Modelado de sistemas software: Introducción

Diciembre 2006

Miguel A. Laguna

Fuentes: Bran Selic, ICSE'03 UML2.0 Tutorial y
número especial sobre MDD, IEEE Software, September 2003, pag. 19-25

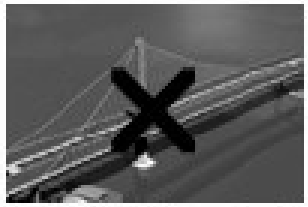
Modelado de ...

- Sistemas...
  - Sistemas web
  - Sistemas de control/tiempo real
- Familias de sistemas
  - Variabilidad
- Patrones de alto nivel
- Restricciones
- Requisitos
- Procesos
- ...Modelos ¿ejecutables?

La importancia de los modelos

- Before they build the real thing...

…they first build models…and then learn from them



Modelos de ingeniería

- Modelo de ingeniería:
  - Representación reducida de un sistema
- Propósito:
  - Ayudar a comprender un problema complejo (o solución)
  - Comunicar ideas acerca de un problema o solución
  - Guiar la implementación

Características de los modelos

- Abstracto
  - Enfatiza los elementos importantes y oculta los irrelevantes
- Comprensible
  - Fácil de comprender por los observadores
- Preciso
  - Representa de forma fiel el sistema que modela
- Predictivo
  - Se pueden usar para deducir conclusiones sobre el sistema que modela
- Barato
  - Mucho más barato y sencillo de construir que el sistema que modela
- Los modelos de ingeniería eficaces deben satisfacer todas estas características

Cómo se usan

- Para detectar errores u omisiones en el diseño antes de comprometer recursos para la implementación
  - Analizar y experimentar
  - Investigar y comparar soluciones alternativas
  - Minimizar riesgos
- Para comunicarse con los “stakeholders”
  - Clientes, usuarios, implementadores, encargados de pruebas, documentadores, etc.
- Para guiar la implementación

Desarrollo guiado por modelos ("Model-Driven development" o MDD)

- Una aproximación al desarrollo de software en el que el enfoque y los artefactos fundamentales son modelos (y no programas)
- Implica la generación automática de programas a partir de modelos
  - Utilizando lenguajes de modelado directamente como herramientas de implementación

"El modelo es la implementación"

Lo esencial en MDD

- En MDD el enfoque y los artefactos fundamentales son modelos (y no programas)
- La mayor ventaja es que los conceptos de modelado están mucho menos ligados a la tecnología de implementación y más cerca del dominio del problema
- Los modelos son más fáciles de especificar, comprender y mantener

Tecnología

- Se generan automáticamente programas completos a partir de modelos
  - (y no sólo esqueletos o fragmentos de código)
- Se “verifican” automáticamente modelos en una computadora
  - (por ejemplo, ejecutándolos)

Estándares: Model-Driven Architecture

- Iniciativa MDA de OMG
- Es un marco para definir estándares:
  - MOF
  - UML
  - XML
  - SOAP
  - SPEM
  - RAS....

La práctica

- Modelos Observables
  - Es necesario que las herramientas nos den información sobre errores, al igual que lo hacen los compiladores (o los depuradores)

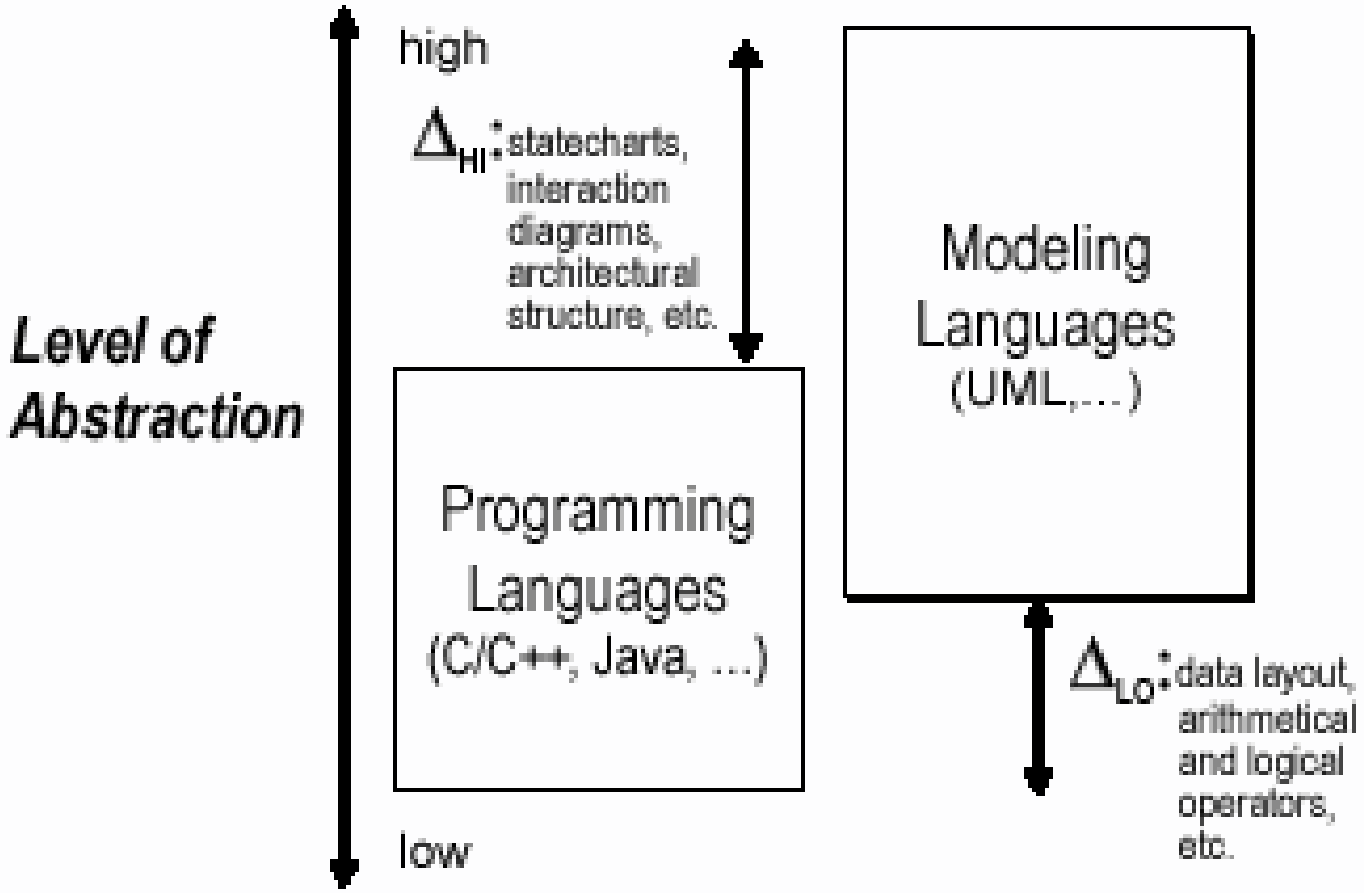
...La práctica

- Modelos ejecutables
  - El “hola_mundo”
  - Debe ser posible trabajar con modelos incompletos (pero bien formados)
- Eficiencia del sistema generado
  - 15 % de diferencia con las herramientas actuales

