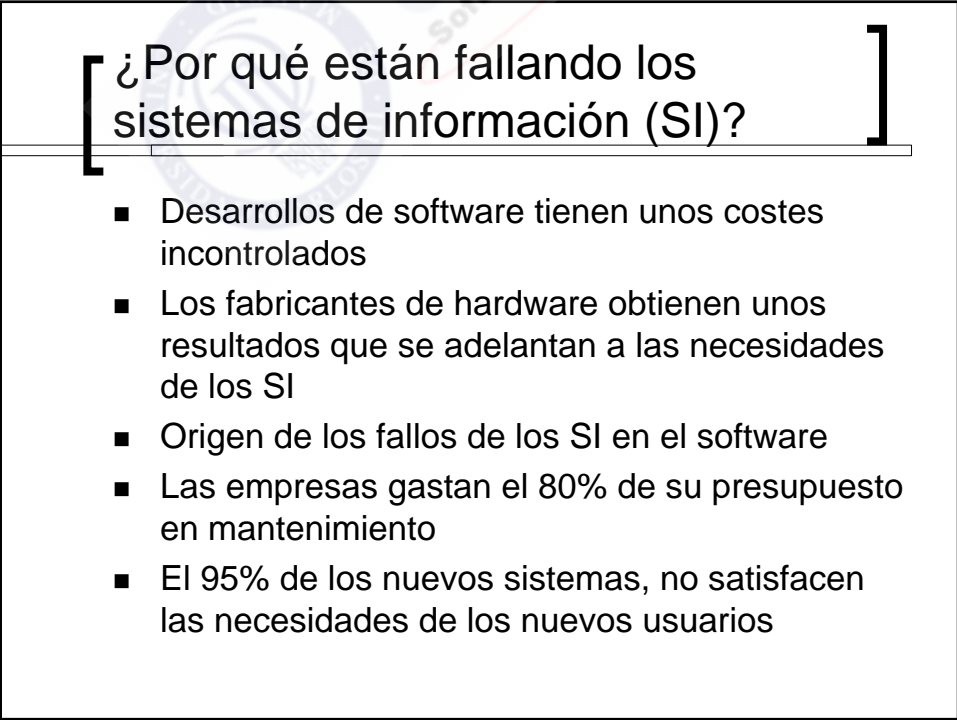


[INTRODUCCIÓN AL PARADIGMA ORIENTADO A OBJETOS]



[¿Por qué están fallando los sistemas de información (SI)?]

- Desarrollos de software tienen unos costes incontrolados
- Los fabricantes de hardware obtienen unos resultados que se adelantan a las necesidades de los SI
- Origen de los fallos de los SI en el software
- Las empresas gastan el 80% de su presupuesto en mantenimiento
- El 95% de los nuevos sistemas, no satisfacen las necesidades de los nuevos usuarios

¿Por qué están fallando los sistemas de información (SI)?(II)

- Los sistemas generados son demasiado complejos e inflexibles para adaptarse a las crecientes necesidades de las empresas
- Diversos estudios han demostrado que un proyecto de desarrollo de software típico:
 - rebasa en un 50% las fechas previstas
 - por cada 6 que entran en fase de producción, dos son cancelados
- Además un estudio realizado en 25 compañías norteamericanas:
 - el 55% de los proyectos software cuestan más de lo previsto
 - el 88% han de ser rediseñados

La orientación a objetos (OO) como solución (I)

- La orientación a objetos, ofrece la posibilidad de una revolución en la producción de software:
 - SI=ensamblaje de componentes (Objetos)
- Entre los objetivos establecidos por la tecnología OO destacan:
 - Corrección: es la capacidad de satisfacer unas necesidades dadas, es decir, grado en el que se satisfacen las expectativas del software
 - Robustez: es la capacidad por la que el software puede seguir en operación aunque se hayan producido entradas no válidas software

