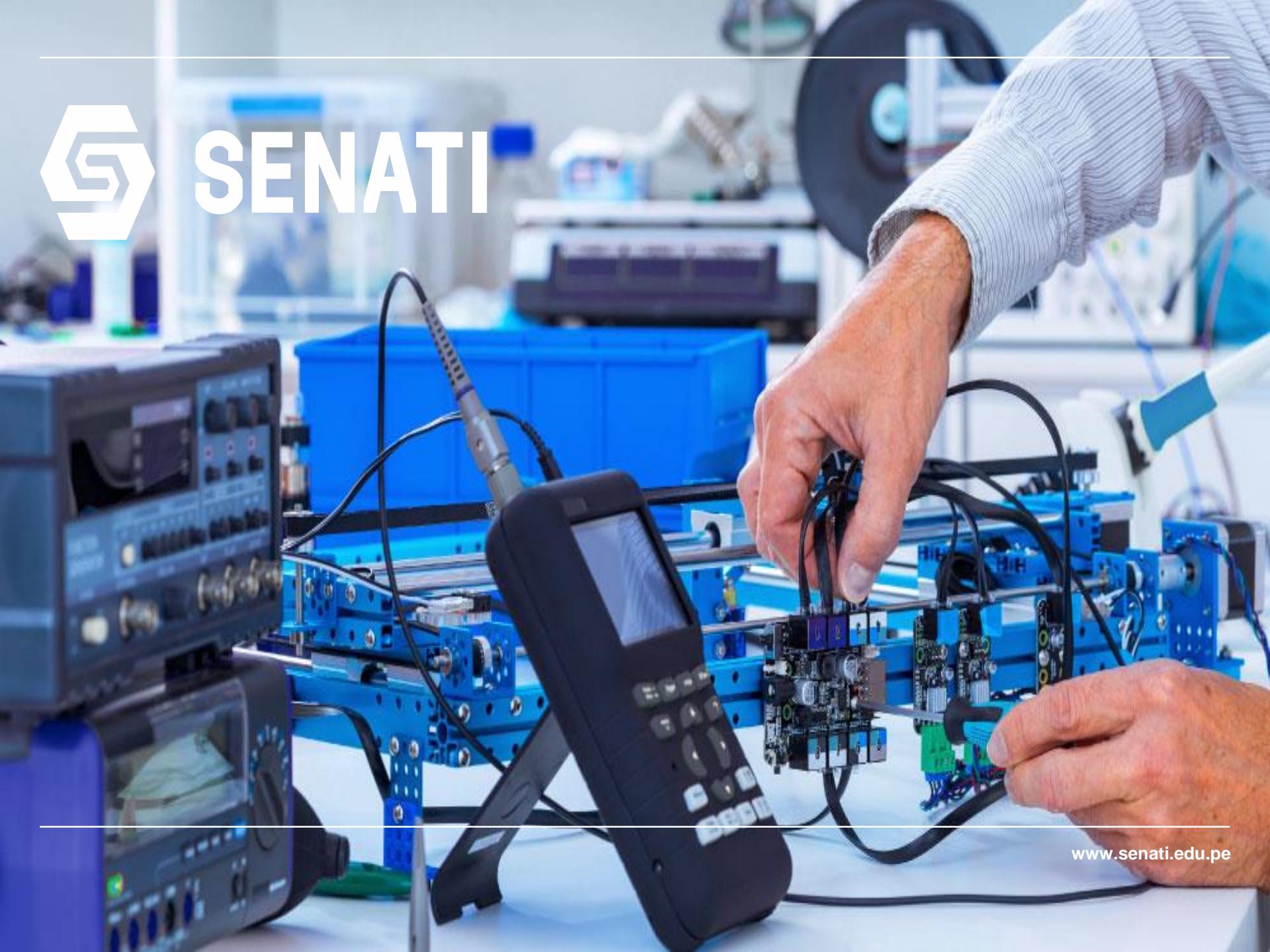




SENATI



Desarrollo de Software

Análisis y Diseño de Sistemas

Instructor Alejandro Ricaldi Rosas

www.senati.edu.pe

OBJETIVO:

Al finalizar el desarrollo del siguiente curso, el alumno será capaz de analizar y diseñar sistemas.

Introducción a UML

Instructor Alejandro Ricaldi Rosas

Construcción de una casa para “fido”



Puede hacerlo una sola persona
Requiere:

Modelado mínimo
Proceso simple
Herramientas simples

Construcción de una casa



Construida eficientemente y en un tiempo razonable por un equipo

Requiere:

Modelado

Proceso bien definido

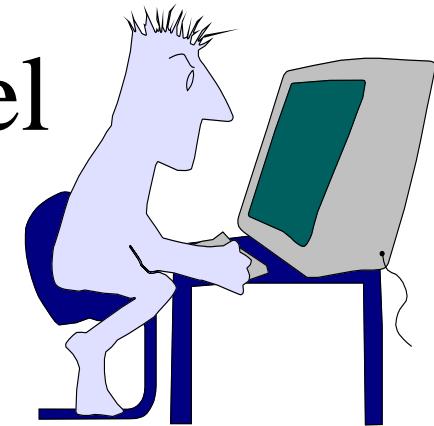
Herramientas más sofisticadas

Construcción de un rascacielos



Problemas de la Industria de Software en la actualidad

- ① Tendencia al **crecimiento** del volumen y complejidad de los **productos**.
- ② **Proyectos** excesivamente **tardes** y se exige **mayor productividad** y **calidad en menos tiempo**.
- ③ **Insuficiente personal calificado.**

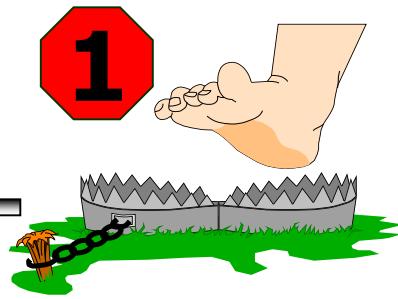




¿ Por qué fallan los Proyectos de Software?

- 1 Planificación Irreal
- 2 Mala Calidad del Trabajo
- 3 Personal Inapropiado
- 4 No Controlar los Cambios

Planificación Irreal



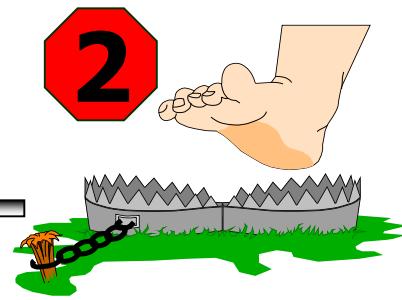
“El sistema es para hoy y con costo 0”

Los ingenieros no son capaces de enfrentar un plan porque:

- **NO** están entrenados para usar métodos de planificación.
- Frecuentemente, las estimaciones **NO** se basan en datos reales.



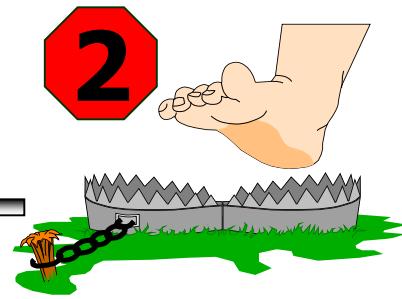
Mala Calidad del Trabajo



CAUSAS

- Prácticas pobres de ingeniería
- Carencia de métricas de calidad
- Inadecuado entrenamiento en calidad
- Decisiones de los directivos guiadas por una planificación irreal

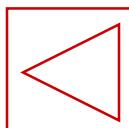
Mala Calidad del Trabajo

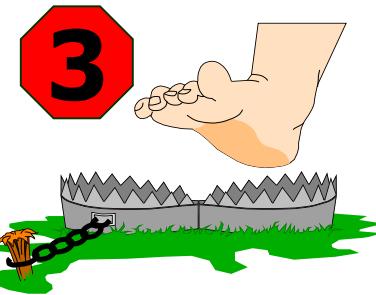


CONSECUENCIAS

- Tiempos de pruebas impredecibles
- Productos con muchos defectos
- Demoras en la aceptación de los usuarios
- Extensa garantía de servicio y reparaciones

“Una pobre calidad afecta la planificación y torna ineficiente el proceso de prueba”





Personal Inapropiado

PROBLEMAS COMUNES

- Demora del personal
- Escaso personal
- Miembros del equipo a tiempo parcial
- Personal con conocimientos inapropiados

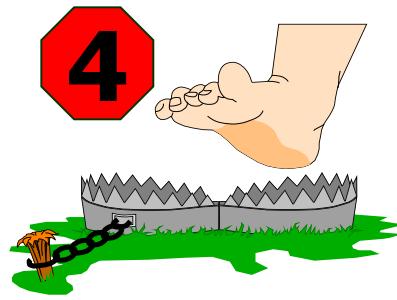
CONSECUENCIAS

- El trabajo se demora o descuida
- Trabajo ineficiente
- Sufre la moral del equipo

Con independencia del plan, los proyectos deben comenzar en tiempo y con todo el personal.



Cambios NO controlados



HECHOS a RECORDAR:

- Siempre ocurren cambios en los requerimientos.
- Los planes del proyecto se basan en el alcance del trabajo conocido.
- Los cambios siempre requieren más trabajo.
- Sin planes detallados, los equipos no pueden estimar el efecto o magnitud de los cambios.
- Si los equipos no controlan cada cambio, se pierde gradualmente el control del plan del proyecto



¿Cómo enfrentarla?

Las organizaciones requieren:

- 1** Desarrollar o adquirir una disciplina en el desarrollo del software.
- 2** Controlar que los ingenieros usen de forma consistente los nuevos métodos.

¿Qué debe hacer una
empresa para obtener
software de buena
calidad?

Cómo?



**Mejorar el proceso de
desarrollo de software**

Cualquier modelo de calidad para mejorar el Proceso de Desarrollo de Software, IMPLICA utilizar los métodos y procedimientos de INGENIERIA Y GESTION DE SOFTWARE

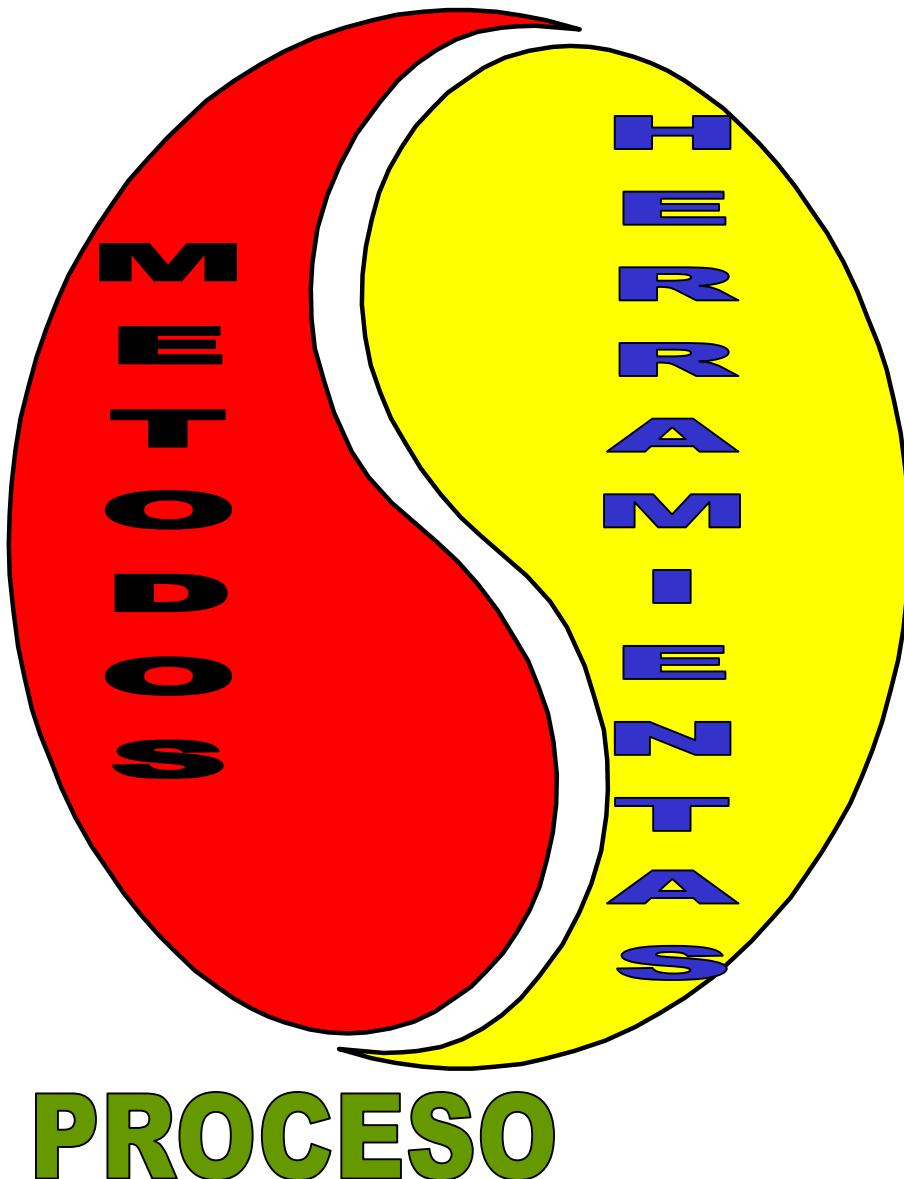
¿Qué es la Ingeniería de Software (IS)?



“...la aplicación de un enfoque sistémico, disciplinado y cuantificable hacia el desarrollo, funcionamiento y mantenimiento de software, es decir la aplicación de ingeniería al software”

IEEE, 1993

IS es una tecnología multicapa



Indican cómo construir técnicamente el Sw.

Soporte automático o semiautomático para el proceso y los métodos.

Es el fundamento de la IS. Es la unión que mantiene juntas las capas de la tecnología.

Síntomas - Causas

Síntomas

Diagnóstico

Causas

- necesidades usuarios
- requerimientos cambiantes
- módulos no calzan
- poco mantenible
- tardía detección
- baja calidad
- baja performance
- versiones y cambios
- liberación y distribución

- requerimientos insuficientes
- comunicación ambigua
- arquitecturas frágiles
- complejidad excesiva
- inconsistencias no detectadas
- prueba pobre
- evaluación subjetiva
- desarrollo en cascada
- cambios no controlados
- automatización insuficiente

...tratar los Síntomas no resuelve el problema

Las Mejores Prácticas de la IS atacan las causas

Desarrolle Iterativamente

**Administre
Requerimientos**

**Use
arquitectura
de
componentes**

**Modele
Visualmente**

**Verifique
Calidad**

Controle Cambios

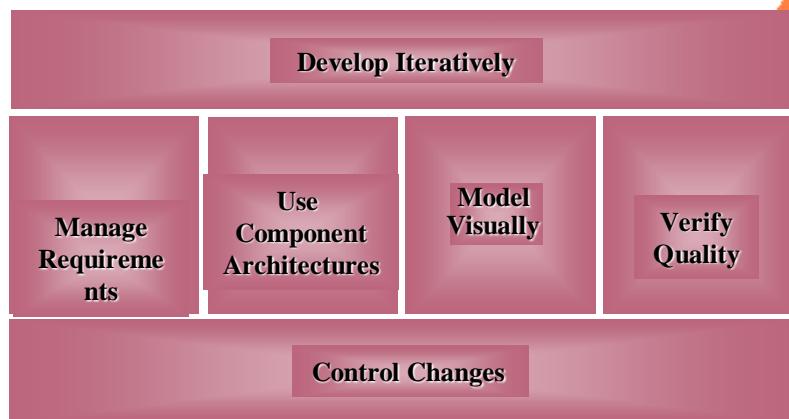
Mejores Prácticas de Software

Son propuestas de desarrollo probadas comercialmente, que usadas en forma combinada atacan la raíz de las causas de las fallas, eliminando los síntomas y permitiendo el desarrollo y mantenimiento de software de calidad de manera predictiva y reiterativa.

