13. El proceso unificado de desarrollo de software

Índice

- Referencias
- Introducción
- Análisis y diseño orientado a objetos
- Requisitos
  - Captura de requisitos.
  - Papel de los requisitos.
  - Modelo del dominio.
  - Modelo del negocio.
  - Requisitos adicionales.

Índice

Índice

- Diseño
  - Introducción.
  - El papel de diseño.
  - Artefactos.
- Implementación
  - Introducción.
  - El papel de la implementación
  - Artefactos

Índice

- Prueba
  - Introducción.
  - El papel de la prueba.
  - Artefactos.
- Modelos en el PUD
- Conclusiones

Referencias

- Jacobson I., Booch G., Rumbaugh J., *El proceso unificado de desarrollo de software*. Addison-Wesley 2000
- Booch G., *Análisis y diseño orientado a objetos con aplicaciones*, Segunda edición, Addison-Wesley/Díaz de Santos, 1996

Referencias

- Booch, G., Rumbaugh, J., Jacobson, I. *El lenguaje unificado de modelado. 2ª edición*. Addison-Wesley, 2006
- Rumbaugh, J., Booch, G., Jacobson, I. *El lenguaje unificado de modelado. Manual de referencia*. Addison-Wesley, 2004
- Sommerville I., *Ingeniería del software. 7ª edición*. Pearson Educación, 2005

Introducción

- Proceso de Jacobson, Booch y Rumbaugh
- Los *autores* de UML
  - Booch: método Booch.
  - Rumbaugh: OMT.
  - Jacobson: proceso *Objectory*.
- También conocido por RUP: *Rational Unified Process*

Introducción

- Modelo *basado en componentes*

  - El sistema software está basado en *componentes* software interconectados a través de *interfaces* bien definidos.

  - *Interfaz*: colección de operaciones que son utilizadas para especificar un servicio de una clase o de un componente.

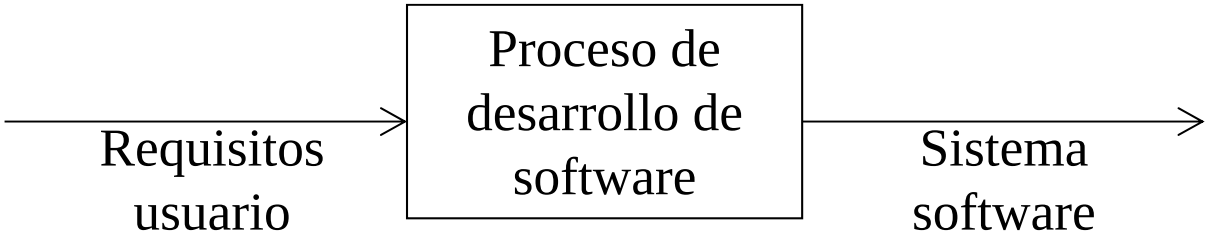
Introducción

- *Componente*: Una parte física y reemplazable de un sistema que se ajusta a, y proporciona la realización de un conjunto de interfaces.

- Muy ligado a UML
- Discusión: ¿cómo diríamos esto en términos de las capas de IS?

Introducción

- Características:
  - Dirigido por casos de uso.
  - Centrado en la arquitectura.
  - Iterativo e incremental.



Un proceso de desarrollo de software

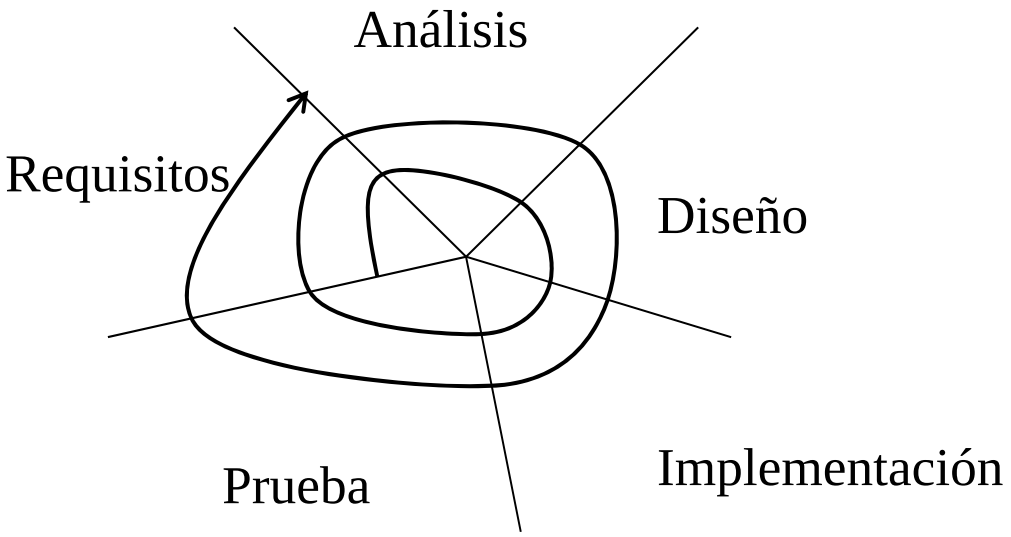
Introducción

- Discusión: ¿todo modelo iterativo es incremental? ¿Y todo incremental es iterativo?

Introducción

- Está formado por cinco *flujos de trabajo* (i.e. AEs) que se iteran:
  - Requisitos.
  - Análisis.
  - Diseño.
  - Implementación.
  - Prueba.

Introducción



El proceso unificado de desarrollo

Introducción

- Cada vuelta en la espiral se denomina *iteración*
- La agrupación de iteraciones se denomina *fase*
  - Inicio.
  - Elaboración.
  - Construcción.
  - Transición.

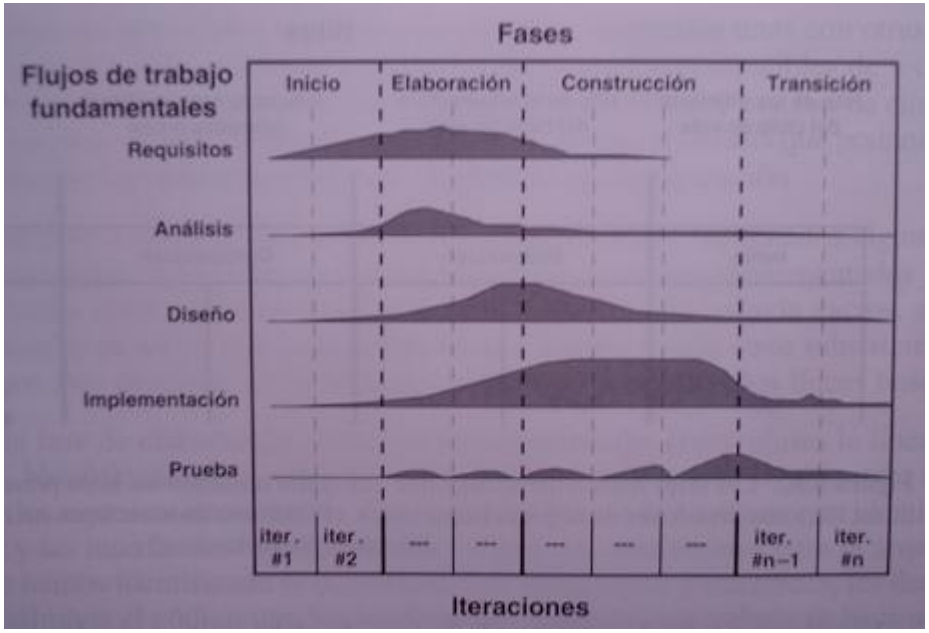
Introducción

- Fase de inicio
  - Se desarrolla una descripción del producto final.
- Fase de elaboración:
  - Se especifican los casos de uso.
  - Se diseña la arquitectura del sistema.

Introducción

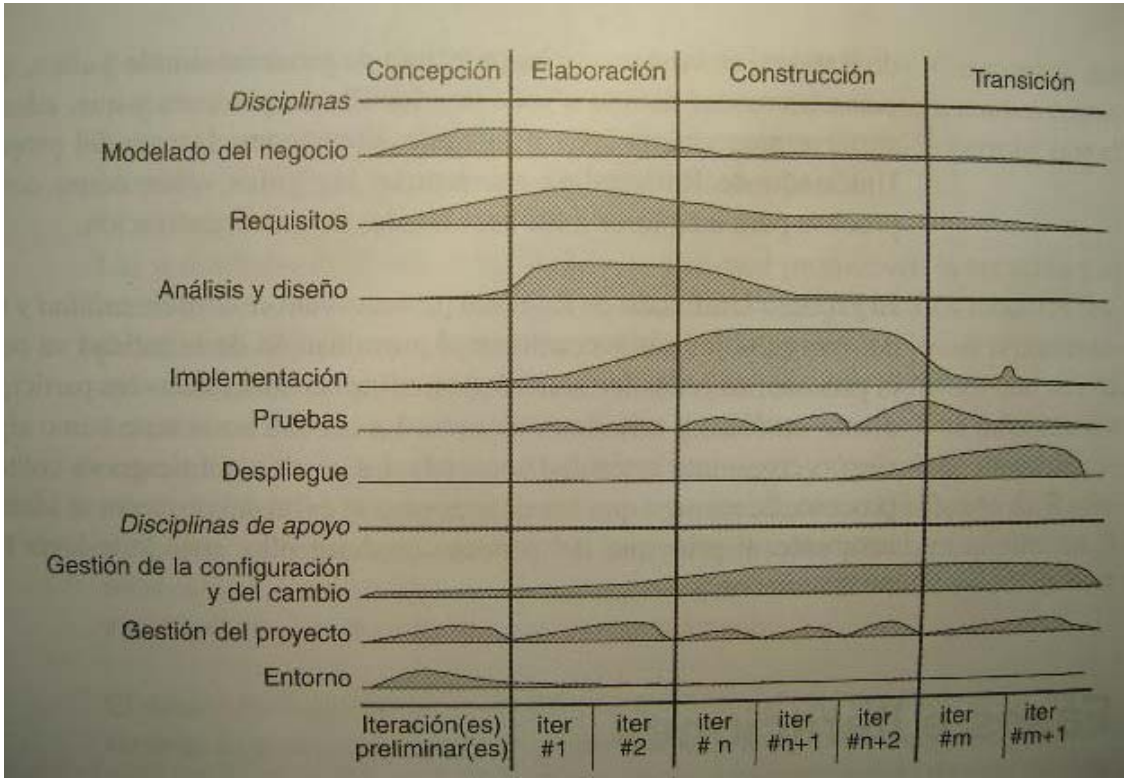
- Fase de construcción
  - Se crea el producto.
- Fase de transición
  - Periodo durante el cual el producto se convierte en versión beta.
- No todos los flujos de trabajo tienen el mismo peso dentro de cada fase

Introducción



Relación entre flujos de trabajo y fases en PUD

Introducción

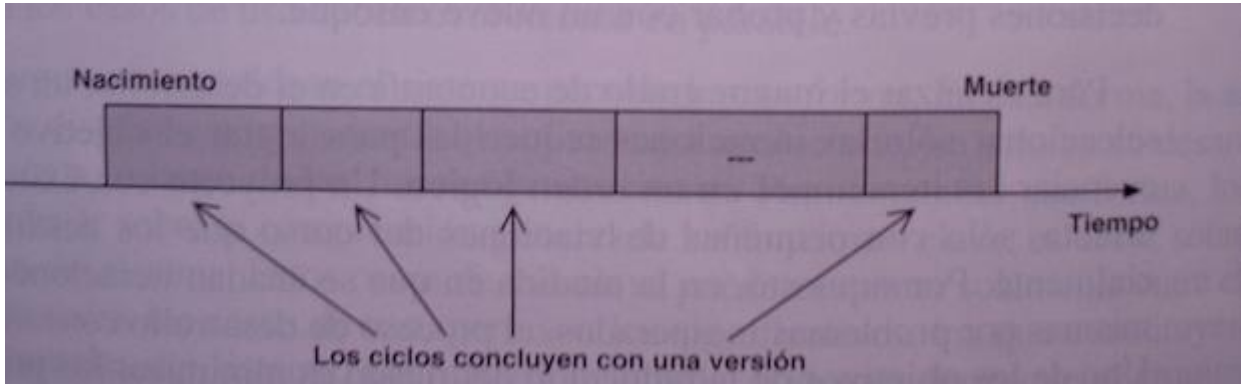


Versión más actual del PUD

Introducción

- Las agrupaciones de fases se denominan *ciclo*
- Cada ciclo concluye con una *versión* del producto
- Discusión: ¿es lo mismo un *ciclo* PUD que un *ciclo* del modelo en espiral?

