

INGENIERÍA DEL SOFTWARE II

TEMA 2

PROCESOS COMPLEJOS DE DESARROLLO DE SOFTWARE

PROCESOS COMPLEJOS DE DESARROLLO DE SOFTWARE

- *Modelos de proceso*
- *Desarrollo evolutivo*
- *Desarrollo incremental*
- *El Proceso Unificado (PU)*
 - *Características*
 - *Vida: Fases*

... este tema también debería de sonar conocido...

Modelos de Proceso

- Existe una gran diversidad de **procesos software** (*conjunto de actividades que conducen a la creación de un producto software*)
- No existe un proceso **ideal** y depende del sistema a desarrollar
- Existen distintos enfoques, distintos **modelos de proceso**, distintas representaciones **abstractas** de procesos
- Los modelos no se excluyen mutuamente y a menudo se usan **conjuntamente**, especialmente para el desarrollo de sistemas grandes

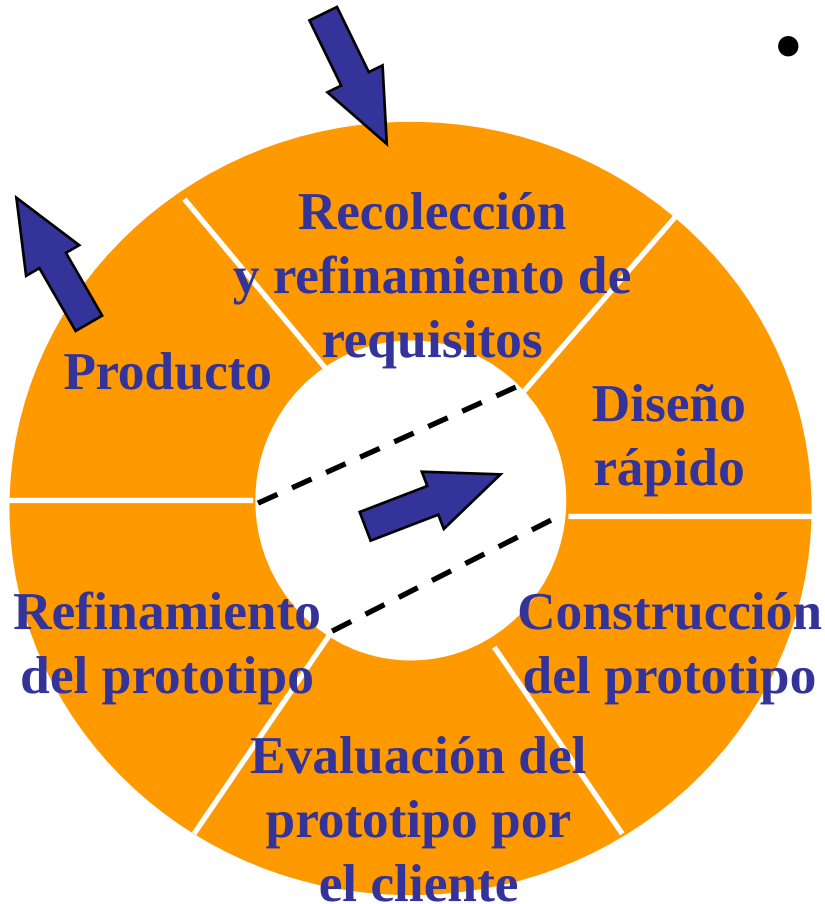
Modelos de Proceso

- En realidad cualquier modelo de proceso es **complejo**, pero es frecuente clasificarlos en dos enfoques diferentes
 - Procesos “**pesados**”, ***dirigidos por un plan***:
 - Gran esfuerzo de planificación, diseño y documentación
 - Grandes equipos, dispersos y proyectos largos
 - Grandes proyectos
 - Procesos “**ligeros**”, ***métodos ágiles***:
 - Entregar software funcional de forma rápida
 - Adaptarse a los requerimientos cambiantes
 - Sistemas de negocio pequeños y de tamaño medio

Modelos de proceso



Desarrollo Evolutivo



- Basado en:
 - Desarrollo de una implementación inicial
 - Exposición a comentarios y crítica del usuario
 - Refinamiento a través de diferentes versiones hasta llegar a un sistema adecuado

Desarrollo Evolutivo

- Dos tipos:
 - **Prototipo evolutivo**
 - trabajo con cliente para explorar sus requerimientos y entregar un sistema final
 - evolución continua del prototipo mediante la agregación de funciones y características propuestas por el cliente
 - **Prototipo desechable**
 - comprensión de las necesidades del cliente
 - desarrollo de una definición mejorada de los requerimientos del sistema
 - prototipos centrados en la experimentación de requisitos poco claros o complejos