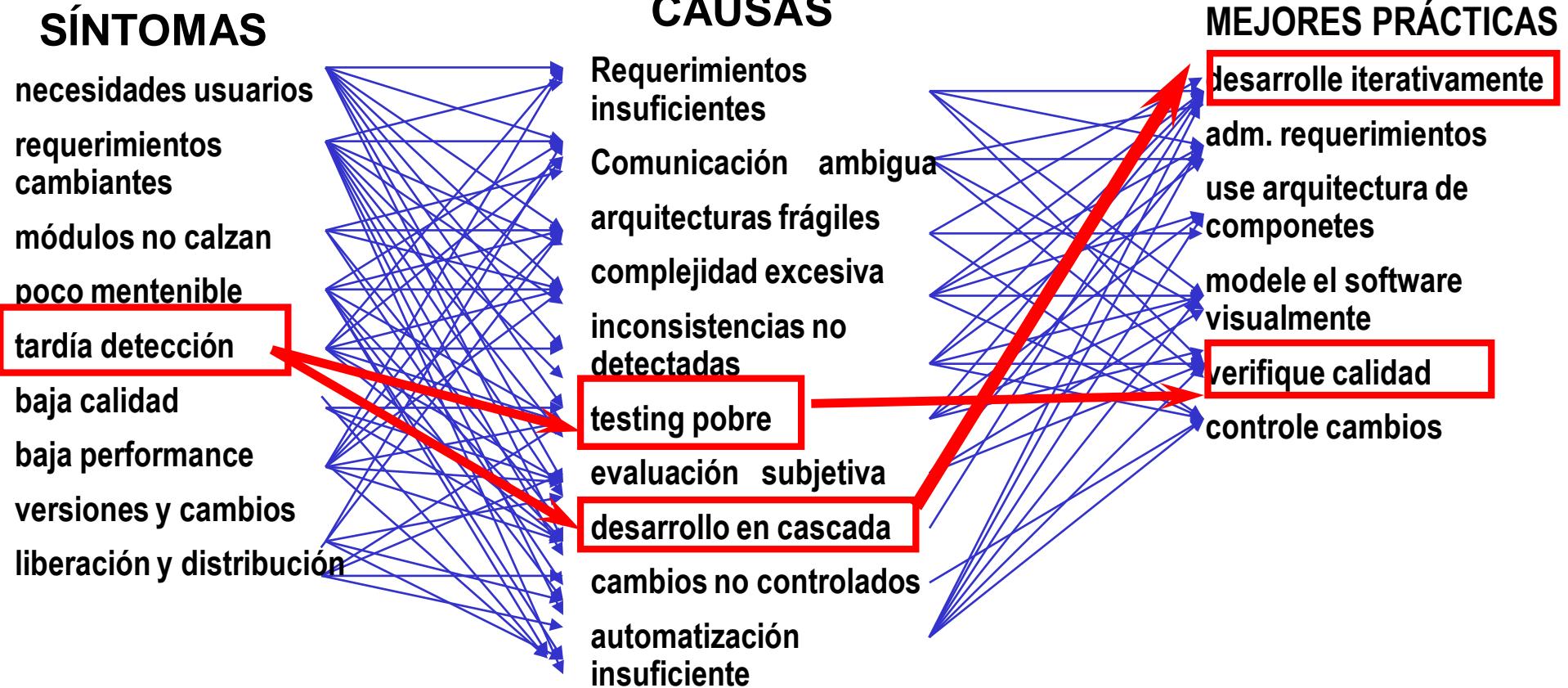
Mejores Prácticas: Equipos de Alto Rendimiento

Resultado

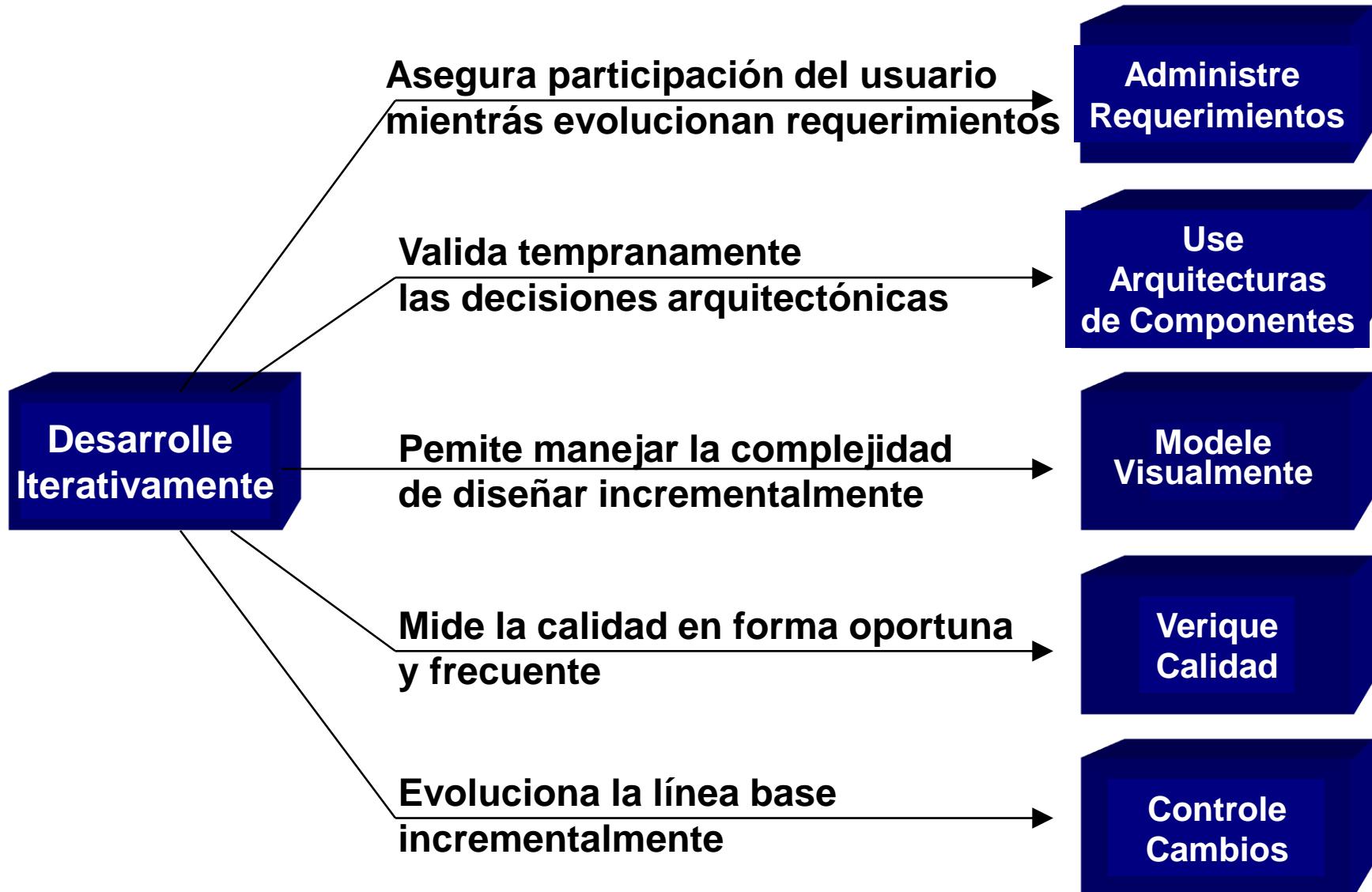
- **Proyectos más exitosos porque están en plazo, en presupuesto y satisfacen las necesidades del usuario**



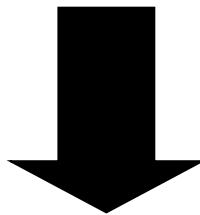
Enfrentando las Causas se eliminan los Síntomas



Mejores Prácticas se refuerzan entre si



Mejores prácticas para el trabajo efectivo de un equipo en el desarrollo del software.



Proceso de desarrollo de software

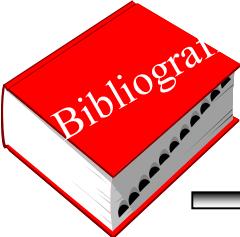
(Dirige y organiza el proceso)

Lenguaje Unificado de Modelación

(Notación)

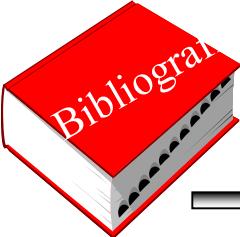
Sumario

- **Conceptos básicos del modelo de objetos**
- **Proceso de desarrollo de software.**
- **El Lenguaje Unificado de Modelación (UML)**
- **El Proceso Unificado de Desarrollo de Software (RUP)**



Lecturas Recomendadas

- BOOCH, Grady, RUMBAUGH, James, JACOBSON, Ivar; “El Lenguaje Unificado de Modelación. Libro introductorio”.2000. Addison Wesley.
- RUMBAUGH, James, JACOBSON, Ivar, BOOCH, Grady; “El Lenguaje Unificado de Modelación. Manual de referencia”.2000. Addison Wesley.
- JACOBSON, Ivar; BOOCH, Grady, RUMBAUGH, James, “El Proceso Unificado de Desarrollo de Software”.2000. Addison Wesley.
- LARMAN, Craig “UML y patrones” 1999, Prentice Hall Iberoamericana.



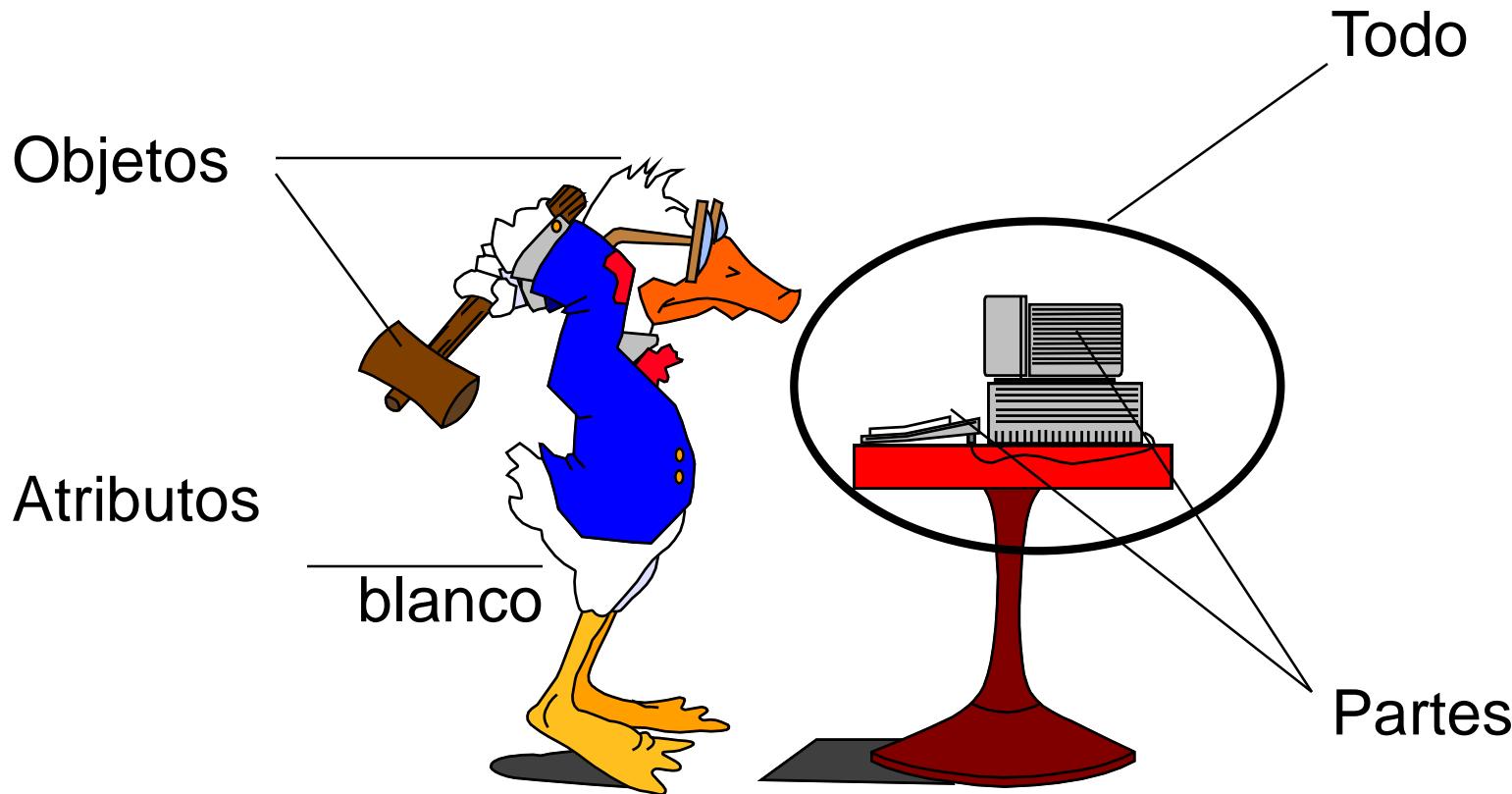
Lecturas Recomendadas

- BOOCH, Grady, MAKSIMCHUCK, Robert, ENGLEm Michael, YOUNG, Bobbi, CONALLEN, Jim, Houston, Kelli; "Object-Oriented Analysis and Design with Applications". Third Edition. Addison Wesley. 2007.
- HAMILTON, Kim, MILES, Russell; "Learning UML 2". 2006. O'Reilly Media
- CONALLEN, Jim, "Building Web Applications with UML". 2002. 2nd edition. Addison Wesley.

Conceptos básicos del Modelo de Objetos

Enfoque Orientado a Objetos

- Se basa en conceptos y relaciones entre ellos.



Enfoque Orientado a Objetos

Tipo de Objeto:

Descripción generalizada que describe una colección de objetos similares.

Clase:

Implementación en software de un tipo de objeto.

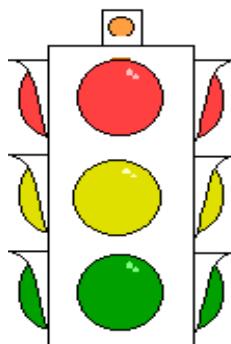


{ **Tlibro = class**
private
• • •
end;

Enfoque orientado a Objetos

Método:

Implementación en software de la operación.



Cambiar luz

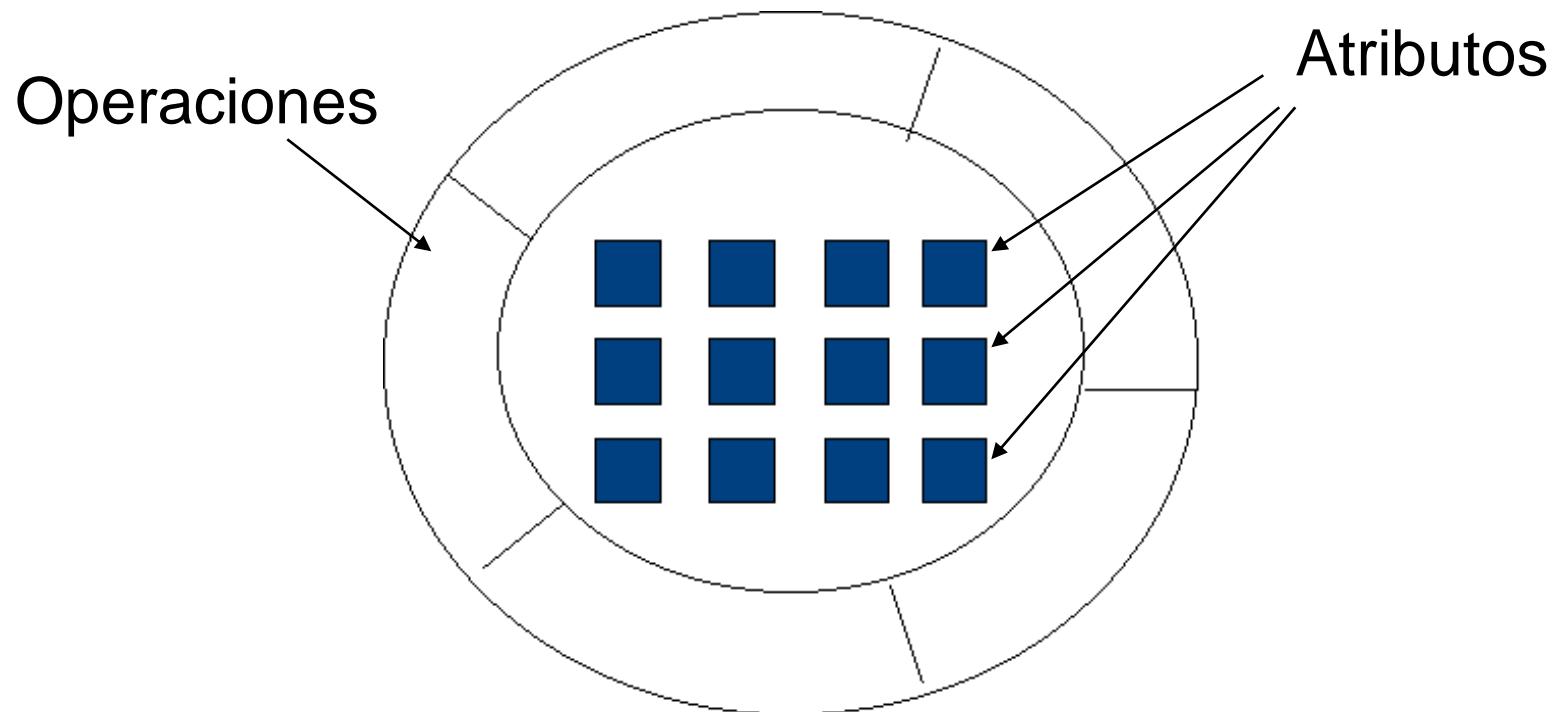
{
 TSemáforo = class

 Cambiar luz();
 end;

Enfoque Orientado a Objetos

Encapsulamiento

Empaque conjunto de datos y métodos.



Enfoque Orientado a Objetos

Herencia: Propiedad de una clase de heredar el comportamiento de sus ancestros.

Polimorfismo: Mecanismo que permite a la subclase implementar la misma operación con un método diferente.

Proceso de Desarrollo de Software

Proceso

Define “**quién**” está haciendo “**qué**”, “**cuándo**” y “**cómo**” para alcanzar un determinado objetivo.

