la información sobre la cual funciona la aplicación (Capa de dominio o negocio)
  - **Vista**: presenta el modelo en un formato adecuado para interactuar, es el mecanismo encargado de realizar la correspondencia entre los datos provenientes del modelo y la interfaz.
  - **Controlador**: realiza el control del flujo de navegación de la aplicación.

Flujo de la Información en MVC

- El usuario interactúa con la interfaz de usuario de alguna forma, por ejemplo pulsando un botón.
- El controlador gestiona el evento que llega.
- El controlador accede al modelo, modificándolo de forma adecuada a la acción solicitada por el usuario.
- El controlador delega a los objetos de la vista la tarea de desplegar la interfaz de usuario. La vista obtiene sus datos del modelo para generar la interfaz apropiada para el usuario donde se refleja los cambios en el modelo.
- La interfaz de usuario espera nuevas interacciones del usuario, comenzando el ciclo nuevamente.

Implementaciones del MVC orientadas a la Web

- JavaServer Faces
- Apache Struts
- XForms

Contenidos

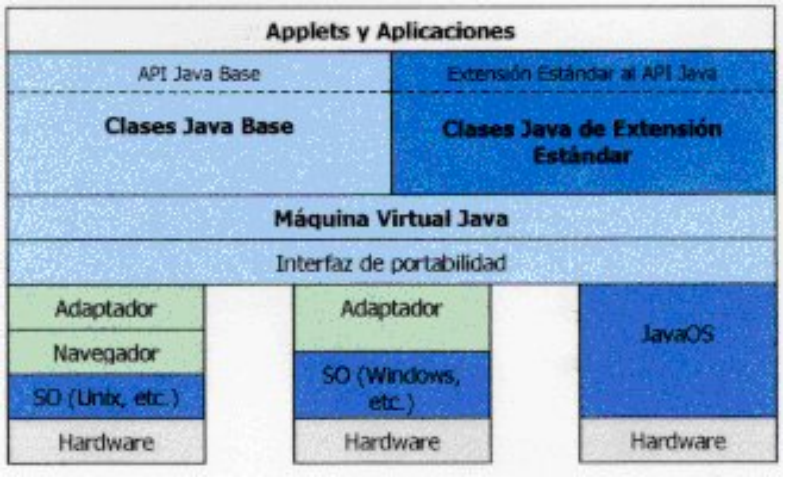
La Plataforma J2EE

- Conjunto de especificaciones diseñadas por Sun que permiten la creación de aplicaciones empresariales

J2EE = JVM + Lenguaje Java + API Java + Utilidades

Características de la JVM (Java Virtual Machine)

- Pila de ejecución y un repertorio de instrucciones que manipulan dicha pila.
- Puede dar soporte a varios hilos (threads) de ejecución concurrente.
- Compilación JIT
- Verificación estática de bytescodes
- Gestión de memoria dinámica
- Dependencia del lenguaje Java

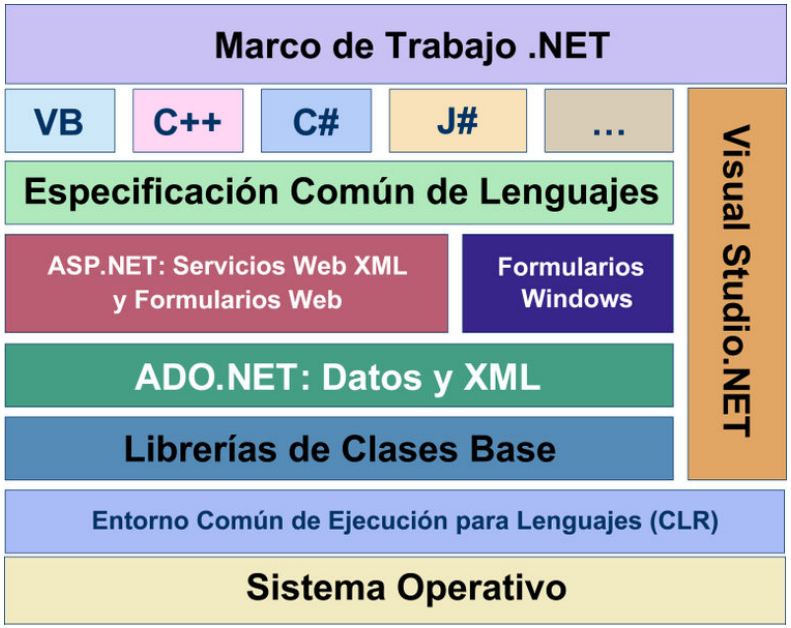


El lenguaje Java

- Este lenguaje fue desarrollado por J. Gosling en 1993
- Características:
  - Sencillo
  - Orientado a Objetos Puro
  - Interpretado y compilado
  - Distribuido y con multihilos
  - Independiente de la plataforma y portable
  - Robusto y seguro

La Plataforma .NET

- Desarrollada por Microsoft.
- Intenta simplificar el desarrollo de aplicaciones en un entorno altamente distribuido como es Internet.

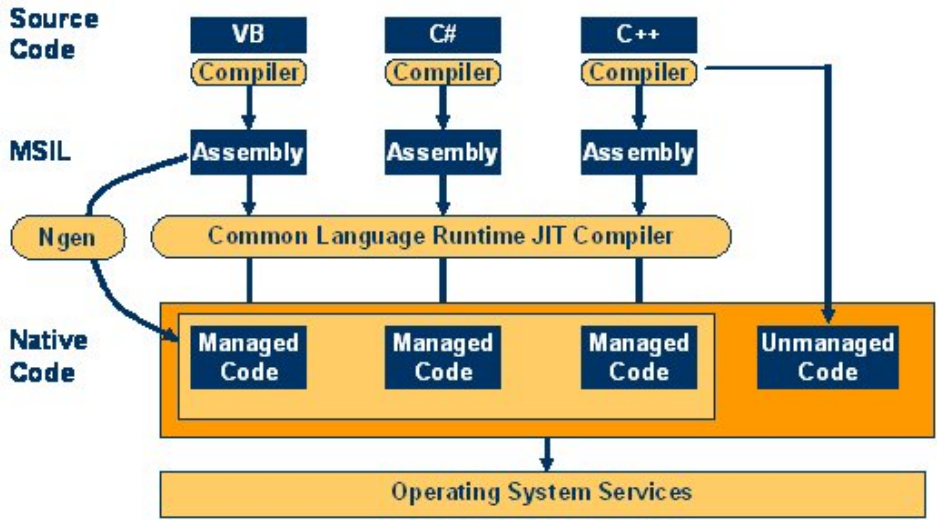


Características de la CLR (Common Language Runtime)

- Utilización de una pila de ejecución
- Compilación JIT
- Generación de código en tiempo de instalación
- Verificación estática de tipos
- Gestión dinámica de memoria
- Independiente del lenguaje
- Sistema de Componentes

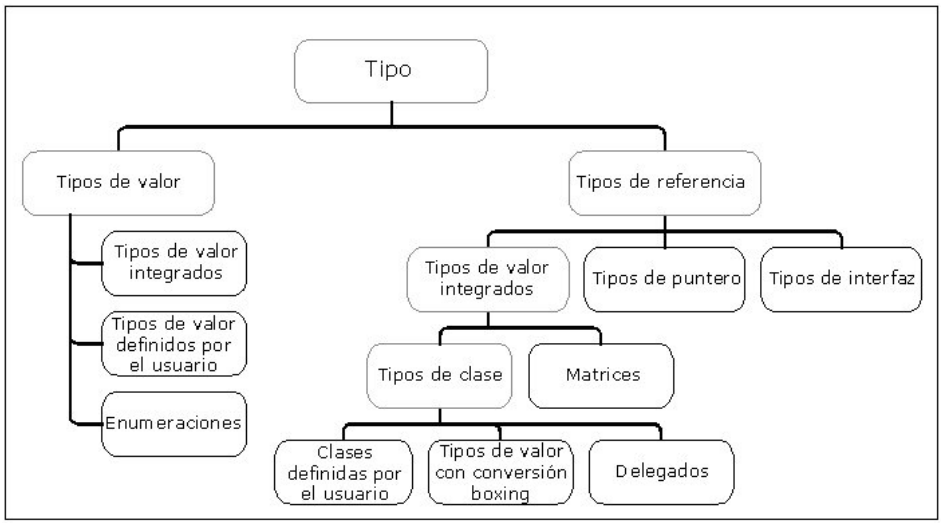
Los Lenguajes .NET

- Lenguaje Visual Basic
- Lenguaje C#
- Lenguaje C++ administrado
- Lenguaje Jscript .NET
- Lenguaje J#



Otros elementos de la Plataforma .NET

- CTS: Sistema Común de Tipos
  - Tipos de valor
  - Tipos de referencia



- Biblioteca de clases de .Net

MDE
Contenidos

Contenidos

- Capacidad de los MDAs con metodologías ágiles frente a los métodos tradicionales.
- Analizar el impacto de los MDAs en el proceso de desarrollo de software.

- Capacidad de los MDAs con metodologías ágiles frente a los métodos tradicionales.
- Analizar el impacto de los MDAs en el proceso de desarrollo de software.