Introducción



Ciclos en RUP

Introducción

- Ventajas
  - Modelo de proceso *racional*.
  - Tecnologías de componentes.
- Inconvenientes
  - *Muy* ligado al método.
  - No incluye explícitamente actividades de *gestión*.

Análisis y diseño OO

- Exactamente, ¿en qué consiste el análisis y diseño orientado a objetos?
- Aunque vimos algunas definiciones, veamos que definiciones proporciona el Manual de Referencia de UML

Análisis y diseño OO

- *Análisis* es la etapa de un sistema que captura los requisitos y el dominio del problema. El análisis se centra en lo qué hay que hacer, mientras que el diseño se centra en cómo hacerlo.

  - En un proceso iterativo estas etapas no tiene porque realizarse de forma secuencial.

Análisis y diseño OO

- El resultado de esta etapa se representa mediante modelos del nivel de análisis, especialmente la vista de casos de uso y la vista estática.

- *Diseño* es la etapa de un sistema que describe cómo se implementará el sistema en un nivel lógico sobre código real.

  - En el diseño, las decisiones estratégicas y tácticas se toman para resolver los requisitos funcionales y de calidad requeridos de un sistema.

Análisis y diseño OO

- Los resultados de esta etapa son representados por los modelos a nivel de diseño, especialmente la vista estática, vista de máquina de estados y vista de interacción.

- Sin embargo, en el modelo de proceso unificado, la salida del análisis es la especificación de los diagramas de casos de uso a diagramas de interacción entre clases de análisis

Análisis y diseño OO

- En Booch *original*, por ejemplo, en diseño proporciona código.
- Pressman en diseño da la representación a nivel de módulos de las clases
- Sommerville identifica:

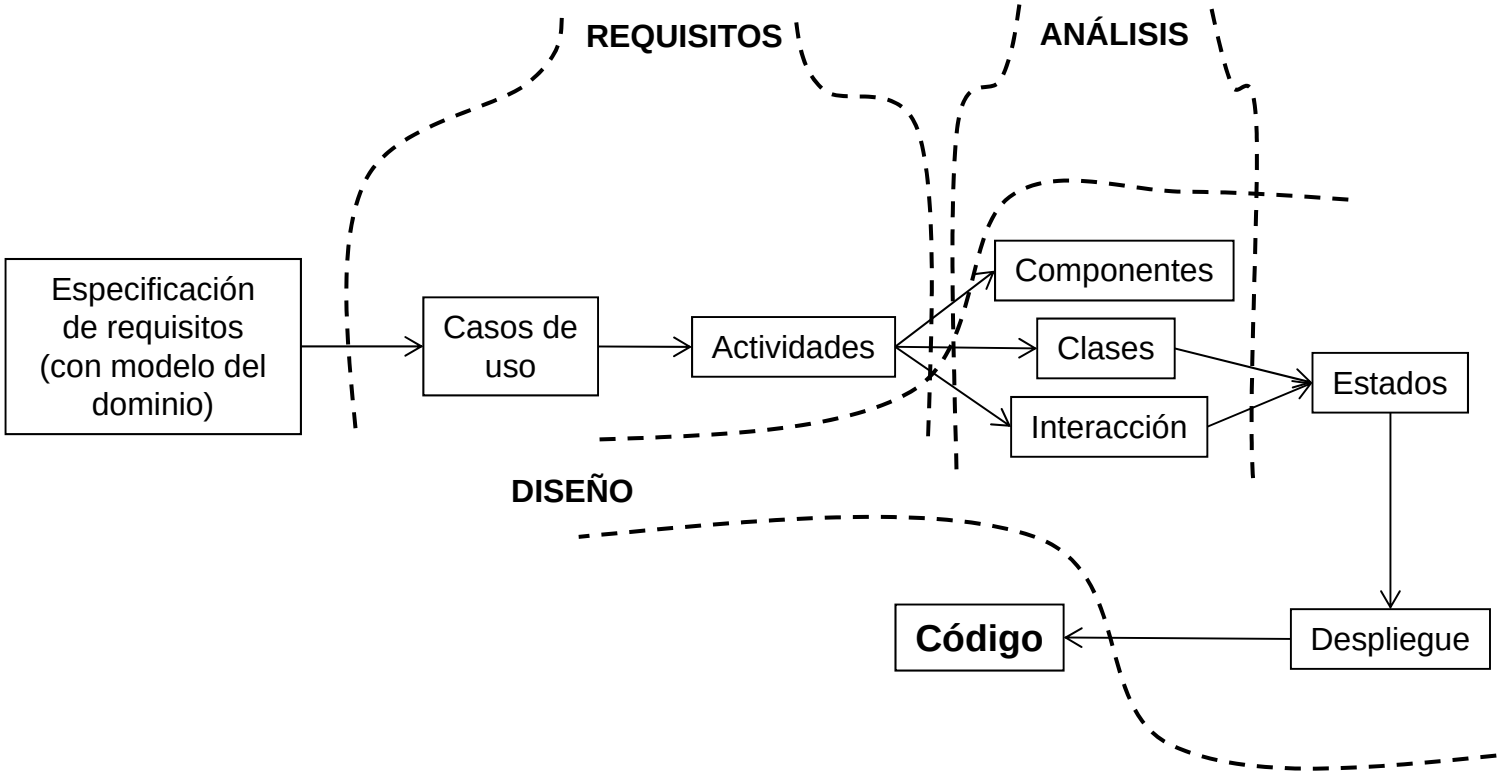
  - *Análisis OO*: centrado en desarrollar un modelo OO del dominio de la aplicación. Los objetos identificados reflejan las entidades y operaciones asociadas con el problema a resolver.

Análisis y diseño OO

- *Diseño OO*: centrado en desarrollar un modelo OO del sistema software para implementar los requisitos identificados. Los objetos en el diseño están relacionados con la solución al problema que se está resolviendo.

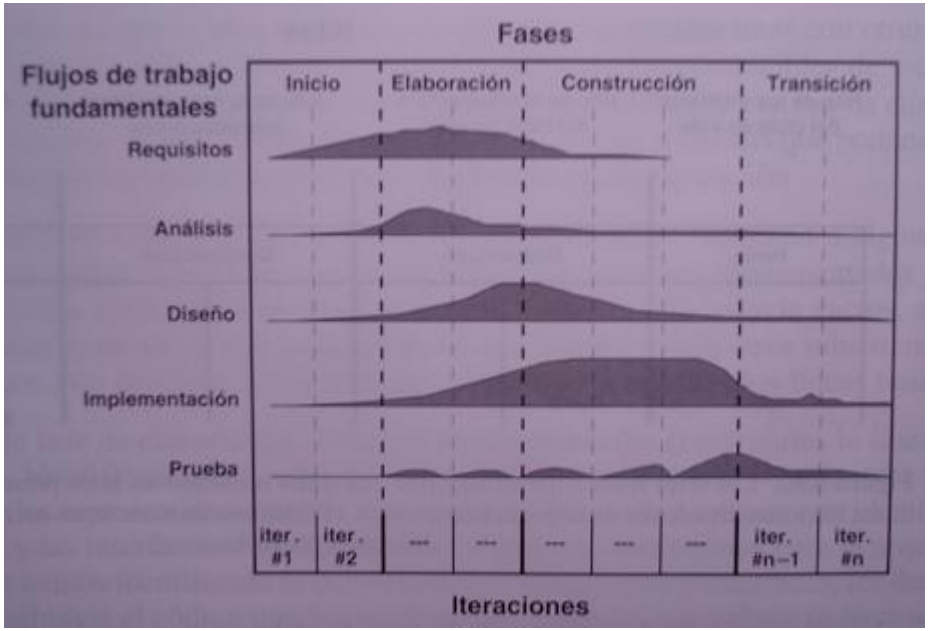
- Con independencia de lo que digan los libros nosotros debemos generar:

Análisis y diseño OO



Análisis y diseño OO en el PUD

Análisis y diseño OO



Relación entre flujos de trabajo y fases en RUP

Requisitos
Captura de requisitos

- Ya hemos hablado en esta asignatura del proceso de captura de requisitos
- El PUD identifica cuatro pasos que no tienen porque llevarse a cabo de forma separada:
  - Enumerar los requisitos candidatos.
  - Comprender el contexto del sistema.
  - Capturar los requisitos funcionales.
  - Capturar los requisitos no funcionales.

Requisitos
Captura de requisitos

- *Enumerar los requisitos candidatos* consiste en identificar todos los posibles requisitos que son susceptibles de ser implementados por el sistema

  - Estos requisitos forman la *lista de características*, en la cual, además del requisito (característica) incluye:

    - *Estado* (propuesto, aprobado, incluido o validado).

    - *Coste estimado de implementación* (tipos de recursos y esfuerzo).

Requisitos
Captura de requisitos

- *Prioridad* (crítico, importante o secundario).
- *Nivel de riesgo* asociado a la implementación de la característica (crítico, significativo u ordinario).

- Esta información sirve para:
  - Estimar el tamaño del proyecto.
  - Decidir cómo dividir el proyecto en una secuencia de iteraciones

Requisitos
Captura de requisitos

- *Comprender el contexto del sistema*, es decir, el dominio de la aplicación

  - Con este fin se proporcionará un:

    - *Modelo del dominio*, que describe los conceptos importantes del contexto como objetos del dominio y enlaza estos objetos entre si.

    - *Modelo del negocio*, que describe los procesos (existentes u observados) con el objetivo de comprenderlos.

Requisitos
Captura de requisitos

- *Capturar requisitos funcionales* identifica los requisitos funcionales utilizando los casos de uso

  - Además de los casos de uso, los analistas deben especificar la apariencia de la interfaz de usuario.

- *Capturar requisitos no funcionales* identifica propiedades del sistema:

  - Restricciones del entorno.

Requisitos
Captura de requisitos

- Restricciones de implementación.
- Rendimiento.
- Dependencias de la plataforma.
- Facilidad de mantenimiento.
- Extensibilidad.
- Fiabilidad.
- Otros.

Requisitos
El papel de los requisitos

- El papel de los requisitos depende de la fase de desarrollo del software en la que nos encontremos
- Durante la fase de *inicio*, los analistas identifican la mayoría de los casos de uso para delimitar el sistema y el alcance del proyecto y para detallar los más importantes (menos del 10%)

Requisitos
El papel de los requisitos

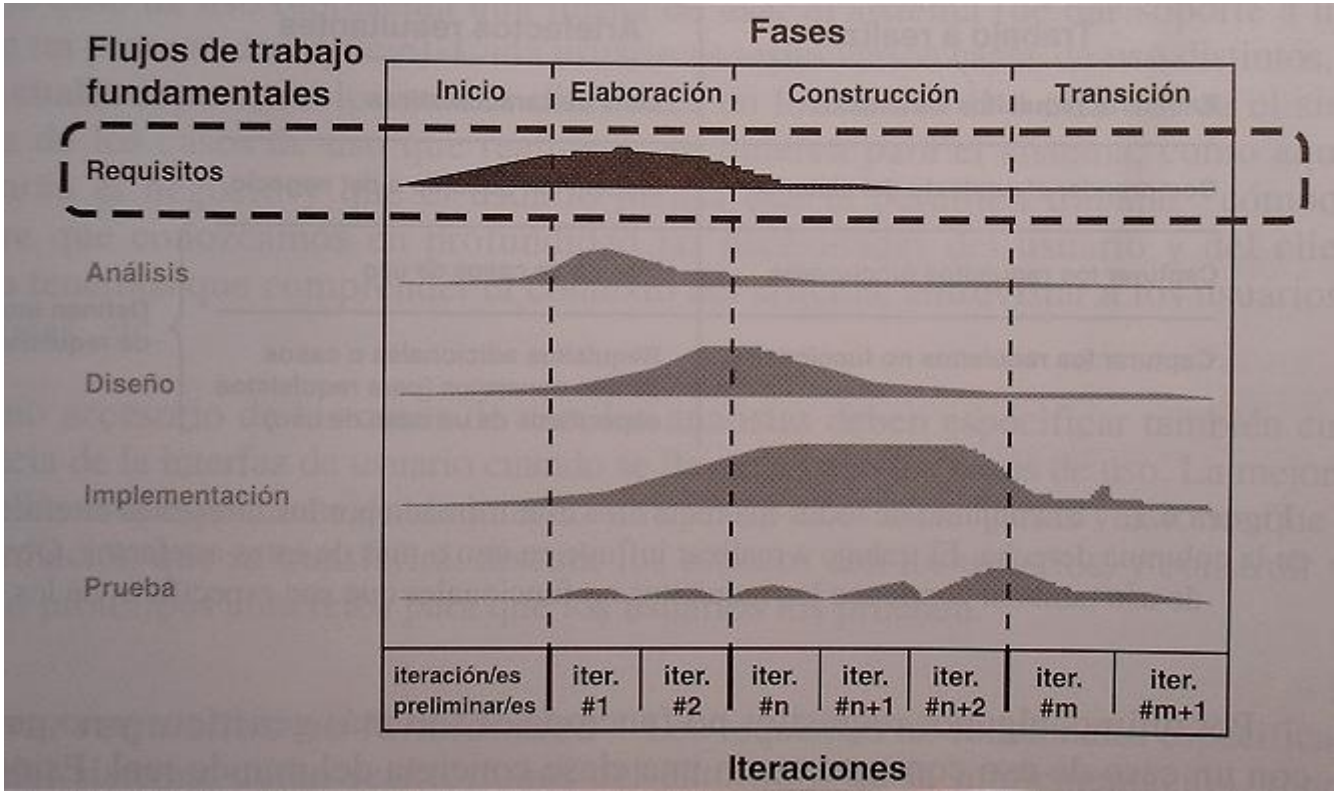
- Durante la fase de *elaboración*, los analistas capturan la mayoría de los requisitos restantes para que los desarrolladores puedan estimar el tamaño del esfuerzo de desarrollo que se requerirá

  - El objetivo es haber capturado un 80% de los requisitos y haber descrito la mayoría de los casos de uso.