Proceso de desarrollo de software (PDS)

CONJUNTO DE ACTIVIDADES



Requisitos del usuario

Sistema informático

Proceso de desarrollo de software (PDS)



La solicitud del usuario



Lo que entendió el líder del proyecto



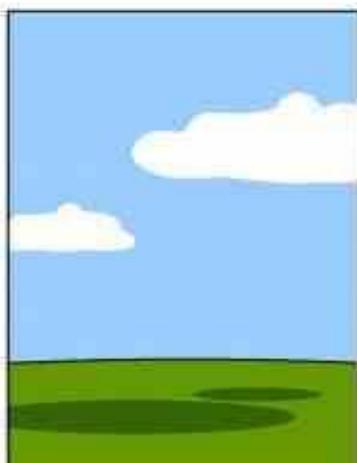
El diseño del analista de sistemas



El enfoque del programador



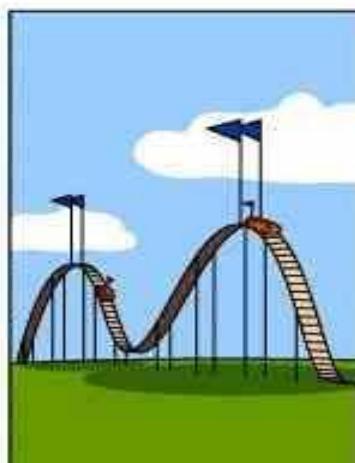
La recomendación del consultor externo



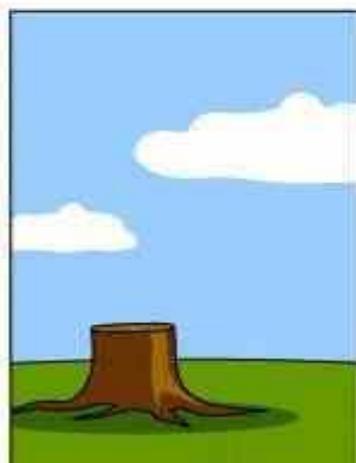
La documentación del proyecto



La implantación en producción



El presupuesto del proyecto



El soporte operativo



Lo que el usuario realmente necesitaba

Proceso de desarrollo de software (PDS)

- **Un proceso software debe especificar:**
 - la secuencia de actividades a realizar por el equipo de desarrollo: flujo de actividades
 - productos que deben crearse: qué y cuándo
 - asignación de tareas a cada miembro del equipo y al equipo como un todo
 - proporcionar heurísticas
 - criterios para controlar el proceso

UML

Lenguaje de modelación

“ Puede ser un seudocódigo, código, imágenes, diagramas, o largos paquetes de descripción; es decir, cualquier cosa que ayude a describir un sistema ”

**Lenguaje de
modelación**  **Notación + Metamodelo**

Lenguaje de modelación

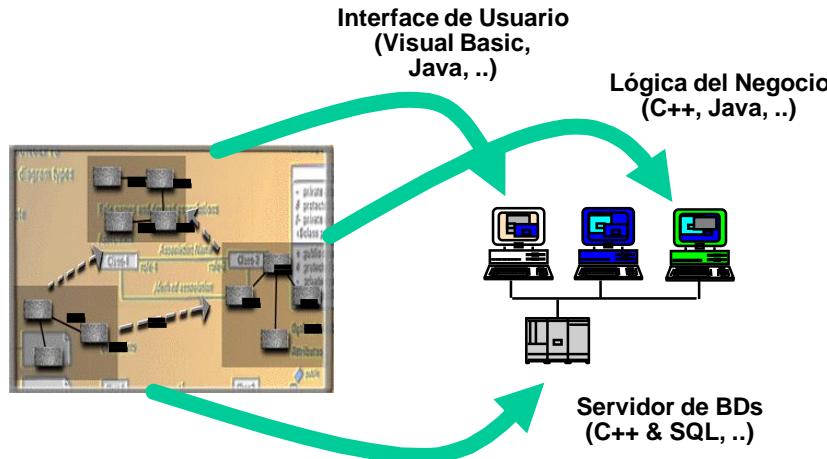
**Lenguaje de
modelación** = **Notación + Metamodelo**

(Forma de expresar el modelo)

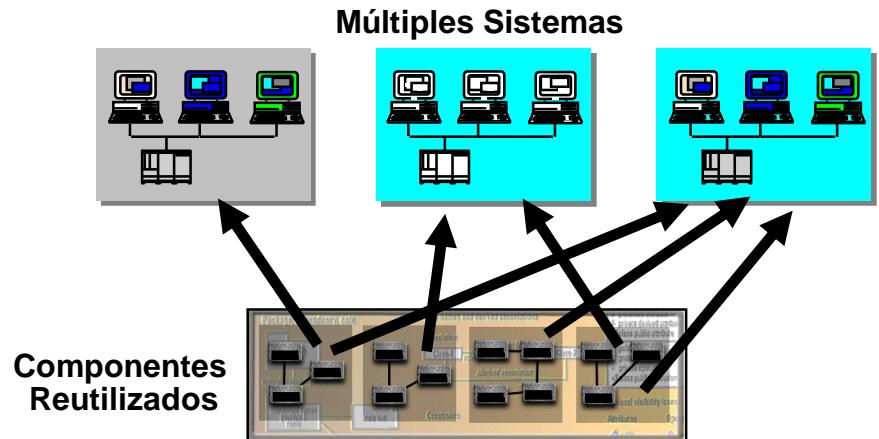
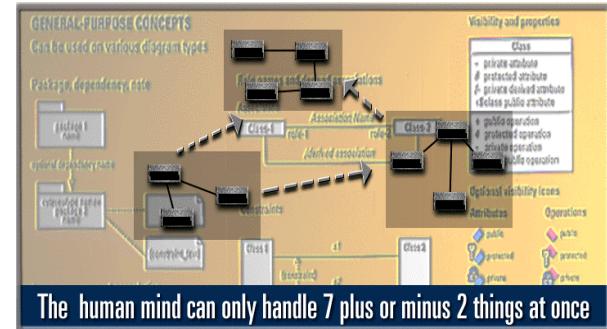
(Descripción de lo que significa esa modelación)

Notación visual

Manejar la complejidad



Modelar el sistema
independientemente
del lenguaje de
implementación



Promover la Reutilización

Situación de partida

- Diversos métodos y técnicas OO, con muchos aspectos en común pero utilizando distintas notaciones
- Inconvenientes para el aprendizaje, aplicación, construcción y uso de herramientas, etc.
- Pugna entre distintos enfoques (y correspondientes gurús)



Necesidad de establecer un estándar





UML

¿Qué es el UML- Unified Modeling Language?

Lenguaje Unificado de Modelación

- Descrito en "The Unified Modeling Language for Object-Oriented Development" de Grady Booch, James Rumbaugh e Ivar Jacobson.
- Basado en las experiencias personales de los autores.
- Incorpora contribuciones de otras metodologías. ⁴⁵



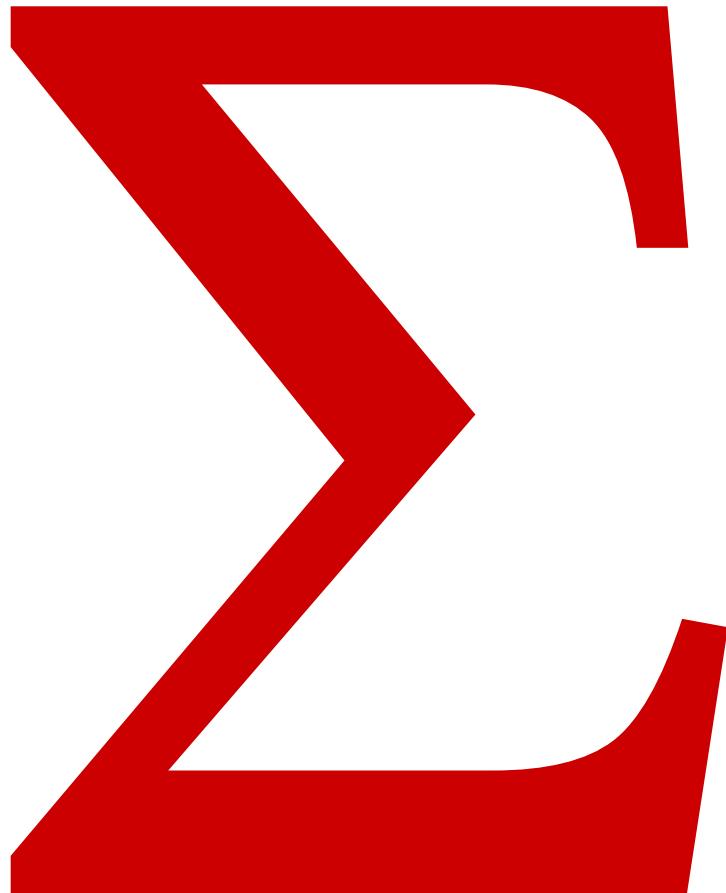
UML

¿Qué es el UML- Unified Modeling Language?

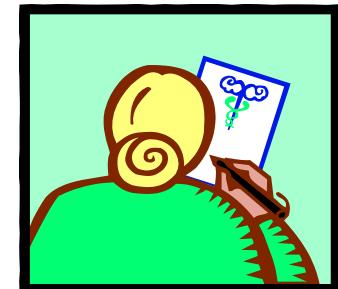
Lenguaje Unificado de Modelación

- Ofrece un estándar para describir un “plano” del sistema.
- Incluye aspectos conceptuales tales como procesos de negocio y funciones del sistema y aspectos concretos como expresiones de lenguajes de programación, esquemas de BD y componentes de software reutilizables.

UML



Visualizar



Especificar



Construir



Documentar

UML no es un método

El UML es una guía al desarrollador para realizar el análisis y diseño orientado a objetos, es un proceso

El UML es un lenguaje que permite la modelación de sistemas con tecnología orientada a objetos



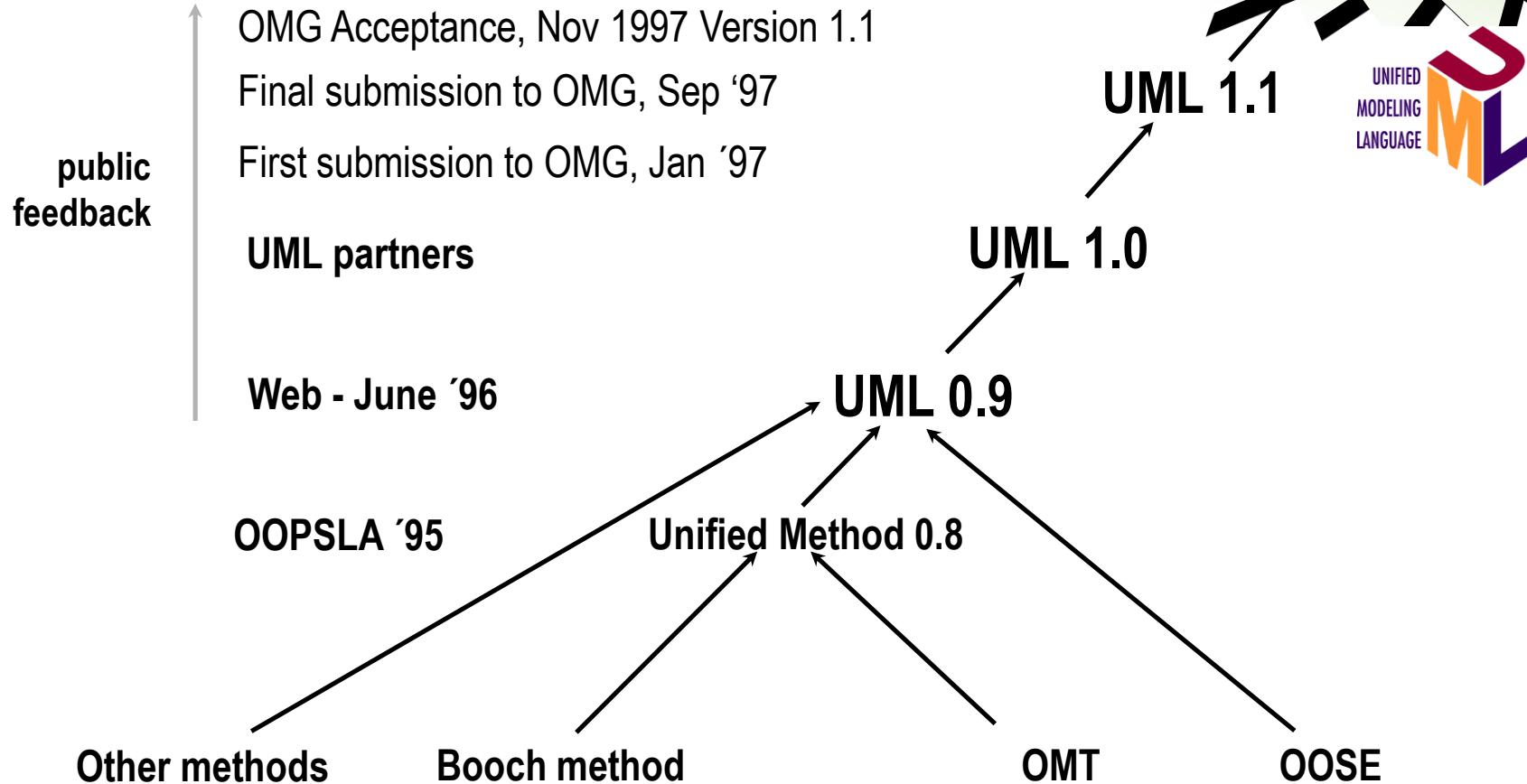
Aprobado como estándar por OMG en noviembre de 1997

Orígenes de UML



- El esfuerzo de UML partió oficialmente en octubre de 1994, cuando Rumbaugh se unió a Booch en Rational.
- El “Método Unificado” en su Versión 0.8, se presentó en el OOPSLA’95
- El mismo año se unió Ivar Jacobson. Los “Tres Amigos” son socios en la compañía Rational Software. Herramienta CASE Rational Rose

Creación del UML



¿Por qué UML 2.0?

- Las primeras versiones de UML estaban más orientadas hacia la modelación del software y ahora se requiere más la modelación del sistema.
- Necesidad de compartir modelos entre diferentes herramientas.
- Nuevas tecnologías: Arquitectura basada en componentes, MDA
- Las primeras versiones estaban más diseñadas para las personas y no para las máquinas, por lo que hay construcciones que no estaban suficientemente formalizadas.

Las Bases de UML



- Booch,
- Rumbaugh
- Jacobson



Rational Software Corporation. (1995)

JAMES
RUMBAUGH

Object Modelling Technique 1991(OMT)

GRADY
BOOCH

Object Oriented Analysis and Design
with Applications 1994

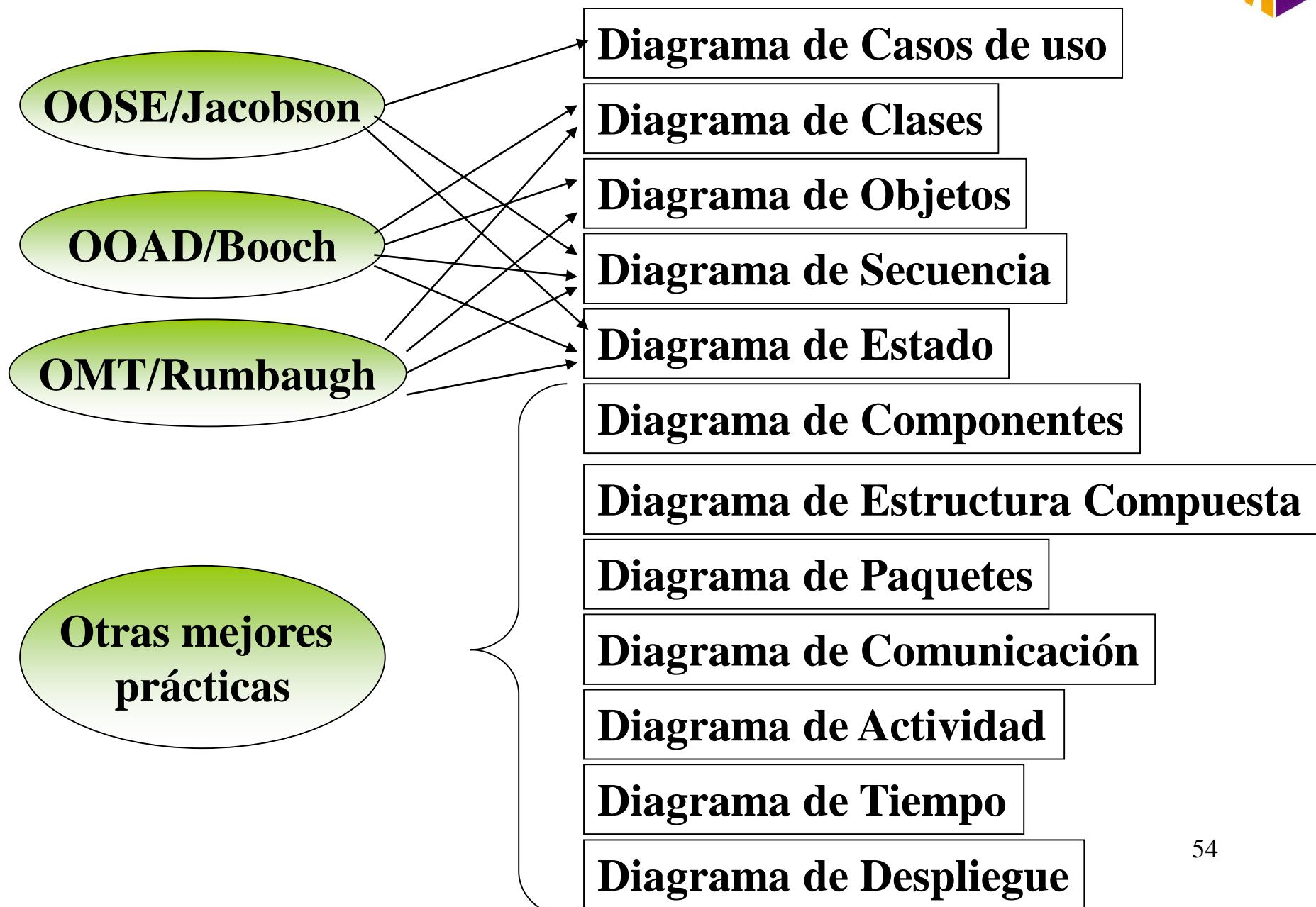
IVAR
JACOBSON

Object Oriented Software Engineering: A
Use Case Driven Approach 1992 (OOSE)

Premisas de UML según Booch, Jacobson y Rumbaugh

- Modelar sistemas, a partir de los conceptos hacia los artefactos ejecutables, utilizando técnicas orientadas a objeto.
- Enfocarnos en las cuestiones de escala inherentes a sistemas complejos, críticos en su misión.
- Crear un lenguaje de modelación utilizable tanto por los humanos, como por las máquinas.

Contribuciones a UML



Ventajas de UML



- Es un lenguaje formal ya que cada elemento del lenguaje tiene un significado fuerte que ayuda a modelar un aspecto particular del sistema.
- Es conciso con una notación simple.
- Es comprensible porque describe todos los aspectos importantes del sistema.
- Es escalable por lo que permite describir proyectos de diferentes tamaños.

Ventajas de UML

- **Contiene las mejores prácticas de la comunidad orientada a objetos de los últimos 15 años.**
- **Es un estándar abierto.**
- **Da soporte a todo el ciclo de vida de desarrollo del software.**
- **Da soporte a diversas áreas de aplicación.**
- **Está soportado por muchas herramientas.**

Limitaciones de UML



- Carece de un semántica precisa, lo que ha dado lugar a que la interpretación de un modelo UML no pueda ser objetiva en ocasiones.
- No se presta con facilidad al diseño de sistemas distribuidos. En estos sistemas son importantes factores como transmisión, serialización, persistencia, etc. No se puede señalar si un objeto es persistente o remoto.