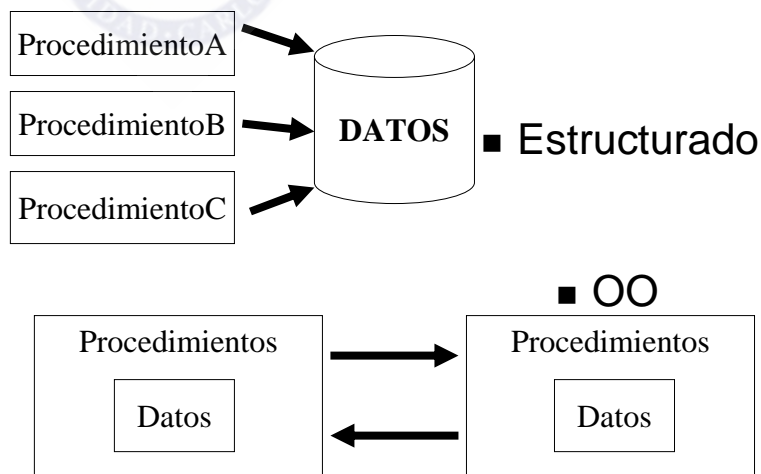
(Definiciones tomadas de la AECC)

La orientación a objetos (OO) como solución (II)

- Reutilización: capacidad por la que un módulo puede utilizarse en múltiples aplicaciones
- Extensibilidad: mejora hecha al software existente para incrementar su alcance y capacidad
- Portabilidad: grado de facilidad con que puede transferirse un software de un sistema o entorno de ordenador a otro
- Facilidad de verificación: facilidad del proceso para determinar si los productos de una fase determinada del ciclo de desarrollo software cumplen o no los requisitos establecidos en la fase previa

(Definiciones tomadas de la AECC)

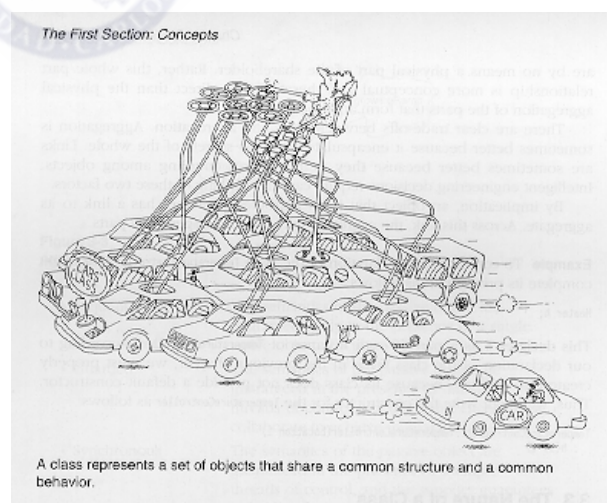
Paradigma de programación estructurado vs. OO



[Elementos básicos de la OO]

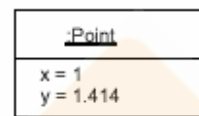
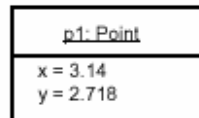
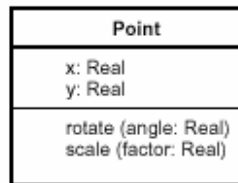
- **Objeto:** cualquier cosa, real o abstracta, acerca de la cual almacenamos datos y los métodos que controlan dichos datos
- **Mensajes:** solicitud para que se lleve a cabo la operación indicada
- **Métodos:** procedimientos que contiene los objetos y que manipulan los datos contenidos en estos
- **Clases:** familia de objetos con las mismas características
- **Herencia:** mecanismo mediante el cual una clase adquiere las propiedades de una clase superior

[CLASE]



Fuente de la imagen: Grady Booch, 2007. Object-Oriented Analysis and Design with Applications (3rd Edition). 3 Edition. Addison-Wesley Professional.

[Clases y Objetos]



[... Objetos]

- Objeto = Identidad + Estado + Comportamiento
- El estado está representado por los valores de los atributos
- Un atributo toma un valor en un dominio concreto

Un coche

Azul
979 Kg
70 CV
...

[Identidad]

- Oid (Object Identifier)

Cada objeto posee un oid. El oid establece la identidad del objeto y tiene las siguientes características:

- Constituye un identificador único y global para cada objeto dentro del sistema
- Es determinado en el momento de la creación del objeto
- Es independiente de la localización física del objeto, es decir, provee completa independencia de localización

[... Identidad]