Requisitos
El papel de los requisitos

- Los requisitos restantes se capturan e implementan durante la fase de *construcción*
- Casi no hay captura de requisitos en la fase de *transición*, a menos que haya requisitos que cambien

Requisitos
El papel de los requisitos



El papel de los requisitos en el proceso de desarrollo

Requisitos
Modelo del dominio

- Un *modelo del dominio* captura los tipos más importantes de objetos en el contexto del sistema
- Los *objetos del dominio* representan las *cosas* que existen o los eventos que suceden en el entorno en el que trabaja el sistema

Requisitos
Modelo del dominio

- Muchos de los objetos del dominio o *clases* pueden obtenerse de una especificación de requisitos o mediante la entrevista con los expertos del dominio
- Las clases del dominio aparecen en tres formas típicas:

  - *Objetos del negocio*, que representan cosas que se manipulan en el negocio, como pedidos, cuentas, contratos, etc.

Requisitos
Modelo del dominio

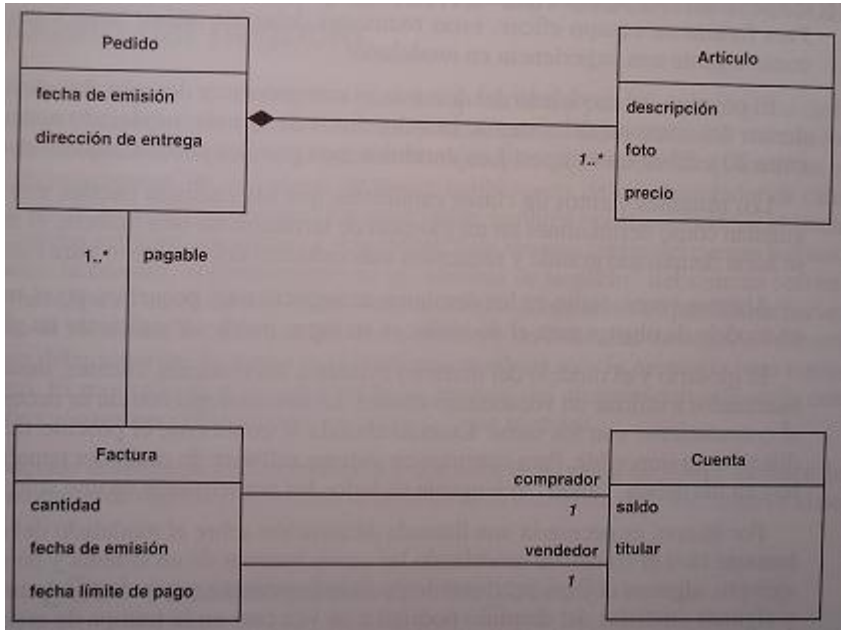
- *Objetos del mundo real* y conceptos de los que el sistema debe hacer un seguimiento, como la aviación enemiga, misiles y trayectorias.

- *Sucesos* que ocurrirán o han ocurrido, como la llegada de un avión, su salida y la hora de la comida.

- El modelo del dominio se describe mediante diagramas UML (especialmente de clases)

Requisitos
Modelo del dominio

- Ejemplo:



Modelo del dominio

Requisitos
Modelo del dominio

- El modelado del dominio se realiza habitualmente en reuniones organizadas por los analistas del dominio
- El objetivo del modelado del dominio es comprender y describir las clases más importantes dentro del contexto del sistema

Requisitos
Modelo del dominio

- Los dominios de tamaño moderado normalmente requieren entre 10 y 50 de estas clases
- Las restantes clases candidatas que los analistas pueden extraer del dominio se guardan como definiciones en un glosario de términos

Requisitos
Modelo del dominio

- En dominios muy pequeños, no es necesario desarrollar un modelo de objetos para el dominio; en su lugar puede ser suficiente un glosario de términos
- El modelo del dominio proporciona un vocabulario común a clientes y desarrolladores

Requisitos
Modelo del dominio

- El objetivo del modelado del dominio es contribuir a la comprensión del contexto del sistema, y por extensión de los requisitos que se desprenden de este contexto
- Es decir, el modelo del dominio contribuye a una comprensión del *problema* que el sistema resuelve en relación a su contexto

Requisitos
Modelo del dominio

- El modo interno por el cual el sistema resuelve este problema se tratará en los flujos de trabajo de análisis, diseño e implementación
- Las clases del dominio y el glosario de términos se utilizan en el desarrollo de los modelos de casos de uso y análisis:

  - Al describir los casos de uso y al diseñar la interfaz de usuario.

Requisitos
Modelo del dominio

- Para sugerir clases internas al sistema de desarrollo durante el análisis.

- También podemos concebir al modelo del dominio como un caso especial de un modelo del negocio más completo

Requisitos
Modelo del negocio

- El *modelado del negocio* es una técnica para comprender los procesos de negocio de la organización
- Se debe modelar el sistema del negocio/entidad que rodea al sistema software a desarrollar

Requisitos
Modelo del negocio

- El modelado del negocio está soportado por dos tipos de modelos UML: modelos de casos de uso y modelos de objetos
- El *modelo de casos de uso del negocio* describe los procesos de negocio de una empresa en términos de casos de uso del negocio y actores del negocio que se corresponden con los procesos del negocio y los clientes, respectivamente.

Requisitos
Modelo del negocio

- El modelo de casos de uso del negocio presenta un sistema desde la perspectiva de su uso, y esquematiza cómo proporciona valor a sus usuarios
- Por ejemplo, en el contexto de un consorcio de compras y ventas, podemos considerar un caso de uso del negocio que comprende la venta desde el pedido a la entrega

Requisitos
Modelo del negocio

- El proceso de venta es el siguiente:
    1. El comprador hace el pedido de bienes o servicios.
    2. El vendedor entrega los bienes o servicios.
    3. El vendedor envía la factura al comprador.
    4. El comprador paga.
- En este contexto, el comprador y el vendedor son los actores del negocio, y utilizan el caso de uso de negocio que la entidad les proporciona.

Requisitos
Modelo del negocio

- El modelo de casos de uso del negocio se describe mediante diagramas de casos de uso.
- Un *modelo de objetos del negocio* es un modelo interno a un negocio
- Describe como cada caso de uso de negocio es llevado a cabo por parte de un conjunto de trabajadores que utilizan un conjunto de entidades del negocio y unidades de trabajo.

Requisitos
Modelo del negocio

- Cada realización de un caso de uso del negocio puede mostrase en diagramas de interacción y diagramas de actividad
- Un *trabajador* es un puesto que puede ser asignado a una persona o equipo, y que requiere responsabilidades y habilidades como realizar determinadas actividades o desarrollar determinados artefactos

Requisitos
Modelo del negocio

- Una *entidad* del negocio representa algo, como una factura, que los trabajadores toman, inspeccionan, manipulan, producen o utilizan en un caso de uso del negocio
- Una *unidad de trabajo* es un conjunto de esas entidades que conforma un todo reconocible para un usuario final

Requisitos
Modelo del negocio

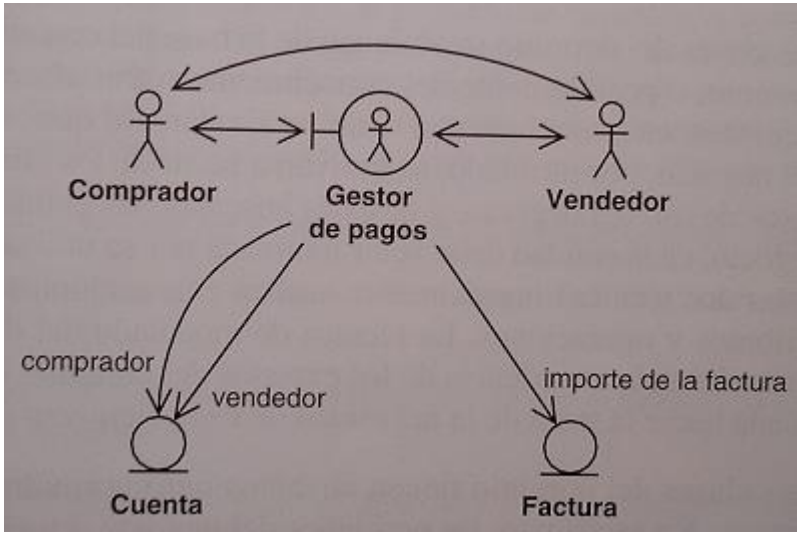
- Las entidades del negocio y las unidades de trabajo se utilizan para representar los mismos tipos de conceptos que las clases del dominio como `Pedido`, `Articulo`, `Factura` y `Cuenta`
- Por tanto, podemos confeccionar un diagrama de las entidades del negocio, muy parecido modelo del dominio

Requisitos
Modelo del negocio

- También tendremos otros diagramas para mostrar los trabajadores, sus interacciones, y cómo utilizan las entidades de negocio y las unidades de trabajo

Requisitos
Modelo del negocio

- Ejemplo:



Interacciones en el modelo del negocio

Requisitos
Modelo del negocio

- Cada trabajador, entidad del negocio, y unidad de trabajo pueden participar en la realización de más de un caso de uso del negocio
- El modelo del negocio se desarrolla en dos pasos:

  1. Desarrollar un modelo de casos de uso del negocio que identifique a los actores del negocio y los casos de uso.

Requisitos
Modelo del negocio

2. Desarrollar un modelo de objetos del negocio compuesto por trabajadores, entidades del negocio y unidades de trabajo que juntos realizan los casos de uso del negocio.

- Nótese que el modelo del dominio se puede ver como una variante simplificada del modelo del negocio, en la cual nos centramos sólo en las “cosas”, es decir, en las clases del dominio o entidades del negocio

Requisitos
Modelo del negocio

- Por tanto, las clases del dominio y las entidades del negocio son conceptos muy parecidos, y utilizamos ambos términos indistintamente
- Sin embargo existen algunas diferencias importantes entre el modelado del negocio y el modelado del dominio que hacen mucho más recomendable realizar el modelado del negocio

Requisitos
Modelo del negocio

- Las clases del dominio se obtienen del conocimiento de expertos del dominio o del conocimiento de sistemas similares. Las entidades del negocio se derivan a partir de los clientes, identificando los casos de uso del negocio y buscando después las entidades.

Requisitos
Modelo del negocio

- Las clases del dominio tienen atributos pero normalmente ninguna o muy pocas operaciones. Por el contrario, las entidades del negocio incluye información de cómo utilizan estas entidades los trabajadores que participan en la realización de los casos de uso del negocio.

Requisitos
Modelo del negocio

- Los trabajadores identificados en el modelado del negocio y los casos de uso del negocio se utilizan como punto de partida para derivar un primer conjunto de actores y *casos de uso del sistema* de información en construcción*.

*En general, *casos de uso de la aplicación* o simplemente *casos de uso*

Requisitos
Requisitos adicionales

- Los *requisitos adicionales* son fundamentalmente requisitos no funcionales que no pueden asociarse a ningún caso de uso concreto, pero que sí pueden a afectar a varios o ningún caso de uso
- Los requisitos adicionales se capturan mediante una lista de requisitos, que se tiene en cuenta durante análisis y diseño

Requisitos
Requisitos adicionales

- Los requisitos adicionales podemos clasificarlos en:

  - *Requisitos de interfaz*. Especifican la interfaz con un elemento externo con el cual debe interactuar el sistema, o establecen restricciones condicionantes en formatos, tiempos u otros factores relevantes en esa interacción.