Un proceso de desarrollo de software debe ofrecer un conjunto de modelos que permitan expresar el producto de software desde cada una de las perspectivas de interés



¿Qué es un producto de software?

Un producto de software es el código máquina y los ejecutables de un sistema

Un producto de software es el conjunto de **artefactos** que se necesitan para representarlo en forma comprensible para:



- Las máquinas.
- Los trabajadores.
- Los usuarios.
- Los interesados.

¿Artefactos?



Término general aplicable a cualquier tipo de información creada, cambiada o utilizada por los trabajadores en el desarrollo del sistema

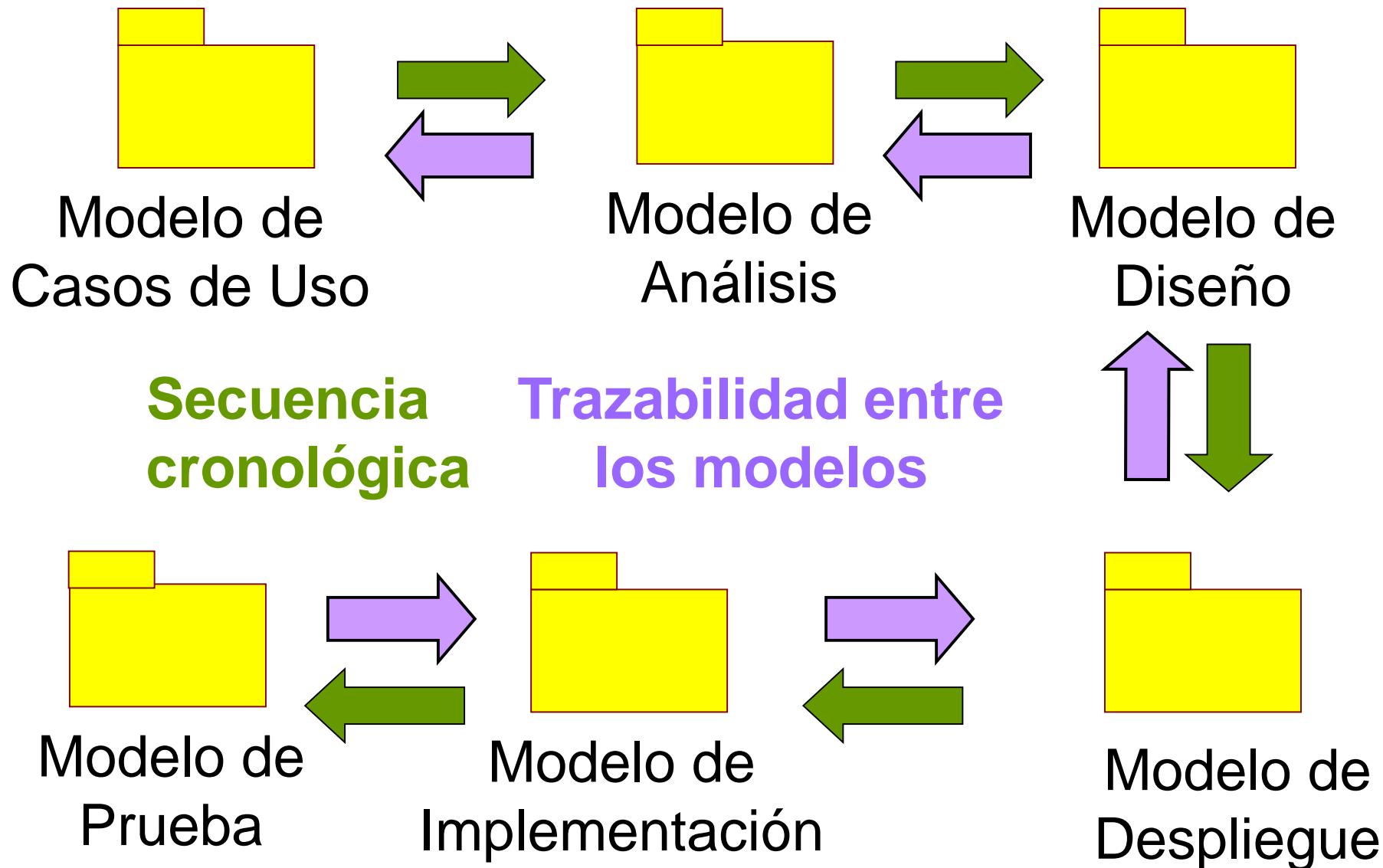
Ejemplos:

- Diagramas UML y su texto.
- Bocetos de interfaz.
- Planes de prueba.

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



Diagramas de UML

Diagrama	¿Dónde aparece?
Caso de uso	1.x
Actividad	1.x
Clase	1.x
Objeto	Informalmente 1.x
Secuencia	1.x
Comunicación	Antes de Colaboración en 1.x
Tiempo	2.0
Estructura interna	2.0
Componente	1.x, pero cambia su significado en 2.0
Paquete	2.0
Estado	1.X
Despliegue	1.x

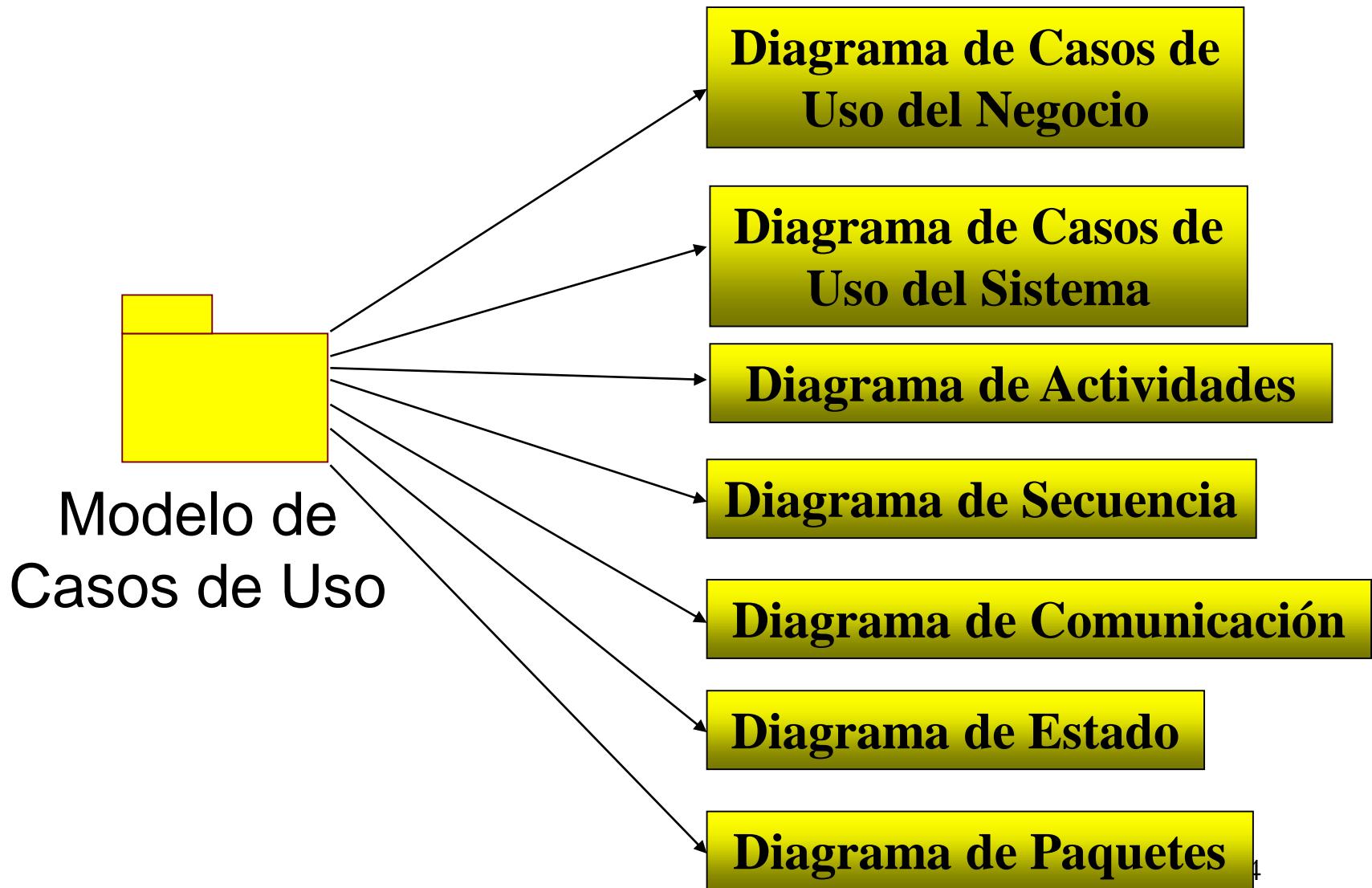
Estructura

Dinámica

Física

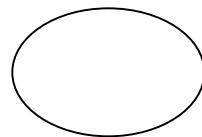
Gestión del modelo

Modelos y Diagramas



Diagramas de UML

- Casos de Uso y Diagramas de Casos de Uso



Casos de uso

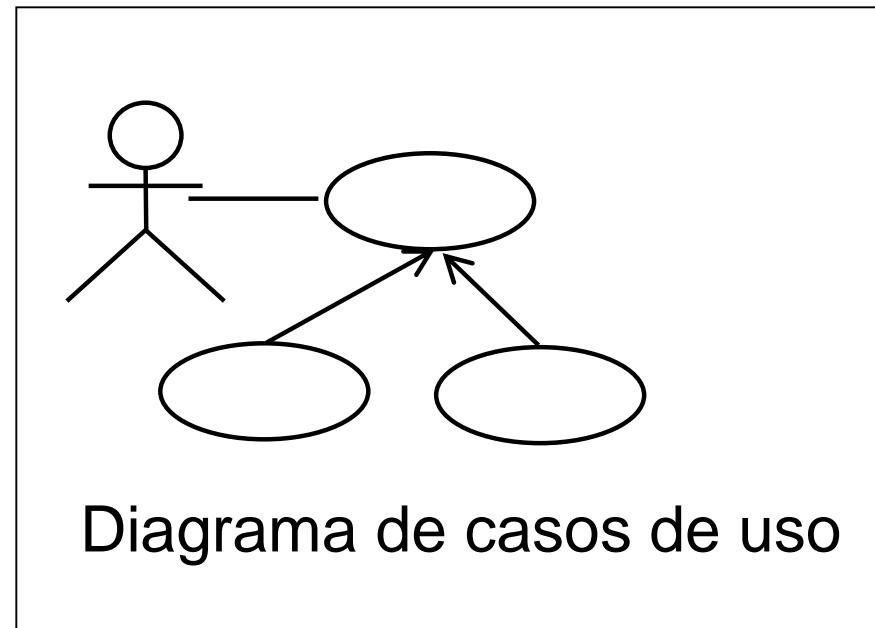
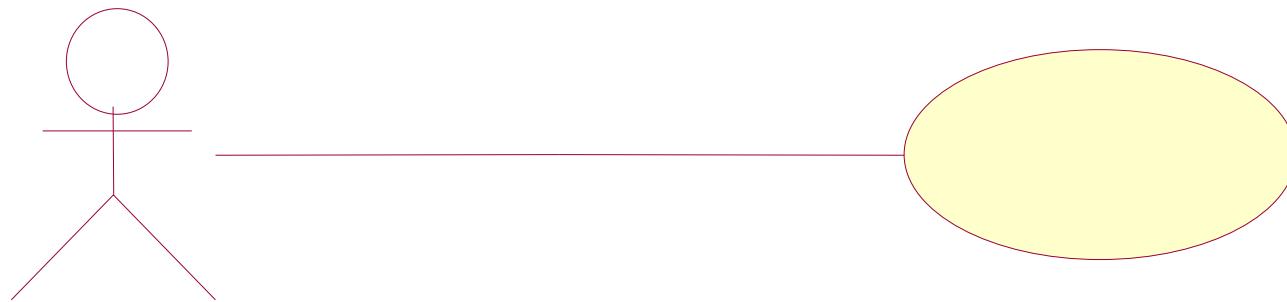


Diagrama de casos de uso

- **Casos de Uso es una técnica para capturar información de cómo un sistema o negocio trabaja, o de cómo se desea que trabaje**
- **No pertenece estrictamente al enfoque orientado a objeto, es una técnica para captura de requisitos**



Cliente

Solicitar Préstamo

Diagramas de UML

- Diagramas de estructura estática
 - Diagrama de clases
 - Diagrama de Objetos

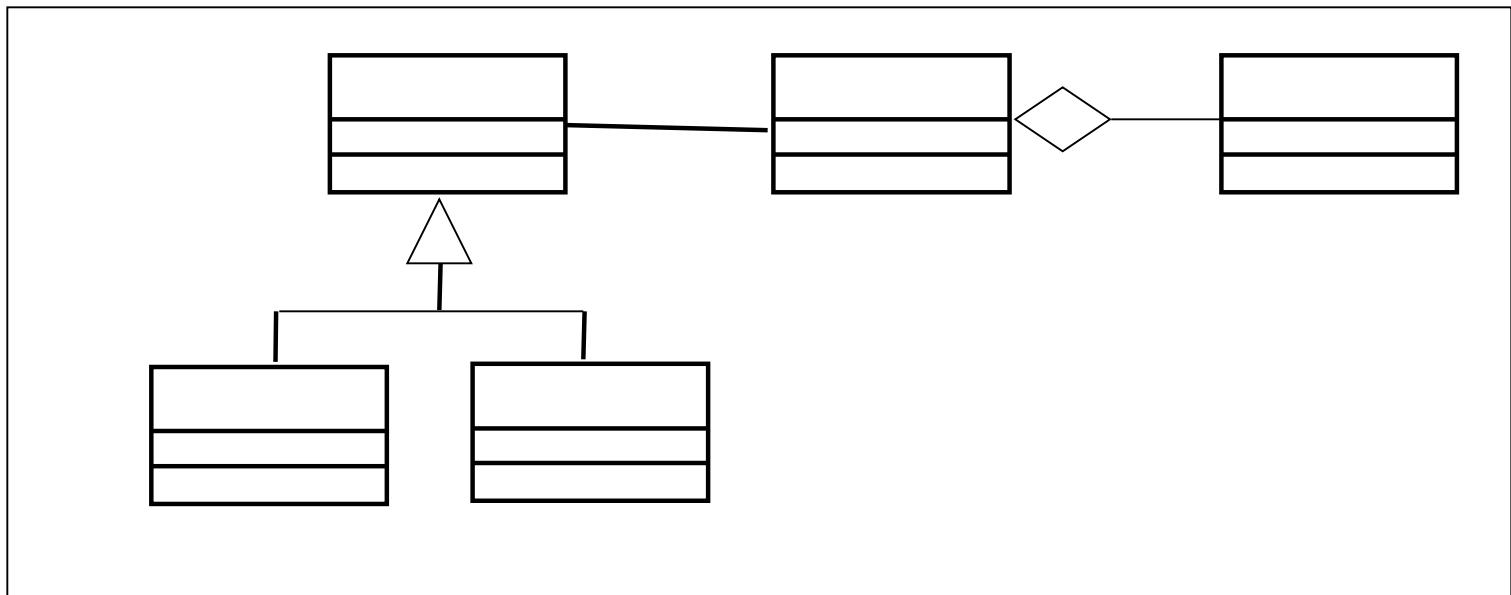


Diagrama de clases

- **Un diagrama de clases presenta las clases del sistema con sus relaciones estructurales y de herencia**
- **La definición de clase incluye definiciones para atributos y operaciones**
- **El modelo de casos de uso aporta información para establecer las clases, objetos, atributos y operaciones**



Diagramas UML

- Diagramas del comportamiento
 - Diagramas de Estado
 - Diagramas de Actividad
 - Diagramas de Secuencia
 - Diagrama de Comunicación
 - Diagrama de tiempo

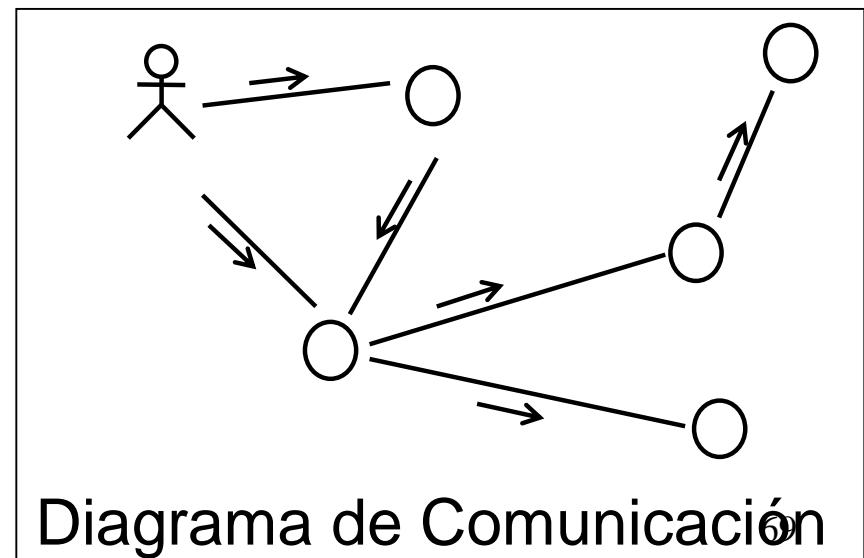
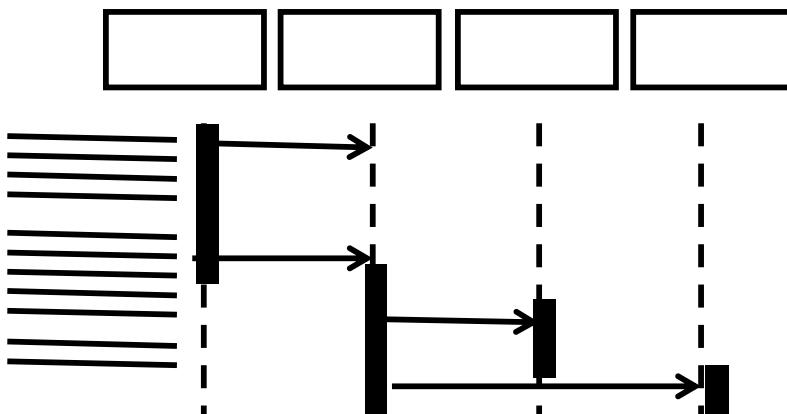