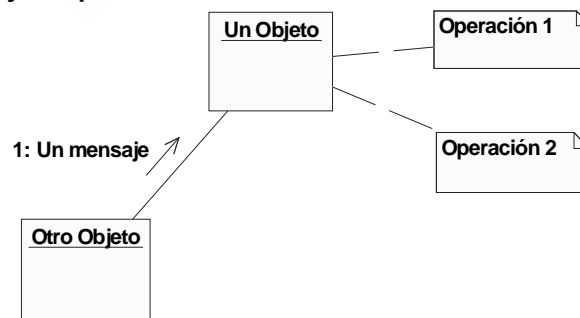
- Es independiente de las propiedades del objeto, lo cual implica independencia de valor y de estructura
- No cambia durante toda la vida del objeto. Además, un oid no se reutiliza aunque el objeto deje de existir
- No se tiene ningún control sobre los oids y su manipulación resulta transparente
- Sin embargo, es preciso contar con algún medio para hacer referencia a un objeto utilizando referencias del dominio (valores de atributos)

[Estado]

- El estado evoluciona con el tiempo
- Algunos atributos pueden ser constantes
- El comportamiento agrupa las competencias de un objeto y describe las acciones y reacciones de ese objeto
- Las operaciones de un objeto son consecuencia de un estímulo externo representado como mensaje enviado desde otro objeto

[Comportamiento]

- Ejemplo de interacción:



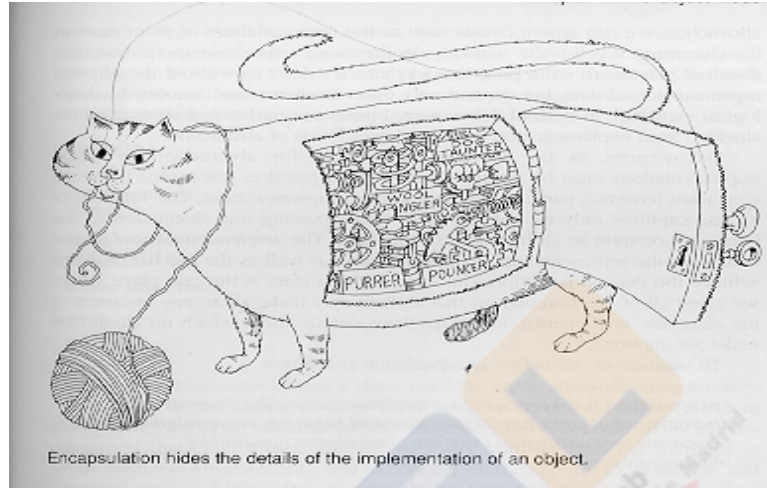
[... Comportamiento]

- Los mensajes navegan por los enlaces, a priori en ambas direcciones
- Estado y comportamiento están relacionados
- Ejemplo: no es posible aterrizar un avión si no está volando. Está volando como consecuencia de haber despegado del suelo

[Propiedades fundamentales]

- Encapsulación: ocultación de la información
- Abstracción: principio de ignorar aquellos aspectos que no son relevantes al propósito de lo que se está realizando
- Persistencia: característica que se refiere a la permanencia del objeto en memoria
- Polimorfismo: propiedad por la cual una misma función puede encontrarse en diferentes clases teniendo distintos parámetros y realizando distintas acciones

Propiedades fundamentales : encapsulación

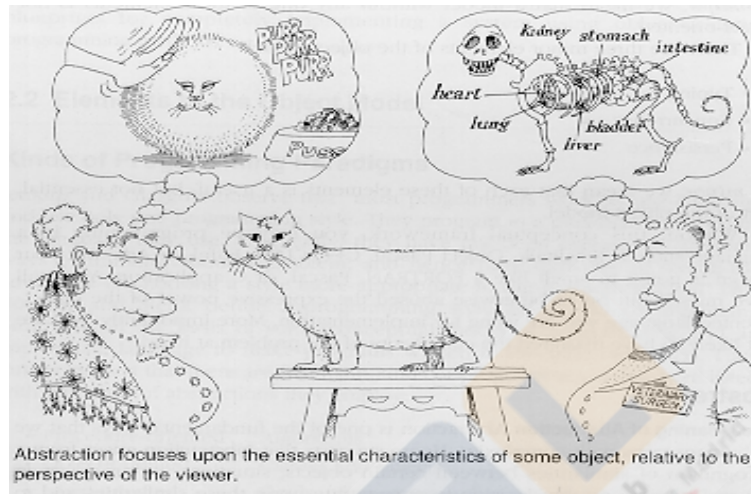


Fuente de la imagen: Grady Booch, 2007. Object-Oriented Analysis and Design with Applications (3rd Edition). 3 Edition. Addison-Wesley Professional.