Requisitos
Requisitos adicionales

- *Requisitos físicos*. Especifican una característica física que debe poseer un sistema, como su material, forma, tamaño o peso.

- *Restricciones de diseño*. Limitan el diseño de un sistema, como lo hacen las restricciones de extensibilidad y mantenibilidad, o las restricciones relativas a la reutilización de sistemas heredados o partes esenciales de los mismos.

Requisitos
Requisitos adicionales

- *Restricciones de implementación*. Especifican o limitan la codificación o construcción de un sistema, tales como estándares requeridos, normas de codificación, lenguajes de programación, etc.

- *Otros requisitos*. Tales como los legales y las normativas

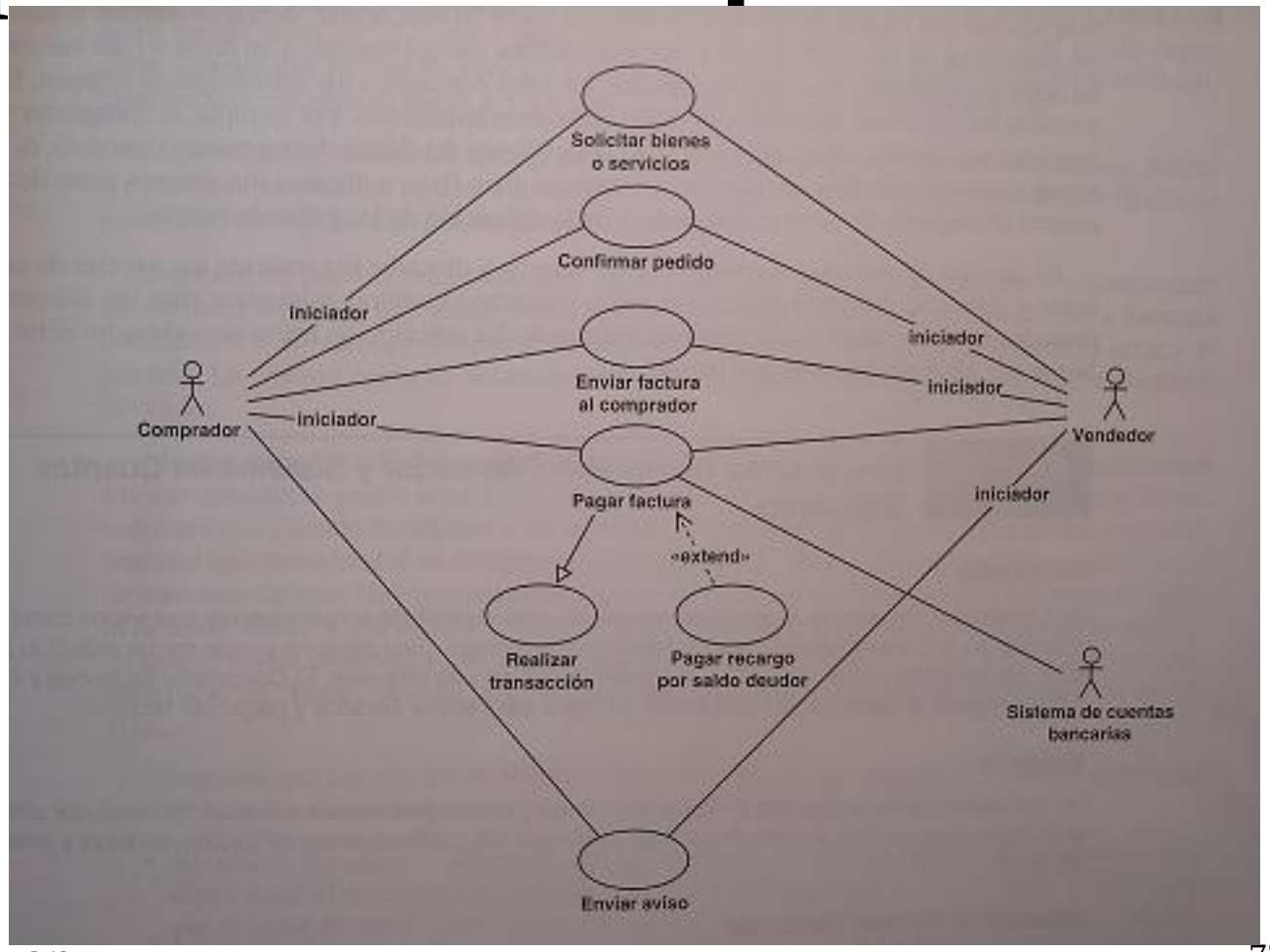
Requisitos
Captura de los requisitos ...

- Los casos de uso de la aplicación proporcionan un medio intuitivo y sistemático para capturar los requisitos funcionales, con un énfasis especial en el valor añadido para cada usuario individual o para cada sistema externo
- Además, sirven como guía para el proceso de desarrollo

Requisitos
Captura de los requisitos ...

- Ej.:

Casos de uso de aplicación de un sistema de Pagos y Facturación

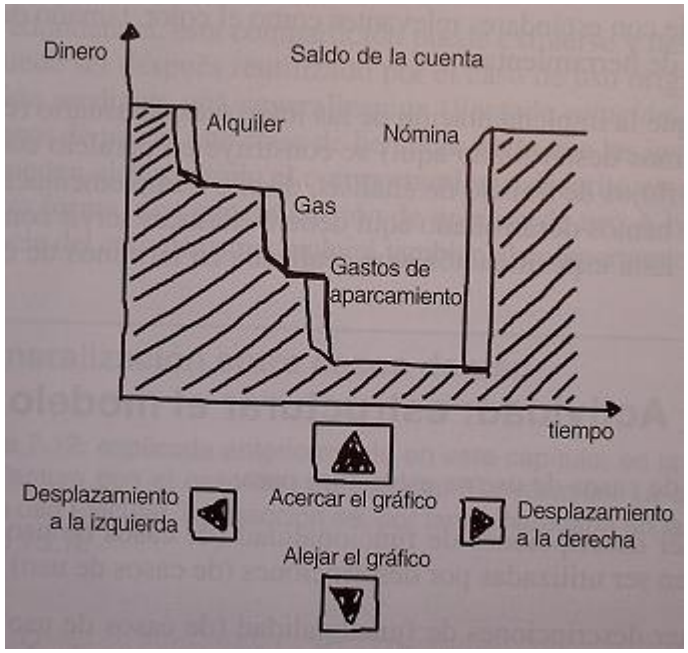


Requisitos
Captura de los requisitos ...

- Como ya hemos comentado, los diagramas de actividades o de interacción pueden especificar a los casos de uso
- Además, durante este proceso se proporciona la caracterización de la interfaz de usuario, bien mediante dibujos en papel, bien mediante prototipos de dicha interfaz

Requisitos
Captura de los requisitos ...

- Ejemplo:



Caracterización de la IGU

Requisitos
Captura de los requisitos ...

- EL artefactos utilizado para capturar los casos de uso del sistema es el *modelo de casos de uso de la aplicación*, formado por:
  - Actores UML.
  - Casos de uso UML.
- Este modelo de casos de uso de la aplicación puede derivarse a partir de los casos de uso del modelo del negocio

Requisitos
Normas

- Podemos considerar las siguientes normas:

  - Los casos de uso deben mantenerse tan independientes los unos de otros como sea posible.

  - Los casos de uso deben describirse utilizando el lenguaje del cliente.

  - Cada caso de uso debe estructurarse para que forme una especificación de funcionalidad completa e intuitiva.

Análisis
Introducción

- Durante el análisis se estudian los requisitos, refinándolos y estructurándolos
- El objetivo es conseguir un comprensión más precisa de los requisitos y una descripción de los mismos que sea fácil de mantener y que ayude a estructurar el sistema

Análisis
Introducción

- Durante el análisis, los casos de uso deben representarse desde la perspectiva de los desarrolladores
- Además, ahora la vista se centra en el sistema, y no en el problema
- Comparativamente:

Análisis
Introducción

Modelo de casos de uso	Modelo de análisis
Descrito con el lenguaje del cliente.	Descrito con el lenguaje del desarrollador.
Vista externa del sistema.	Vista interna del sistema.
Estructurado por los casos de uso; proporciona la estructura a la vista externa.	Estructurado por clases y paquetes estereotipados; proporciona la estructura a la vista interna.
Utilizado fundamentalmente como contrato entre el cliente y los desarrolladores sobre qué debería y qué no debería hacer el sistema.	Utilizado fundamentalmente por los desarrolladores para comprender cómo debería darse forma al sistema, es decir, cómo debería ser diseñado e implementado.
Puede contener redundancias, inconsistencias, etc., entre requisitos.	No debería contener redundancias, inconsistencias, etc., entre requisitos.
Captura la funcionalidad del sistema, incluida la funcionalidad significativa para la arquitectura.	Esboza cómo llevar a cabo la funcionalidad dentro del sistema, incluida la funcionalidad significativa para la arquitectura; sirve como una primera aproximación al diseño.
Define casos de uso que se analizarán con más profundidad en el modelo de análisis.	Define realizaciones de casos de uso, y cada una de ellas representa el análisis de un caso de uso del modelo de casos de uso.

Comparación entre requisitos y análisis

Análisis
Introducción

- Analizar los requisitos en forma de un modelo de análisis es importante, ya que:

  - Un modelo de análisis ofrece una especificación más precisa de los requisitos que el modelo de casos de uso.

  - Un modelo de análisis se describe utilizando el lenguaje de los desarrolladores, y puede por tanto introducir un mayor formalismo y ser utilizado para razonar sobre el funcionamiento interno del sistema.

Análisis
Introducción

- Un modelo de análisis estructura los requisitos de un modo que facilita su comprensión, su preparación, su modificación, y en general, su mantenimiento
- Un modelo de análisis puede considerarse como una primera aproximación al modelo de diseño, aunque es un modelo por sí mismo.

Análisis
El papel del análisis

- El papel del análisis depende de la fase de desarrollo del software en la que nos encontremos
- Las iteraciones iniciales de *elaboración* se centran en el análisis
- Más adelante, al término de la fase de *elaboración* y durante la *construcción* el énfasis pasa a diseño e implementación

Análisis
El papel del análisis

- Además, hay tres formas diferentes de considerar al modelo de análisis:

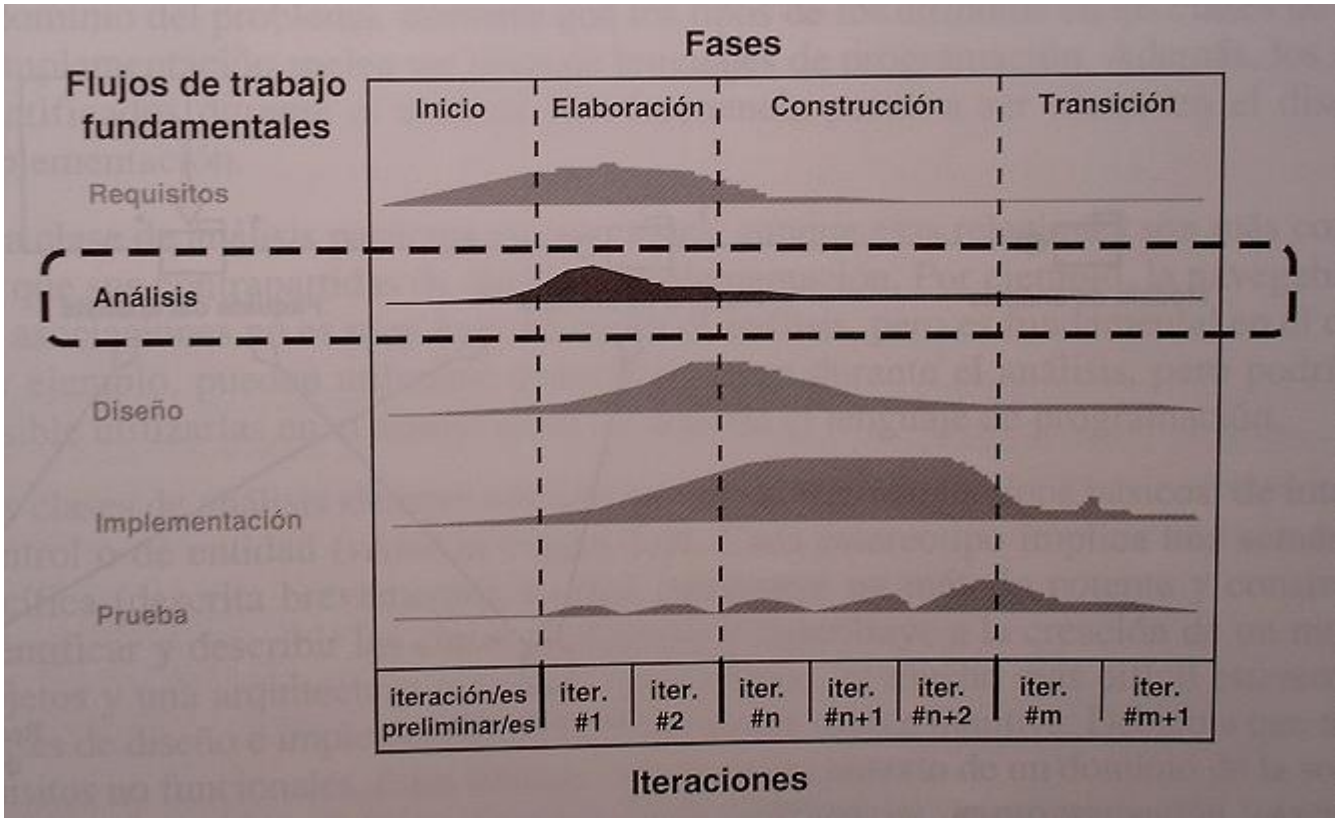
  - Para describir los resultados del análisis y mantener la consistencia de este modelo durante todo el ciclo de vida del software.