Diagrama de estado

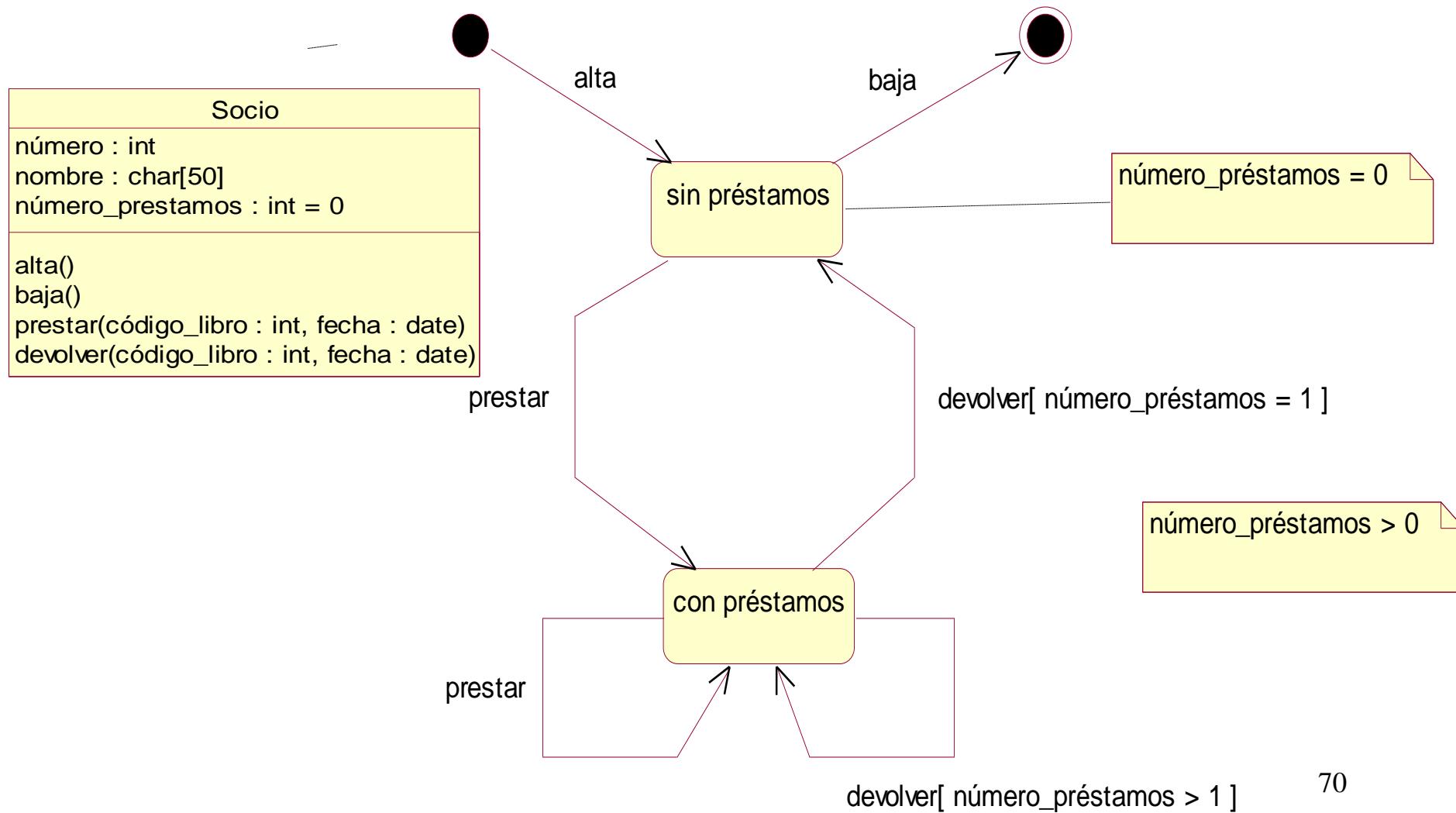


Diagrama de actividades

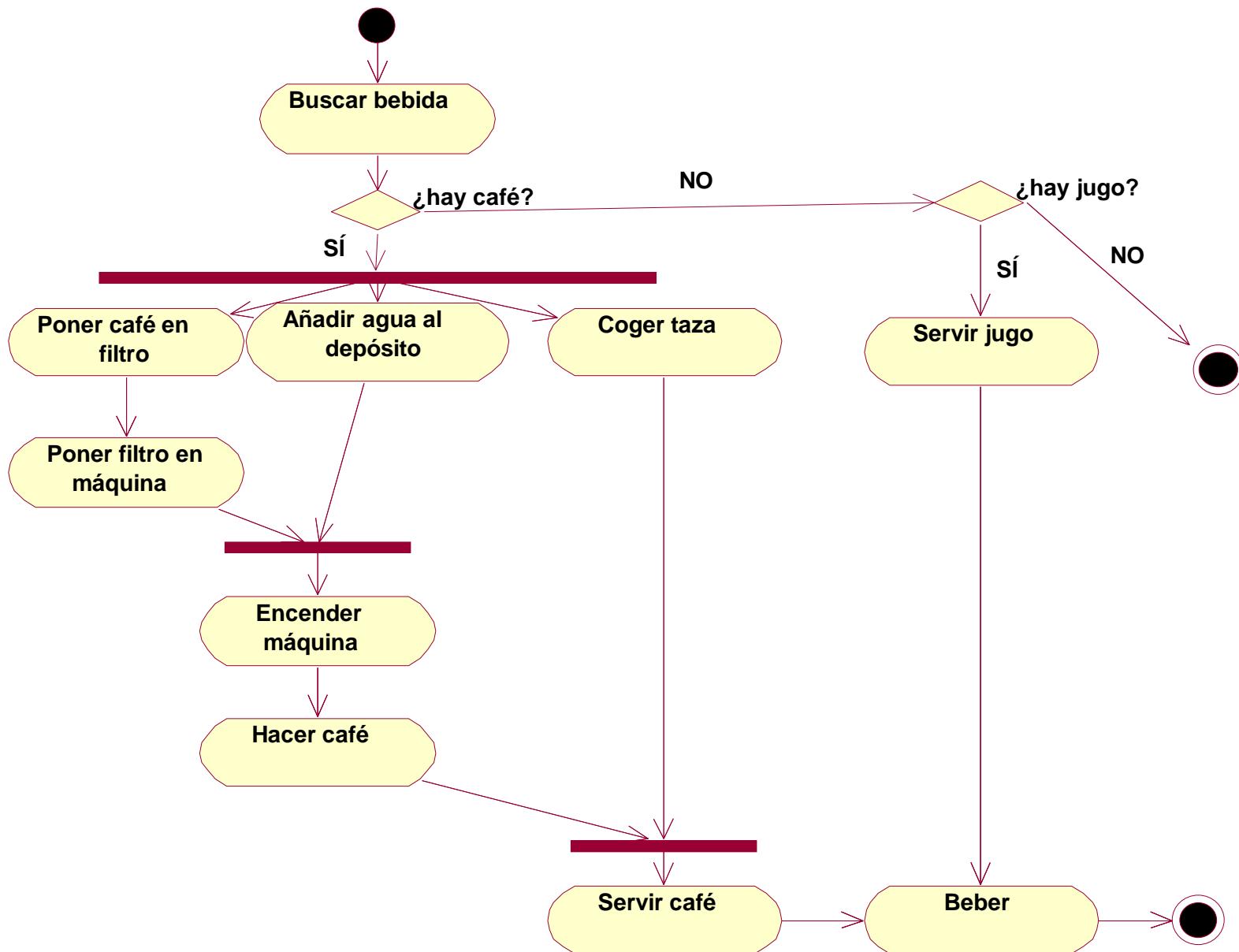


Diagrama de secuencia

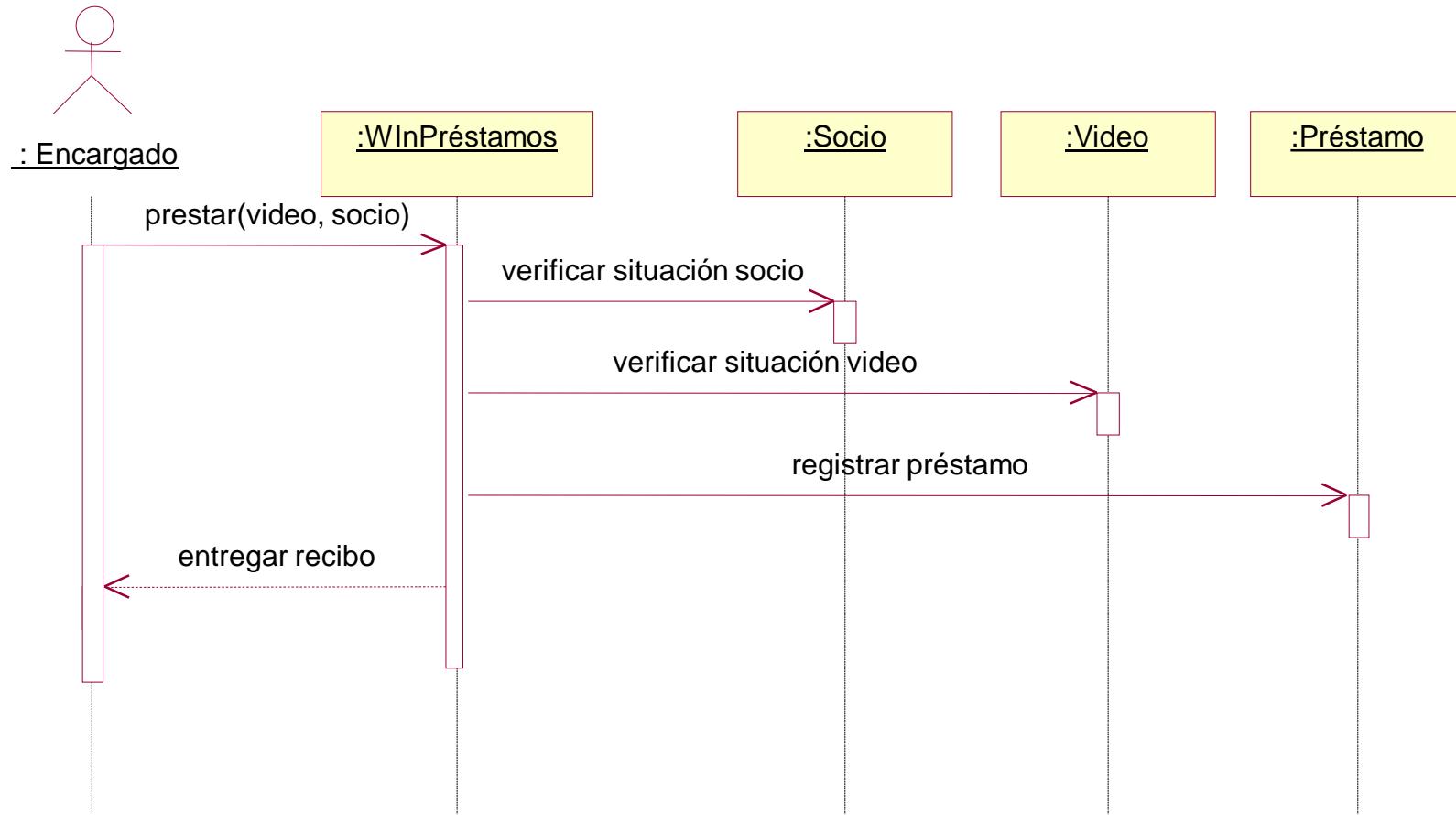
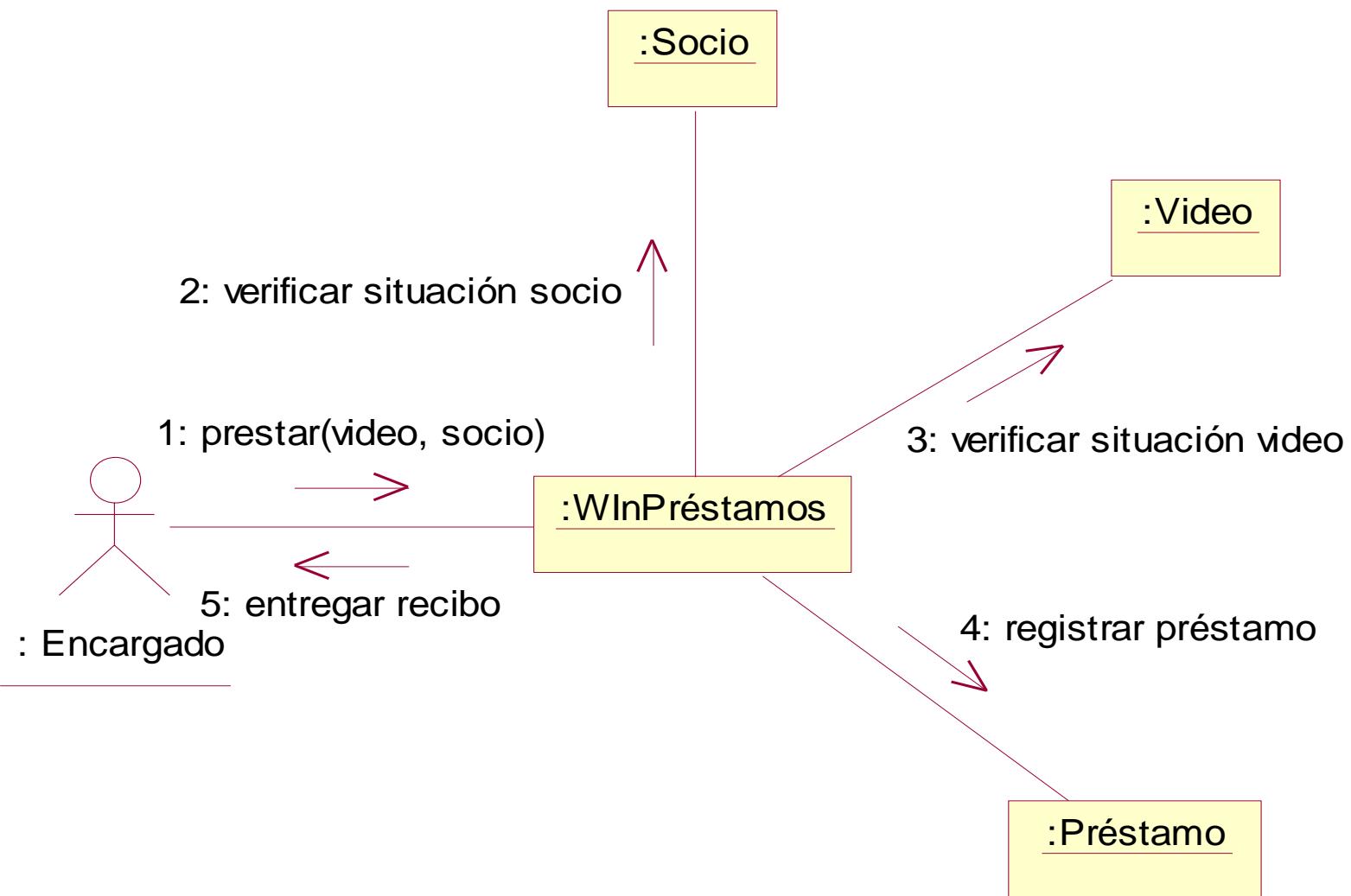


Diagrama de comunicación



Diagramas de UML

- Diagramas de implementación
 - Diagramas de componentes
 - Diagramas de instalación/Distribución
(Despliegue)