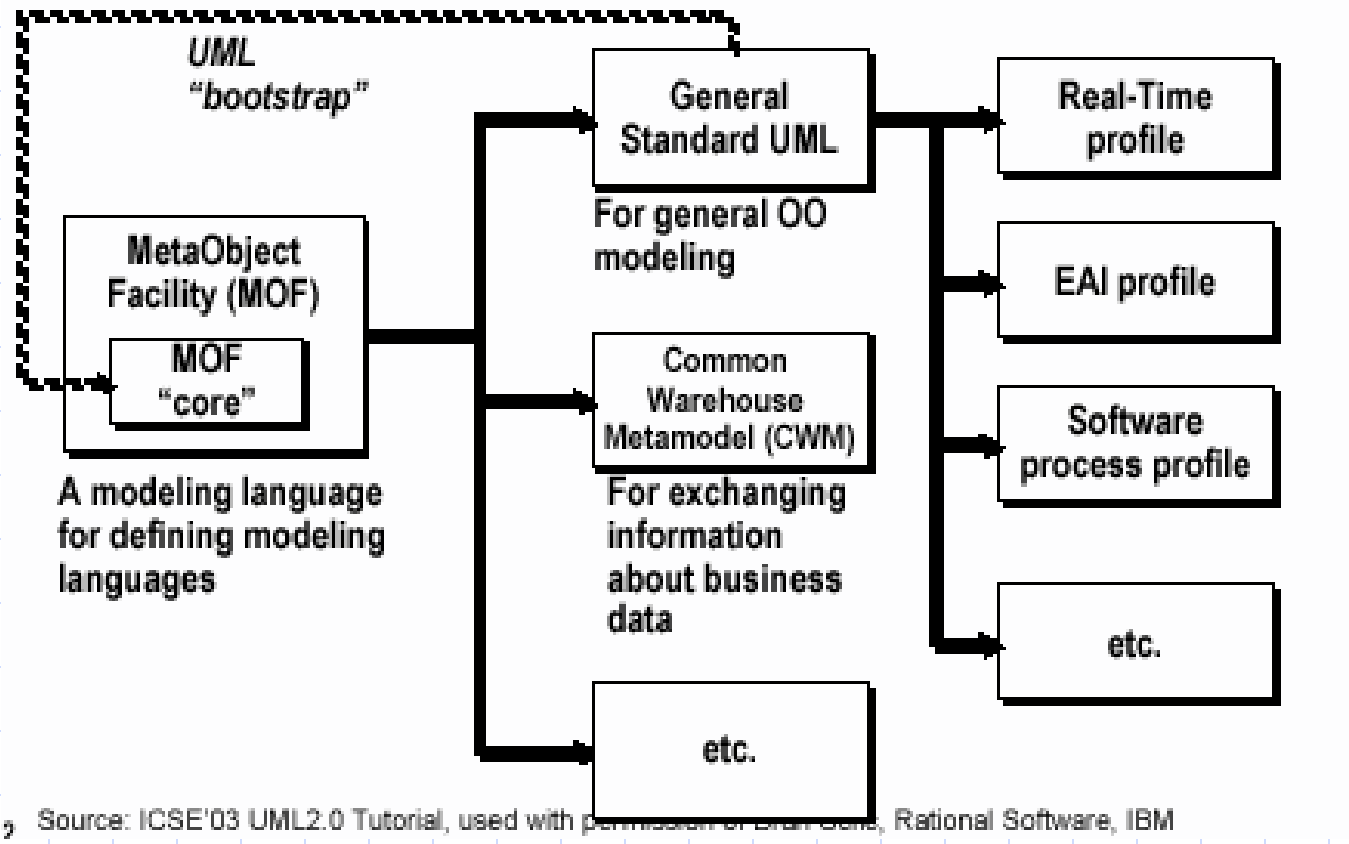
...La práctica

- Escalabilidad
  - Grandes sistemas:
    - Tiempo de generación/compilación del sistema
    - Tiempo de generación/compilación de cada incremento
  - En realidad el tiempo de generación representa un 10 %
- Integración con sistemas legados

Modelado y lenguajes

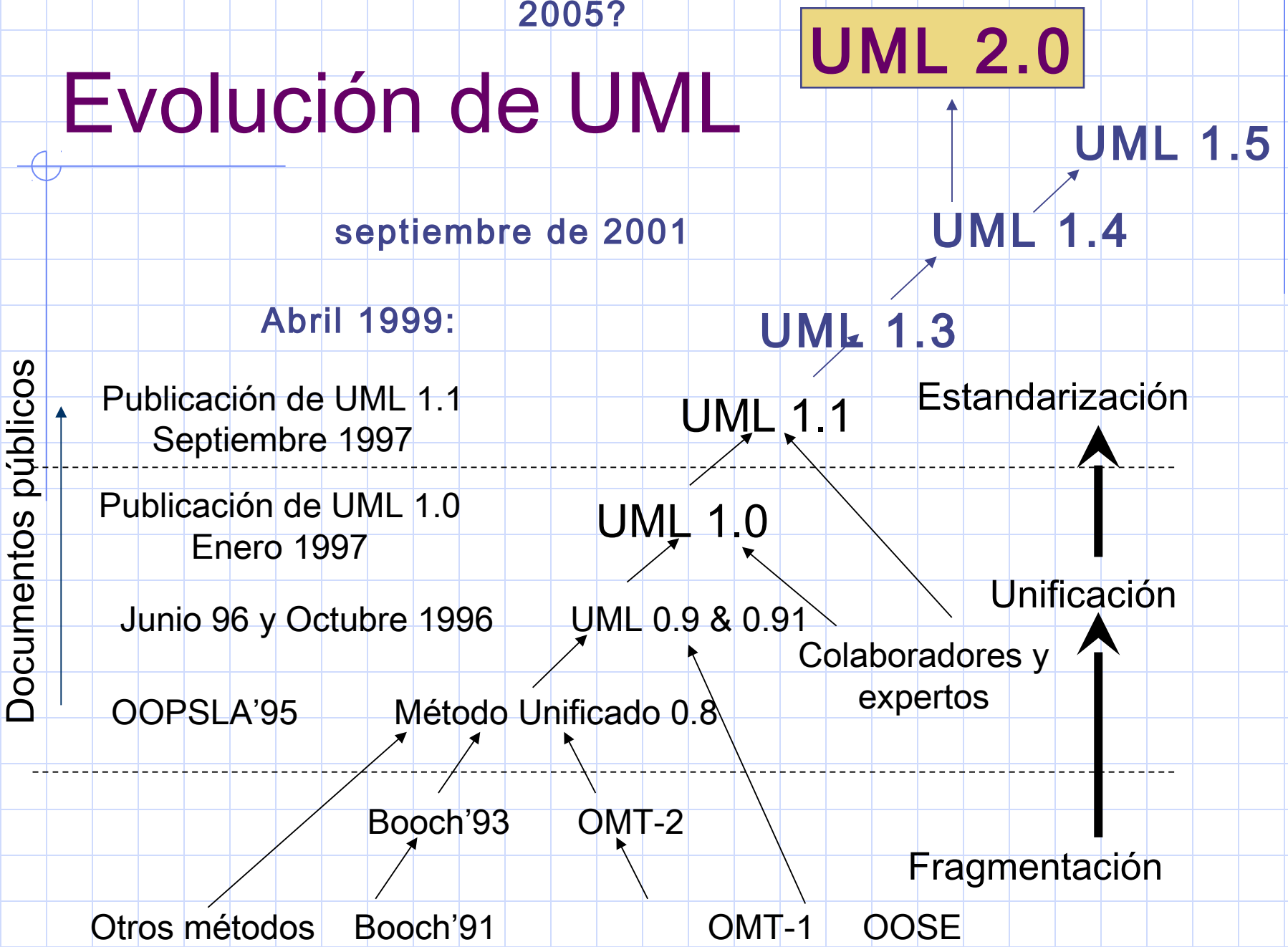


Lenguajes para MDA/MDD



Source: ICSE'03 UML2.0 Tutorial, used with permission of Brian Selic, Rational Software, IBM

Evolución de UML



Críticas a UML 1.X

- excesivo tamaño,
- complejidad gratuita,
- semántica imprecisa,
- personalización limitada,
- Soporte inadecuado para desarrollos basados en componentes,
- implementaciones no estándar
- falta de soporte para intercambio de diagramas.

Qué ha ido mal en UML 1.X

- Does not fully exploit MDD potential of models,
  - E.g., “C++ in pictures”
- Inadequate modeling capabilities
  - Business and similar processes modeling
  - Large-scale systems
  - Non-functional aspects (quality of service specifications)
- Too complex
  - Too many concepts
  - Overlapping concepts
- Inadequate semantics definition
  - Vague or missing (e.g., inheritance, dynamic semantics)
  - Informal definition (not suitable for code generation or executable models)
- No diagram interchange capability
- Not fully aligned with MOF
  - Leads to model interchange problems (XMI)

Requisitos para UML 2.0

- Requisitos de la infraestructura:
  - se refieren a la arquitectura, reestructuración y mecanismos de extensión.
  - Indican cómo UML 2.0 es definido y estructurado como un metamodelo.
- Requisitos de la superestructura:
  - se refieren al refinamiento y extensión de la notación y la semántica de UML 1.x.
- Requisitos generales:
  - afectan tanto a los cambios en la infraestructura como a los de la superestructura.

Qué se le pide a UML 2.0

- Se ha dividido la petición en varios RPF (peticiones de propuestas)

UML 2.0 RPF

- "UML 2.0 Infrastructure RFP". Documento OMG ad/2000-09-01.
  - UML interno
  - base conceptual precisa para soporte de MDA
- "UML 2.0 Superstructure RFP". Documento OMG ad/2000-09-02.
  - Características para el usuario
  - Capacidades nuevas para sistemas grandes
  - Consolidación

...UML 2.0 RPF

- "UML 2.0 OCL RPF". Documento OMG ad/2000-09-03.
  - Lenguaje de restricciones
  - Alineamiento con UML
- "UML 2.0 Diagram Interchange RFP". Documento OMG ad/2001-02-39.
  - Estándar de intercambio de diagramas
  - Incluye información gráfica

UML 2.0 Infrastructure RFP

- Solicitaba propuestas que mejorasen las bases arquitectónicas de UML, incluyendo su núcleo y sus mecanismos de extensión:
  - - Mejorar la alineación arquitectónica con otros estándares de modelado del OMG, como MOF (Meta Object Facility) y XMI (XML Metadata Interchange).
  - - Reestructurar la arquitectura del lenguaje, para que fuera más sencillo de entender, implementar y extender, manteniendo la semántica que ya había sido contrastada.
  - - Proporcionar perfiles y mecanismos de extensión de primera clase (metaclases) que fueran consistentes con la arquitectura del metamodelo.

UML 2.0 Superstructure RFP

- Solicitaba propuestas que actualizasen y mejorasen el soporte que UML proporciona al desarrollo del software, en áreas tales como
  - desarrollo basado en componentes,
  - modelado de procesos de negocios,
  - modelado arquitectónico modelos ejecutables
- Requería la revisión de ciertos aspectos estructurales y de comportamiento.

Componentes y arquitectura

- Mejorar el soporte para desarrollos basados en componentes. Era necesario demostrar que se podían especificar contenedores de ejecución y perfiles para las principales arquitecturas de componentes, como EJB y COM+
- Aumentar el soporte para arquitecturas de tiempo de ejecución (comparar modelos ejecutables) incluyendo la especificación de estructuras jerárquicas y comportamientos dinámicos.

Revisión de ciertos aspectos...

- Refinar la semántica de las relaciones, incluyendo generalización, asociación y dependencia.
- Mejorar el encapsulamiento y la escalabilidad en los modelos de comportamiento, especialmente en los diagramas de estado y en los diagramas de interacción.
- Refinar la semántica gráfica de las actividades, centrándose en la gestión de eventos y el flujo de control y de objetos.