  - Para describir los resultados del análisis, considerando a este modelo como una herramienta transitoria e intermedia hasta el diseño.

Análisis
El papel del análisis

- El proyecto no utiliza en absoluto el modelo de análisis para describir los resultados del análisis. El proyecto concibe los requisitos como parte integrada de la captura de requisitos o en el diseño. Suele aplicarse en el caso de tratarse de sistemas simples

Análisis
El papel del análisis



El papel del análisis en el proceso

Análisis
Artefactos

- El artefacto utilizado para capturar el análisis es el *modelo de análisis*, formado por:
  - Clases de análisis.
  - Realizaciones de casos de uso.
  - Paquetes de análisis.
  - Vista arquitectónica del modelo de análisis.

Análisis
Artefactos

- Una *clase de análisis* representa una abstracción de una o varias clases y/o subsistemas del diseño del sistema. Posee las siguientes características:

  - Una clase de análisis se centra en el tratamiento de los requisitos funcionales y pospone los no funcionales, hasta llegar a las actividades de diseño e implementación.

Análisis
Artefactos

- Esto hace que una clase de análisis sea más evidente en el contexto del dominio del problema, menos específica que sus contrapartidas en diseño e implementación.
- Una clase de análisis raramente define u ofrece una interfaz en términos de operaciones y de sus signaturas. Su comportamiento se define mediante responsabilidades en un nivel más alto y menos formal. Una responsabilidad es una descripción textual del comportamiento de una clase.

Análisis
Artefactos

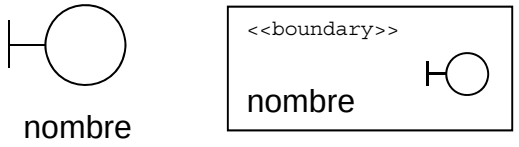
- Una clase de análisis define atributos, aunque esos atributos también son de un nivel bastante alto. Normalmente los tipos de esos atributos son conceptuales y reconocibles en el dominio del problema, mientras que en diseño suelen ser tipos. Con frecuencia, los atributos identificados durante el análisis pasan a ser clases durante el diseño.

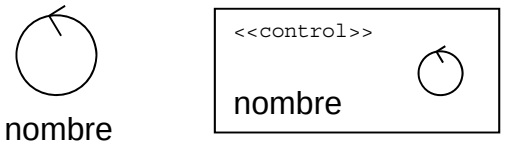
Análisis
Artefactos

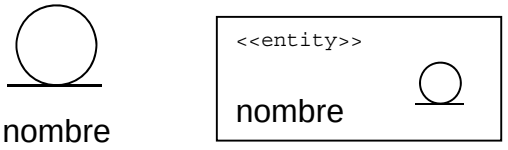
- Una clase de análisis participa en relaciones, aunque esas relaciones son más conceptuales que sus contrapartidas de diseño. Por ejemplo, la navegabilidad de las asociaciones no es vital en análisis, pero sí en diseño.

- Las clases de análisis están estereotipadas en tres tipos, mientras que es más difícil estereotipar todas las clases de diseño.

Análisis
Artefactos

- Los estereotipos de clases de análisis son:
  - Clase de interfaz: 

  - Clase de control: 

  - Clase de entidad: 

Análisis
Artefactos

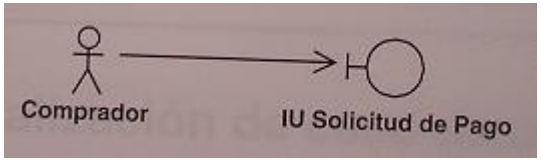
- Las *clases de interfaz* se utilizan para modelar la interacción entre el sistema y sus actores
- Esta interacción a menudo implica recibir información y peticiones de y hacia los usuarios y los sistemas externos

Análisis
Artefactos

- Las clases de interfaz modelan las partes del sistema que dependen de sus actores, lo cual implica que clarifican y reúnen los requisitos en los límites del sistema
- A menudo representan abstracciones de ventanas, formularios, paneles, interfaces de comunicación, etc.

Análisis
Artefactos

- No es necesario que describan cómo se ejecuta físicamente la interacción, eso se retrasa a actividades de diseño.
- Ejemplo:



Una clase de interfaz

Análisis
Artefactos

- Las *clases de entidad* se utilizan para modelar información que posee una vida larga y que a menudo es persistente
- Modelan la información y el comportamiento asociado de algún fenómeno o concepto, como una persona, un objeto del mundo real, o un suceso del mundo real

Análisis
Artefactos

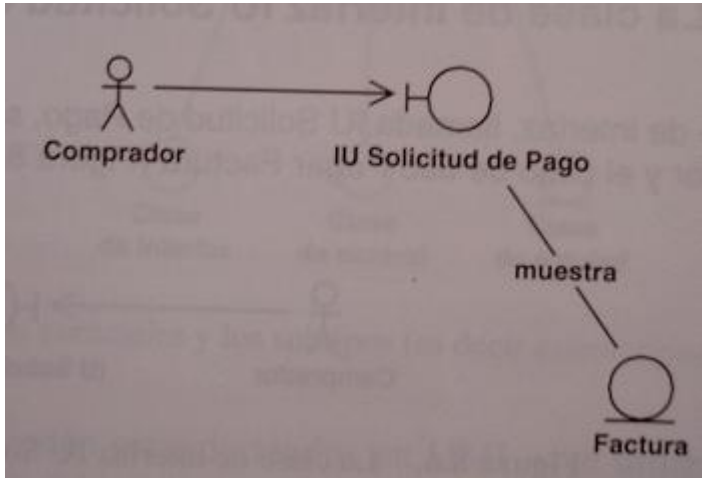
- En la mayoría de los casos, las clases de entidad se derivan directamente de una clase de entidad del negocio (o de una clase del dominio). Sin embargo, las clases entidad representan objetos manejados por el sistema en consideración y no del negocio

Análisis
Artefactos

- Por lo tanto, las clases de entidad reflejan la información de tal forma que indica como diseñar el sistema, incluyendo el soporte de persistencia
- Un objeto de entidad no ha de ser necesariamente pasivo y puede tener en ocasiones un comportamiento relativo a la información que representa

Análisis
Artefactos

- Las clases de entidad suelen mostrar una estructura de datos lógica y contribuyen a comprender de qué información depende el sistema
- Ejemplo:



Clase de entidad en un caso de uso

Análisis
Artefactos

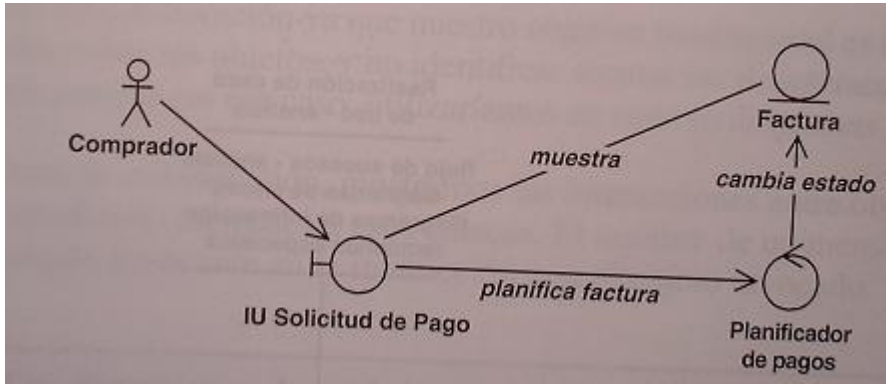
- Las *clases de control* representan coordinación secuencia, transacciones y control de otros objetos.
- Se usan con frecuencia para encapsular el control de un caso de uso en concreto
- También encapsulan procesamientos complejos que no pueden asociarse con una clase entidad concreta

Análisis
Artefactos

- Los aspectos dinámicos del sistema se modelan con clases de control, debido a que ellas manejan y coordinan las acciones y los flujos de control principales, y delegan trabajo a otros objetos (de interfaz y/o entidad)

Análisis
Artefactos

- Ejemplo:



Clase de control en un caso de uso

Análisis
Artefactos

- Una *realización de caso de uso-análisis* es una colaboración dentro del modelo de análisis que describe cómo se lleva a cabo y se ejecuta un caso de uso determinado en términos de las clases de análisis y sus objetos de análisis en interacción

Análisis
Artefactos

- Una realización de caso de uso-análisis contiene:
  - Descripción textual del flujo de sucesos.
  - Diagramas de clases.
  - Diagramas de interacción
- Ejemplo:

Análisis
Artefactos

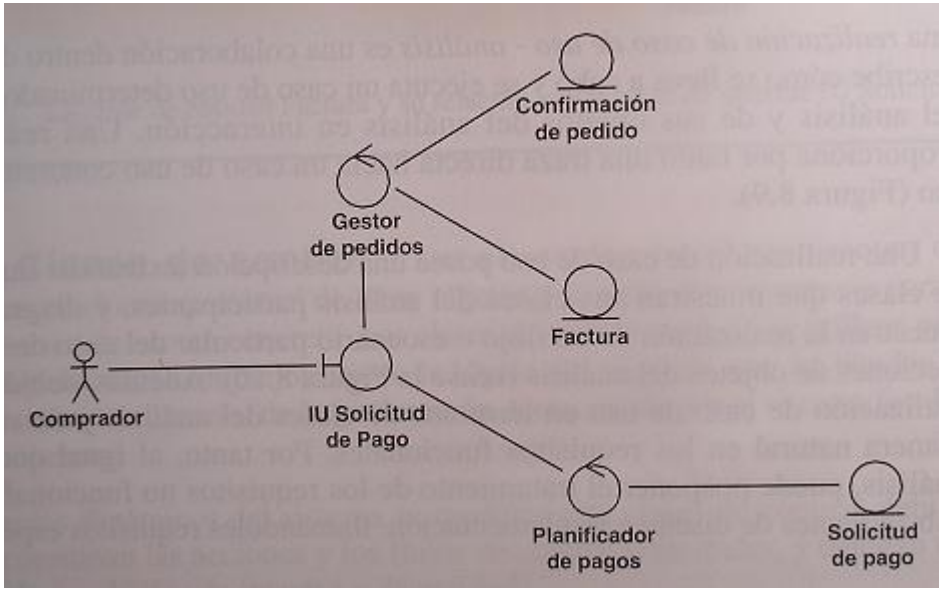
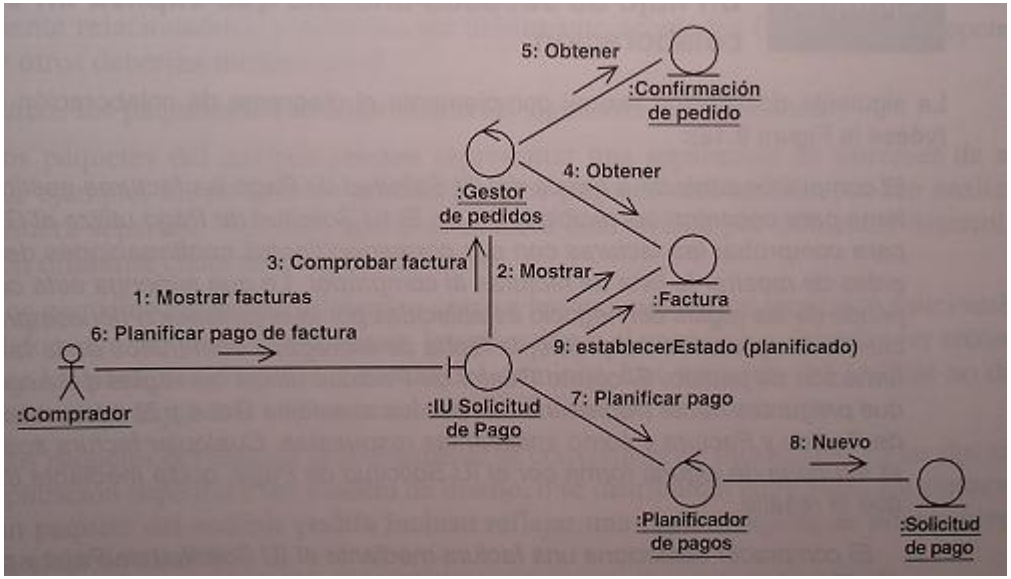


Diagrama de clases de análisis

Análisis
Artefactos



Pagar factura

Análisis
Artefactos

- Los *paquetes de análisis* proporcionan un medio para organizar los artefactos del modelo de análisis en piezas manejables
- Pueden contener:
  - Clases de análisis.
  - Realizaciones de casos de uso.
  - Otros paquetes de análisis.