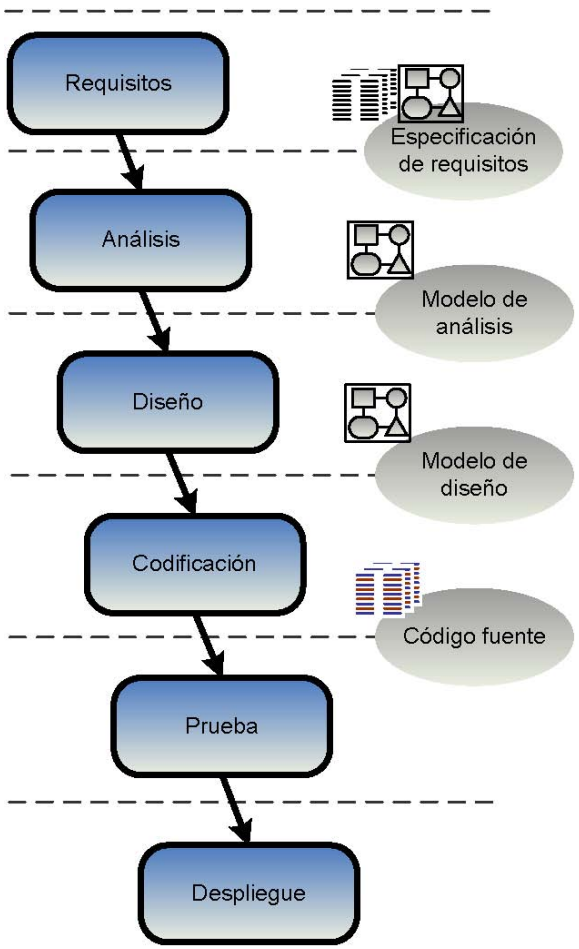
Disciplina de Ingeniería

Algunos autores consideran que el desarrollo de herramientas MDA es un requisito indispensable para convertir el proceso de crear software en una verdadera disciplina de ingeniería. [Fra03] [KWB03] [RFW+04]

Contenidos

Proceso de Desarrollo Tradicional (1/9)

Etapas básicas



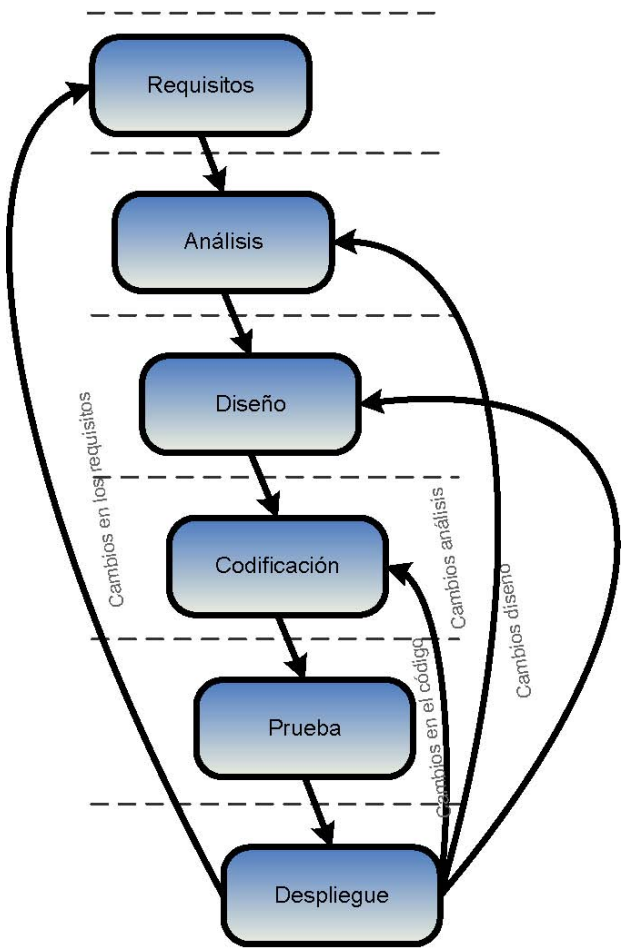
- Análisis de requisitos
- Modelo de análisis
- Modelo de diseño
- Escritura del código fuente
- Despliegue del sistema

Proceso de Desarrollo Tradicional (2/9)

- Etapas básicas del proceso de desarrollo de Software, propuesto inicialmente por Royce [Roy70]
  - Modelo de ciclo de vida en cascada *(waterfall model)*
- Modelo en espiral de Boehm [Boe88]
- Evolucionario [MZ96]
- Modelo incremental [Som01]
- Unified Proccess (UP), la implementación más adoptada de este método es el Racional Unified Proccess (RUP) [Kru00]

Proceso de Desarrollo Tradicional (3/9)

La iteración como respuesta al cambio



- Todos estos modelos adoptan la iteración cómo herramienta principal para abordar los cambios en el software
- Se acepta que los cambios en los requisitos del software son inevitables [Bec99a]
- El proceso de desarrollo debe estar preparado para absorberlos

Problemas con la Iteración

- Los métodos con ciclo de vida iterativo producen gran cantidad de artefactos en forma básicamente de documentos de texto y diagramas, pero estos están desconectados entre sí y del código fuente que representan [KWB03]
- Dos alternativas opuestas:
  - Retroceder a la fase de desarrollo apropiada y acometer el cambio, provoca un *"exceso de burocracia"* [Fow05]
  - Realizar los cambios únicamente en el código. Las tareas de análisis, modelado y documentación previas a la codificación dejan de considerarse imprescindibles

Proceso de Desarrollo Tradicional (5/9)

Métodos ágiles

- Acortan el plazo de iteración y apuestan por un desarrollo evolutivo
- Utilizan una planificación adaptativa
- Apuestan por la entrega incremental
- Incorporan prácticas de desarrollo que buscan permitir una respuesta rápida y flexible a los cambios

Proceso de Desarrollo Tradicional (5/9)

Métodos ágiles

- Acortan el plazo de iteración y apuestan por un desarrollo evolutivo
- Utilizan una planificación adaptativa
- Apuestan por la entrega incremental
- Incorporan prácticas de desarrollo que buscan permitir una respuesta rápida y flexible a los cambios

Unified Proccess (UP)

Considerarlo pesado o ágil dependerá de su implantación concreta. Martin Fowler en [Fow05] señala que en la industria pueden encontrarse utilizaciones del UP que van desde rígidos modelos en cascada a instancias perfectamente ágiles

Proceso de Desarrollo Tradicional (6/9)

Problemas de los Métodos ágiles

- Falta de datos empíricos que prueben su efectividad
- No escalabilidad a la hora de su aplicación en proyectos grandes [TFR02]
- Excesiva dependencia del conocimiento tácito de los miembros del equipo de desarrollo debido a la ausencia de una documentación formal [Coc01]
- Alto acoplamiento entre las diferentes prácticas de XP y la falta de documentación de sus interrelaciones [Van05]

Proceso de Desarrollo Tradicional (6/9)

Problemas de los Métodos ágiles

- Falta de datos empíricos que prueben su efectividad
- No escalabilidad a la hora de su aplicación en proyectos grandes [TFR02]
- Excesiva dependencia del conocimiento tácito de los miembros del equipo de desarrollo debido a la ausencia de una documentación formal [Coc01]
- Alto acoplamiento entre las diferentes prácticas de XP y la falta de documentación de sus interrelaciones [Van05]

Métodos ágiles = Métodos anárquicos

Esta consideración es errónea dado que en el *"Manifiesto Ágil" (Agile Manifesto)* [Bec01], en el principio número 10 se señala que la excelencia técnica y los buenos diseños aumentan la agilidad

Proceso de Desarrollo Tradicional (7/9)

Modelado y Codificación

- Un error en muchos enfoques metodológicos de desarrollo software ha sido obviar la importancia de la actividad de codificación.
- Royce equiparó la codificación a la fase de construcción de otras ingenierías [Roy70]
- Fowler [Fow05] señala que en otras ingenierías:
  - El diseño es más difícil de predecir y requiere la gente con mayor formación y creatividad
  - La actividad de construcción es más fácil de predecir, requiere gente con menos formación y es, en general, mucho más costosa
  - ¿Puede obtenerse un diseño que haga de la codificación una actividad de construcción predecible? Y si esto es posible ¿será rentable en términos de costes?

Proceso de Desarrollo Tradicional (8/9)

Modelado y Codificación

- El problema fundamental es la gran distancia existente entre:
  - Los modelos software o descripciones del software realizadas a un elevado nivel de abstracción.
  - El código fuente que comprende la aplicación y que realiza los modelos desarrollados sobre una plataforma tecnológica concreta.

Proceso de Desarrollo Tradicional (8/9)

Modelado y Codificación

- El problema fundamental es la gran distancia existente entre:
  - Los modelos software o descripciones del software realizadas a un elevado nivel de abstracción.
  - El código fuente que comprende la aplicación y que realiza los modelos desarrollados sobre una plataforma tecnológica concreta.