Desarrollo Evolutivo

- Problemas:

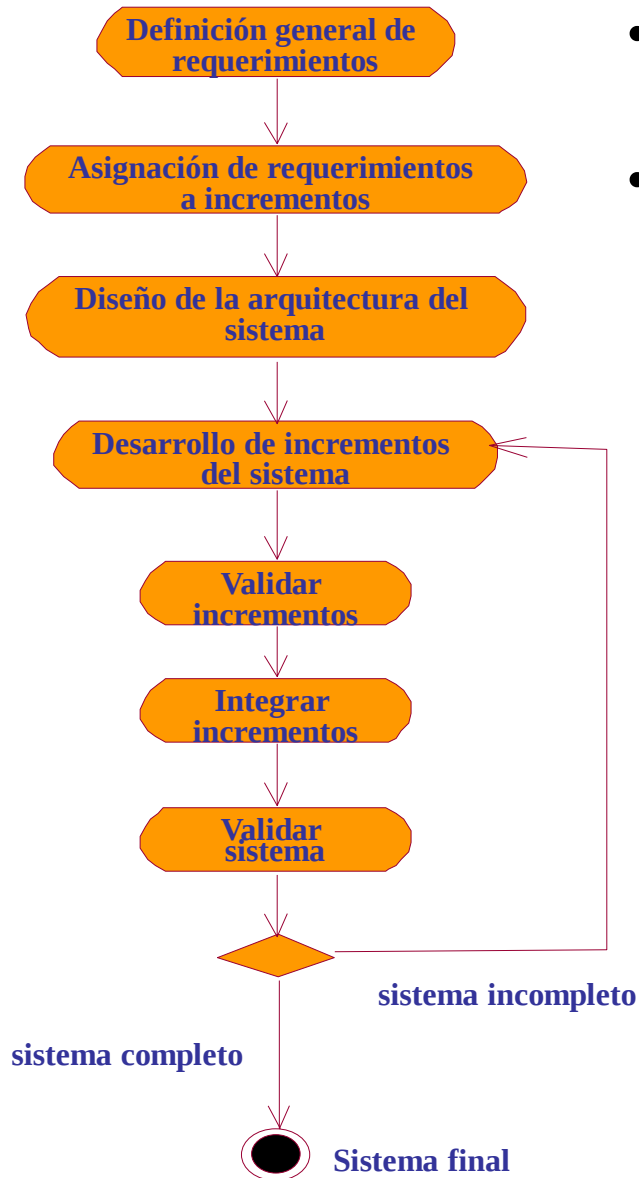
- prisas del cliente (utilización del prototipo como sistema final)
- pasar elecciones debidas al prototipo a la implementación final (entorno, sistema operativo,...)
- estructura deficiente, los cambios continuos tienden a corromper el sistema
- evolución del proceso difícil de gestionar y cuantificar
- requiere herramientas y técnicas especiales
- poco adecuado para grandes sistemas, desarrollado por diferentes equipos

Desarrollo Evolutivo

- **Ventajas:**
 - satisface las necesidades inmediatas del cliente
 - la especificación de requisitos se puede realizar de un modo creciente
 - la mejora en el entendimiento por parte de los usuarios de lo que se pretende construir se puede reflejar en el software
- En el caso de proyectos grandes se podría adoptar un **enfoque mixto** *por ejemplo...*
 - prototipo desechable para los requisitos poco claros
 - posteriormente se sigue el modelo de cascada
 - prototipo evolutivo para la interfaz de usuario

Desarrollo Incremental

- Es un enfoque **intermedio** entre el modelo en **cascada** y desarrollo **evolutivo**
- Pasos
 - identificación y priorización de funciones y servicios
 - definición de varios requerimientos que proporcionan parte de la funcionalidad, según la prioridad (los más importantes se entregan antes)
 - definición detallada de requerimientos del incremento y desarrollo con el proceso más adecuado
 - congelación de requerimientos de incrementos desarrollados
 - **puesta en explotación** de los incrementos completados y entregados



Desarrollo Incremental

- Ventajas

- puesta en marcha temprana
- los incrementos iniciales permiten refinar requerimientos de incrementos posteriores
- satisfacción del cliente (bajo riesgo de fallo)
- sistema final muy probado y con pocos fallos