Análisis
Artefactos

- Además, sus características son:

  - Pueden representar una separación de intereses de análisis.

  - Deberían crearse en base a los requisitos funcionales y el dominio del problema.

  - Probablemente se convertirán en subsistemas en diseño.

Análisis
Artefactos

- La *vista arquitectónica del modelo de análisis* muestra los artefactos de análisis significativos para la arquitectura:

  - Descomposición en paquetes de análisis y sus dependencias.

  - Clases fundamentales de análisis.

  - Realizaciones de casos de uso que describen una funcionalidad crítica o importante.

Diseño
Introducción

- En el diseño se modela el sistema y se le proporciona su forma para que soporte todos los requisitos
- Una entrada fundamental del diseño es el modelo de análisis, que proporciona una comprensión detallada de los requisitos e impone una estructura al sistema

Diseño
Introducción

- Los objetivos del diseño son:

  - Adquirir una comprensión en profundidad de los aspectos relacionados con los requisitos no funcionales y restricciones técnicas.

  - Crear una entrada apropiada y un punto de partida para actividades de implementación subsiguientes.

Diseño
Introducción

- Ser capaces de descomponer los trabajos de implementación en partes más manejables que puedan ser llevadas a cabo por diferentes equipos de desarrollo, teniendo en cuenta la posible concurrencia.

Diseño
Introducción

Modelo de análisis	Modelo de diseño
Modelo conceptual, porque es una abstracción del sistema y permite aspectos de la implementación.	Modelo físico, porque es un plano de la implementación.
Genérico respecto al diseño (aplicable a varios diseños).	No genérico, específico para una implemetación.
Tres estereotipos conceptuales sobre las clases: Control, Entidad e Interfaz.	Cualquier número de estereotipos (físicos) sobre las clases, dependiendo del lenguaje de implementación.
Menos formal.	Más formal.
Menos caro de desarrollar (ratio al diseño 1:5).	Más caro de desarrollar (ratio al análisis 5:1).
Menos capas.	Más capas.
Dinámico (no muy centrado en la secuencia).	Dinámico (muy centrado en las secuencias).
Bosquejo del diseño del sistema, incluyendo su arquitectura.	Manifiesto del diseño del sistema, incluyendo su arquitectura (una de sus vistas).
Creado principalmente como "trabajo de a pie" en talleres o similares.	Creado principalmente como "programación visual" en ingeniería de ida y vuelta; el modelo de diseño es realizado según la ingeriería de ida y vuelta con el modelo de implemetación (descrito en el Capítulo 10).
Puede no estar mantenido durante todo el ciclo de vida del software.	Debe ser mantenido durante todo el ciclo de vida del software.
Define una estructura que es una entrada esencial para modelar el sistema —incluyendo la creación del modelo de diseño.	Da forma al sistema mientras que intenta preservar la estructura definida por el modelo de análisis lo más posible.

Comparación modelo análisis y diseño

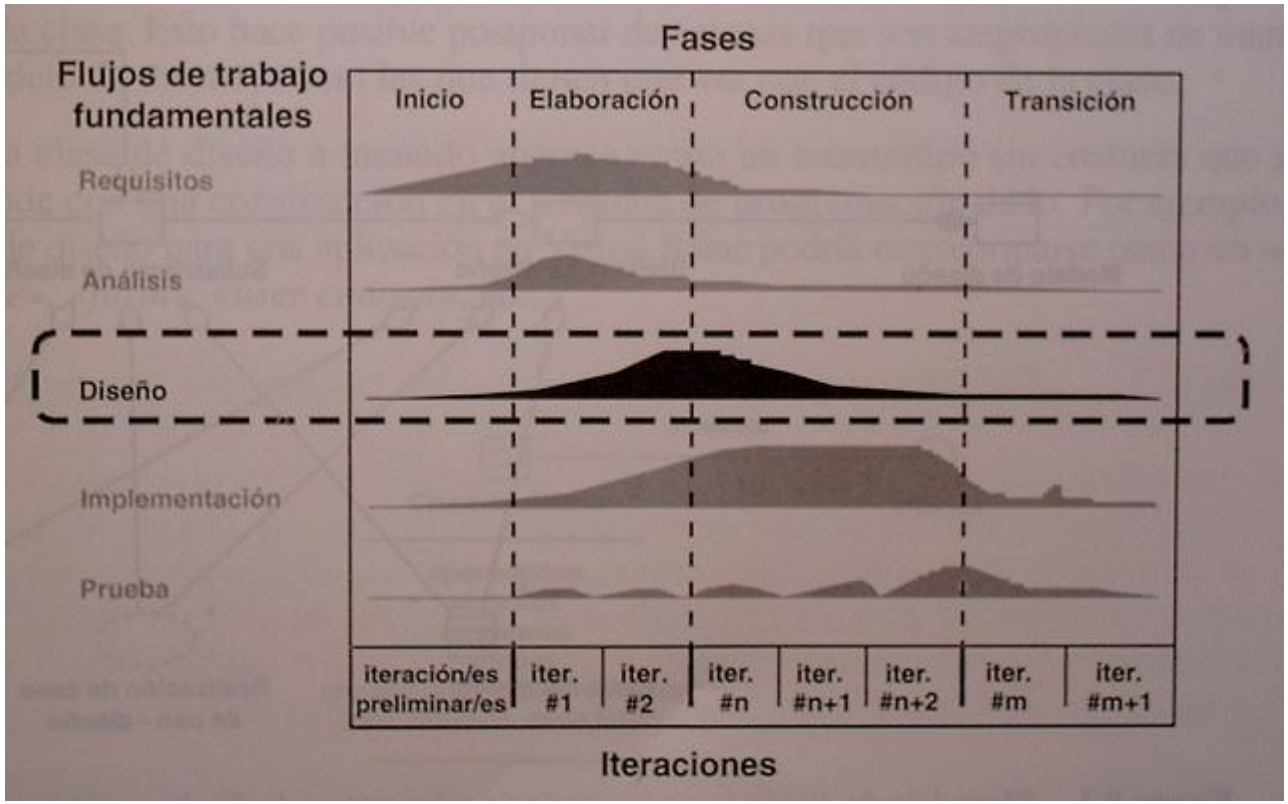
Diseño
El papel del diseño

- El diseño es el centro de atención al final de la fase de elaboración y el comienzo de las iteraciones de construcción
- Esto contribuye a una arquitectura estable y sólida y a crear un plano del modelo de implementación.

Diseño
El papel del diseño

- Más tarde, durante la fase de construcción cuando la arquitectura es estable y los requisitos están bien entendidos, el centro de atención se desplaza a implementación
- El modelo de diseño está muy ligado al de implementación, por lo que se suele mantener actualizado al modelo durante todo el proceso de desarrollo

Diseño
El papel del diseño



El papel del diseño en el proceso de desarrollo

Diseño
Artefactos

- El artefacto utilizado para capturar el diseño es el *modelo de diseño*, formado por:
  - Clases de diseño.
  - Realización de caso de uso-diseño.
  - Subsistema de diseño.
  - Interfaces.
  - Vista arquitectónica del modelo de diseño.
  - Modelo de despliegue.
  - Vista arquitectónica del modelo de despliegue.

Diseño
Artefactos

- El *modelo de diseño* es un modelo de objetos que describe la realización física de los casos de uso centrándose en el impacto en el sistema de los requisitos
- La *clase de diseño* es una abstracción de una clase o construcción similar en la implementación

Diseño
Artefactos

- Las características de las clases de diseño son:

  - El lenguaje de especificación de la clase coincide con el de implementación*.

  - Normalmente se especifica la visibilidad de los atributos.

  - Las relaciones entre clases de diseño se traducen directamente en implementación.

*Hoy en día esto está matizado por la *Model Driven Architecture* http://www.omg.org/mda/

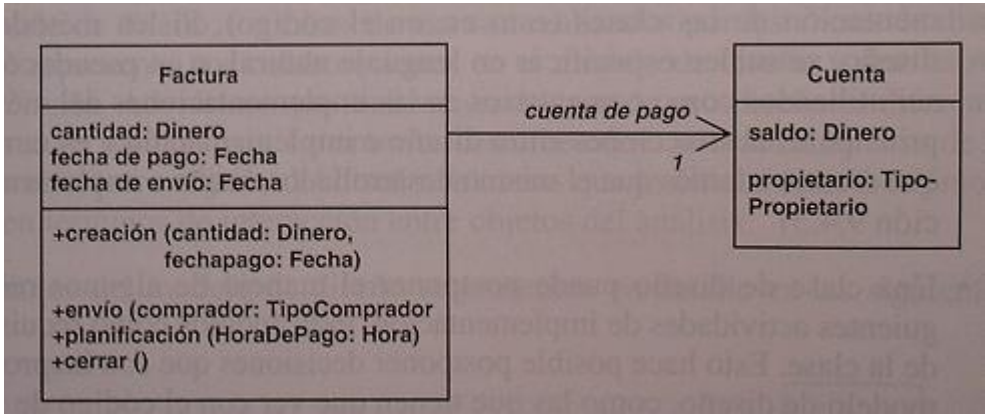
Diseño
Artefactos

- Los métodos de una clase de diseño tienen correspondencia directa con el código.
- Una clase de diseño puede posponer decisiones que son inapropiadas de manejar en el modelo de diseño.
- Una clase de diseño se puede corresponder con clases existentes en el lenguaje de implementación.

Diseño
Artefactos

- Una clase de diseño puede realizar interfaces si tiene sentido hacerlo en el lenguaje de programación.
- Una clase de diseño puede ser activa.

- Ej.:



Clases de diseño

Diseño
Artefactos

- Una *realización de caso de uso-diseño* es una colaboración en el modelo de diseño que describe cómo se realiza un caso de uso específico, y cómo se ejecuta, en términos de clases de diseño y sus objetos
- Se construyen sobre los casos de uso de análisis

Diseño
Artefactos

- Una realización de caso de uso-diseño tiene:

  - Una descripción del flujo de eventos textual.

  - Diagramas de clases de diseño.

  - Diagramas de interacción entre objetos de diseño.

- Los *subsistemas* de diseño son una forma de organizar los artefactos del modelo de diseño en piezas más manejables

Diseño
Artefactos

- Un subsistema puede constar de:
  - Clases de diseño.
  - Realizaciones de caso de uso.
  - Interfaces.
  - Otros subsistemas.
- Características de subsistemas:
  - Deben ser cohesivos.
  - Deben tener bajo acoplamiento.

Diseño
Artefactos

- Pueden representar una separación de aspectos del diseño.
- Suelen tener relaciones con las clases y/o paquetes de análisis.
- Pueden representar productos software reutilizados que han sido encapsulados en ellos.
- Pueden representar sistemas heredados (o parte de ellos) encapsulándolos.

Diseño
Artefactos

- Los subsistemas pueden representar componentes de grano grueso en la implementación del sistema, es decir, componentes que proporcionan varios interfaces compuestos a partir de otros componentes de grano más fino, como los que especifican clases de implementación individuales, y que se convierten ellos mismos en ejecutables, ficheros binarios o entidades similares que pueden distribuirse en diferentes nodos.

Diseño
Artefactos

- Las *interfaces* se utilizan para especificar las operaciones que proporcionan las clases y los subsistemas de diseño
- Una clase de diseño que realice una interfaz debe proporcionar métodos que realicen las operaciones del interfaz

Diseño
Artefactos

- Un subsistema que realice a un interfaz debe contener también clases de diseño u otros subsistemas que realicen la interfaz
- Las interfaces constituyen una forma de separar la especificación de la funcionalidad en términos de operaciones de sus implementaciones (métodos)

Diseño
Artefactos

- Esta distinción hace independiente de la implementación de la interfaz a cualquier cliente que dependa de ella
- Así, podemos sustituir una implementación concreta de una interfaz, como puede ser una clase o un subsistema de diseño, por otra implementación sin tener que cambiar los clientes

Diseño
Artefactos

- La *vista arquitectónica del modelo de diseño* muestra los artefactos relevantes para la arquitectura de la aplicación:

  - Descomposición del modelo en subsistemas, sus interfaces y las dependencias entre ellos.