Enfoque pragmático: Modelado ágil [AJ02]

El modelado es importante, como herramienta de comunicación, NO de documentación
No se construyen modelos detallados del software
Los modelos sólo especifican las partes más complejas de la aplicación, los problemas sencillos de diseño se resuelven directamente en la fase de codificación

Proceso de Desarrollo Tradicional (9/9)

Problemas en los Procesos de Desarrollo Actuales

- La desconexión entre los artefactos de modelado, y entre éstos y el código fuente del programa, hace desaconsejable la realización de un modelado formal, detallado e integral.
- Si no se realiza un modelado formal se plantea un problema importante de mantenimiento.
- Otros problemas comunes a los métodos de desarrollo iterativos son:
  - Productividad
  - Calidad
  - Trazabilidad de requisitos
  - Reutilización

Contenidos

Proceso de Desarrollo utilizando MDA (1/4)

El Enfoque MDA de OMG

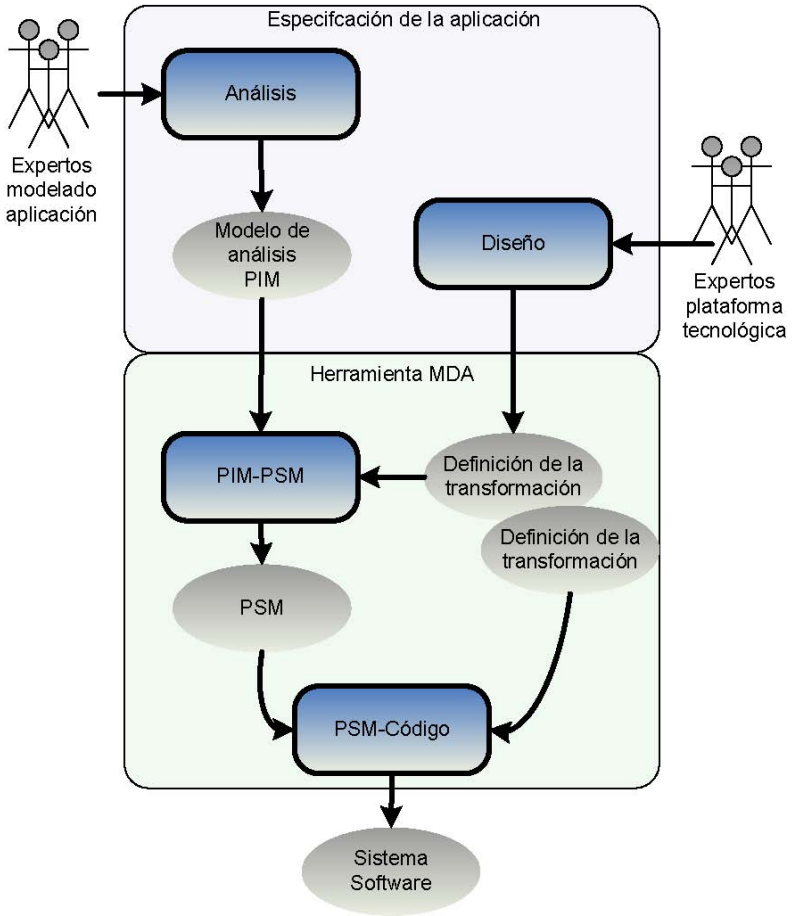
- Importantes cambios en cómo se acometen las fases en el proceso de desarrollo
- Cambios sustanciales en las fases del análisis y del diseño debido a la doble transformación automática:
  - entre un PIM y un PSM
  - entre un PSM y el código fuente de la aplicación
- PIM se corresponde con la fase de diseño del modelo tradicional
- La transformación es el conocimiento experto sobre la plataforma tecnológica

El Enfoque MDA de OMG

- Se favorece la reutilización de:
  - Mismos modelos con diferentes definiciones de transformación
  - Mismas definiciones de transformación con diferentes modelos
- La doble transformación automática (PIM-PSM y PSM-código fuente) implica que los modelos se perciban como ejecutables:
  - Permite validar los modelos realizados
  - Permite determinar cuándo se ha terminado de modelar: cuando el PIM verifique todos los casos de pruebas diseñados [RFW+04]

Proceso de Desarrollo utilizando MDA (3/4)

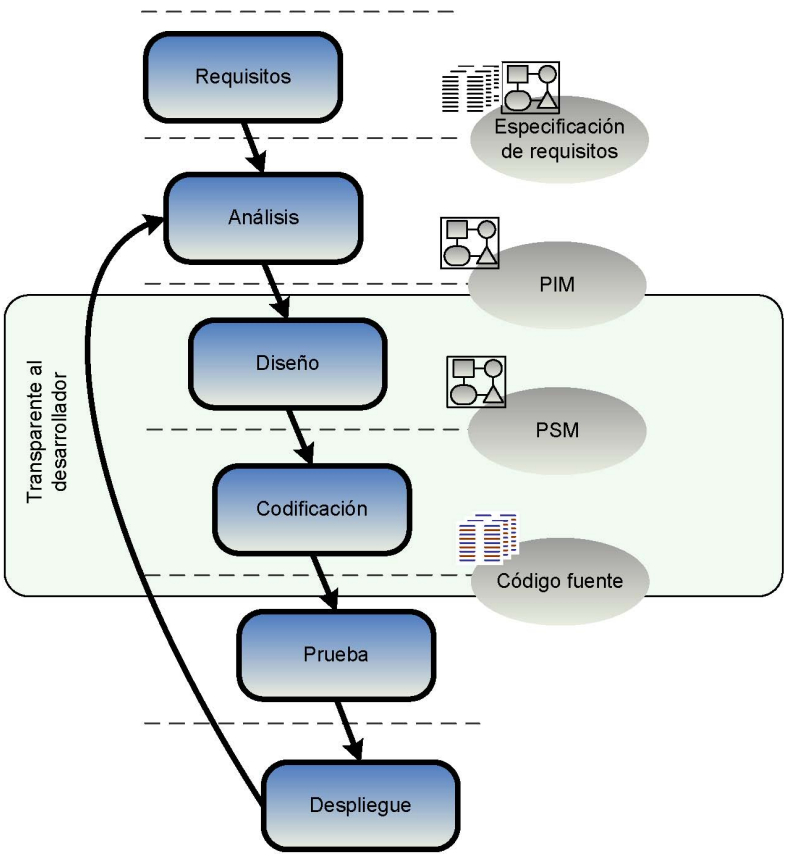
Especificación e implementación con MDA



- Separación del análisis y del diseño, realizados por grupos de especialistas diferentes
- La herramienta MDA acepta como entradas:
  - Modelo de diseño (PIM)
  - Definición de la transformación
- A partir de las entradas se genera el PSM
- El PSM será transformado a código fuente

Proceso de Desarrollo utilizando MDA (4/4)

Ciclo de Vida con MDA



- Iteración para afrontar el cambio en los requisitos, pero son más efectivas
  - Cambio en requisitos tecnológicos ⇒ ajuste en las transformaciones que son reutilizables para otras aplicaciones
  - Cambio en requisitos de negocio ⇒ modificaciones en el modelo de análisis.

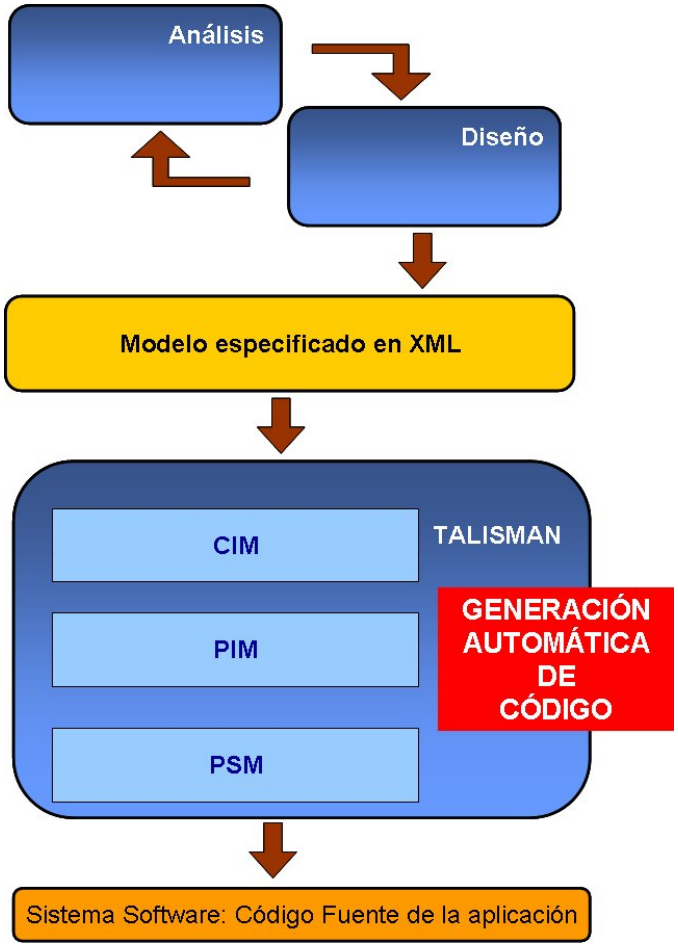
- Como la transformación PIM - Código fuente es automática se pueden validar los modelos de análisis ejecutando los PIM