UML 2.0 OCL RFP

- Solicitaba propuestas que definiesen un metamodelo de Lenguaje de Restricciones de Objetos (OCL) acorde al metamodelo de UML.
- Esto incrementaría la precisión y consistencia de las implementaciones OCL y facilitaría el intercambio de constructores OCL entre distintas herramientas.
- Aunque se la Infraestructura como la Superestructura utilizan OCL para especificar sus reglas, ninguno de sus respectivos RFP dependen de éste.

UML 2.0 Diagram Interchange RFP

- Solicitaba propuestas que definieran un metamodelo para el intercambio de elementos de diagramas entre herramientas UML.
- Este metamodelo necesitaría soportar el intercambio de características tales como la posición de los elementos, el agrupamiento de elementos, la alineación de elementos, las configuraciones de las fuentes, los caracteres y los colores.

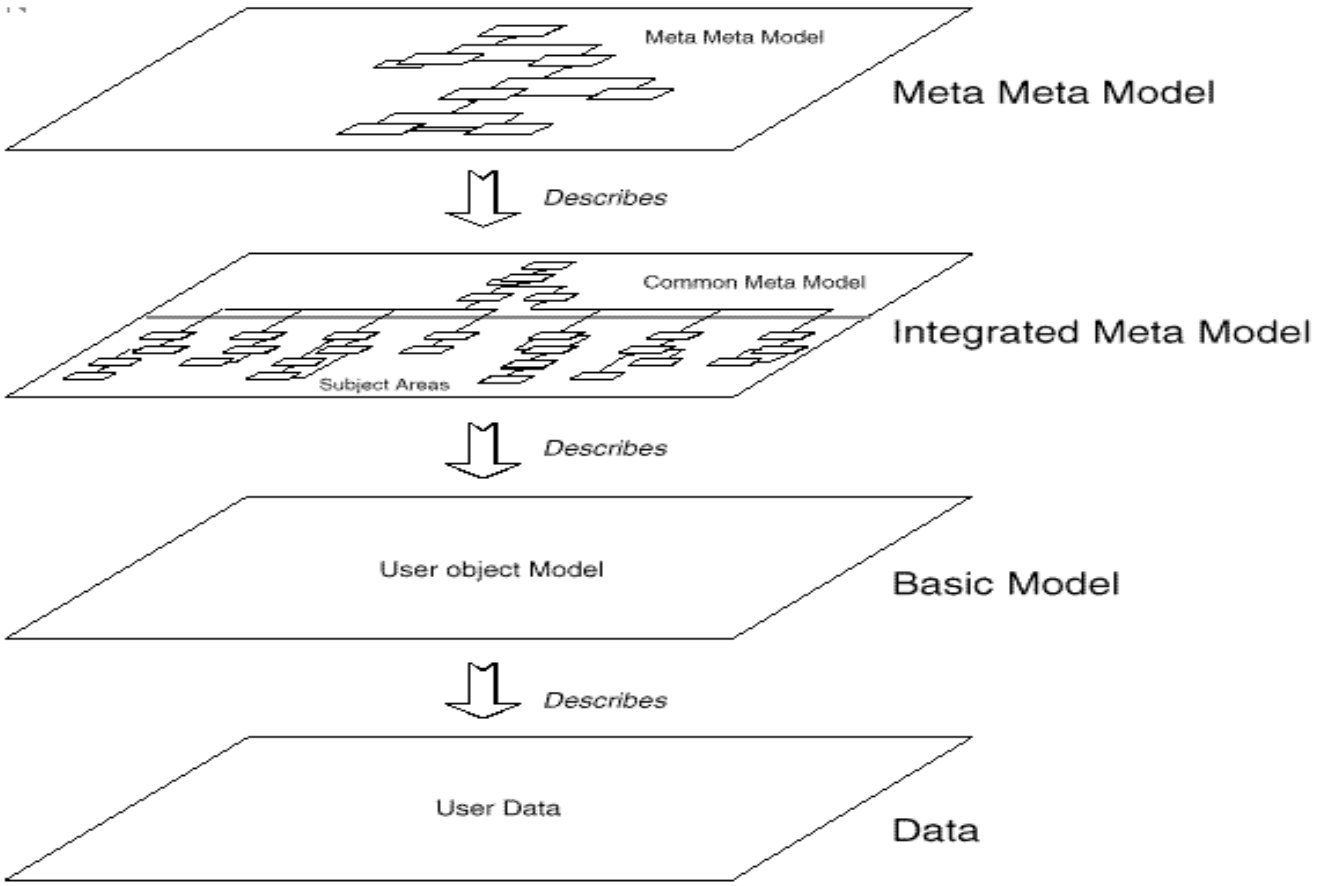
Facilidad Meta-Objetos (MOF)

- MOF, Meta-Object Facility es un lenguaje para definir lenguajes de modelado
  - Permite a los usuarios definir totalmente nuevos lenguajes a partir de metamodelos
- Fue también definido por el OMG y actualmente se encuentra en su versión 2.0
- La alineación del metamodelo UML 2.0 con el metamodelo MOF simplificará el intercambio de modelos vía XMI y la interoperabilidad cruzada entre herramientas.
- La especificación del núcleo unificado MOF 2.0 debe estar arquitectónicamente alineada con la Infraestructura de UML

Arquitectura de Lenguajes de Modelado

- MOF define una Arquitectura de Lenguajes de Modelado en la que existen 4 capas o niveles:
  - – Nivel M3: MOF.
  - – Nivel M2: UML.
  - – Nivel M1: Modelo del usuario.
  - – Nivel M0: Instancias en tiempo de ejecución.

Arquitectura de UML/MOF



Situación actual: finalización

- UML 2.0 Infrastructure RFP: adoptado en agosto de 2003 la especificación final
- UML 2.0 Superstructure RFP: adoptada en agosto de 2003 la especificación final
- UML 2.0 OCL RFP: adoptado en agosto de 2003 el borrador de la especificación,
- UML 2.0 Diagram Interchange RFP: adoptado en julio de 2003 el borrador de la especificación,

Se aprobó en agosto de 2005

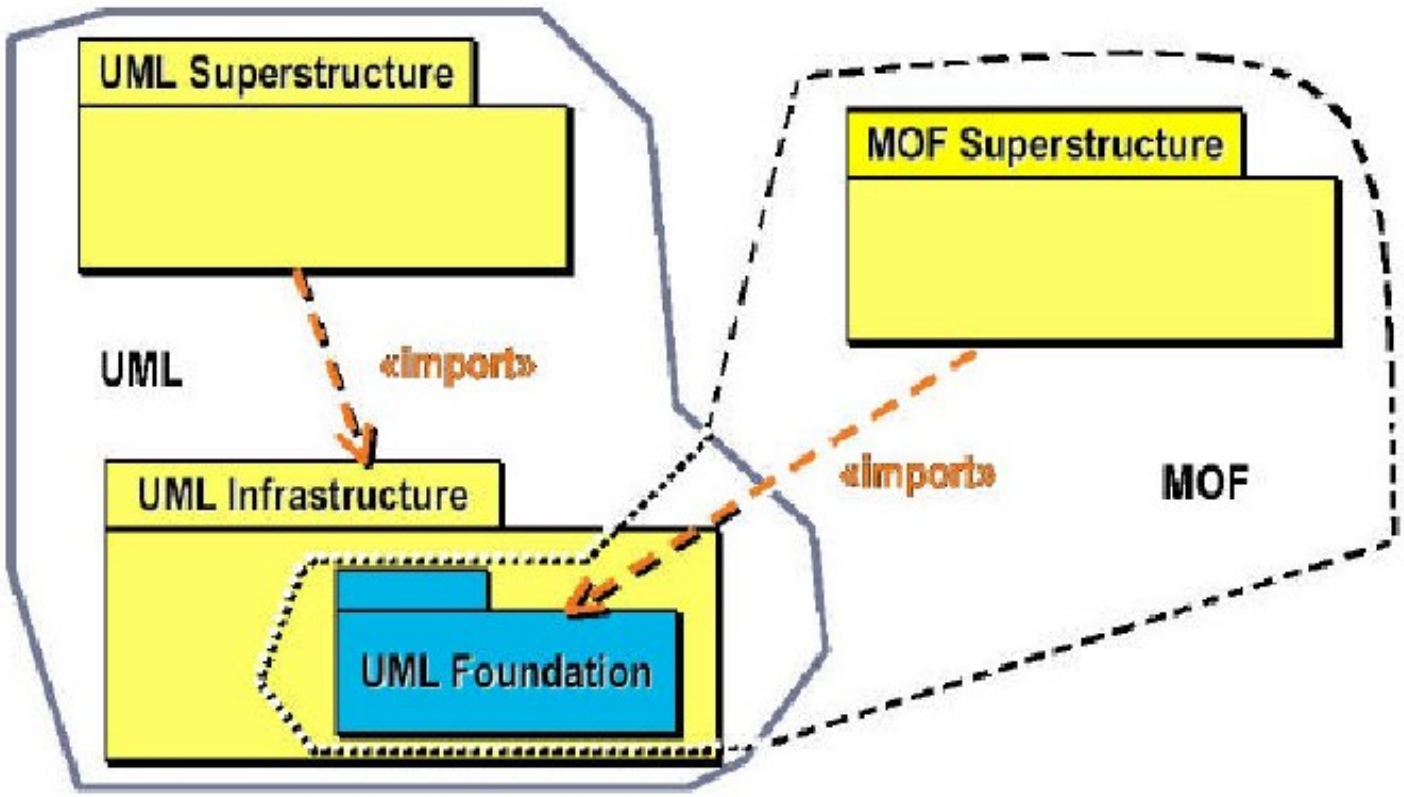
Infraestructura

- a) Alineación arquitectónica y reestructuración
- b) Extensibilidad

a) Alineación arquitectónica y reestructuración

- Aunque el metamodelo UML 1.x era compatible con el metamodelo MOF, no se ceñía estrictamente al patrón de metamodelo de 4 niveles, en el que cada metamodelo es una instancia de sólo un meta-metamodelo
- En UML 2.0 el metamodelo UML está perfectamente alineado con el metamodelo MOF
- Además, el núcleo de UML y el núcleo de MOF deben compartir los mismos elementos de metamodelo,

UML 2.0 y MOF 2.0



b) Extensibilidad

- Los perfiles UML incorporan mecanismos de extensión (estereotipos, valores etiquetados y restricciones) que permiten personalizarlo para distintas aplicaciones y tecnologías.
  - En el OMG se está trabajando con ellos, algunos ya han sido adoptados y otros están en proceso de adopción.
  - Por ejemplo existen perfiles para: CORBA IDL, Modelo de Componentes CORBA (CCM), Computación de Empresa de Objetos Distribuidos (EDOC).
- Se ha incluido un mecanismo de extensibilidad de primera clase, que permite a los desarrolladores definir y añadir sus propias metaclases (que serán instancias de las meta-metaclases MOF), dando así soporte a la "familia de lenguajes“ basada en UML.