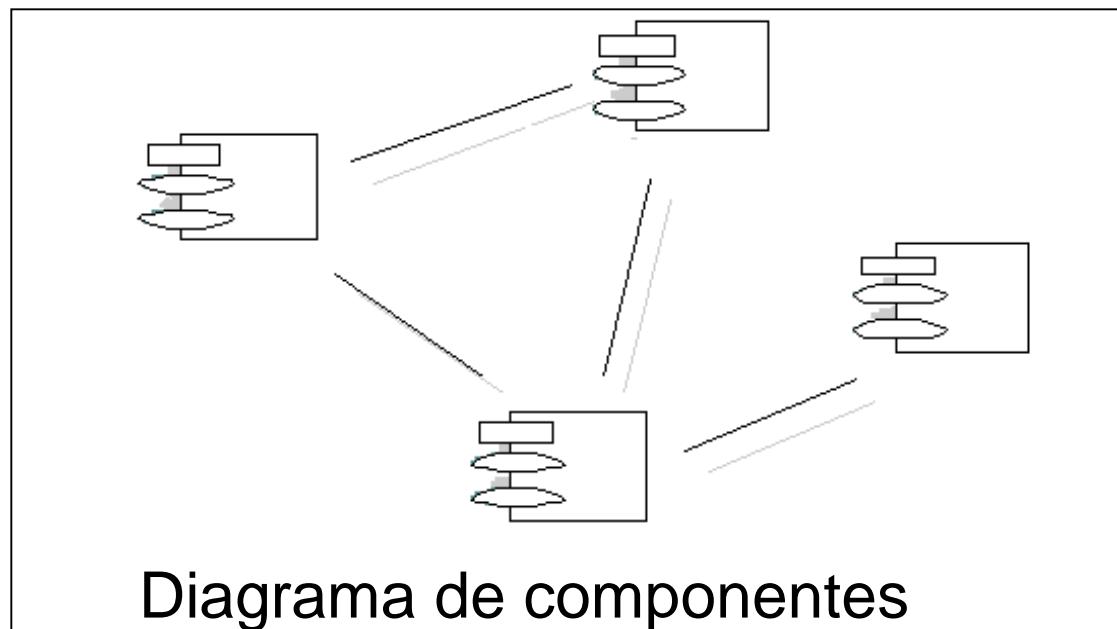


Diagrama de componentes

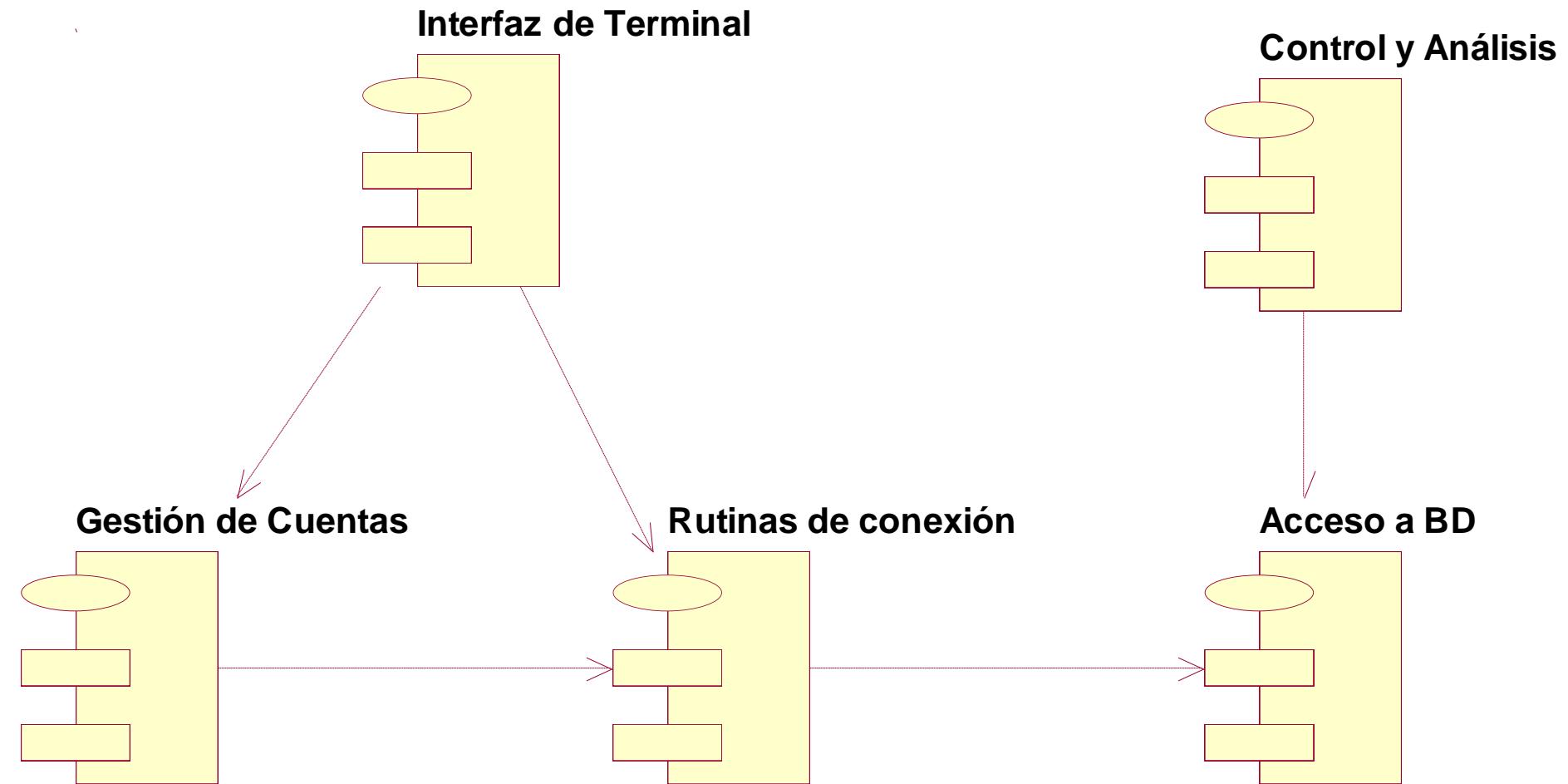
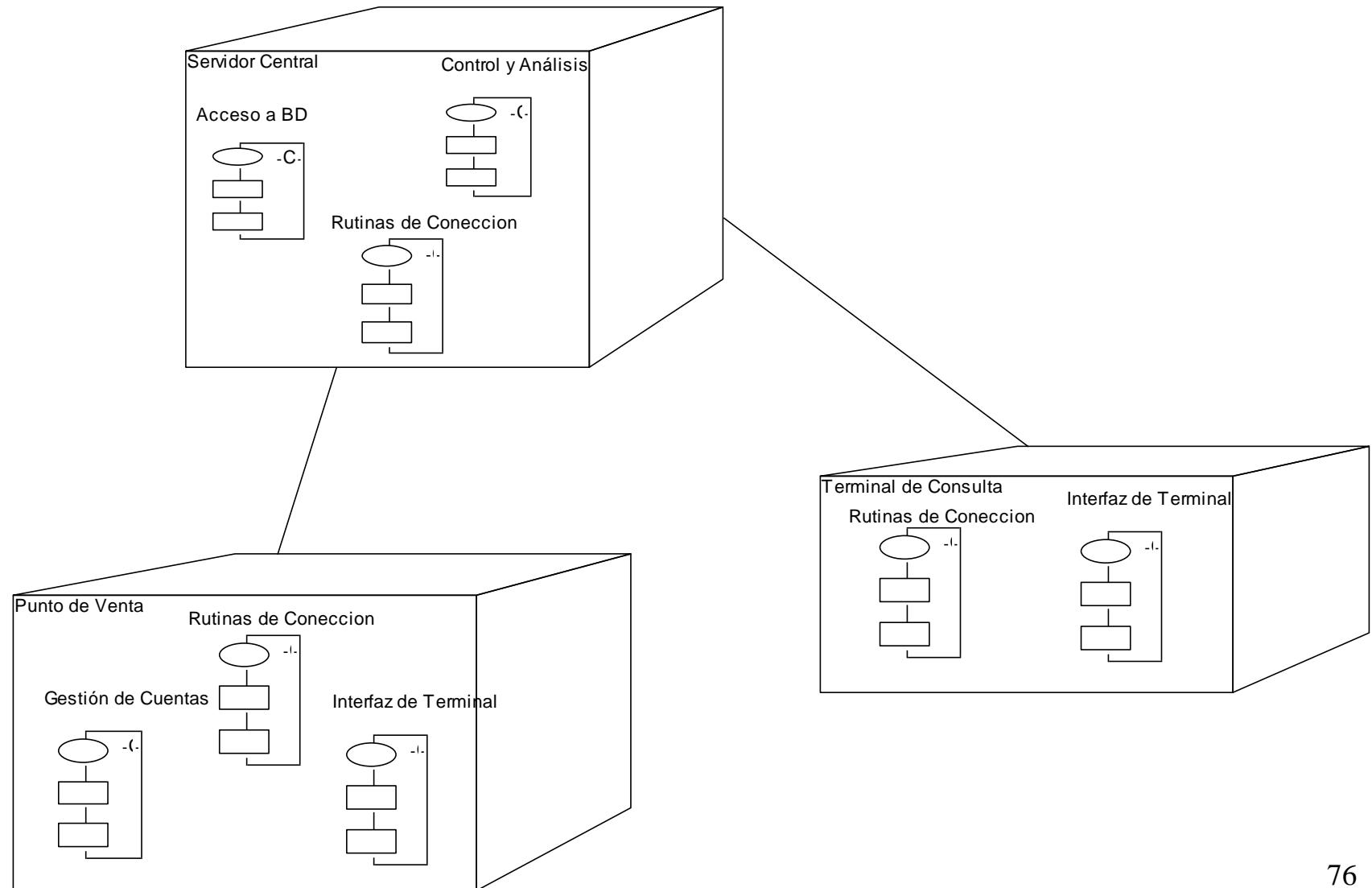
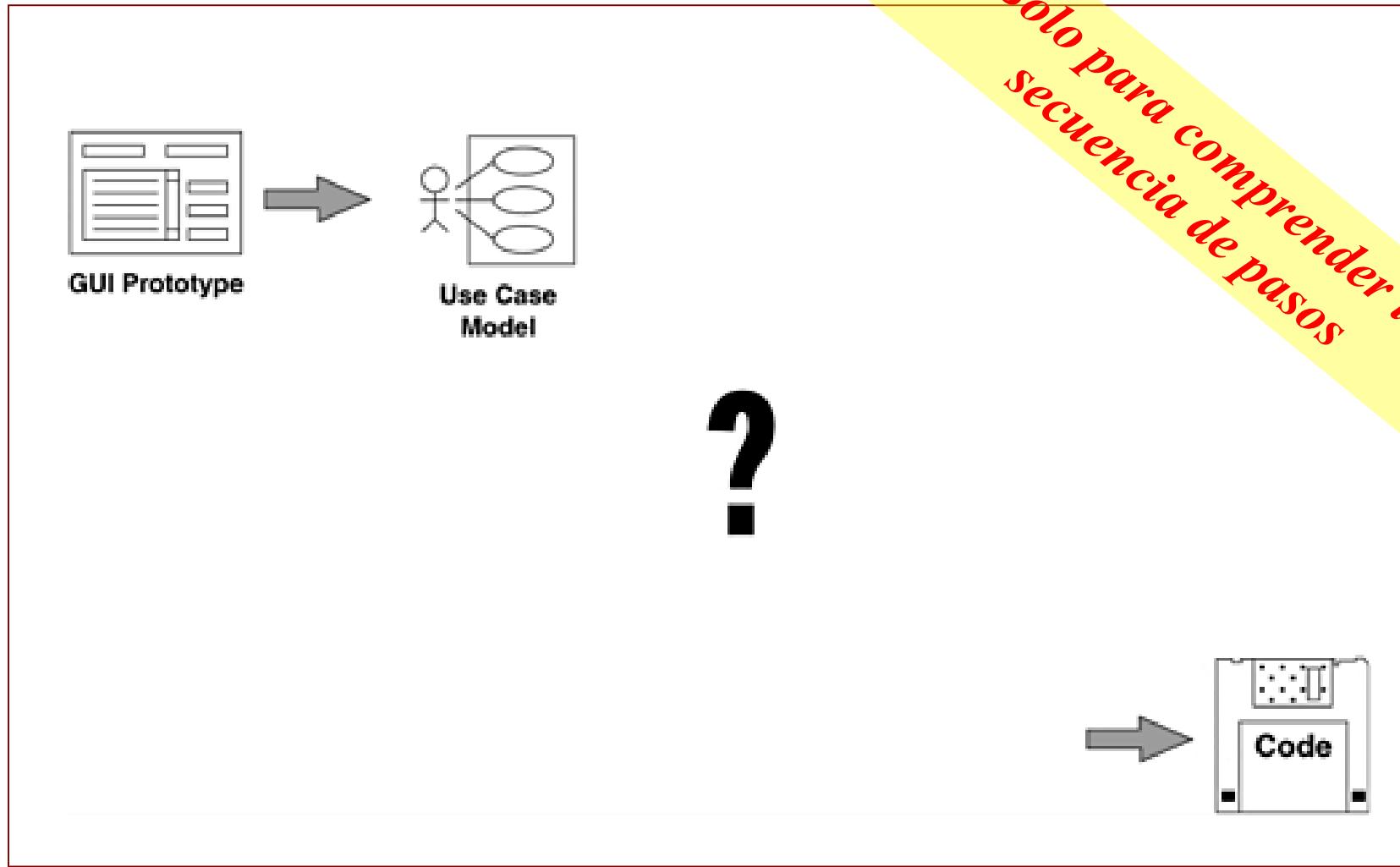


Diagrama de despliegue



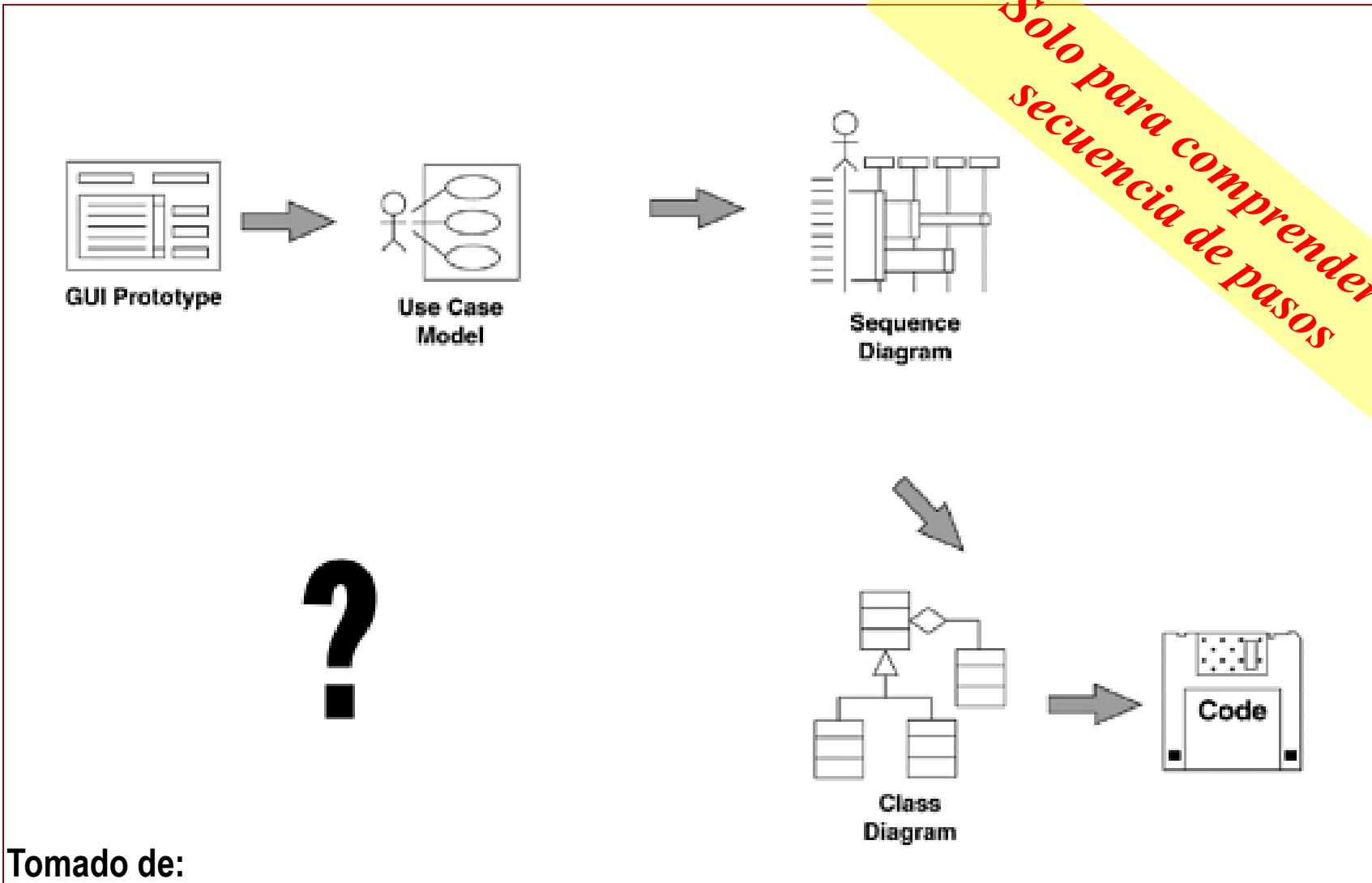
¿Cómo se relacionan los diagramas?



Tomado de:

Rosenberg, Doug, Kendall Scott. Applying use case driven object modeling with UML: an annotated e-commerce example.

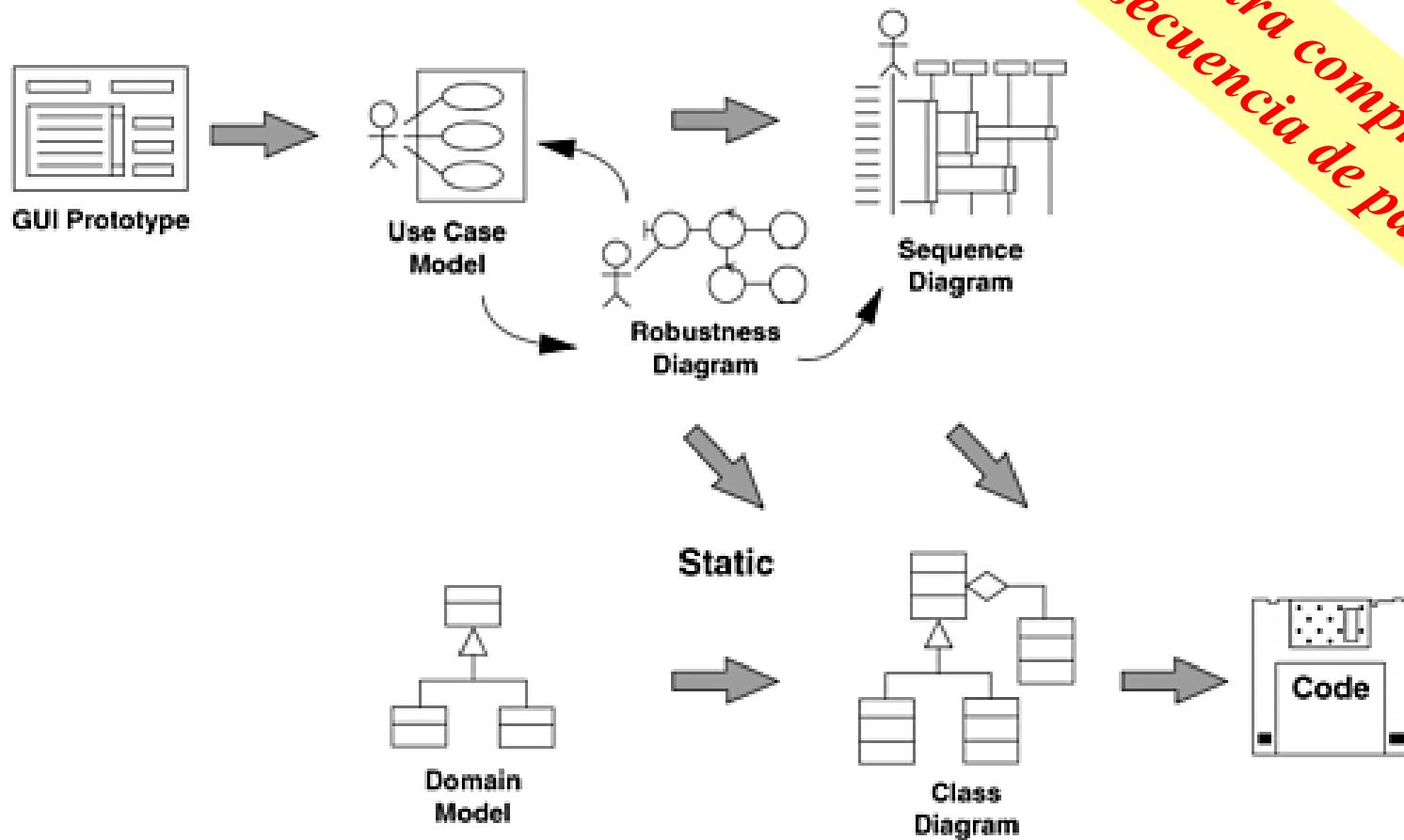
¿Cómo se relacionan los diagramas?