- Problemas

- incrementos deben ser relativamente pequeños
- adaptación de requerimientos a incrementos del tamaño apropiado es difícil
- identificación de recursos comunes a todos los incrementos también es difícil

Proceso Unificado de Desarrollo (PU)

- Propuesto por Jacobson, Booch & Rumbaugh (creadores también **del UML**, *el lenguaje unificado de desarrollo*)
- Describe un **enfoque** para la construcción, desarrollo y mantenimiento del software
- Pensado principalmente para la construcción de software **orientado a objetos**
- Agrupa **buenas prácticas**, comúnmente aceptadas como adecuadas
- Es un marco de trabajo **genérico** que puede **especializarse, adaptarse**

Proceso Unificado: Características

- Tres principales:
 - Dirigido por casos de uso
 - Centrado en la arquitectura
 - Iterativo e incremental
- Otras:
 - **Arquitectura** basada en *componentes* interconectados a través de *interfaces*
 - **Utiliza UML** para desarrollar las vistas estáticas y dinámicas del sistema software
 - **Proceso integrado** de desarrollo, abarca:
 - ciclos, fases, flujos de trabajo
 - mitigación de riesgos, control de calidad, gestión del proyecto y control de configuración
 - automatización del proceso

PU: Dirigido por casos de uso

- ¿Qué son los casos de uso? Para construir un sistema con éxito hay que conocer las necesidades y deseos de los futuros usuarios

– usuario

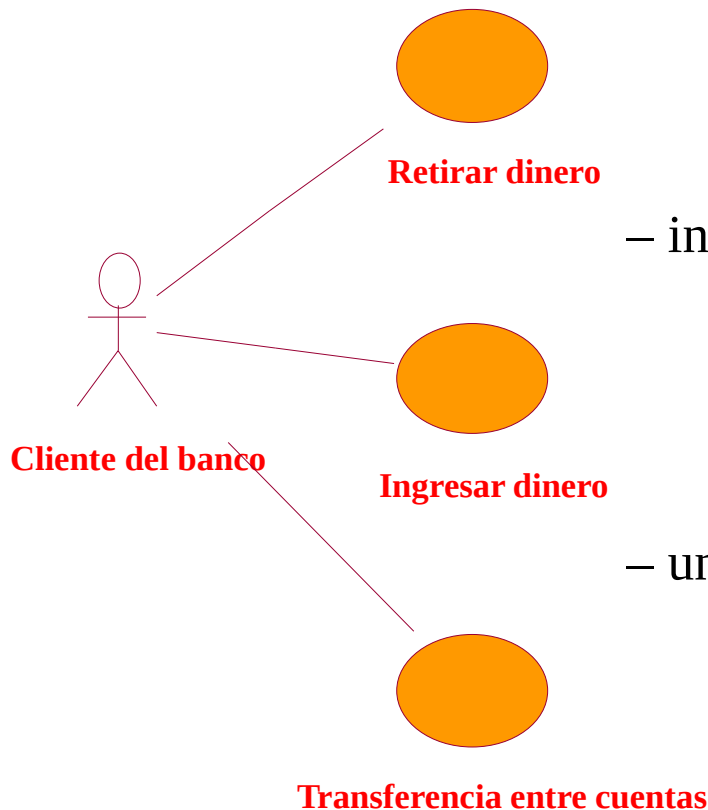
- personas que trabajan y necesitan el sistema
- otros sistemas que interactúan con el que estamos desarrollando

– interacción:

- usuario inserta tarjeta en cajero automático
- usuario responde sobre la pantalla a las preguntas que realiza el cajero
- el usuario recibe una cantidad de dinero y su tarjeta

– una interacción así es un **caso de uso**

- fragmento de funcionalidad del sistema que proporciona al usuario un resultado importante