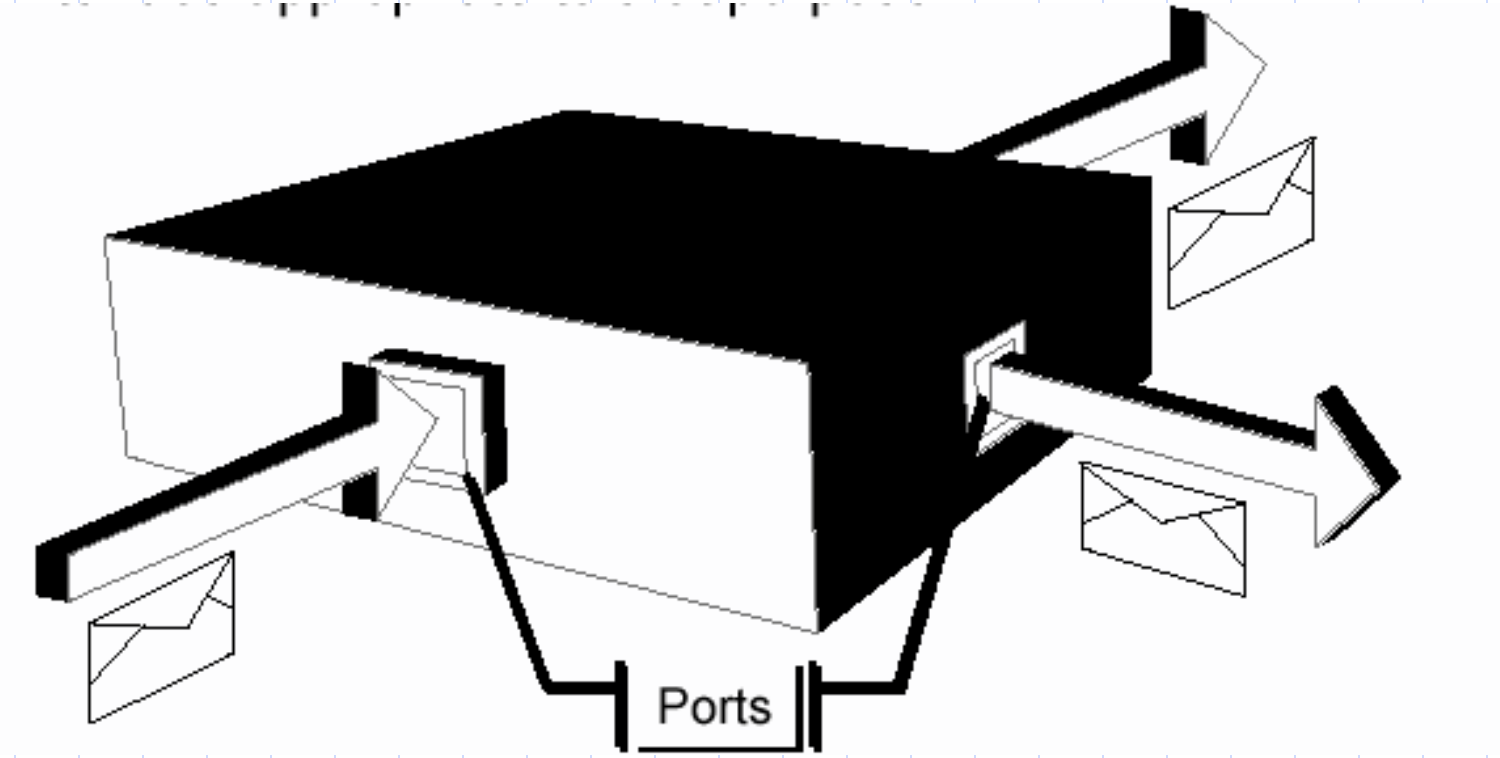
Superestructura

- Pensada para el modelado arquitectónico
  - Objetos con estructura externa e interna (objetos arquitectónicos)
  - Modelado de sistemas complejos
- La estructura deseada es declarada (asserted) más que construida
  - Constructor de clase crea la estructura deseada automáticamente
  - El destructor de la clase hace la limpieza automáticamente
  - Expresividad, fiabilidad y productividad
- Basado en lenguajes de descripción arquitectónica (ADLs)
  - UML-RT profile: Selic and Rumbaugh (1998)
  - ACME: Garlan et al.
  - SDL (ITU-T standard Z.100)

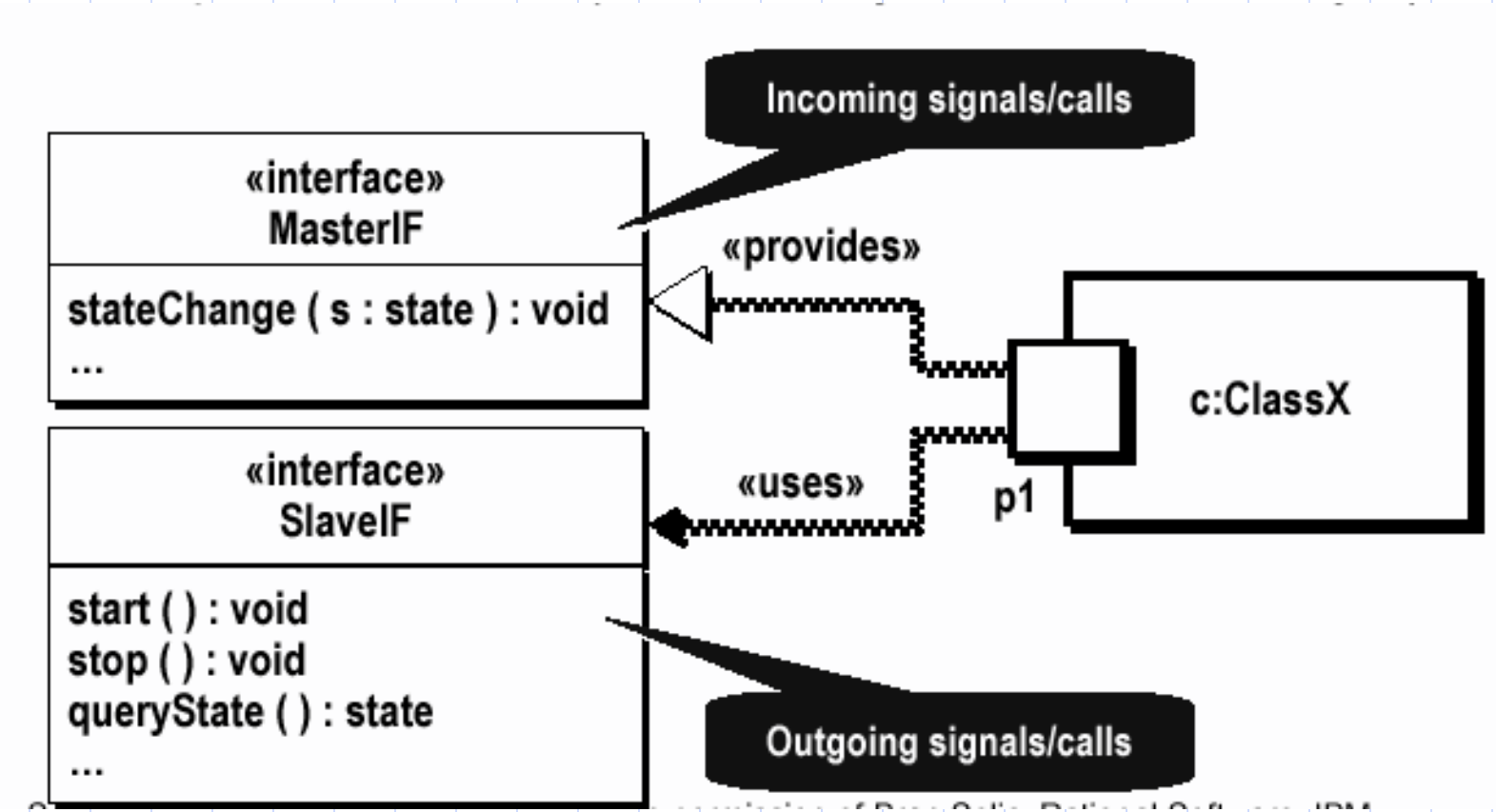
Nuevos elementos

- Clases estructuradas
- Puertos
- Protocolos
- Componentes
- ...