Ventajas e inconvenientes

- **Ventajas**
 - Modelos
 - Modularidad
 - Extensibilidad
 - Eliminación de redundancias
 - Reutilización
- **Inconvenientes**
 - Dificultad de aplicación de una tecnología
 - Necesidad de mayor preparación del personal

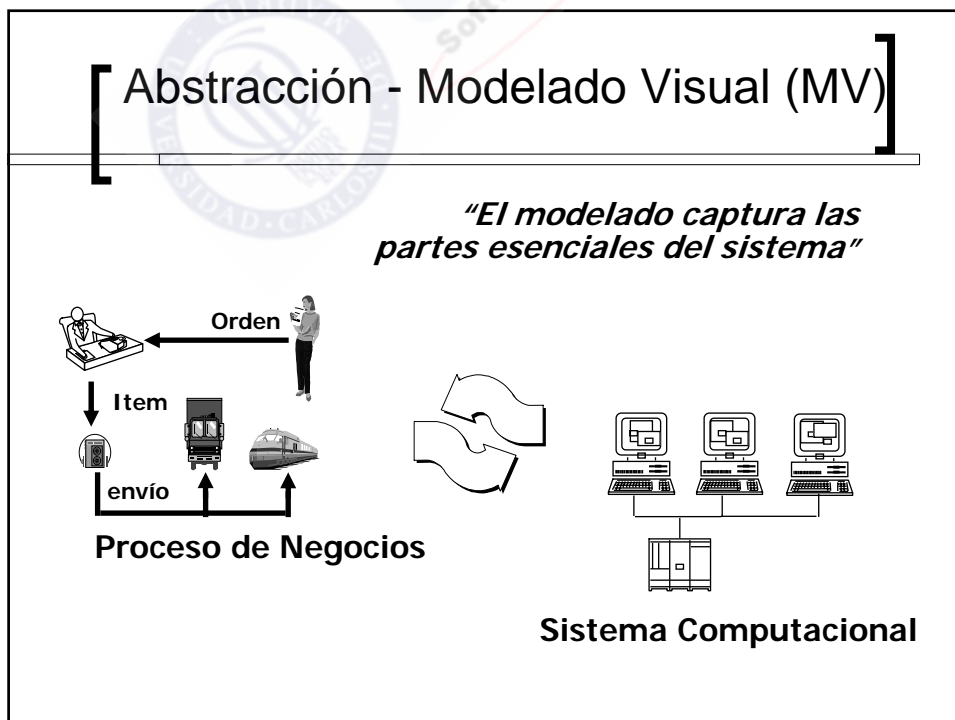
Propiedades fundamentales: abstracción



Fuente de la imagen: Grady Booch, 2007. Object-Oriented Analysis and Design with Applications (3rd Edition). 3 Edition. Addison-Wesley Professional.

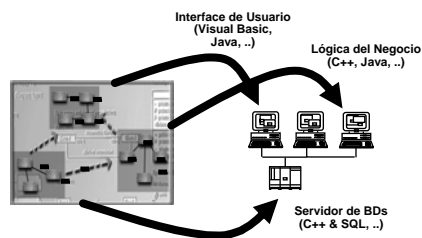
Persistencia

- La persistencia de los objetos designa la capacidad de un objeto trascender en el espacio/tiempo
- Podremos después reconstruirlo, es decir, cogerlo de memoria secundaria para utilizarlo en la ejecución (materialización del objeto)
- Los lenguajes OO no proponen soporte adecuado para la persistencia, la cual debería ser transparente, un objeto existe desde su creación hasta que se destruya

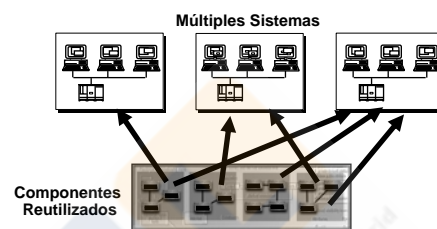


II. Notación (Visual) - Beneficios

Manejar la complejidad



“Modelar el sistema independientemente del lenguaje de implementación”