Tomado de:

Rosenberg, Doug, Kendall Scott. Applying use case driven object modeling with UML: an annotated e-commerce example.

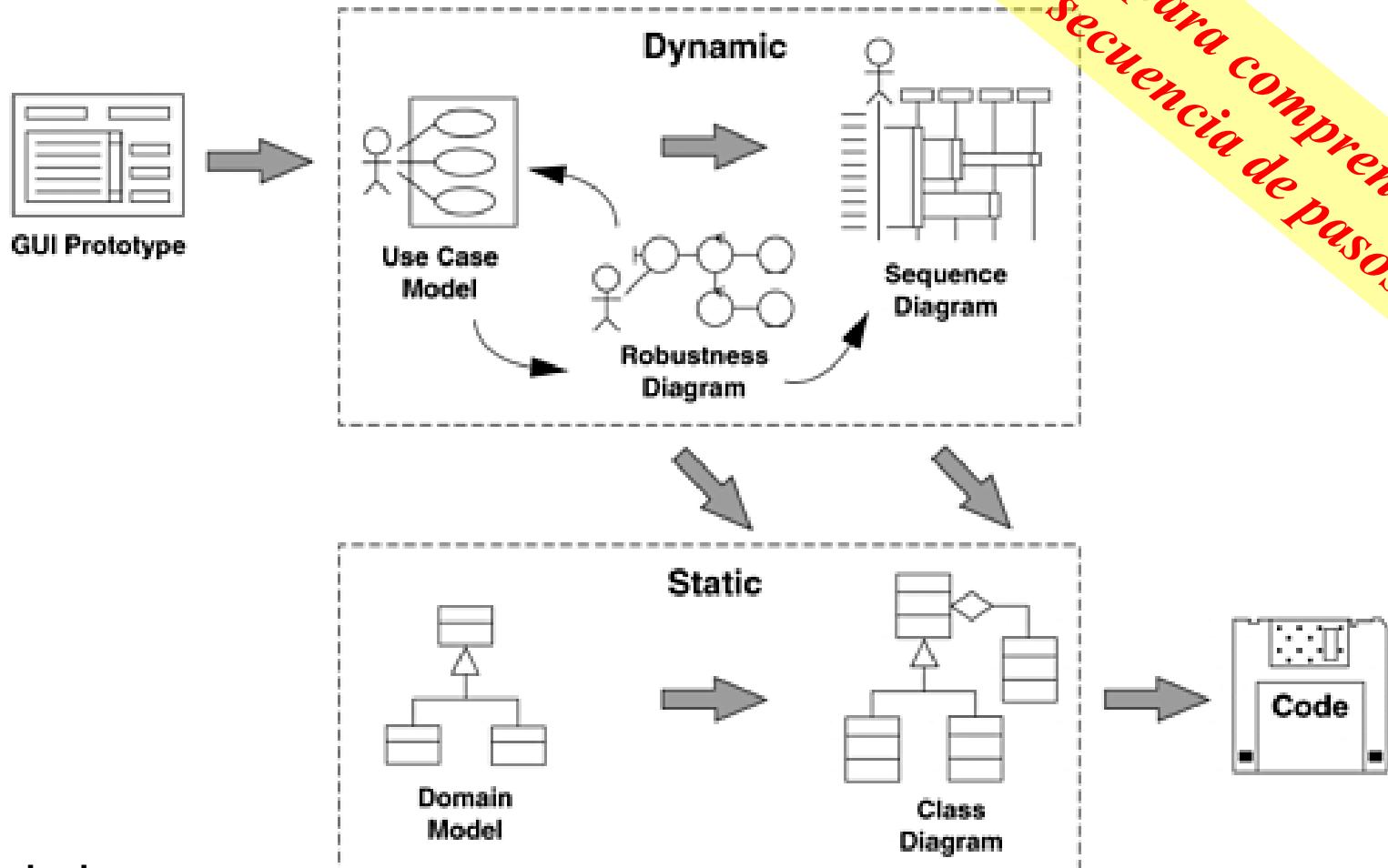
¿Cómo se relacionan los diagramas?



Tomado de:

Rosenberg, Doug, Kendall Scott. *Applying use case driven object modeling with UML: an annotated e-commerce example.*

¿Cómo se relacionan los diagramas?



Tomado de:

Rosenberg, Doug, Kendall Scott. Applying use case driven object modeling with UML: an annotated e-commerce example.

Resumen

- UML define una notación que se expresa como diagramas sirven para representar modelos/subsistemas o partes de ellos
- *El 80 por ciento de la mayoría de los problemas pueden modelarse usando alrededor del 20 por ciento de UML-- Grady Booch*

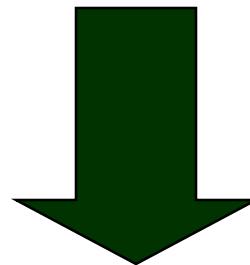


El Proceso Unificado de Desarrollo

(Rational Unified Process- RUP)

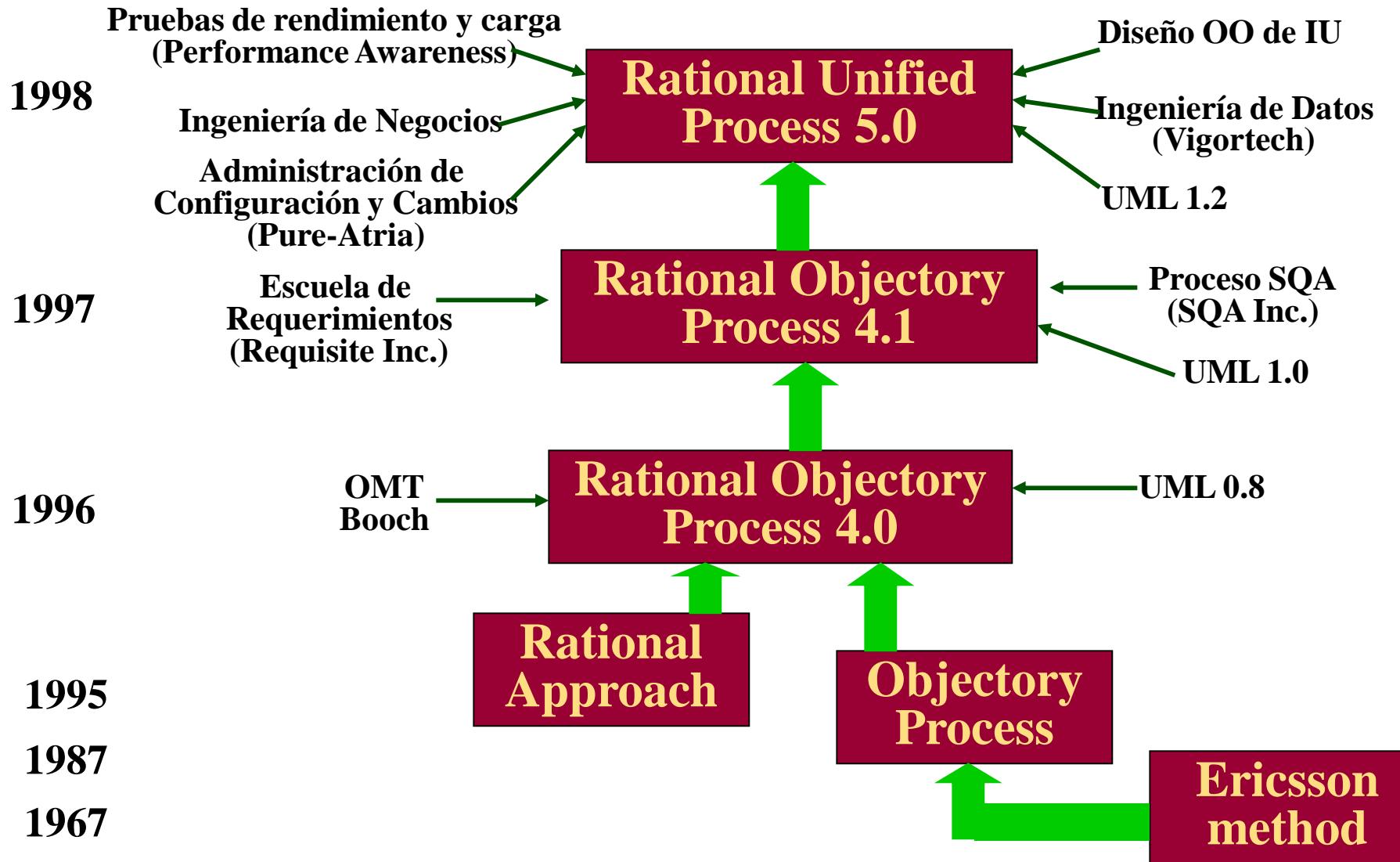
Rational Unified Process

RUP es un proceso para el desarrollo de software orientado a objeto que utiliza UML para describir un sistema



Rational Software Corporation, 1998

Evolución de RUP



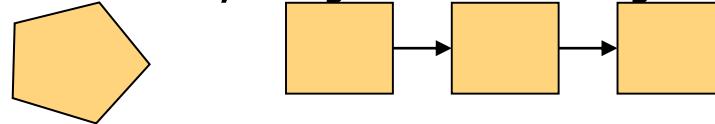
Estructura Estática de RUP

Fases e Iteraciones

¿Cuándo ocurre el proceso?

Disciplinas del Proceso

Actividades, flujo de trabajo



¿Cómo ocurre el proceso y sus detalles?

Artefactos



Modelos, reportes, documentos

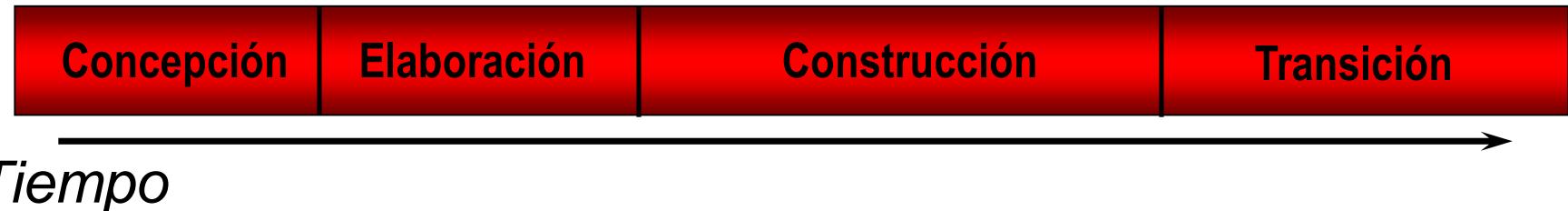
¿Qué se produce u obtiene?

Trabajadores



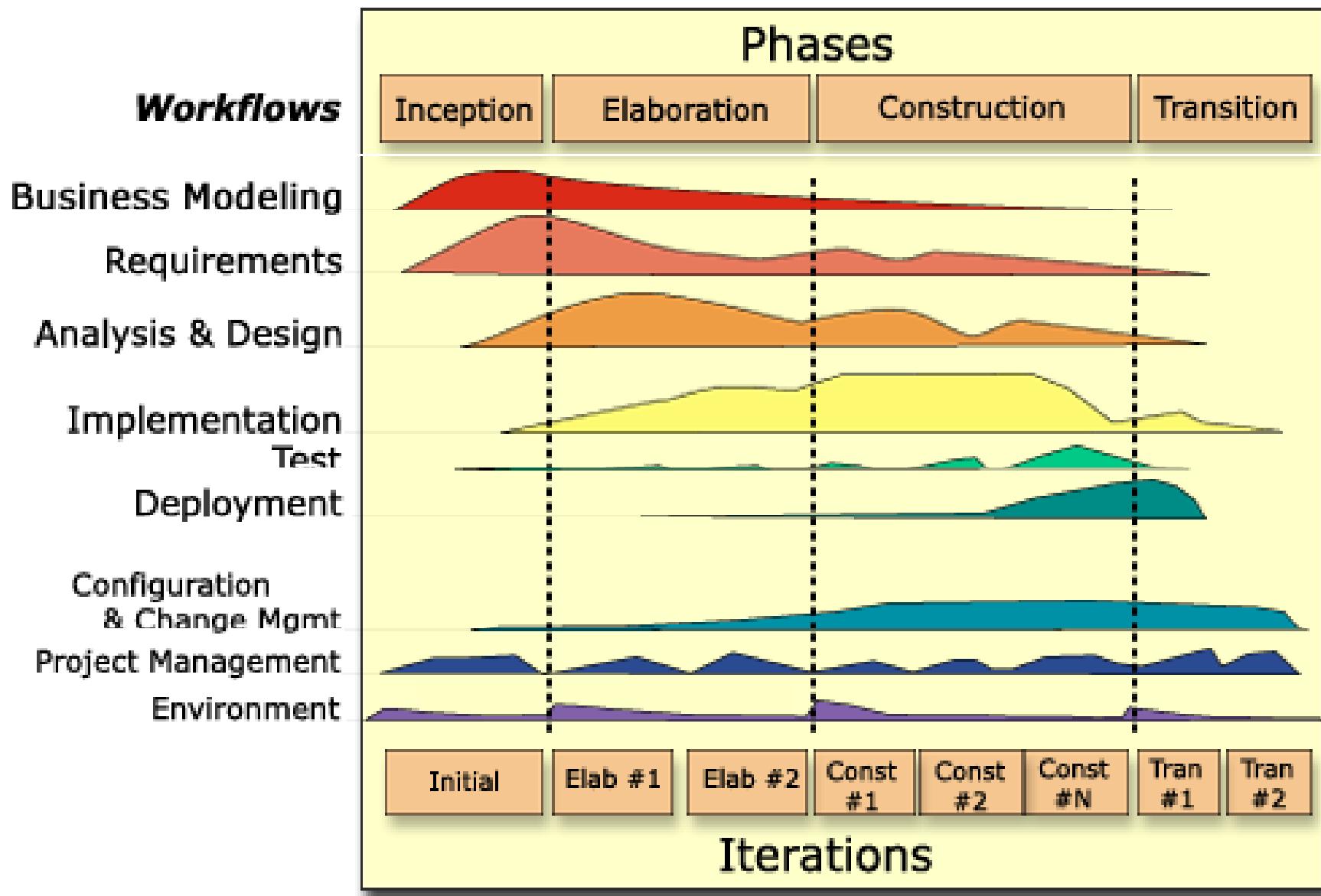
¿Quién lo hace o se responsabiliza?

Estructura Dinámica



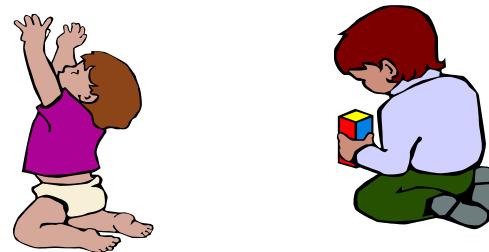
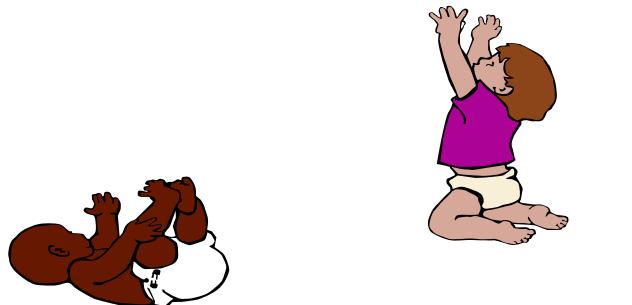
- Concepción Define el alcance del proyecto y el desarrollo de los casos del negocio.
- Elaboración Planifica el proyecto, especifica las características y define los cimientos de la arquitectura.
- Construcción Construye el producto.
- Transición Implementa el producto a sus usuarios.

Fases-Flujos de trabajo de RUP



Características del ciclo de vida de RUP

- Dirigido por Casos de Uso.
- Centrado en la arquitectura.
- Iterativo e incremental.