Clases estructuradas



Puertos


Incoming signals/calls
«interface»
MasterIF
stateChange (s : state) : void
...
«provides»
c:ClassX
«interface»
SlaveIF
«uses»
p1
start ( ) : void
stop ( ) : void
queryState ( ) : state
...
Outgoing signals/calls

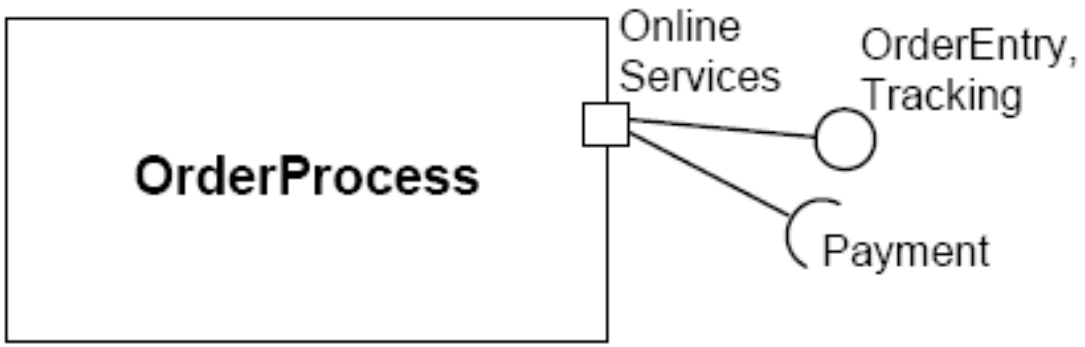


Figure 9.19 - Port notation showing multiple provided interfaces

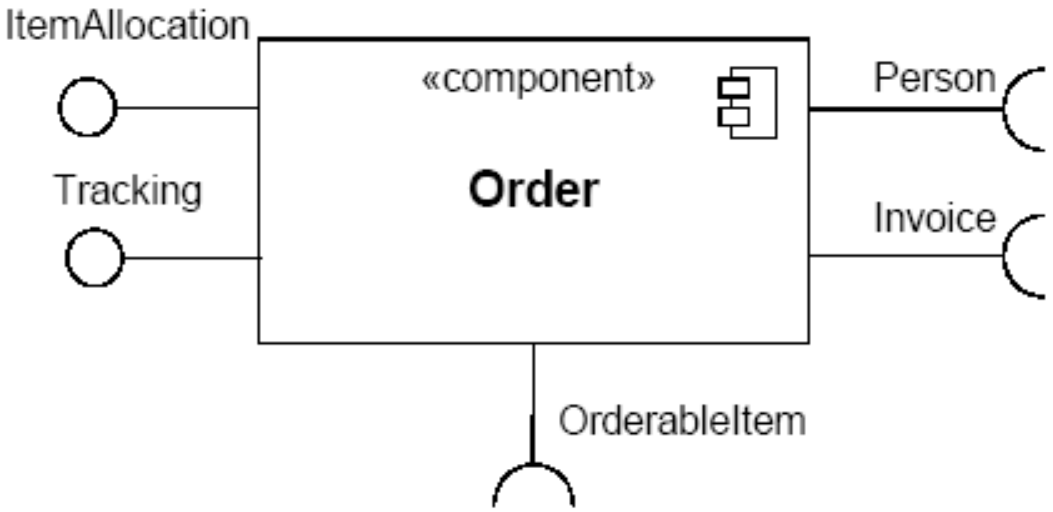
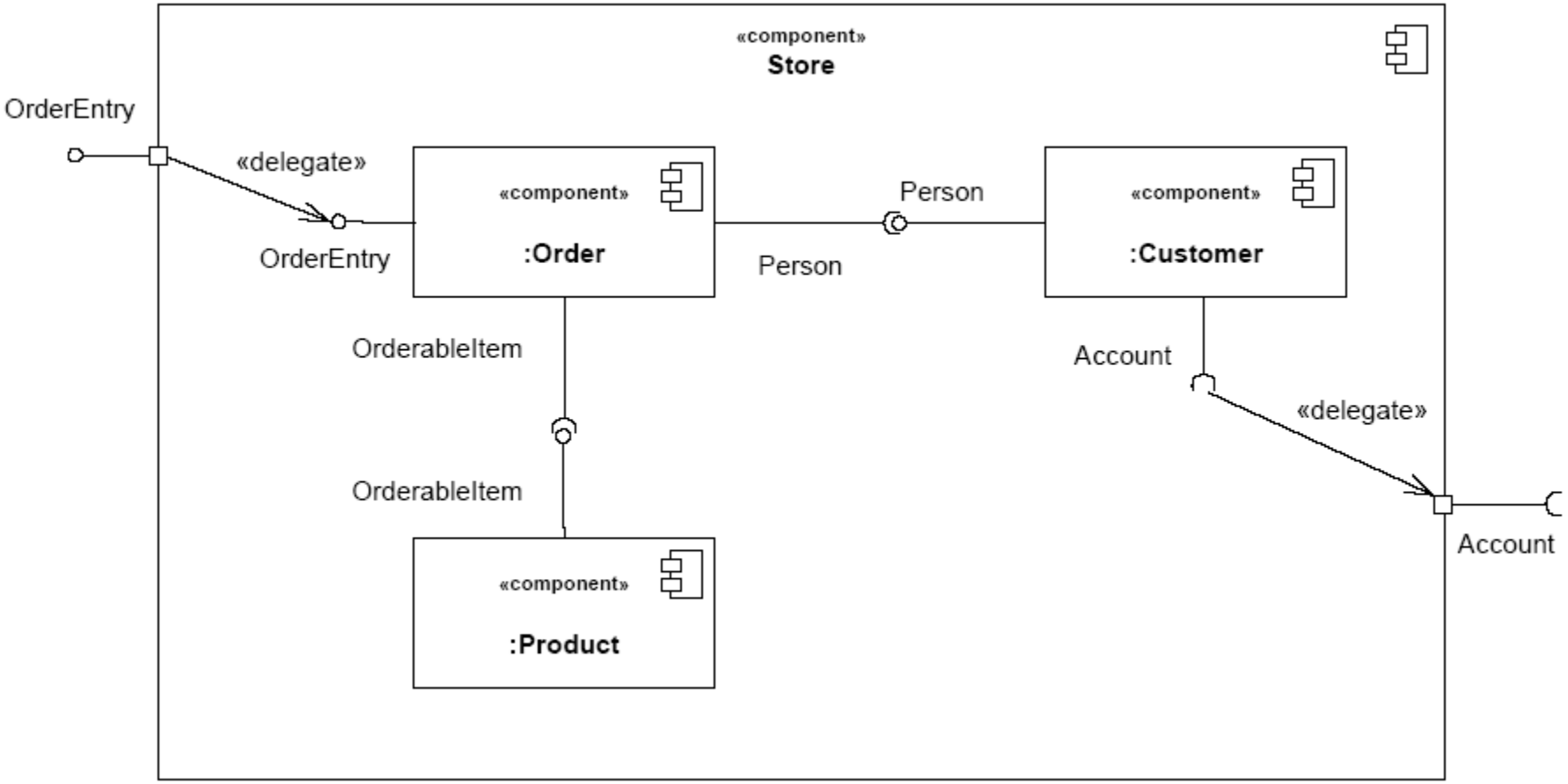
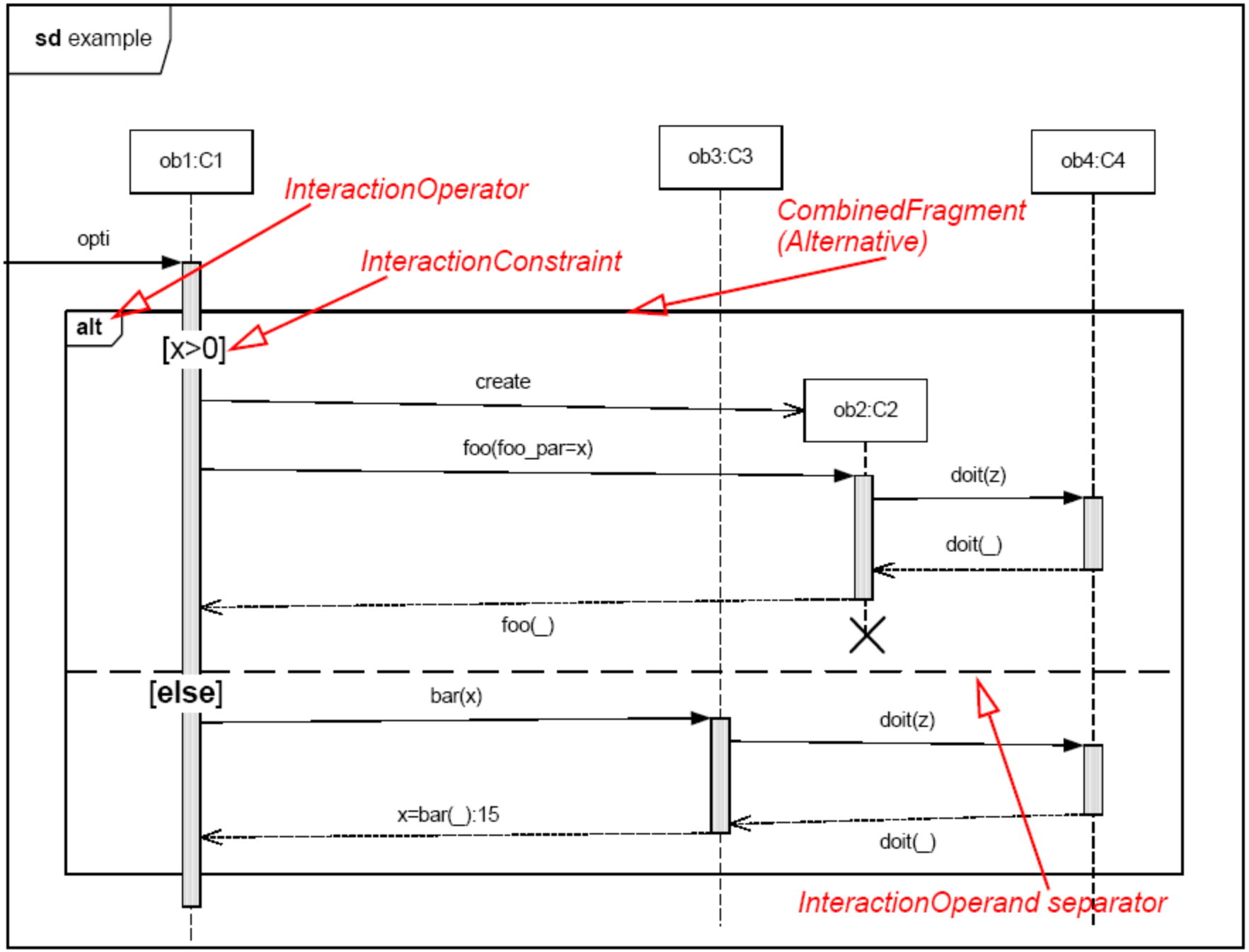
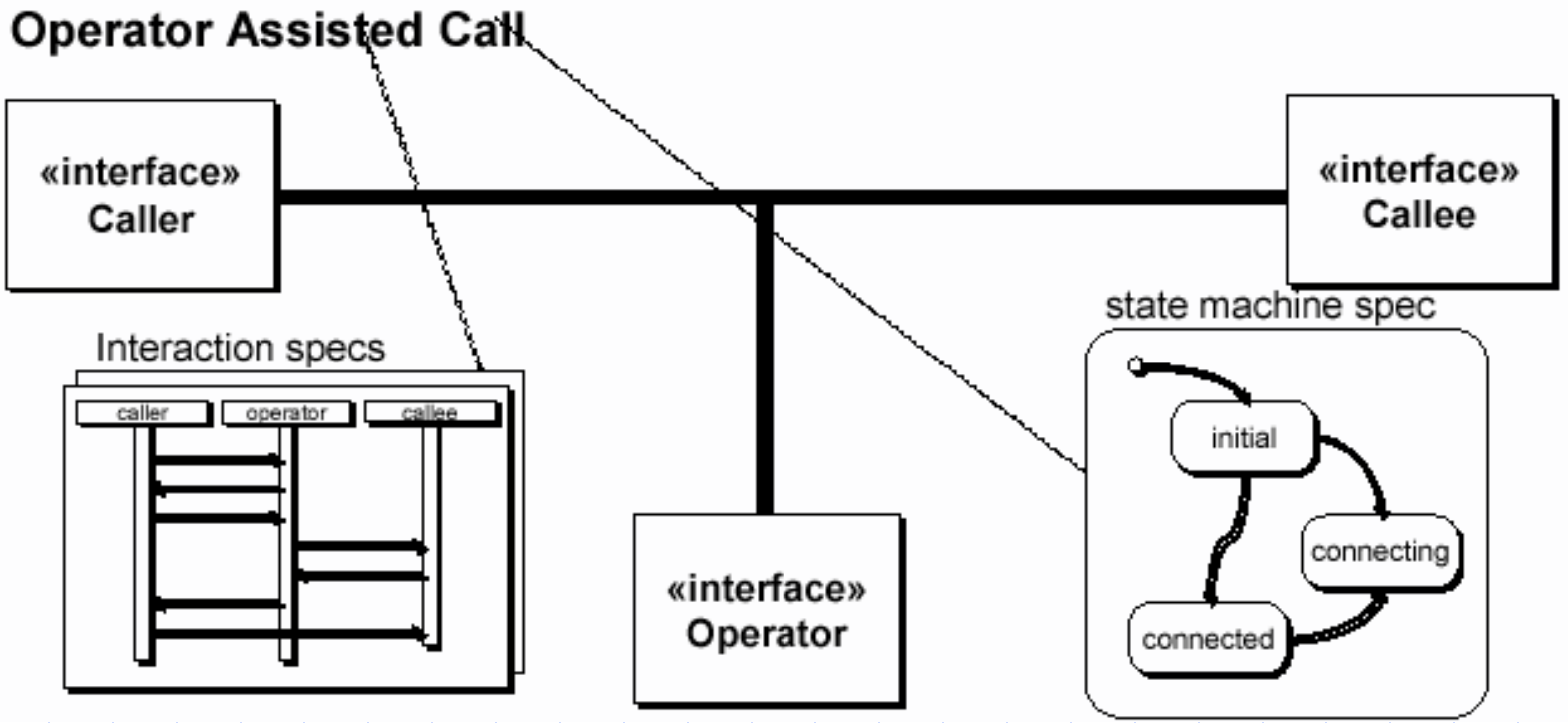