Promover la Reutilización

Modelos y Diagramas

- Un **modelo** captura una vista de un sistema del mundo real. Es una abstracción de dicho sistema, considerando un cierto propósito. Así, el modelo describe completamente aquellos aspectos del sistema que son relevantes al propósito del modelo, y a un apropiado nivel de detalle.
- **Diagrama**: una representación gráfica de una colección de elementos de modelado, a menudo dibujada como un grafo con vértices conectados por arcos

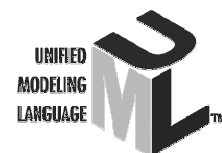
[... Modelos y Diagramas]

- Un proceso de desarrollo de software debe ofrecer un conjunto de modelos que permitan expresar el producto desde cada una de las perspectivas de interés
- El código fuente del sistema es el modelo más detallado del sistema (y además es ejecutable). Sin embargo, se requieren otros modelos ...



- Cada modelo es completo desde su punto de vista del sistema, sin embargo, existen relaciones de trazabilidad entre los diferentes modelos

[Lenguaje de Modelado UML]



[UML]

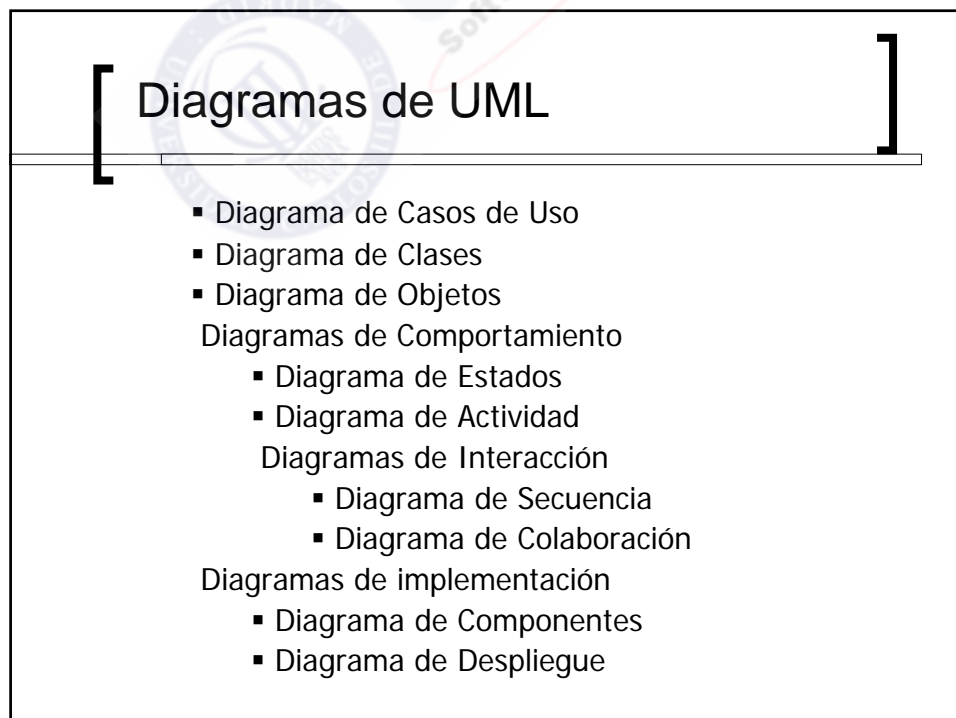
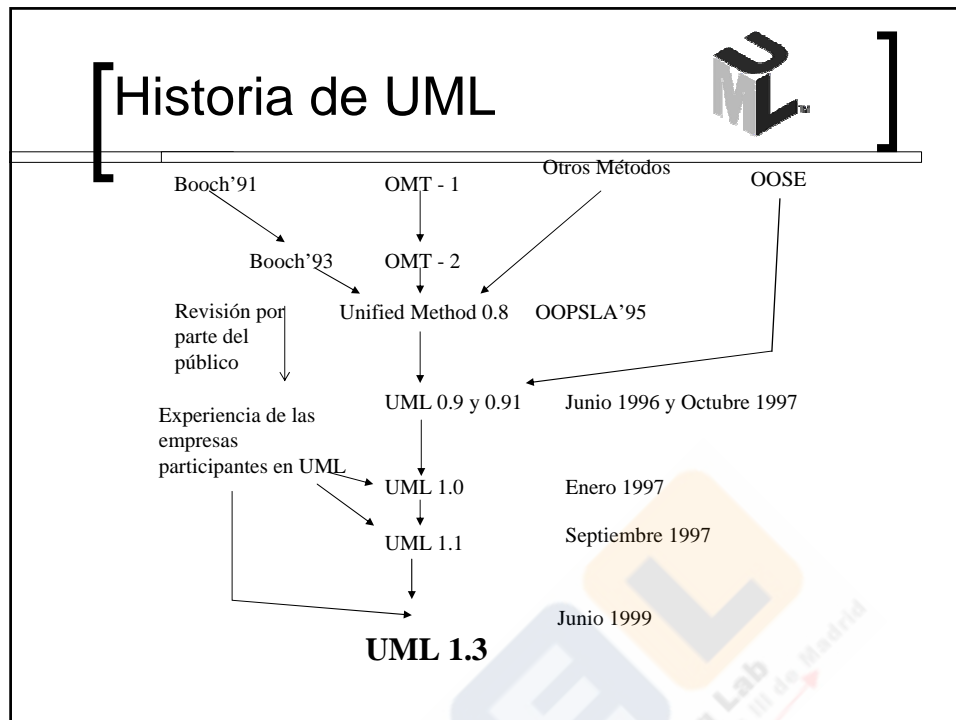