Contenidos

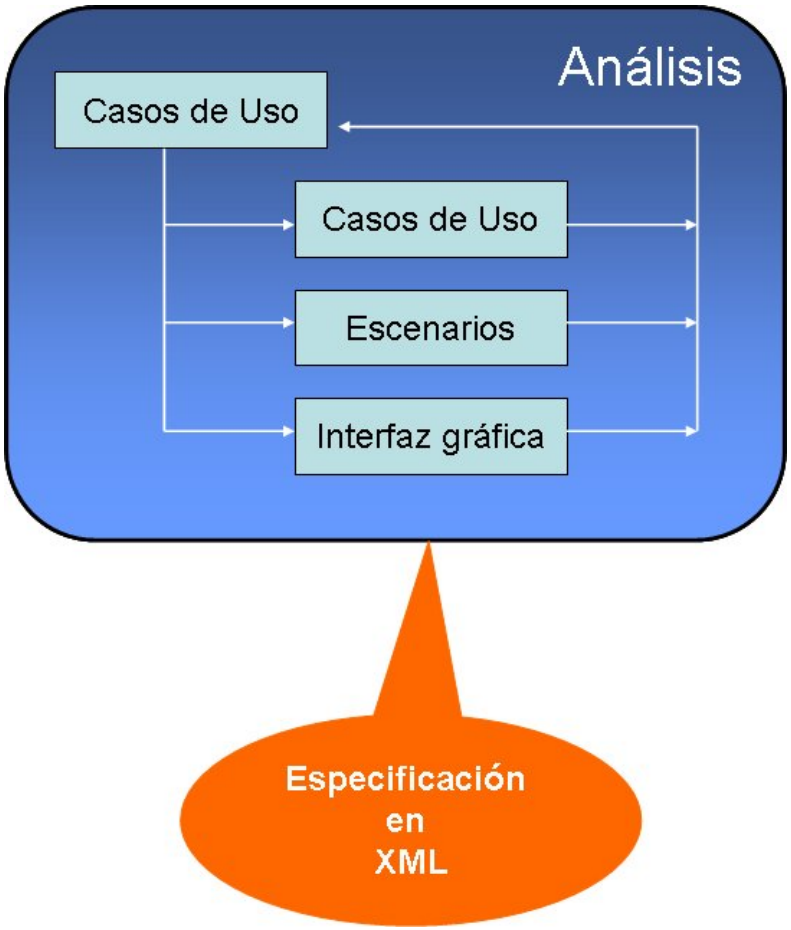
Metolodología propuesta (1/11)

- Los métodos ágiles de desarrollo constituyen una respuesta a la realidad cambiante
- El proceso de desarrollo propuesto en TALISMAN se caracteriza por:
  - Generación de código automática a partir de especificaciones en XML
  - Ser ágil en base a XP basada en la simplicidad, comunicación y retroalimentación
  - Ser dirigido por modelos especificados en XML
  - Ser un proceso AMDD *(Agile Model Driven Development)*

Proceso de Desarrollo Ágil con TALISMAN

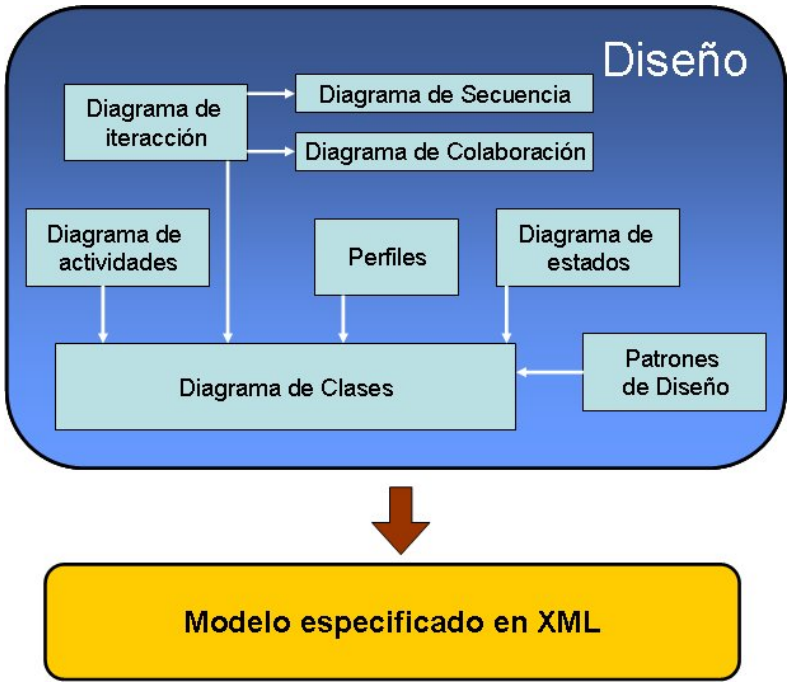


Análisis



- Basado en Casos de Uso
- Escenarios que especifican los Casos de Uso
- Especificación completa del proceso *(story board)* de cada Escenario
- Interfaz Gráfica ligada a cada Escenario

Diseño

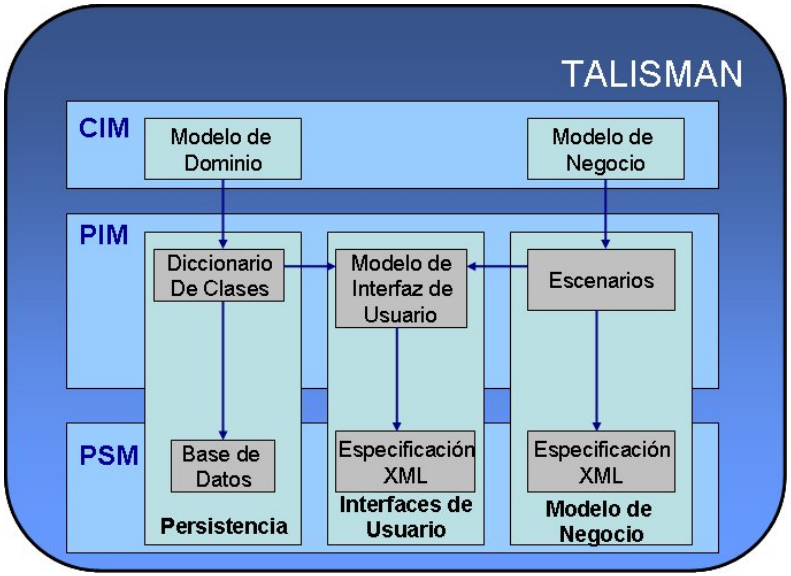


- Diccionario de Clases
- Comportamiento interno de los métodos de cada clase
- Especificación de la arquitectura software
- Especificación de la arquitectura hardware
- Otros tipos de Modelos

Modelo especificado en XML

- Especificación de requisitos en XML
- Aporta contenido semántico a UML a través de extensiones
- Especificación del comportamiento de los métodos (algorítmica de los métodos)
- Información de los perfiles
- Implementación de Patrones de Diseño

Arquitectura de TALISMAN



- Utiliza el Modelo especificado en XML para generar código
- Realiza la validación semántica
- Construye el sistema de persistencia, los interfaces de usuario con ayuda de herramientas generadoras de interfaces y genera el código por medio de transformaciones

Arquitectura de TALISMAN

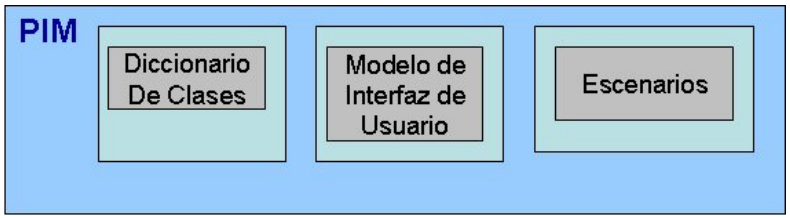
- La arquitectura tiene varios modelos
  - CIM
  - PIM
  - PSM
- Guías de transformación entre modelos:
  - PIM a PIM
  - PIM a PSM
  - PSM a PSM

Modelo CIM de la Arquitectura de TALISMAN

- Centrado en el dominio del negocio
- Especificado en XML

Modelo PIM de la Arquitectura de TALISMAN

- Especificación del funcionamiento propias del sistema junto a la especificación para la implementación en un medio informático
- Especificado en XML
- N-Capas
  - Diccionario de Clases
  - Modelo de interfaz de usuario
  - Escenarios que se utilizan para definir la lógica de negocio



Modelo PSM de la Arquitectura de TALISMAN

- Se combina el punto de vista independiente de la plataforma con los detalles y características propias del uso de una plataforma de desarrollo
- N-Capas
  - Base de Datos a través de especificación XML
  - Interfaces de usuario a través de especificación XML
  - Modelo de negocio a través de especificación XML

Guías de transformación entre modelos de TALISMAN

- CIM a PIM: los aspectos independientes de computación se transforman en una especificación formal de la estructura y forma del sistema
- PIM a PIM: con objeto de refinamiento
- PIM a PSM: un PIM refinado se transforma en un modelo dependiente de la infraestructura final de ejecución
- PSM a PSM: con objeto de refinamiento
- PSM a Código: un PSM refinado se transforma en código fuente en la plataforma de destino

Contenidos

Construcción del prototipo (1/16)

Objetivos

- Construir un prototipo utilizando la Metodología TALISMAN
- Orientado al mundo Web
- Plataforma .NET
- Generación automática de código ASP.NET y C#
- Se crean los modelos en XML y a partir de ellos se realiza el proceso de desarrollo, transformación y construcción del sistema hasta obtener el código de forma automática