PU: Dirigido por casos de uso

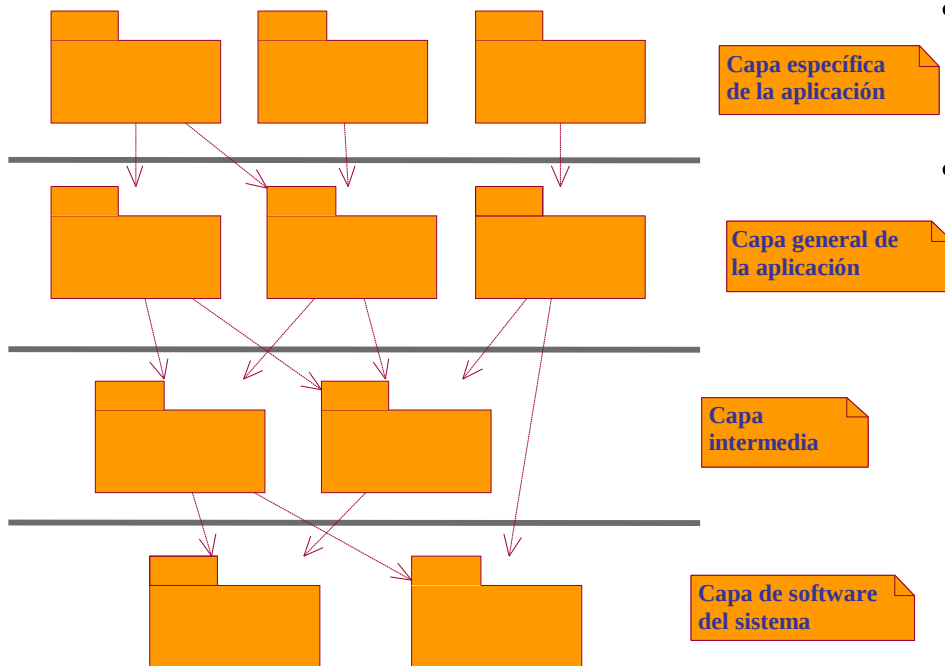
- ¿Por qué dirigido por? (casos de uso en el PU)
 - Aparte de ser una herramienta para especificar los requisitos **funcionales** de un sistema, y que todos los casos de uso juntos constituyen el **modelo de casos de uso**, que describe la **funcionalidad total del sistema**
 - Se dice que guían el proceso de desarrollo (diseño, implementación y prueba) ya que sirven también para
 - identificar y especificar **clases**, **subsistemas** e **interfaces**
 - identificar y especificar casos de **prueba**
 - planificar las **iteraciones** e **integración** del sistema
 - Por tanto no sólo inician el proceso de desarrollo sino que éste **sigue un hilo de trabajo** que parte de los casos de uso
 - Y además se utilizan como punto de partida para escribir el **manual de usuario**

PU: Centrado en la arquitectura

- La arquitectura de un sistema software se describe mediante *diferentes vistas* del sistema en construcción

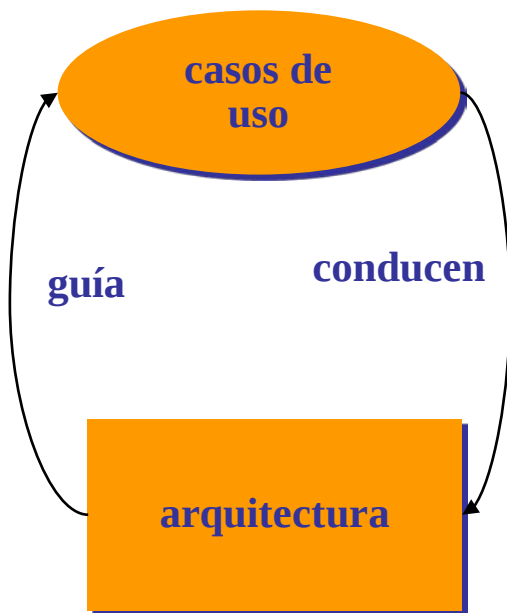
– influida por diversos factores

- necesidades de la empresa, tal y como las perciben los usuarios y clientes
- otros factores, como plataforma de explotación (arquitectura hardware, sistema operativo, gestor de bases de datos, protocolos de comunicación,...), componentes reutilizables, sistemas heredados, requisitos no funcionales,...



PU: Centrado en la arquitectura

- es una **vista del diseño** completo con las características más importantes resaltadas, dejando los detalles de lado
- hay una constante **interacción** entre los casos de uso y la arquitectura, que evolucionan en paralelo



- los casos de uso deben encajar en la arquitectura cuando se realizan
- la arquitectura debe permitir el desarrollo de todos los casos de uso requeridos

PU: Centrado en la arquitectura

–Se dice centrado en porque el arquitecto

- realiza un esquema en **borrador** de la arquitectura que no es específica de los casos de uso (por ejemplo, la plataforma)
- trabaja con un subconjunto de los casos de uso principales del sistema, especificándolo en detalle y realizándolo en términos de **subsistemas**, **clases** y **componentes**
- a medida que los casos de uso se especifican y maduran, se descubre más de la arquitectura, lo que **a su vez** lleva a la maduración de más casos de uso
- este proceso continúa hasta que se considera que se dispone de una **arquitectura estable**