- Notación para especificación, construcción y documentación de artefactos de sistemas de software, modelos de negocio y otros sistemas.
- Propone técnicas para:
 - Especificaciones
 - Diseño
 - Implementación
- Aplicable a distintos dominios.

[Motivación de UML]

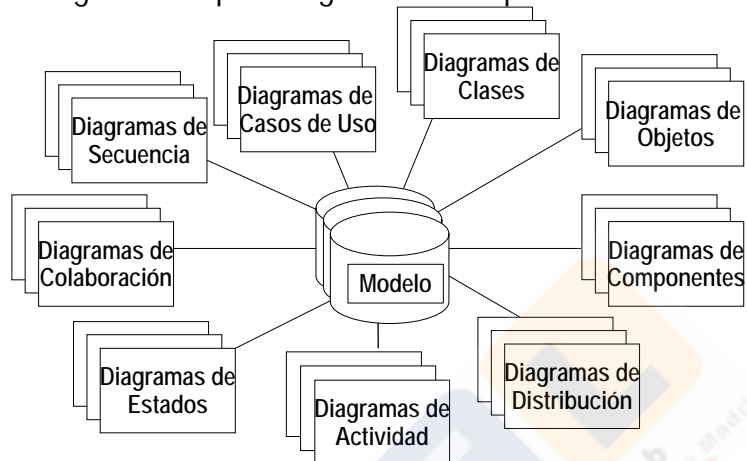


- Proporcionar a los desarrolladores un lenguaje que permita intercambiar modelos de conocimiento.
- Permitir extensibilidad y especialización para poder trabajar en diferentes entornos.
- Unificar las distintas notaciones existentes en Orientación a Objetos en un entorno estándar, estable y configurable.
- Incorporar las ventajas principales de los métodos más conocidos (Booch, OMT y OOSE).



[... Diagramas de UML]

Los diagramas expresan gráficamente partes de un modelo



[... Organización de Modelos]

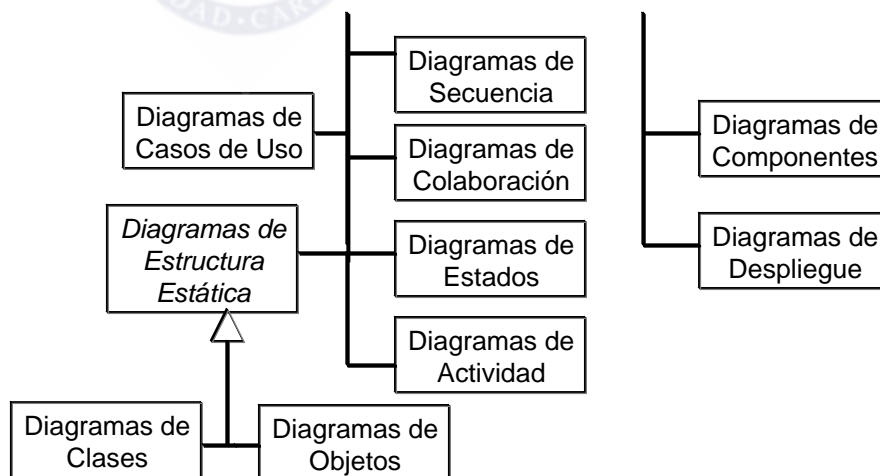
Propuesta de Rational Unified Process (RUP)