Características de las aplicaciones Web

- En el desarrollo de aplicaciones Web se mezclan:
  - Métodos tradicionales de desarrollo
  - Propuestas de ingeniería Web
  - Métodos para hipermedia
- La agilidad en los desarrollos Web es fundamental
  - Pronta disponibilidad del software en la red
  - Ciclos de desarrollo generalmente más cortos
  - Requisitos de usuario desconocidos
  - Desarrollos aparentemente muy sencillos
  - Necesidad de entrega de versiones previas

Adaptación de TALISMAN a los Sistemas Web:

- La adaptación de TALISMAN se realiza definiendo nuevos artefactos adaptados a la Ingeniería Web
- Con los Diagramas y la información del Análisis y del Diseño se genera la documentación en XML con información de:
  - Clases con la información del Diagrama de Clases
  - Fragmentos con la información semántica de las clases del Diagrama de Fragmentos
  - Navegación con la información sobre la navegación del Diagrama de Navegación
  - Usuarios con sus roles que definidos en el Diagrama de Usuarios
  - Servicios Web definidos en los Diagrama de servicios Web y Servicios Web Cliente
  - Modelo de la lógica de Negocio

Construcción del prototipo (4/16)

Clases más representativas del Prototipo

- Model
- Resources
- Generator
- GeneratorASPNET2
- Utils
- Associations
- Class
- MultipleAssociation
- ClassTest
- Compiles
- DataAccessBase
- Links

Clase Generator

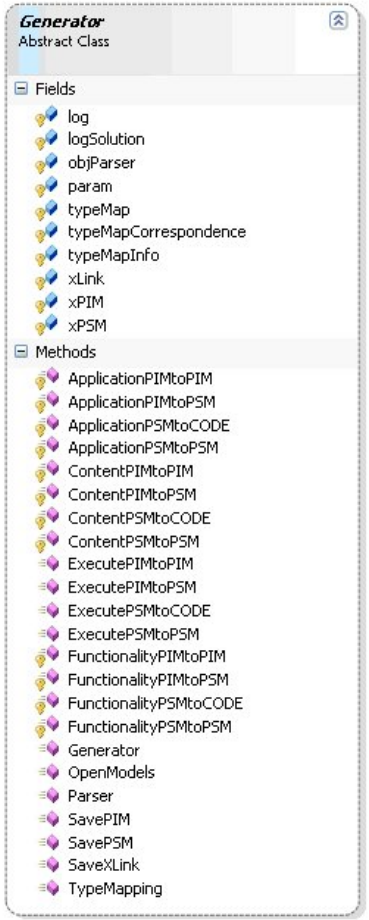
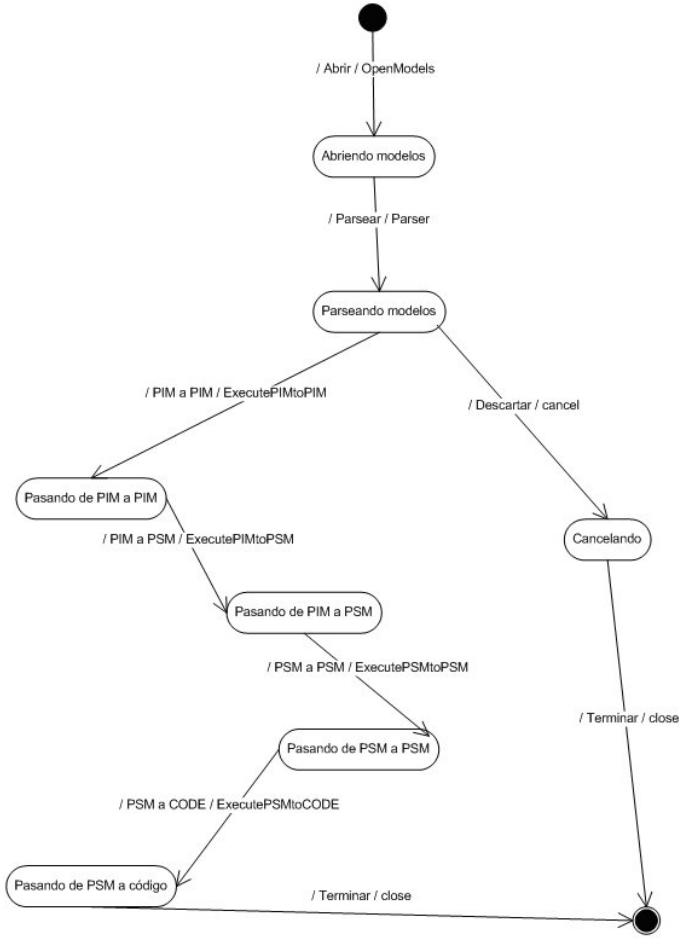
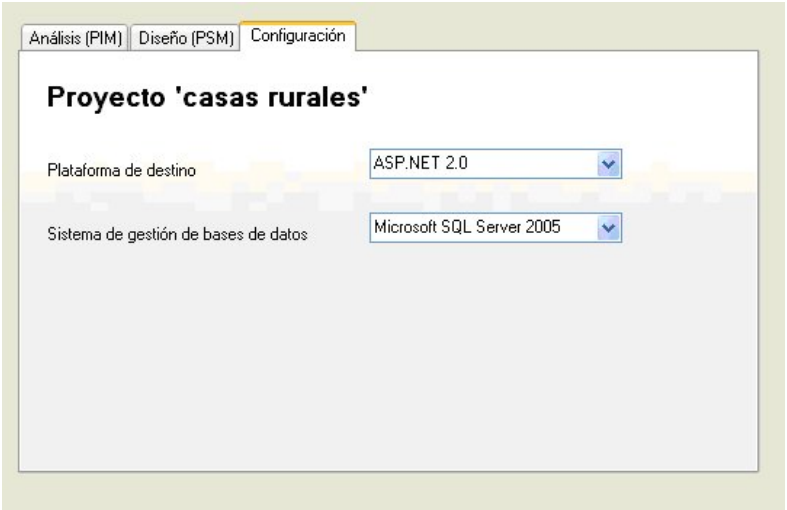


Diagrama de Estados de la Clase Generator

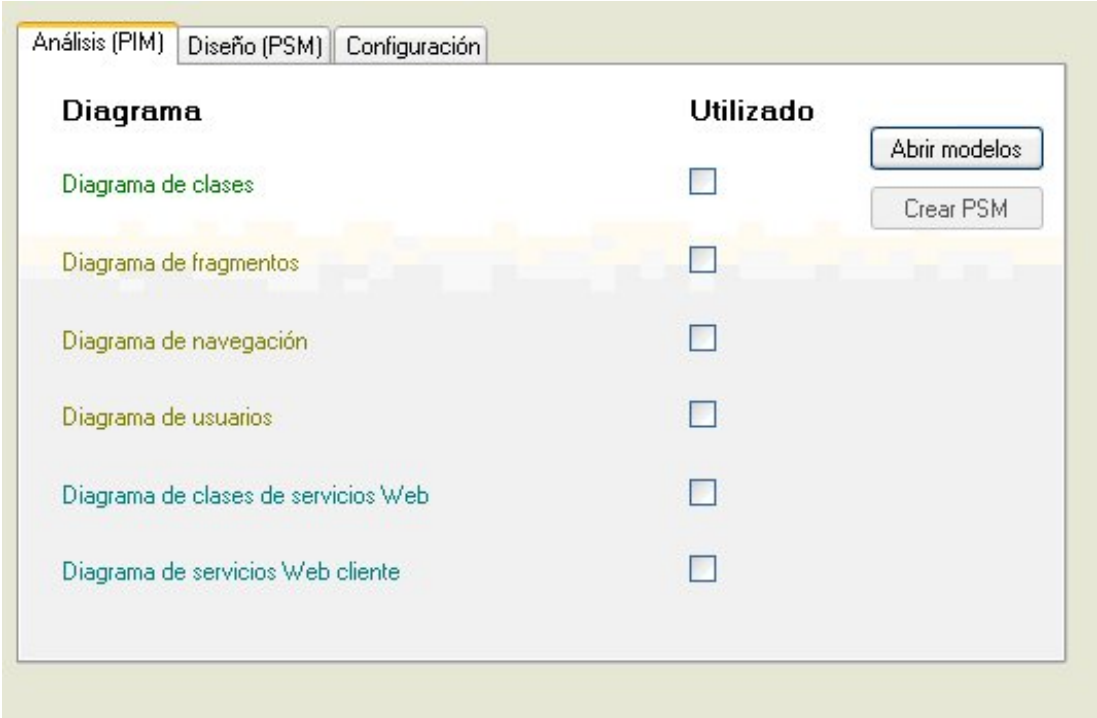


Configuración del Prototipo



Construcción del PIM

- El PIM de TALISMAN con la información de la aplicación se almacena en un archivo "pim.xml"
- Se permite selecionar los modelos de entrada que serán utilizados

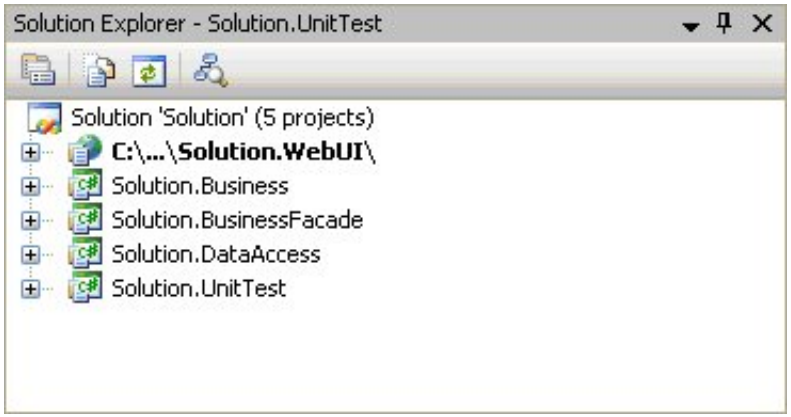


Construcción del PSM

- A partir del PIM se genera el PSM
- A partir del PSM se genera el código fuente de la aplicación
- Proceso completamente automático

Generación de Código fuente

- Se genera un proyecto de Visual Studio 2005 denominado “solution.sln”
- La solución consta de cinco proyectos



Construcción del prototipo (10/16)

Interfaz de Usuario (WebUI)

- Proyecto principal
- Páginas WebASPX generadas
- Web.config con la configuración de la aplicación
- Servicios Web que se ofrecen y que se consumen
- Archivos generadores de menús.