PU: Centrado en la arquitectura

- Ventajas:
 - Permite ganar y retener control intelectual sobre el proyecto, para administrar su **complejidad**
 - Facilita el desarrollo basado en **componentes**
 - Provee una base efectiva para la **reutilización** de software
 - Provee una base para la **gestión** del proyecto

PU: Iterativo e incremental

- Porque el trabajo se divide en partes más pequeñas o miniproyectos
 - **miniproyecto**: una iteración que resulta en un incremento (***dura pocas semanas, 2 a 6...***)
 - iteración: pasos en el flujo de trabajo
 - incremento: crecimiento del producto
 - las iteraciones están controladas y planificadas
 - factores para **seleccionar** lo que se implementará en una iteración
 - la iteración se centra en un grupo de casos de uso que juntos amplían la utilidad del producto desarrollado hasta ahora
 - la iteración trata los riesgos más importantes

PU: Iterativo e incremental

- las iteraciones sucesivas se construyen sobre los **artefactos** de desarrollo tal como quedaron al final de la última iteración
 - en las primeras iteraciones los incrementos implican modificaciones
 - en las últimas iteraciones los incrementos implican menos modificaciones y más añadidos a los modelos
- **pasos** para cada iteración:
 - identificación y especificación de los casos de uso relevantes
 - creación de un diseño utilizando la arquitectura seleccionada como guía
 - implementación del diseño mediante componentes
 - verificación de que los componentes satisfacen los casos de uso
 - si una iteración cumple con sus objetivos, el desarrollo continúa con la siguiente iteración, si no, se revisan las decisiones previas y se adopta un nuevo enfoque

PU: Iterativo e incremental

–**ventajas** proceso iterativo controlado

- reducción del coste del riesgo a los costes de un solo incremento
- reducción del riesgo de no sacar al mercado el producto en el calendario previsto
- se acelera el ritmo del esfuerzo de desarrollo en su totalidad, para obtener resultados claros a corto plazo
- los requerimientos del usuario se van refinando en iteraciones sucesivas, por lo que se pueden acomodar mejor los cambios

–**equilibrio** entre **estabilizar** un conjunto de requisitos y la realidad de los requisitos **cambiantes** a lo largo del proyecto

Vida del Proceso Unificado

- El proceso unificado se puede repetir a lo largo de una serie de **ciclos**, cada ciclo concluye con una *versión* del producto y consta de **cuatro fases** (*normalmente sólo un ciclo*)
 - Inicio
 - Elaboración
 - Construcción
 - Transición
- Cada fase se divide a su vez en **iteraciones**

Fases del Proceso Unificado

- **Inicio:** descripción del producto a partir de una idea inicial y **análisis de negocio** para el producto
 - principales funciones del sistema y usuarios más importantes (**modelo de casos de uso**)
 - hay que identificar, encontrar, la mayoría de los casos de uso
 - como mucho se describe algún caso de uso al completo
 - posible arquitectura del sistema
 - plan del proyecto, coste, identificación y priorización de **riesgos**
 - otros artefactos: **Visión**, Especificación Complementaria, Glosario

Fases del Proceso Unificado

- **Elaboración:**

- se especifican en detalle **todos** los principales casos de uso
- se diseña el núcleo central de la arquitectura del sistema
- continuar observando los riesgos
- obtener el **modelo del dominio**, modelo conceptual del sistema a desarrollar
- implementación y prueba de la arquitectura
- planificación de las actividades y estimación de los recursos necesarios para finalizar el proyecto **más precisa**
- se crean **nuevas versiones** de todos los artefactos, manual de usuario preliminar

Fases del Proceso Unificado

- **Construcción:**

- las otras fases pueden considerarse de investigación
- se crea el producto añadiendo el software a la arquitectura (construir el código)
- alcanzar niveles de calidad y desarrollar versiones rápidamente **totalmente probadas**
- al final se dispone de todos los casos de uso acordados aunque puede que queden defectos no detectados en el entorno del desarrollador (capacidad **operativa**)
- además plan de proyecto para la transición y versiones nuevas de todos los artefactos