Figure 8.6 - A Component with two provided and three required interfaces

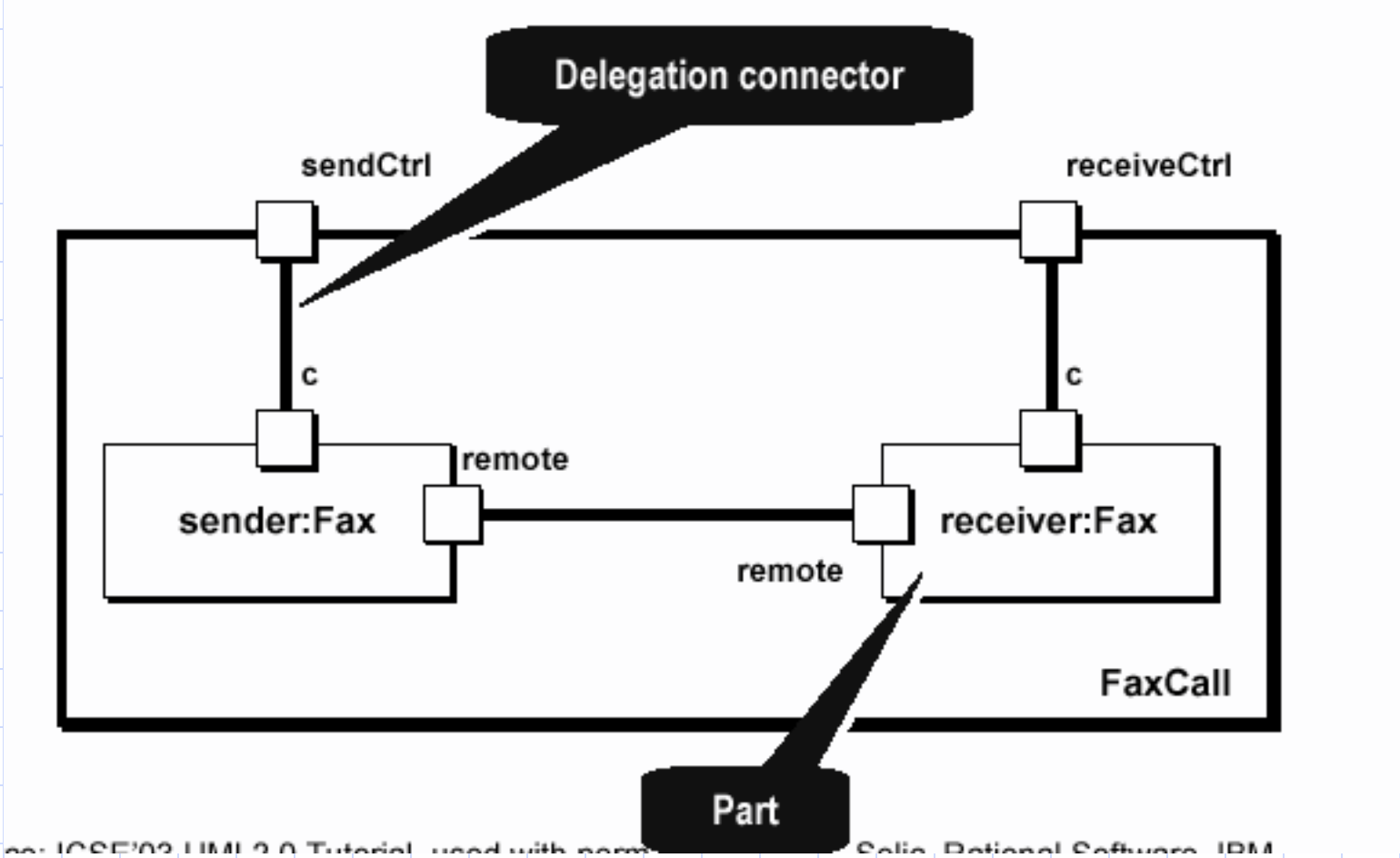

«component»
Store
OrderEntry
«delegate»
«component»
:Order
OrderEntry
Person
Person
«component»
:Customer
OrderableItem
Account
«delegate»
OrderableItem
Account
«component»
:Product


sd example
ob1:C1
ob3:C3
ob4:C4
InteractionOperator
CombinedFragment
(Alternative)
opti
InteractionConstraint
alt
[x>0]
create
ob2:C2
foo(foo_par=x)
doit(z)
doit(_)
foo(_)
[else]
bar(x)
doit(z)
x=bar(_):15
doit(_)
InteractionOperand separator

Protocolos


Operator Assisted Call
«interface»
Caller
«interface»
Callee
Interaction specs
caller
operator
callee
«interface»
Operator
state machine spec
initial
connecting
connected