Construcción del prototipo (11/16)

Lógica de Negocio (Business)

- Contiene las clase para las tablas que se generarán en la base de datos
- Contiene los archivos que utiliza NHibernate para independizar el acceso a datos de la lógica de negocio

Lógica de Negocio (BusinessFacade)

- Puente entre la capa de negocio y la capa de acceso a datos

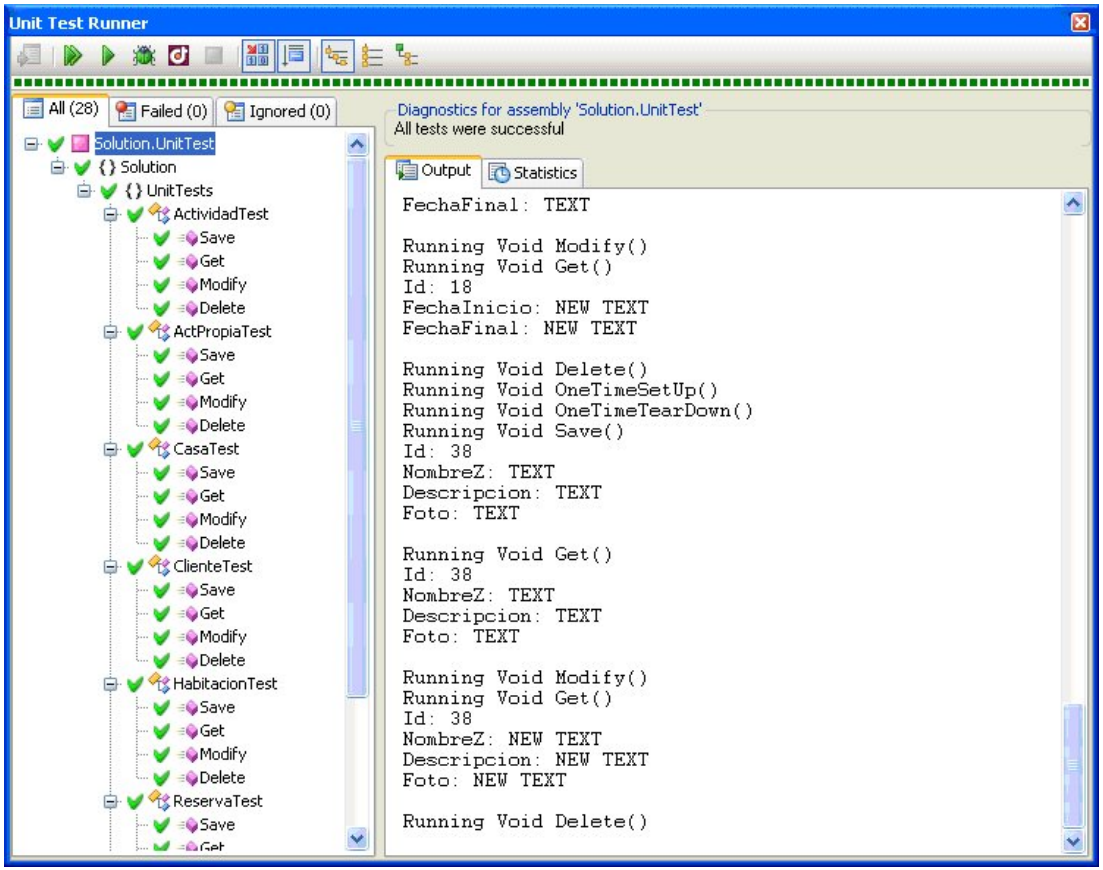
Construcción del prototipo (12/16)

Persistencia (DataAccess)

- Independiza el acceso a la base de datos del resto de la aplicación
- Contiene métodos para guardar, obtener o borrar elementos de la base de datos

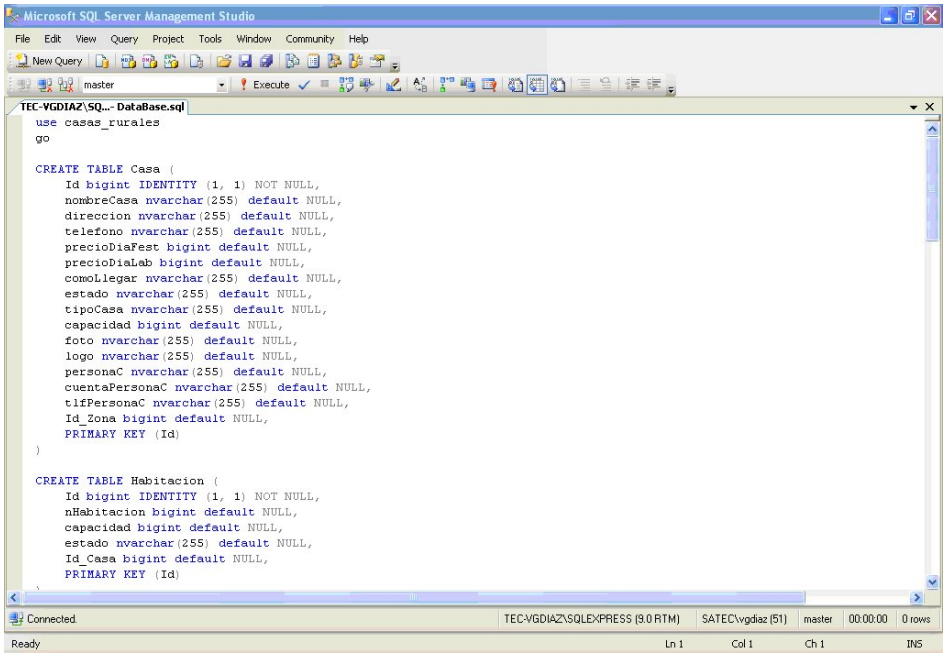
Pruebas (TestUnit)

- Pruebas unitarias generadas automáticamente



Tablas de la base de datos

- Script que genera la estructura de la base de datos
- Para Sql Server 2005 se generará un archivo llamado DataBase.sql



Construcción del prototipo (15/16)

Administración de usuarios y perfiles de usuario

- Se realiza la gestión de usuarios a través de la información generada en el archivo web.config
- Se especifican roles y permisos de acceso a toda o parte de la aplicación

Fragmento de web.config

```
<location path="Reserva.aspx">
 <system.web>
  <authorization>
   <allow roles="Admin" /><deny users="?" />
  </authorization>
 </system.web>
</location>
```

Construcción del prototipo (16/16)

Registros(logs)

- Registro de proyecto generado con todas las actividades de TALISMAN dividido en secciones:
  - Beginning: creación directorios y copia de archivos comunes a todos los proyectos
  - PIM to PIM
  - PIM to PSM
  - PSM to PSM
  - PSM to Code
- Registro de la ejecución del proyecto generado
  - Acciones que realiza el usuario de la aplicación
  - Mensajes de NHibernate generados durante la ejecución

Contenidos

Ejemplo de utilización del Prototipo (1/11)

Objetivo y Características generales

- Objetivo: Comprobar la validez del prototipo creado siguiendo TALISMAN
- Características:
  - Plataforma ASP.NET 2.0
  - Sistema de gestión de bases de datos Sql Server 2005

Ejemplo de utilización del Prototipo (1/11)

Objetivo y Características generales

- Objetivo: Comprobar la validez del prototipo creado siguiendo TALISMAN
- Características:
  - Plataforma ASP.NET 2.0
  - Sistema de gestión de bases de datos Sql Server 2005

Casas Rurales en Asturias

Aplicación Web con información de casas rurales de diferentes zonas en Asturias y de las actividades que en ellas se pueden realizar