- M. de Casos de Uso del Negocio (Business Use-Case Model)
- M. de Objetos del Negocio (Business Object Model)
- M. de Casos de Uso (Use-Case Model)
- M. de Análisis (Analysis Model)
- M. de Diseño (Design Model)
- M. de Despliegue (Deployment Model)
- M. de Datos (Data Model)
- M. de Implementación (Implementation Model)
- M. de Pruebas (Test Model)

[Participantes en UML 1.0]

- Rational Software
(Grady Booch, Jim Rumbaugh y Ivar Jacobson)
- Digital Equipment
- Hewlett-Packard
- i-Logix (David Harel)
- IBM
- ICON Computing
(Desmond D'Souza)
- Intellicorp and James Martin & co. (James Odell)
- MCI Systemhouse
- Microsoft
- ObjecTime
- Oracle Corp.
- Platinum Technology
- Sterling Software
- Taskon
- Texas Instruments
- Unisys

[Diagramas de UML]

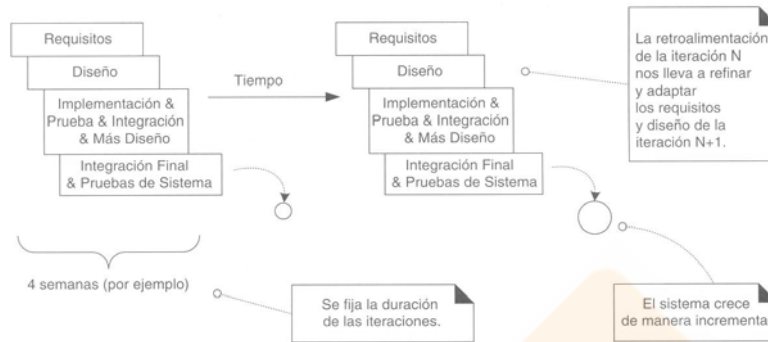


[El proceso de desarrollo iterativo incremental]

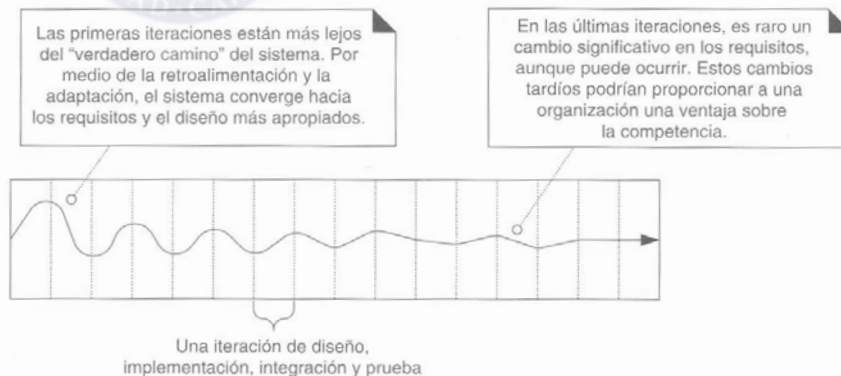
[Características del Proceso]

- Iterativo
- Incremental:
 - Menos tiempo hasta tener algo funcionando
 - Contrastable con el cliente / usuario
- Dirigido por Casos de Uso:
 - Desarrollar el software desde dentro hacia fuera.

Desarrollo iterativo incremental



Aceptando los cambios: retroalimentación y adaptación



[Beneficios del desarrollo iterativo]

- Mitigación tan pronto como sea posible de los riesgos altos
- Progreso visible en las primeras etapas
- Una temprana retroalimentación, compromiso de los usuarios y adaptación
- Gestión de la complejidad: el equipo no se ve abrumado por la parálisis del análisis
- El conocimiento adquirido en una iteración se puede utilizar metódicamente para mejorar el propio proceso de desarrollo