  - Clases de diseño fundamentales.

  - Realizaciones de casos de uso-diseño que describan alguna funcionalidad crítica

Diseño
Artefactos

- Ejemplo:

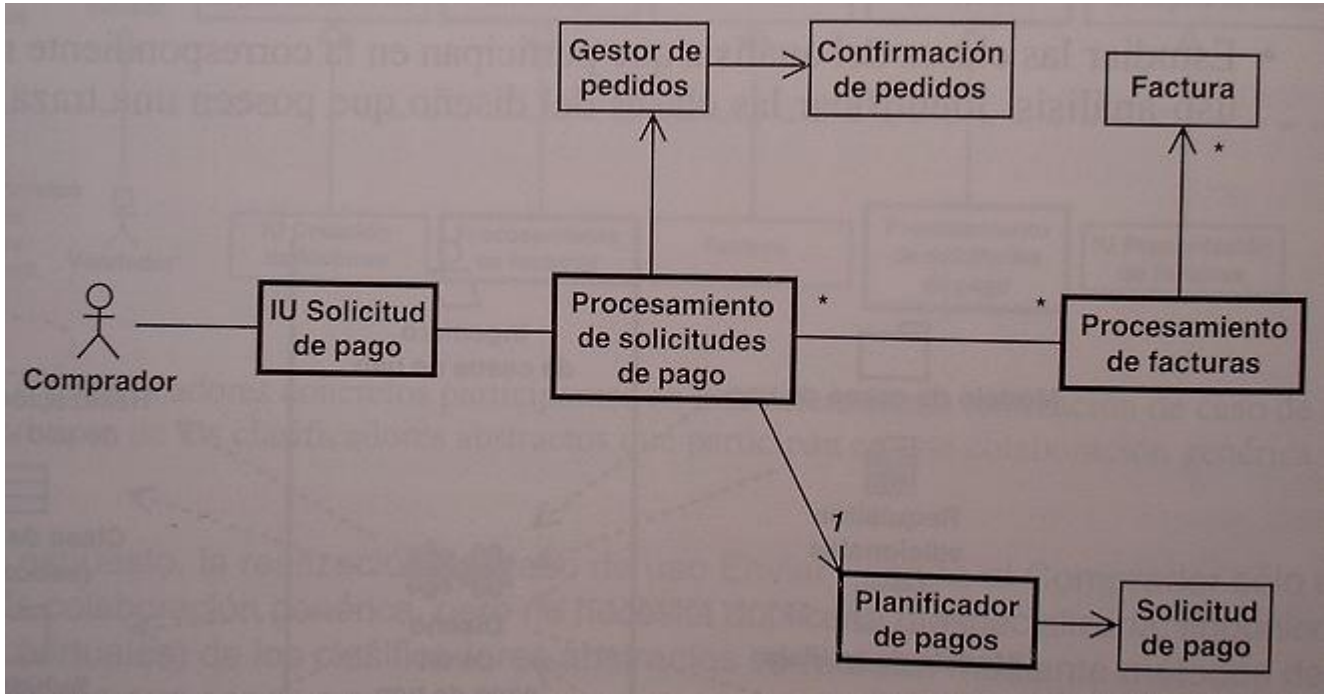
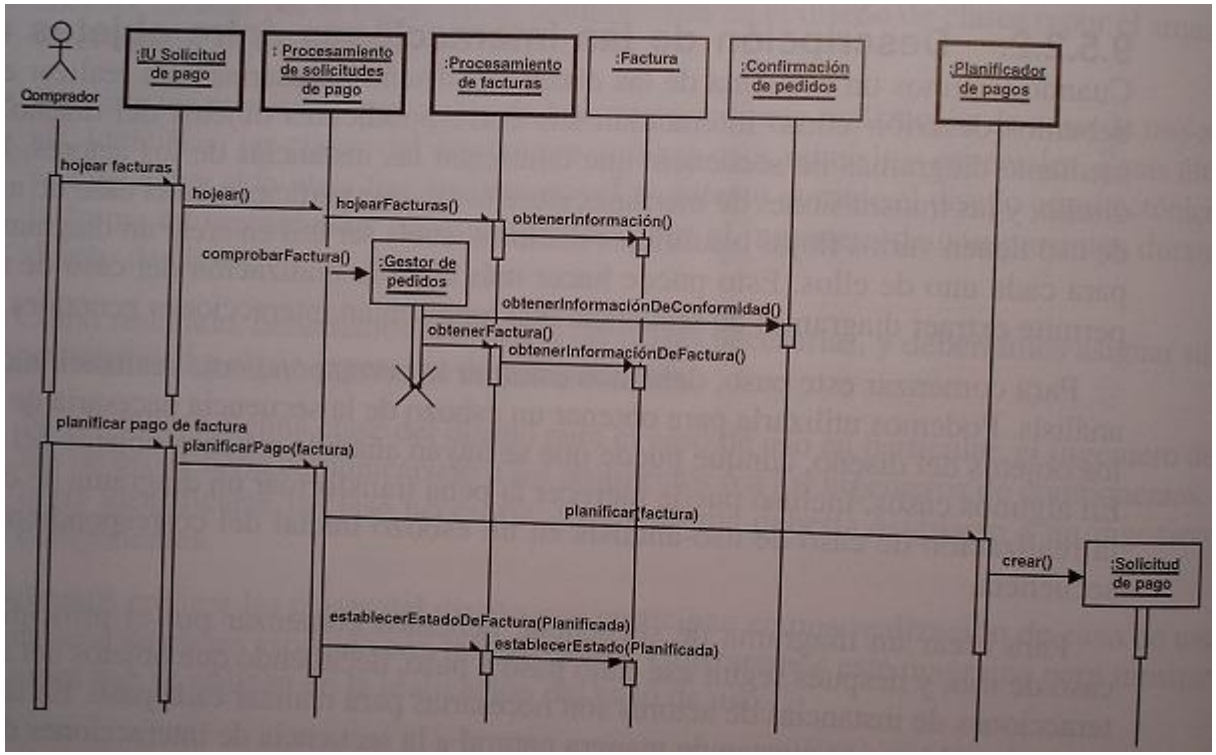


Diagrama de clases de diseño

Diseño
Artefactos



Pagar factura

Diseño
Artefactos

- El *modelo de despliegue* es un modelo de objetos que describe la distribución física del sistema en términos de cómo se distribuye la funcionalidad entre los nodos de cómputo
- Cada nodo representa un recurso de cómputo, normalmente un procesador o dispositivo hardware similar

Diseño
Artefactos

- Los nodos poseen relaciones que representan medios de comunicación entre ellos, tales como *Internet*, *intranet*, *bus* y similares
- El modelo de despliegue puede describir diferentes configuraciones de red, incluidas las configuraciones para pruebas y simulación

Diseño
Artefactos

- La funcionalidad (los procesos) de un nodo se definen por los componentes que se distribuyen sobre ese nodo
- El modelo de despliegue en sí mismo representa una correspondencia entre la arquitectura software y la arquitectura del sistema (el hardware)

Diseño
Artefactos

- La *vista arquitectónica del modelo de despliegue* muestra los artefactos relevantes del modelo de despliegue para su arquitectura

Implementación
Introducción

- En la *implementación* empezamos con el resultado del diseño e implementamos el sistema en términos de componentes, es decir, ficheros de código fuente, *scripts*, ficheros de código binario, ejecutables y similares

Implementación
Introducción

- Los propósitos de la implementación son:

  - Planificar las integraciones de sistema necesarias en cada iteración.

  - Distribuir el sistema asignando componentes ejecutables a nodos en el diagrama de despliegue.

  - Implementar las clases y subsistemas encontrados durante el diseño.

  - Probar los componentes individualmente e integrarlos.

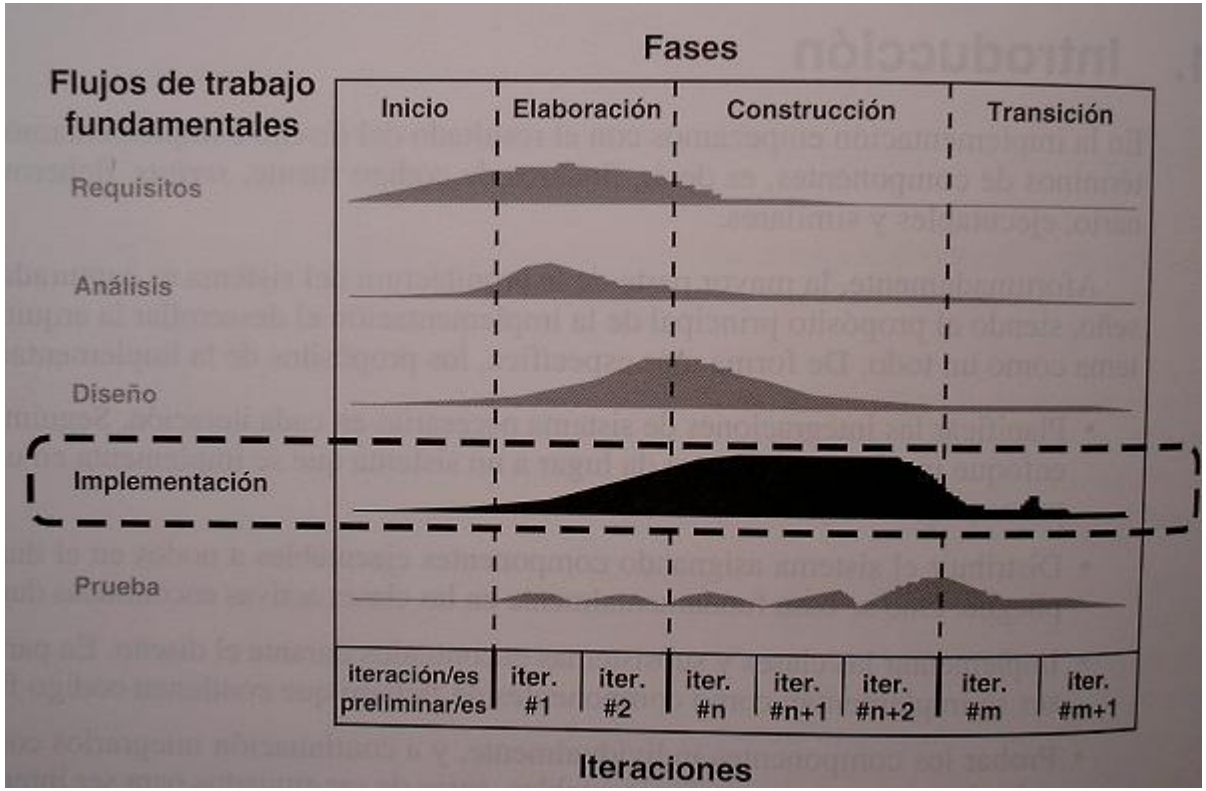
Implementación
El papel de la implementación

- La implementación es el centro durante las iteraciones de construcción.
- También se lleva a cabo trabajo de implementación durante la fase de elaboración, para crear la línea base ejecutable de la arquitectura
- Durante la fase de transición puede tratar defectos tardíos

Implementación
El papel de la implementación

- Ya que el modelo de implementación denota la implementación actual del sistema en términos de componentes y subsistemas de implementación, es natural mantener el modelo de implementación a lo largo de todo el ciclo de vida del software

Implementación
El papel de la implementación



El papel de la implementación en el proceso de desarrollo

Implementación
Artefactos

- El artefacto utilizado para capturar la implementación es el *modelo de implementación*, formado por:
  - Componentes (artefactos UML 2.x).
  - Subsistemas de implementación.
  - Interfaces.
  - Visión arquitectónica del modelo de implementación.
  - Plan de integración de construcciones.

Implementación
Artefactos

- El *modelo de implementación* describe cómo los elementos del modelo de diseño, como las clases, se implementan en términos de componentes, como ficheros de código fuente, ejecutables, etc.
- También describe cómo se organizan los componentes y sus dependencias

Implementación
Artefactos

- Un *componente (artefacto* UML 2.x) es el empaquetamiento físico de los elementos de un modelo, como son las clases en el modelo de diseño
- Algunos estereotipos de componentes son:
  - `executable` es un programa ejecutable en un nodo.
  - `file` es un fichero que contiene código fuente o datos.
  - `library` es una librería estática o dinámica.
  - `table` es una tabla de una base de datos.
  - `document` es un documento.