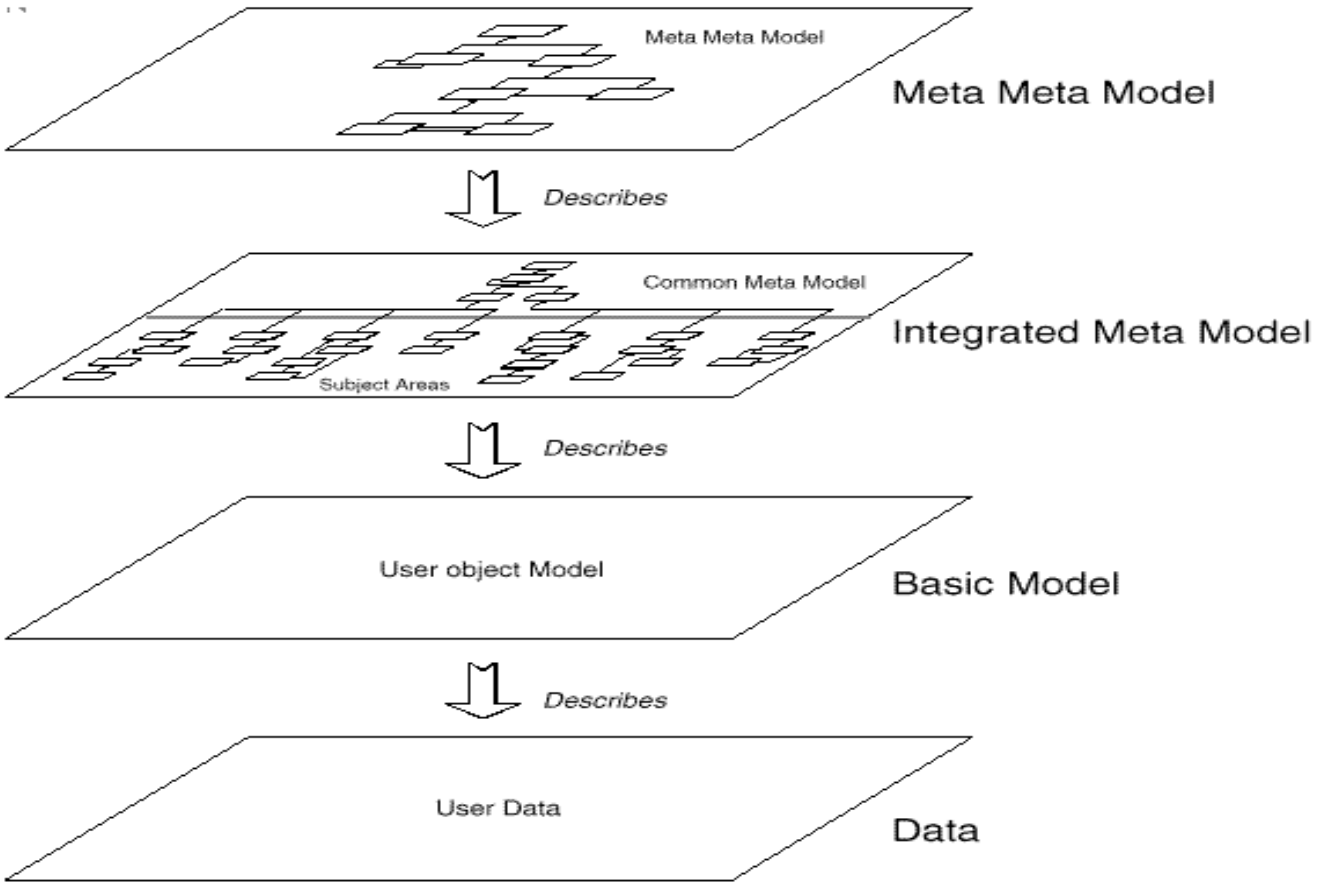
Componentes



Sumario de UML 2.0

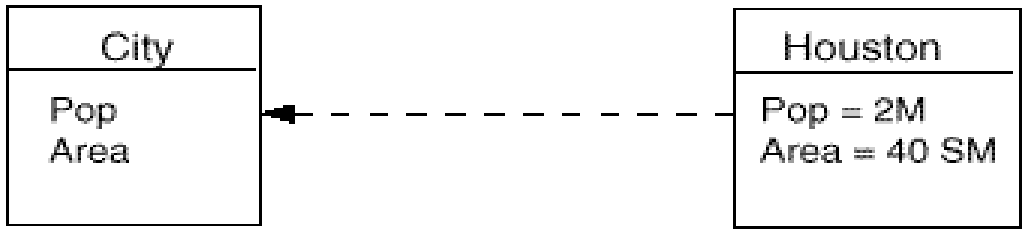
- First major revision of UML
- Original standard had to be adjusted to deal with
  - MDD requirements (precision, code generation, executability)
- UML 2.0:
  - Small number of new features + consolidation of existing ones
  - Scaleable to large software systems (architectural modeling
  - Modular structure for easier adoption (core + optional)
  - Increased semantic precision and conceptual clarity
  - Suitable foundation for MDA (executable models, full code generation)

Arquitectura de UML/MOF



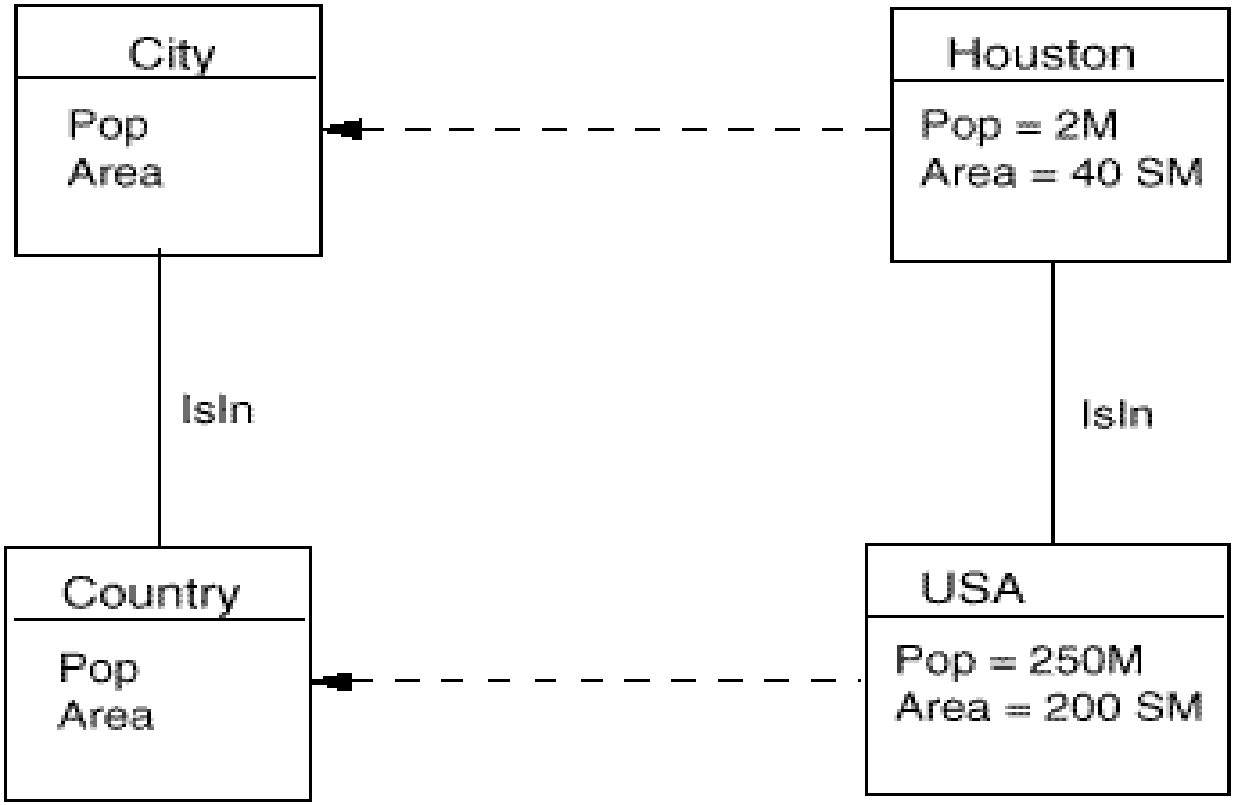
Modelado de objetos

- Descripción de los objetos en términos de sus propiedades y de sus relaciones
- Idea básica: describir un grupo de objetos similares en términos de clases, que son instanciadas para crear objetos individuales
- Los objetos se relacionan con las clases de las que son creados por la relación "**SerInstanciaDe**" ("*IsInstanceOf*")



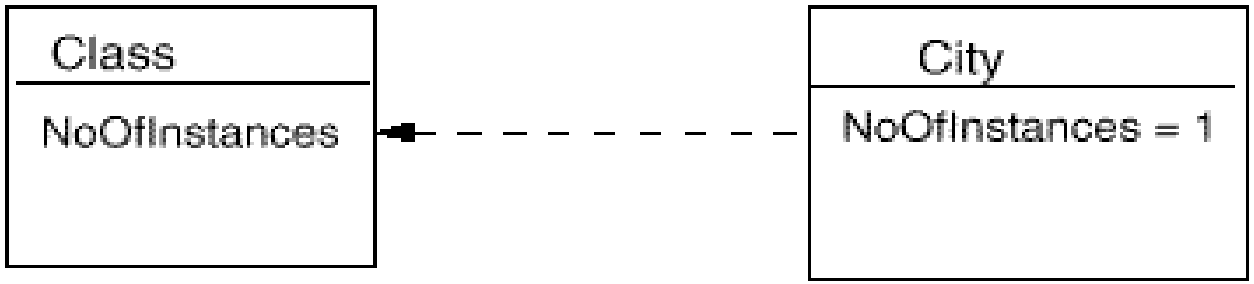
...Modelado de objetos

- Una situación parecida ocurre con las relaciones. Una clase define los tipos de relaciones que sus instancias pueden tener con instancias de otras clases