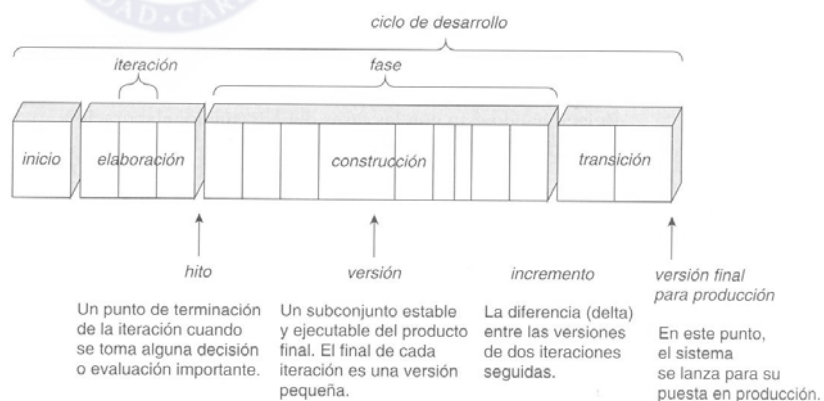
[Buenas prácticas adicionales]

- Abordar cuestiones de alto riesgo y muy valiosas durante las primeras iteraciones
- Involucrar continuamente a los usuarios para la evaluación, retroalimentación y requisitos
- Construir en las primeras iteraciones una arquitectura que constituya un núcleo central consistente
- Verificar la calidad continuamente: pruebas muy pronto, con frecuencia y de manera realista
- Aplicar casos de uso
- Modelar software visualmente
- Gestionar los requisitos con cuidado
- Manejar las peticiones de cambio y gestión de configuración

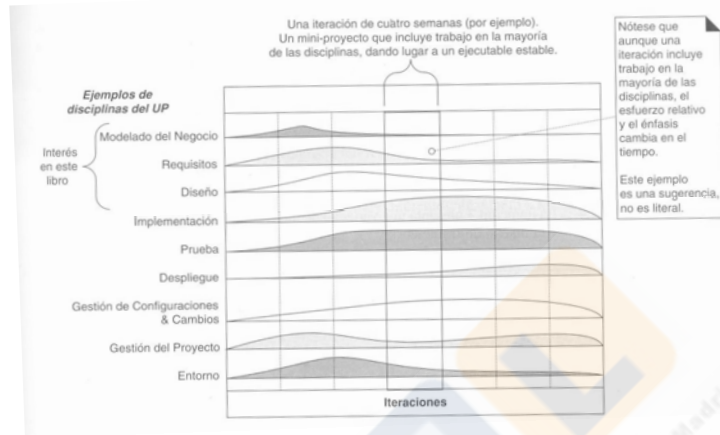
[El Proceso Unificado. Fases (I)]

1. Inicio
2. Elaboración
3. Construcción
4. Transición

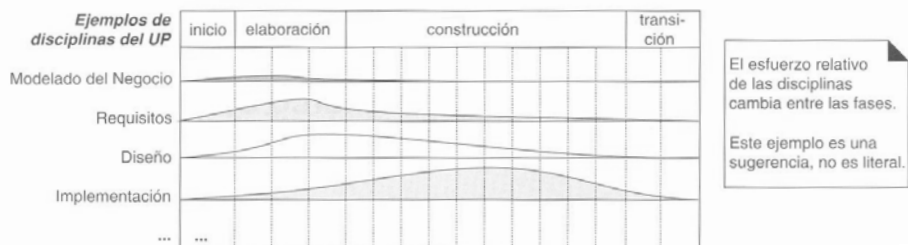
[El Proceso Unificado. Fases (II)]



Disciplinas del Proceso Unificado



Disciplinas y fases



Artefactos del proceso de desarrollo

Disciplina	Artefacto Iteración →	Inicio I1	Elab. E1...En	Const. C1...Cn	Trans. T1...T2
Modelado del Negocio	Modelo del Dominio		c		
Requisitos	Modelo de Casos de Uso	c	r		
	Visión	c	r		
	Especificación Complementaria	c	r		
	Glosario	c	r		
Diseño	Modelo de Diseño		c	r	
	Documento de Arquitectura SW		c		
	Modelo de Datos		c	r	
Implementación	Modelo de Implementación		c	r	r
Gestión del Proyecto	Plan de Desarrollo SW	c	r	r	r
Pruebas	Modelo de Pruebas		c	r	
Entorno	Marco de Desarrollo	c	r		

¿Cómo trabajaremos en la asignatura?