Implementación
Artefactos

- Los *subsistemas de implementación* proporcionan una forma de organizar los artefactos del modelo de implementación en trozos más manejables
- Un subsistema puede estar formado por:
  - Componentes.
  - Interfaces.
  - Otros subsistemas.

Implementación
Artefactos

- Un subsistema de implementación se manifiesta a través de un *mecanismo de empaquetamiento* concreto en un entorno de implementación determinado, tales como:
  - Un paquete Java.
  - Un proyecto Visual Basic.
  - Un directorio de ficheros en un proyecto C++.
  - Un paquete en una herramienta Rational Rose.

Implementación
Artefactos

- Los subsistemas de implementación están muy ligados a los subsistemas de diseño
- Los *interfaces de implementación* se corresponden con los interfaces de diseño
- La *visión arquitectónica del modelo de implementación* representa los artefactos significativos arquitectónicamente:

Implementación
Artefactos

- La descomposición del modelo de implementación en subsistemas, sus interfaces y las dependencias entre ellos.
- Componentes claves.

- El *plan de integración de construcciones* describe la secuencia de construcciones necesarias en una iteración, es decir:
  - La funcionalidad que se espera de la construcción.

Implementación
Artefactos

- Las partes del modelo de implementación afectadas por la construcción

- Una *construcción* es una versión ejecutable del sistema, usualmente, una parte específica del mismo.
- En una iteración puede haber una o más construcciones

Prueba
Introducción

- Durante la prueba se verifica el resultado de la implementación probando cada construcción, incluyendo tanto construcciones internas como intermedias, así como las versiones finales del sistema a ser entregadas a terceros

Prueba
Introducción

- Los objetivos de la prueba son:

  - Planificar las pruebas necesarias en cada iteración, incluidas las de integración y sistema. Las pruebas de integración son necesarias para cada construcción dentro de la iteración, mientras que las pruebas de sistema son necesarias sólo al final de la iteración.

  - Diseñar e implementar las pruebas.

Prueba
Introducción

- Realizar las diferentes pruebas y manejar los resultados de cada prueba sistemáticamente.

- Nótese que se supone que las pruebas de unidad se realizan según se termina la clase/subsistema correspondiente

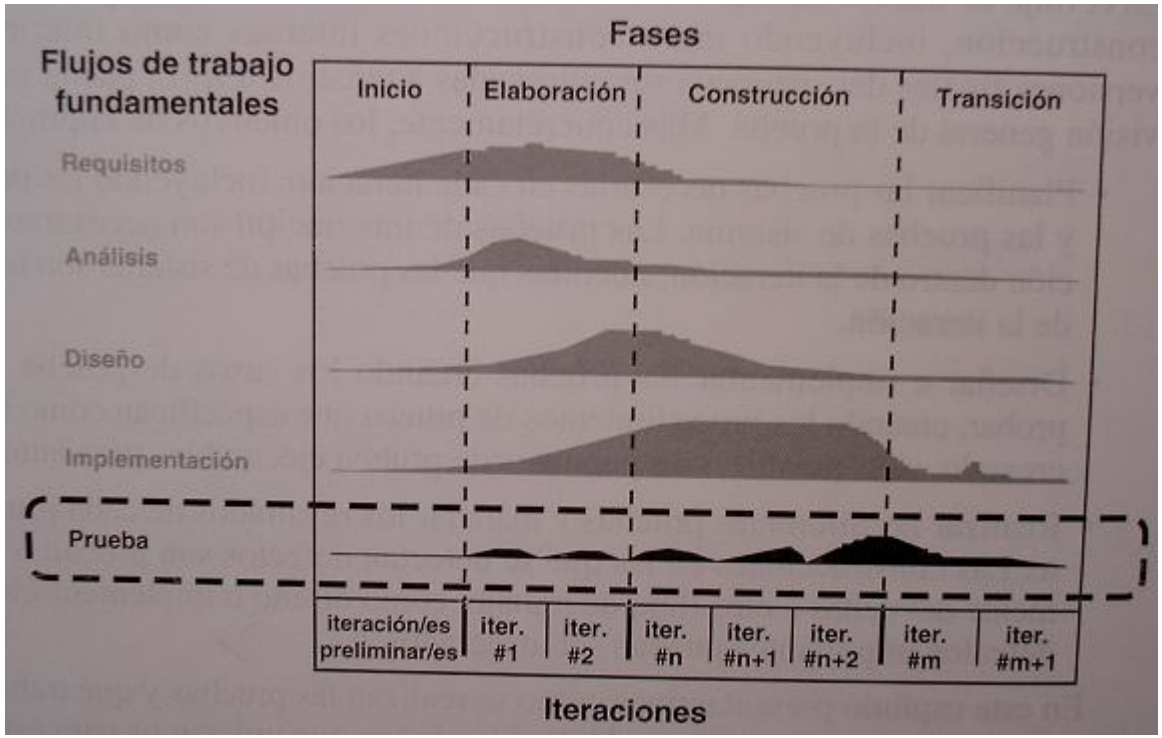
Prueba
El papel de la prueba

- Durante la fase de inicio puede hacerse parte de la planificación inicial de las pruebas cuando se define el ámbito del sistema.
- Sin embargo, las pruebas se llevan a cabo sobre todo cuando una construcción es sometida a pruebas de integración y de sistema

Prueba
El papel de la prueba

- Por tanto, la realización de pruebas se centra en las fases de elaboración y construcción.
- Durante la fase de transición la prueba sirve para la corrección de defectos durante los primeros usos.

Prueba
El papel de la prueba



El papel de la prueba en el proceso de desarrollo

Prueba
Artefactos

- El artefacto utilizado para capturar la prueba es el *modelo de prueba*, formado por:
  - Casos de prueba.
  - Procedimientos de prueba.
  - Componente de prueba.
  - Plan de prueba.
  - Defecto.
  - Evaluación de prueba.

Prueba
Artefactos

- Un *caso de prueba* especifica una forma de probar el sistema, incluyendo la entrada o resultado con la que se ha de probar y las condiciones bajo las que ha de probarse
- Lo que se prueba puede venir dado por un requisito o colección de requisitos del sistema

Prueba
Artefactos

- Casos de prueba comunes:

  - Caso de prueba de un caso de uso.

  - Caso de prueba de la realización de un caso de uso-diseño.

- Un *caso de prueba de caso de uso* incluye:

  - La verificación del resultado de la interacción entre los actores y el sistema.

  - Que se satisfacen las pre y postcondiciones del caso de uso.

Prueba
Artefactos

- Que se sigue la secuencia de acciones especificadas por el caso de uso.

- Un caso de prueba basado en un caso de uso especifica típicamente una prueba del sistema como *caja negra*, es decir, una prueba del comportamiento observable externamente del sistema

Prueba
Artefactos

- Un *caso de prueba de la realización de un caso de uso-diseño* incluye:

  - La verificación de la interacción entre los componentes que implementan dicho caso de uso.

- Los casos de prueba basados en la la realización de un caso de uso típicamente especifican una prueba del sistema como *caja blanca*, es decir, una prueba de la interacción interna de los componentes del sistema

Prueba
Artefactos

- Se pueden especificar otros casos de prueba para probar el sistema como un todo:

  - *Pruebas de instalación*. Verifican que el sistema puede ser instalado en la plataforma del cliente, y que el sistema funciona correctamente en dicha plataforma.

  - *Pruebas de configuración*. Verifican que el sistema funciona correctamente en diferentes configuraciones.

Prueba
Artefactos

- *Pruebas negativas*. Intentan provocar que el sistema falle para poder así revelar sus debilidades. Se intenta probar el sistema en formas para los que no ha sido diseñado.

- *Pruebas de tensión o estrés*. Identifican problemas con el sistema cuando hay recursos insuficientes o cuando hay competencia por los recursos

Prueba
Artefactos

- El *procedimiento de prueba* especifica cómo realizar uno o varios casos de prueba o partes de éstos
- El *componente de prueba* automatiza uno o varios procedimientos de prueba o partes de ellos
- El *plan de prueba* describe las estrategias, recursos y planificación

Prueba
Artefactos

- La estrategia de prueba incluye:

  - La definición del tipo de pruebas a realizar para cada iteración y sus objetivos.

  - El nivel de cobertura de prueba y de código necesario.

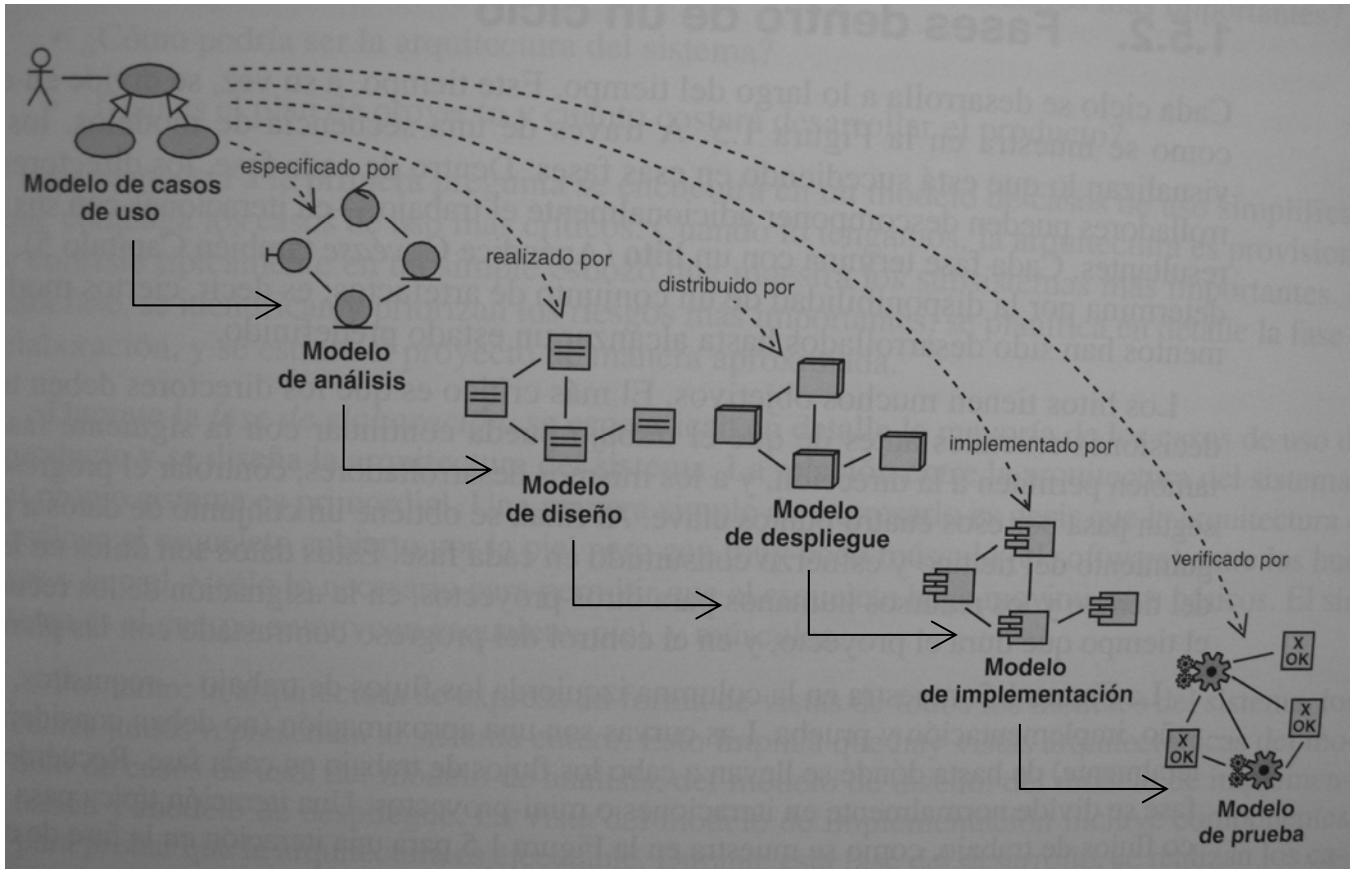
  - El porcentaje de pruebas que deberían ejecutarse con un resultado específico.

Prueba
Artefactos

- Un *defecto* es una anomalía del sistema.
- Una *evaluación de prueba* es una evaluación de los resultados de los esfuerzos de prueba

Modelos en el PUD

- Modelos en el PUD



Modelos en el PUD

Conclusiones

- Proceso unificado de desarrollo
- Ventaja: especifica los diagramas UML que hay que generar en cada actividad estructural
- Inconveniente: muy ligada al método
- Análisis y diseño OO

Conclusiones

- Requisitos
- Análisis
- Diseño
- Implementación
- Prueba
- Modelos en el PUD