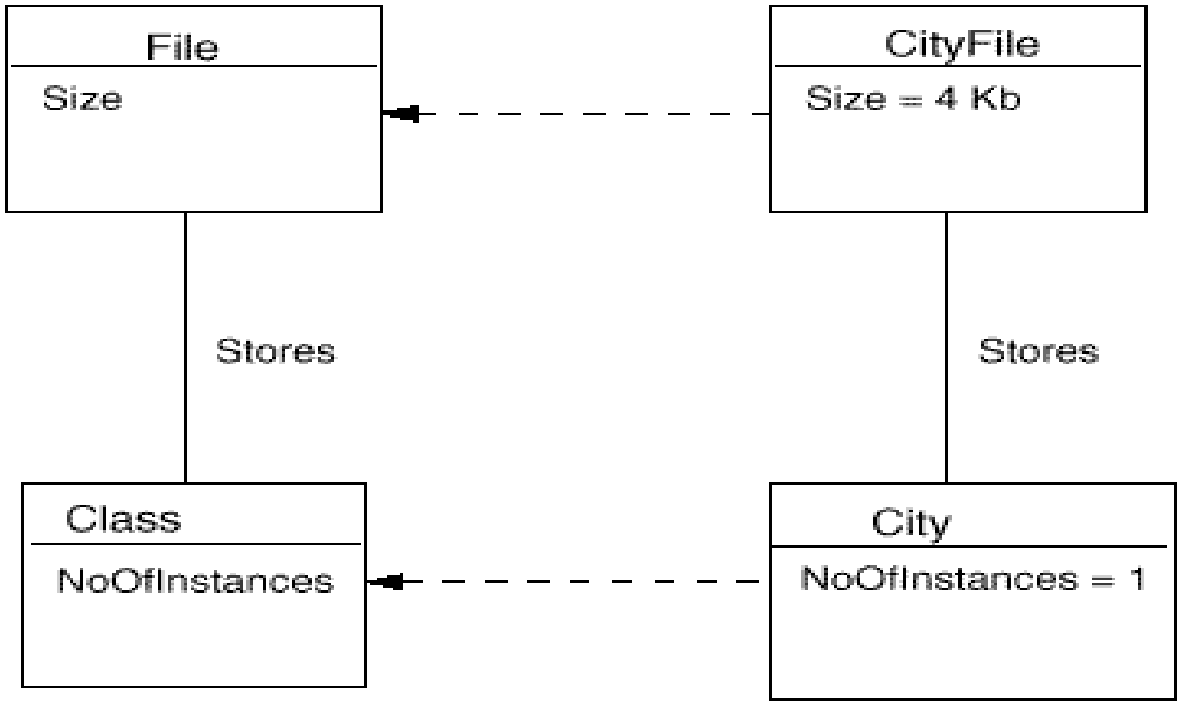
Metamodelado...

- Se basa en la idea de reificar las entidades que forman un cierto tipo de modelo y describir las propiedades comunes del tipo de modelo en forma de un modelo de objetos
- Cuando se ve la clase como un objeto, la clase es una instancia de otra clase



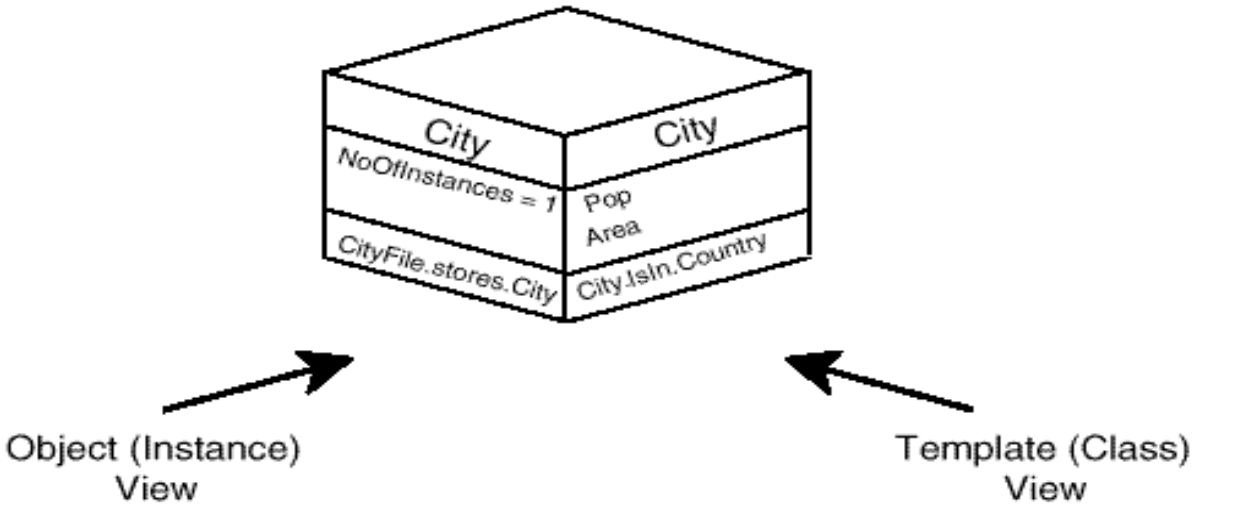
Metamodelado...

- Las clases pueden participar en relaciones con otros objetos



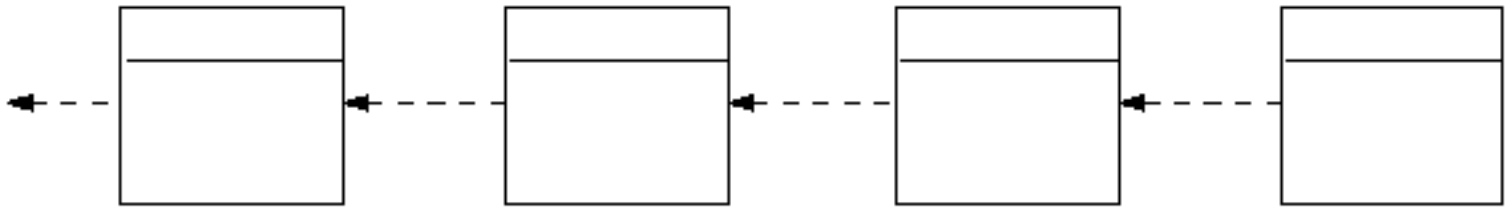
...Metamodelado

- La idea fundamental en el metamodelado es que las entidades del modelo (**clases**) juegan dos papeles: el papel de plantilla (cuando se ve como una clase) y el papel de instancia (cuando se ve como objeto



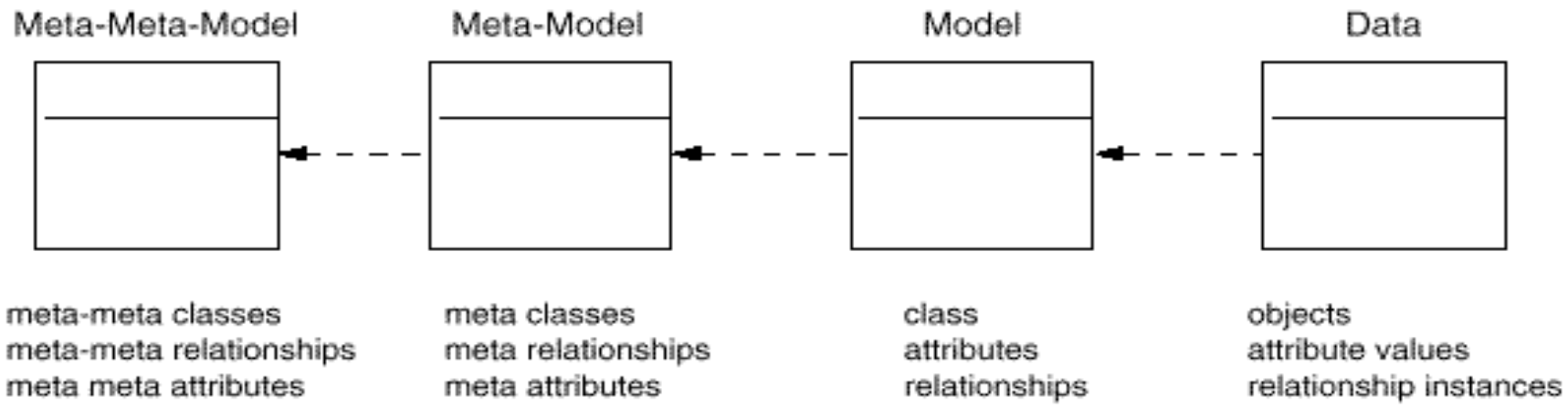
Terminología de metamodelado...

- La idea de ver las clases como objetos puede ser aplicada repetidamente para crear una jerarquía de instanciación del clases y objetos
- En principio está jerarquía podría continuar hasta cualquier profundidad arbitraria, pero en la práctica no se extiende más allá de cuatro niveles de profundidad



Terminología de metamodelado...

- Si la jerarquía tiene una profundidad fijada, se puede utilizar un esquema de nombres para describir el nivel en que reside una entidad dada en la jerarquía de instanciación



...Terminología de metamodelado