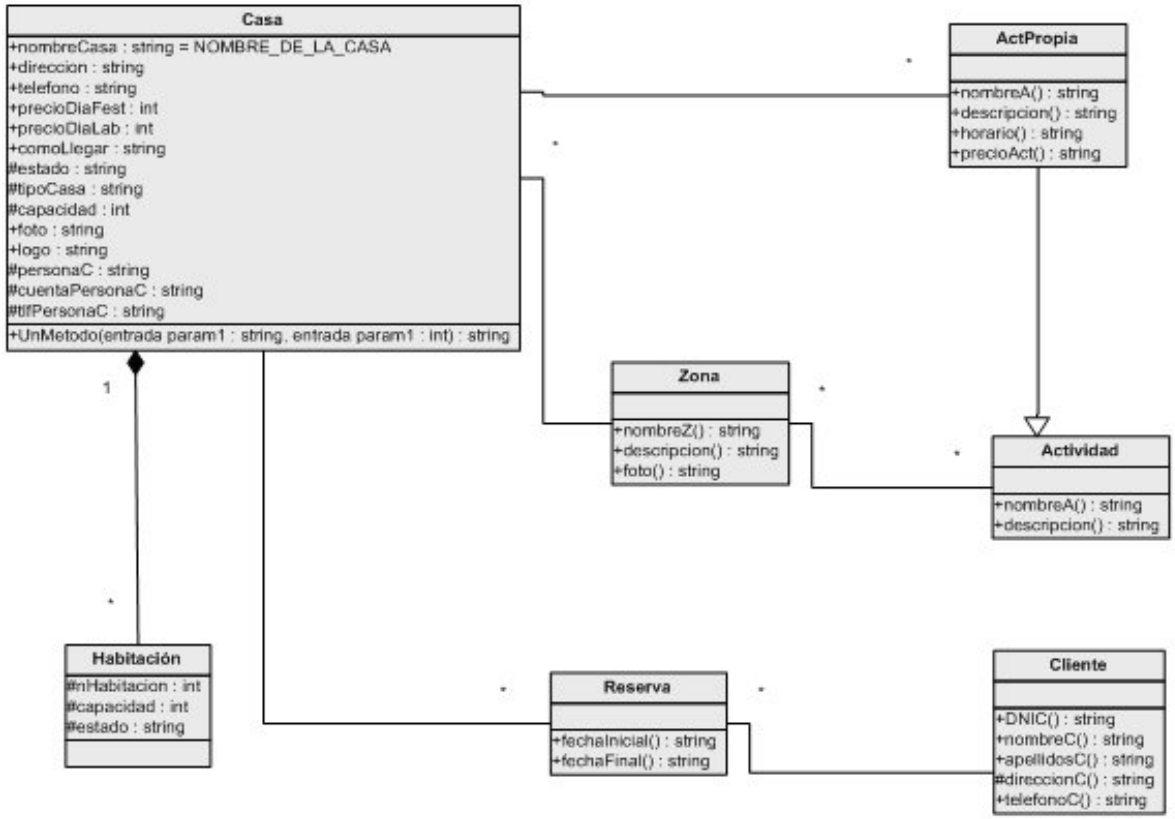
Ejemplo de utilización del Prototipo (2/11)

Análisis y Diseño

- Especificación en XML de los Requisitos
- Diagramas:
  - Diagrama de Clases
  - Diagramas de Interfaz de Usuario a través de:
    - Diagrama de Fragmentos
    - Diagrama de Navegación
  - Diseño de la Lógica de Negocio a través de:
    - Diagrama de clases de servicios Web
    - Diagrama de servicios Web Cliente
    - Diagrama de usuarios

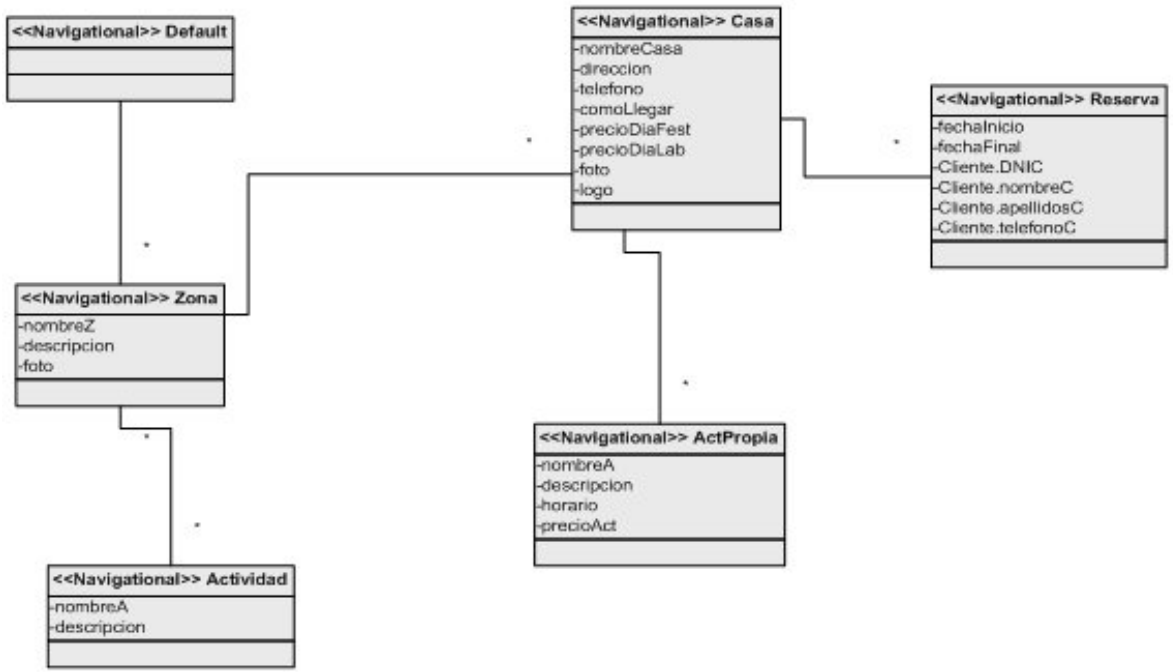
Ejemplo de utilización del Prototipo (3/11)

Análisis y Diseño: Diagrama de Clases



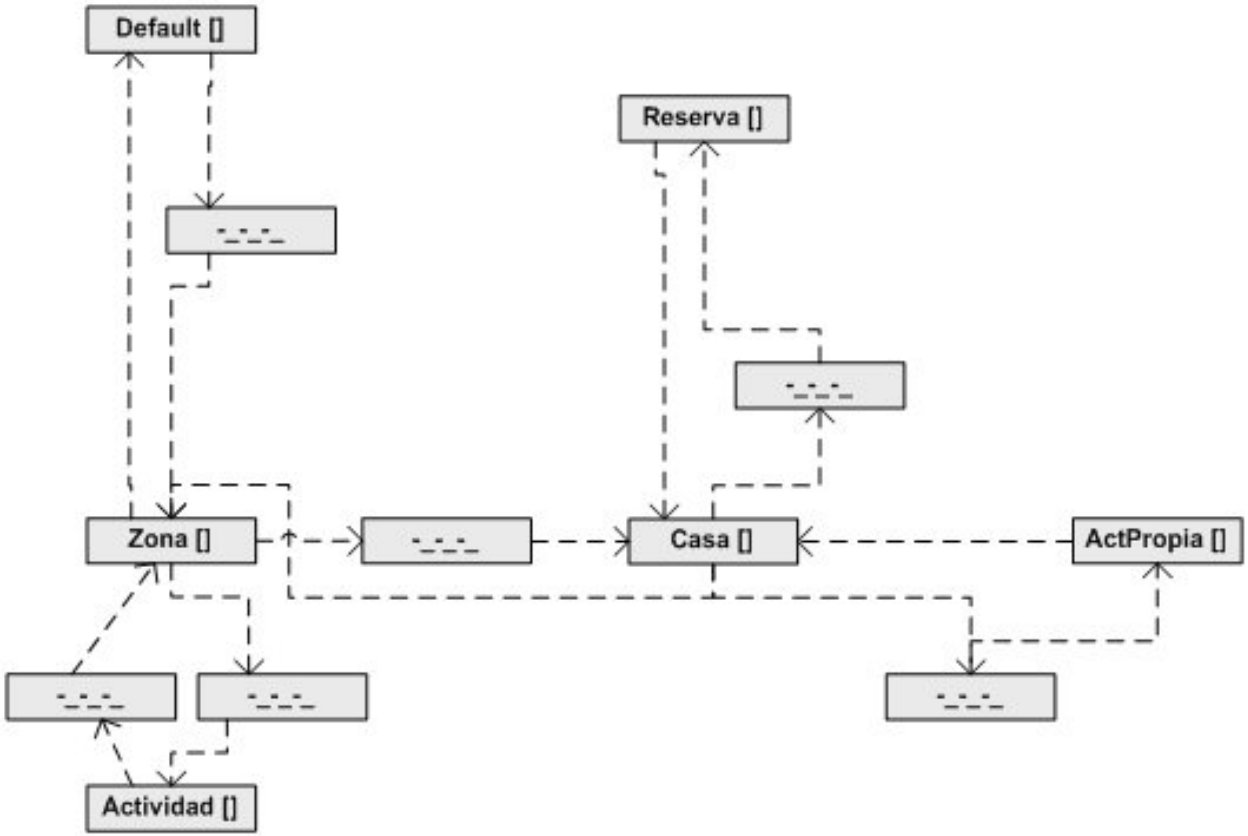
Ejemplo de utilización del Prototipo (4/11)