Ciclo de vida de RUP

Dirigido por Casos de Uso
**(Reflejar lo que los futuros usuarios
necesitan y desean)**

Caso de Uso

Requisitos

Análisis

Diseño

Implementación

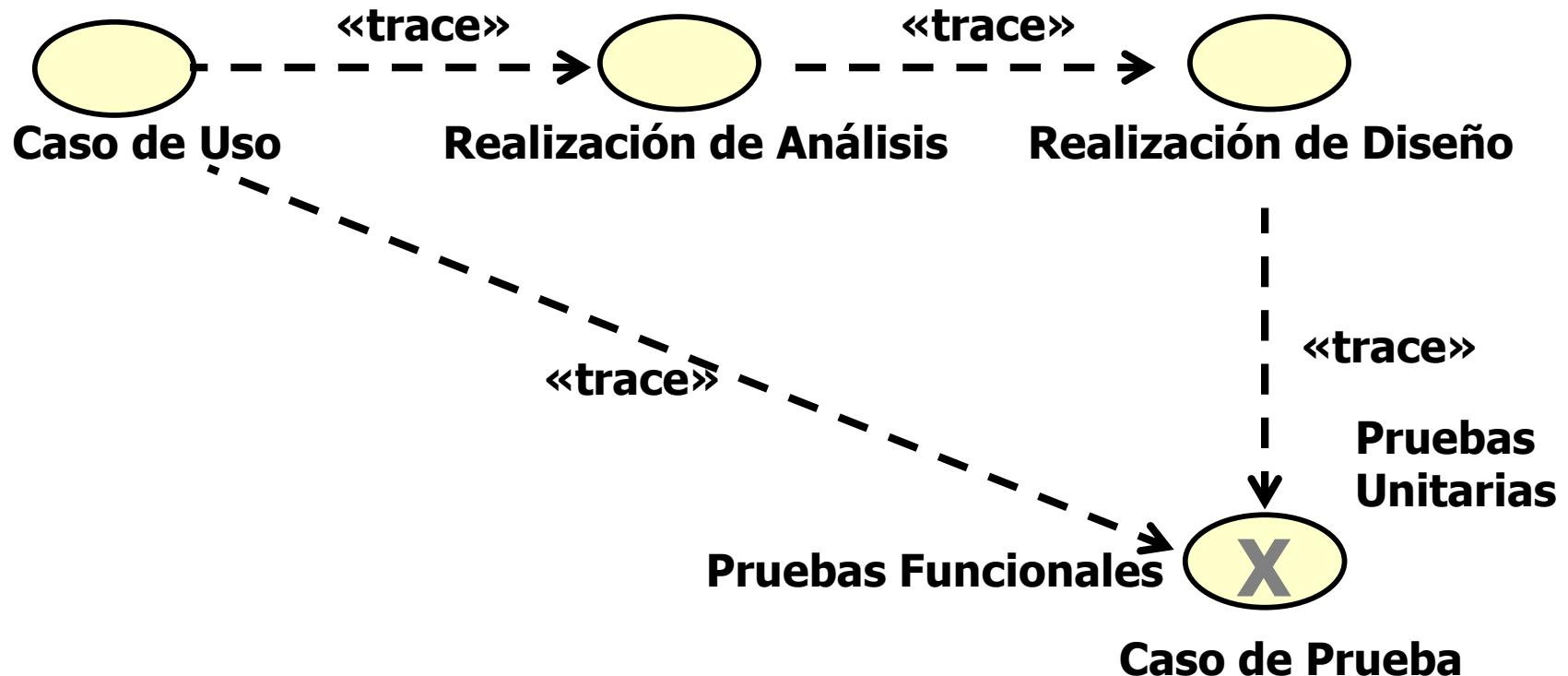
Prueba

Representa los
requerimientos
funcionales

Guían el proceso de desarrollo porque
los modelos que se crean llevan a
cabo los casos de uso.

Ciclo de vida de RUP

Dirigido por Casos de Uso



Ciclo de vida de RUP

Dirigido por Casos de Uso

Estado de aspectos de los Casos de Uso al finalizar cada fase

	Modelo de Negocio Terminado	Casos de Uso Identificados	Casos de Uso Descritos	Casos de Uso Analizados	Casos de Uso Diseñados, Implementados y Probados
Fase de Concepción	50% - 70%	50%	10%	5%	Muy poco, puede que sólo algo relativo a un prototipo para probar conceptos
Fase de Elaboración	Casi el 100%	80% o más	40% - 80%	20% - 40%	Menos del 10%
Fase de Construcción	100%	100%	100%	100%	100%
Fase de Transición					

The Unified Software Development Process. I. Jacobson, G. Booch y J. Rumbaugh. página 358. Addison-Wesley, 1999.

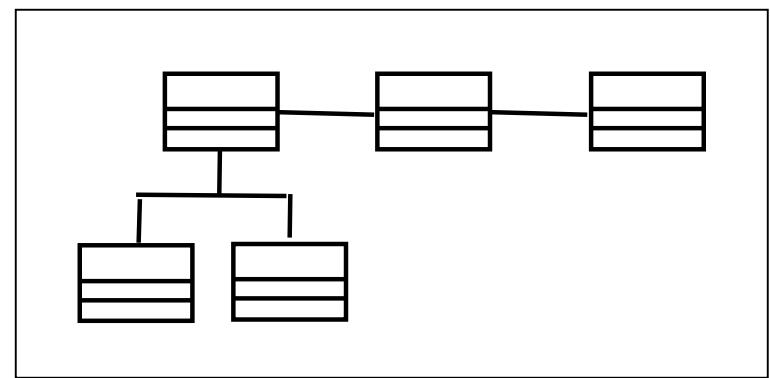
Ciclo de vida de RUP

Centrado en la arquitectura

En la construcción,
vista de:

- A) Estructura.
- B) Calefacción.
- C) Plomería.
- D) Electricidad.

Aspectos



Estáticos

Dinámicos

Ciclo de vida de RUP

Centrado en la arquitectura

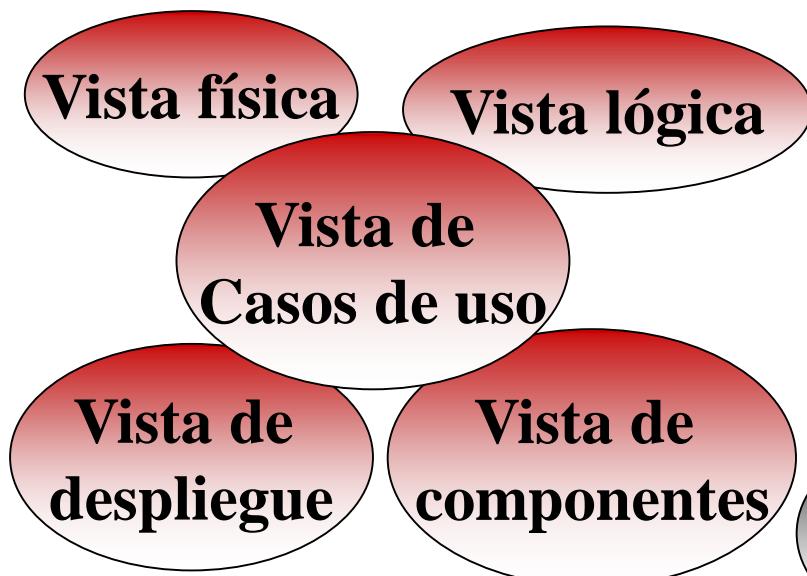
(Visión común del sistema completo en la que desarrolladores y usuarios deben estar de acuerdo)

- Describe los elementos del modelo que son más importantes para su construcción, los cimientos del sistema que son necesarios como base para comprenderlo, desarrollarlo y producirlo económicoamente.
- Se desarrolla mediante iteraciones comenzando por los CU relevantes desde el punto de vista de la arquitectura.

Ciclo de vida de RUP

Centrado en la arquitectura

UML 1.x

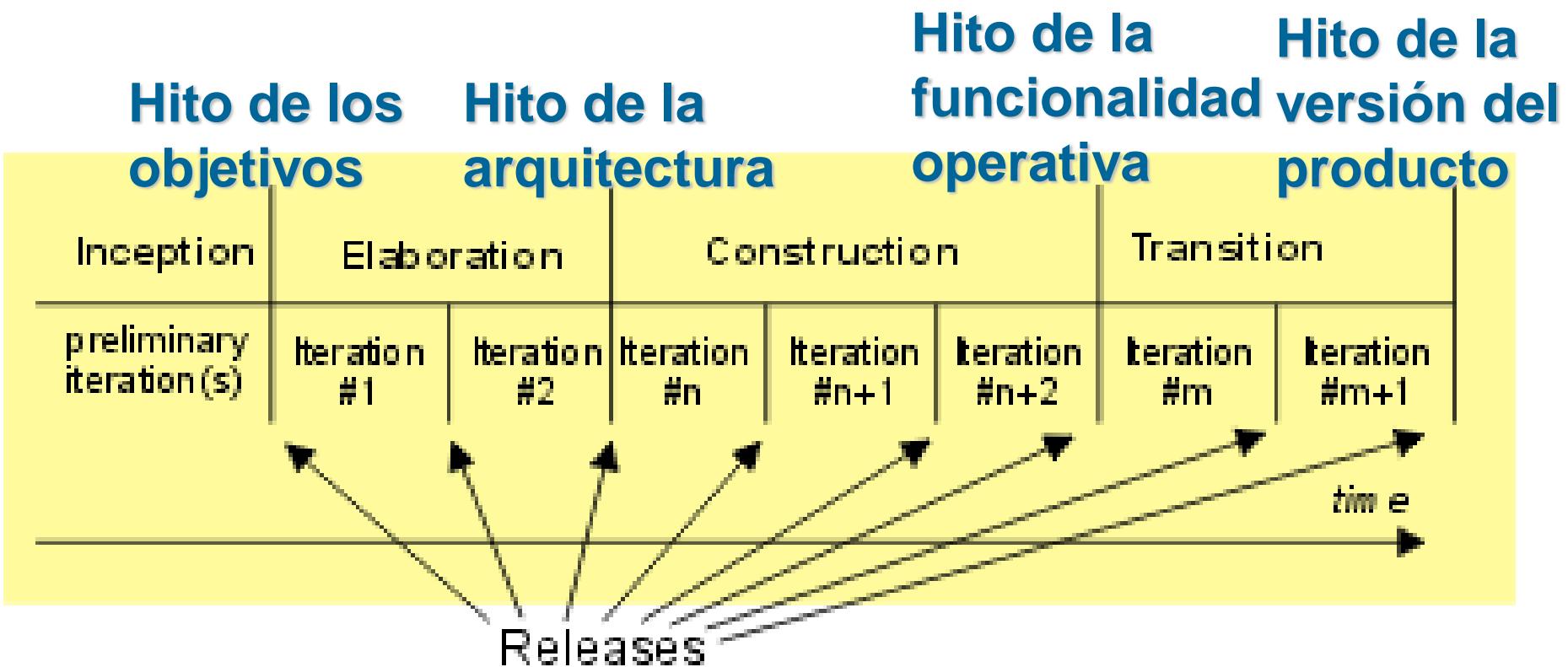


UML 2.0



Ciclo de vida de RUP

Iterativo e incremental

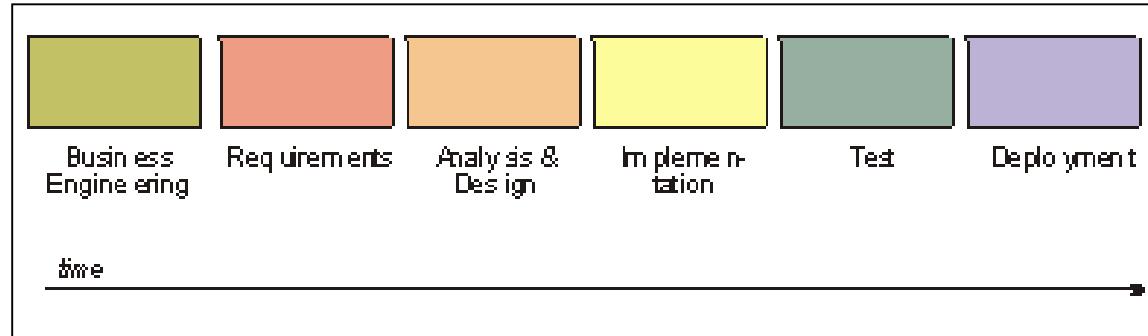


- Dentro de cada fase hay hitos (artefactos a construir) asociados a resultados de cada iteración.
- La terminación de una iteración produce un nuevo incremento.

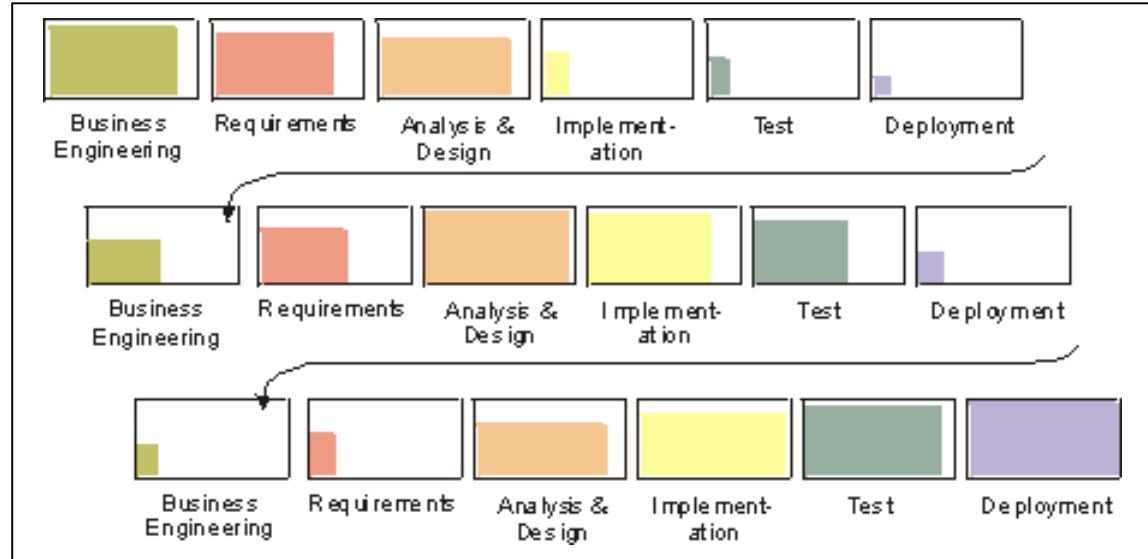
Ciclo de vida de RUP

Iterativo e incremental

Enfoque
Cascada



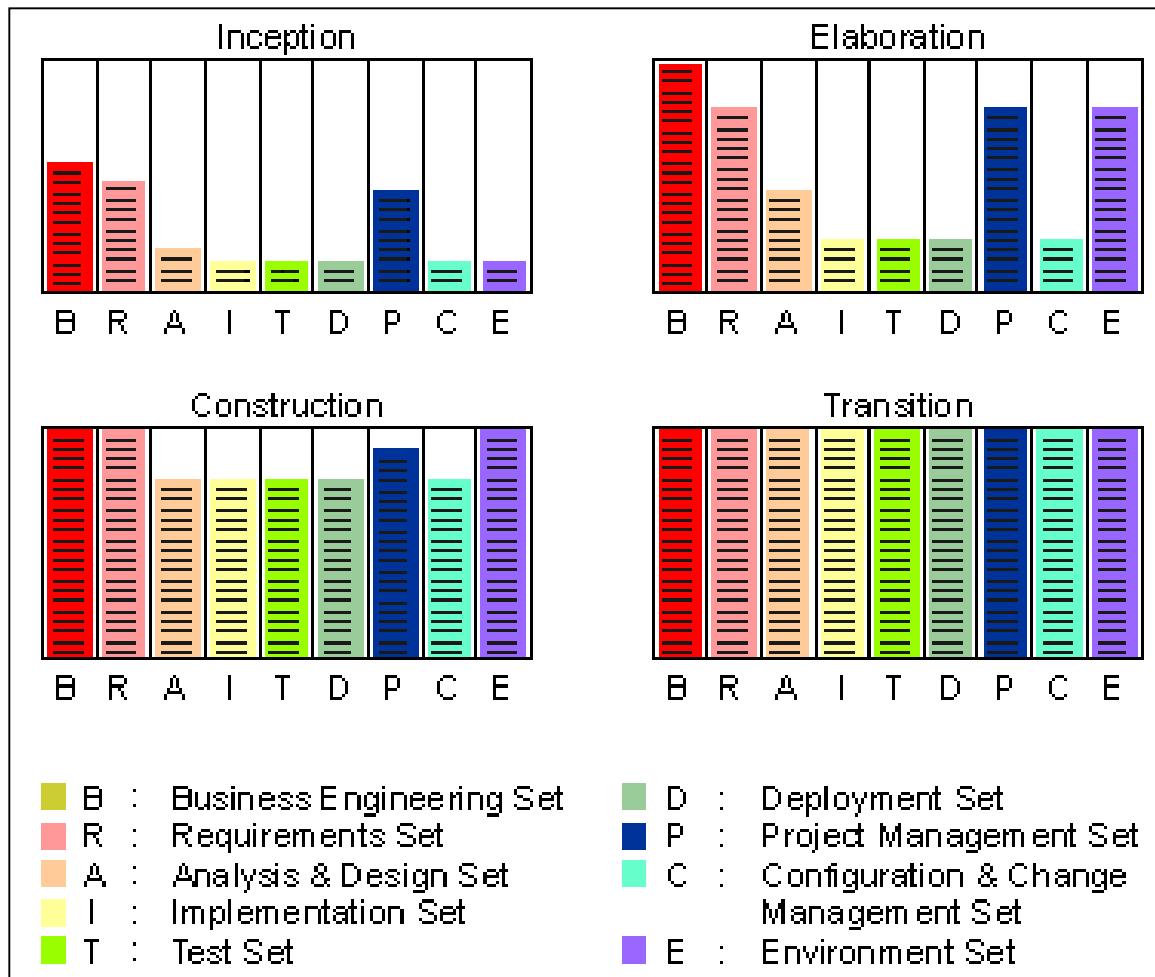
Enfoque
Iterativo e
Incremental



Ciclo de vida de RUP

Iterativo e incremental

Grado de Finalización de Artefactos



Beneficios de la iteración

- Reduce el coste del riesgo al coste de un solo incremento.
- Menos riesgo de no sacar el producto al mercado en fecha.
- Acelera el ritmo de desarrollo.
- Las necesidades del usuario y correspondientes requisitos no pueden definirse completamente al principio. Se requieren iteraciones sucesivas.

RUP

- RUP es un proceso que garantiza la elaboración de todas las fases de un producto de software orientado a objetos.
- RUP es dirigido por los casos de uso, centrado en la arquitectura, iterativo e incremental.
- RUP utiliza el UML.
- UML es un lenguaje que permite la modelación de sistemas con tecnología orientada a objetos

UML y RUP