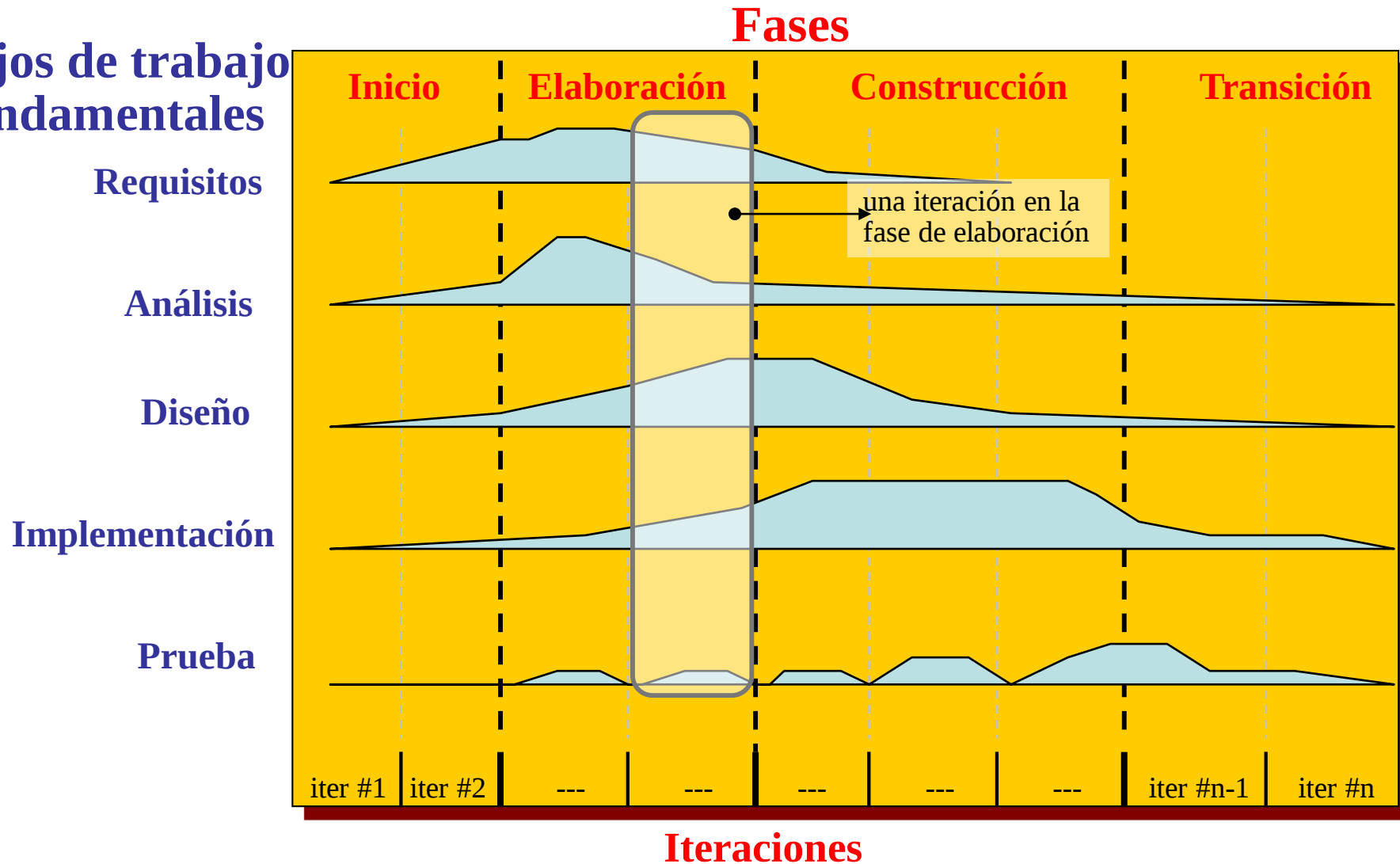
Fases del Proceso Unificado

- Transición

- periodo durante el cual el producto se lleva a un entorno real, en la que usuarios prueban el producto e informan de defectos y deficiencias
- se corrigen problemas e incorporan sugerencias
- finalización de todos los artefactos
- software ejecutable incluyendo software de instalación
- contratos, licencias, referencias de soporte
- manuales de usuario, operador y administrador
- incluye actividades como la formación del usuario, proporcionar una línea de ayuda y asistencia

Vida del Proceso Unificado

Flujos de trabajo fundamentales



Bibliografía

- *Sommerville*, “Ingeniería de Software,” Pearson, 2011
- *Pressman*, “Ingeniería del Software,” McGraw-Hill, 2005
- *Jacobson, Booch & Rumbaugh*, “El Proceso Unificado de Desarrollo de Software,” Pearson, 2000
- *Larman*, “UML y Patrones: una Introducción al Análisis y Diseño Orientado a Objetos y al Proceso Unificado,” Pearson, 2003