Análisis y Diseño: Diagrama de Fragmentos



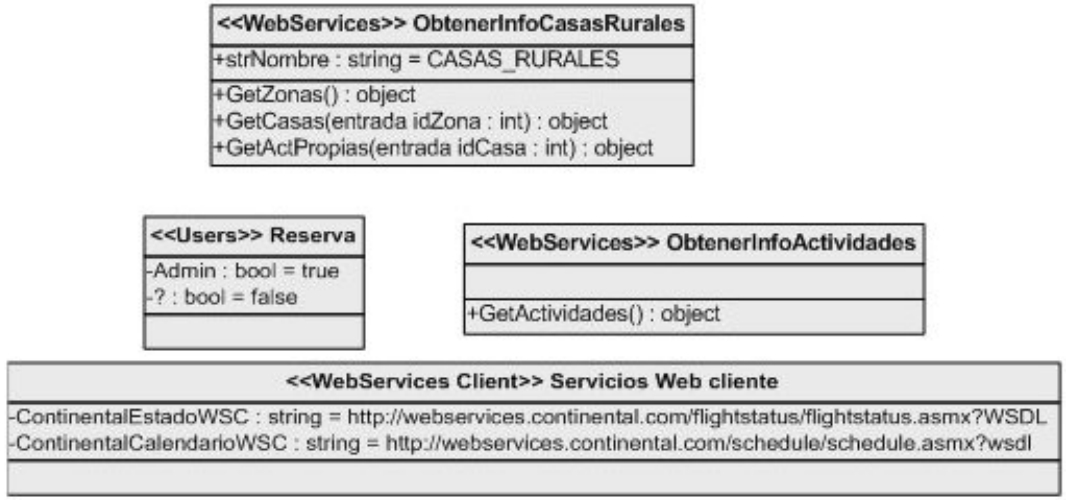
Ejemplo de utilización del Prototipo (5/11)

Análisis y Diseño: Diagrama de Navegación



Ejemplo de utilización del Prototipo (6/11)

Análisis y Diseño: Lógica de Negocio



Ejemplo de utilización del Prototipo (7/11)

Modelo especificado en XML

- A partir de los Diagramas del Análisis y Diseño se realiza la documentación en formato XML
  - Clases
  - Fragmentos
  - Navegación
  - Usuarios
  - Servicios Web
  - Servicios Web Cliente
- Un único archivo que constituye la entrada a TALISMAN

Ejemplo de utilización del Prototipo (8/11)

TALISMAN: Modelo PIM

- Almacenado en el archivo “pim.xml”

Esqueleto del PIM

```
<?xml version="1.0" encoding="utf-8"?>
 <PIM>
   <classes>
   <fragments>
   <navigation>
   <users>
   <webservices>
   <webservicesClients>
 </PIM>
```

Ejemplo de utilización del Prototipo (9/11)

TALISMAN: Transformaciones mediante archivos XSLT y CSS

- Especificación de transformaciones con archivos XSLT (eXtensible Stylesheet Language Transformations)
  - Actividad.xslt
  - ActPropia.xslt
  - Casa.xslt
  - Default.xslt
  - Error.xslt
  - Reserva.xslt
  - Zona.xslt
- Estilo de la aplicación Web a través de archivos CSS de Hojas de Estilo (StyleSheet.css).

Ejemplo de utilización del Prototipo (10/11)

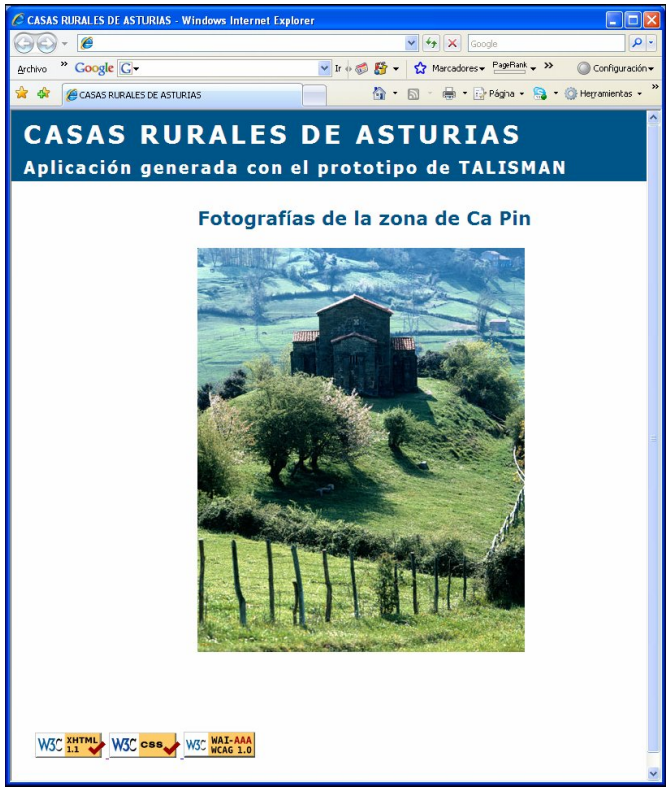
TALISMAN: Recursos adicionales

- Para crear la interfaz de usuario se proporcionan imágenes

Ejemplo de utilización del Prototipo (11/11)

Salida Generada

- El prototipo genera todos los archivos necesarios para la ejecución de la aplicación sobre la plataforma ASP .NET.
- Se generan más de 60 archivos



Contenidos

Aportaciones en el ámbito de MDA

- Definición de un entorno de desarrollo basado en MDA
- Definición de modelos CIM a partir de los procesos del negocio
- Directrices para la creación de modelos PIM a partir de modelos CIM
- Una metodología que mejora la conexión entre las reglas de negocio y las tecnologías MDA

Aportaciones en el ámbito del software ágil

- Una metodología para el desarrollo de software ágil que pueda adaptarse a las necesidades y características particulares de cada empresa
- Una metodología de desarrollo de software que combina las ventajas del MDA y el desarrollo ágil

Contenidos

Referencias

- [Amb04] S.W. Ambler Agile Model Driven Development with UML 2. Cambridge University Press, 3rd edition, 2004.
- [BBC+06] L. Balmelli, D. Brown, M. Cantor y M. Mott Model-driven systems development. IBM Systems Journal, volumen 45 número 3, páginas 569 a 585. 2006.
- [Bec01] K. Beck. Manifesto for agile software development.Technical Report, 2001. http://www.agilemanifesto.org/
- [Fra03] David S. Frankel. Model Driven Architecture. Applying MDA to Enterprise Computing. OMG Press, 2003
- [KWB03] A. Kleppe, J. Warmer y W. Bast MDA Explained. The Model Driven Architecture: Practice and Promise. Addison-Wesley. 2003.
- [Mel04] S. J. Mellor Agile MDA.The MDA Journal, páginas 144 a 160, 2004
- [MM03] J. Miller y J. Mukerji MDA Guide Version 1.0.1. Object Management Group, 2003

Referencias

- [MSU+04] S.J. Mellor, K. Scott, A. Uhl, y D. Weise. MDA Distilled. Principles of Model-Driven Architecture. Object Technology. Addison-Wesley, 2004.
- [PCJ06] B. C. Pelayo García-Bustelo, J. M. Cueva Lovelle, y A.A. Juan Fuente. C3NET: Smart Environment For .NET Code Generation Using MDA. WORLDCOMP'06 Proceedings: The 2006 World Congress in Computer Science, Computer Engineering, and Applied Computing. Las Vegas, USA (2006).
- [PC06a] B. C. Pelayo García-Bustelo y J. M. Cueva Lovelle. Arquitecturas dirigidas por modelos (MDA). El framework C3NET. II Congreso Internacional de Ingeniería de Computación y Sistemas (IICIIS). Trujillo, Perú (2006).
- [PC06b] B. C. Pelayo García-Bustelo y J. M. Cueva Lovelle. Usabilidad, accesibilidad y métricas de sitios Web. II Congreso Internacional de Ingeniería de Computación y Sistemas (IICIIS). Trujillo, Perú (2006).

Referencias

- [PCS+05] B.C. Pelayo García-Bustelo, J.M. Cueva Lovelle, M.C. Suárez Torrente, y A.A. Juan Fuente. C3NET: Framework para la construcción de MDA en la Plataforma .NET. III Simposio Internacional de Sistemas de Información en la Sociedad del Conocimiento (SISOF 2005). Instituto Tecnológico de las Américas, República Dominicana (2005)
- [PC05] B. C. Pelayo García-Bustelo y J. M. Cueva Lovelle. Evolución de las directrices de accesibilidad del W3C. III Simposio Internacional de Sistemas de Información en la Sociedad del Conocimiento (SISOF 2005). Instituto Tecnológico de las Américas, República Dominicana (2005)
- [RFW+04] C. Raistrick, P. Francis, J. Wright, C. Carter y I. Wilkie. Model Driven Architecture with Executable UML. Cambridge University Press, 2004.

Referencias

- [Sch06] Douglas C. Schmidt. Model - Driven Engineering. IEEE Computer Society, páginas 25 a 31. Febrero 2006
- [SFP04] Richard Soley, David S. Frankel, y John Parodi. The MDA Journal: Model Driven Architecture Straight From The Masters. Meghan Kiffer Pr, Noviembre 2004