

INTRODUCCION

Proceso Unificado

DISEÑO Y CONSTRUCCIÓN DE SOFTWARE



Tendencia actual

■ Organización de un empresa



Habilidades de individuos altamente cualificados



Sin dirección sobre las políticas y procedimientos a seguir



Tendencia actual

■ Organización de un empresa



sin dirección
políticas
a seguir

e individuos
calificados



Tendencia actual

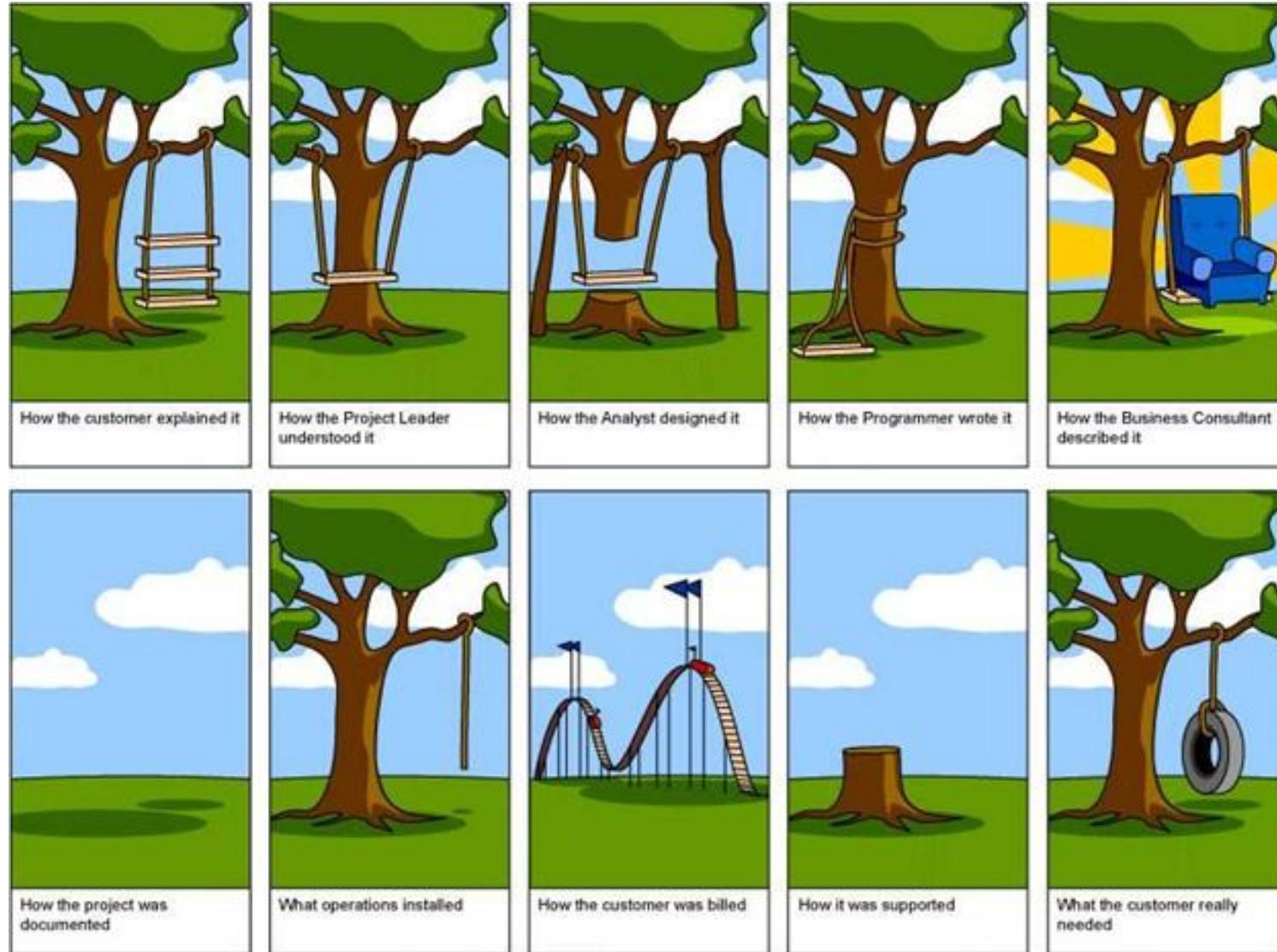
■ Software

- ◆ Construcción de sistemas más grandes y más potentes
 - Computadores más potentes
 - Se espera más de ellos
 - Uso creciente de Internet
- ◆ Mejor adaptado a nuestras necesidades
 - Más complejo
- ◆ Menor tiempo de salida al mercado

Problemas en el desarrollo de software

Problema	Solución
El programa no hace lo que se supone que debe de hacer	Definir bien la especificaciones
El programa se cuelga	Definir bien el diseño y las pruebas
Hace falta un 200% del tiempo planificado para terminar el programa	Planificar bien
Si el programador se va de la empresa... ¡Vaya! Hay que hacer un nuevo programa	Definir bien la documentación

Problema de comunicación



¿Entonces?

- Necesidad de organización
 - ◆ Forma coordinada de trabajar
 - ◆ Pautas de cómo trabajar
 - ◆ Proceso que integre las diferentes fases del desarrollo



¿Qué tipo de proceso?

■ Características

- ◆ Proporcione una guía para ordenar las actividades de un equipo
- ◆ Dirija las tareas de cada desarrollador por separado y del equipo como un todo
- ◆ Especifique los artefactos que deben desarrollarse
- ◆ Ofrezca criterios para el control y la medición de los productos y actividades del proyecto

Resultado

Proceso bien definido y gestionado

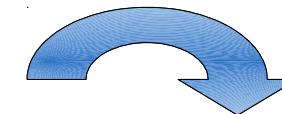


Proyecto productivo

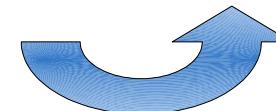
Proceso de desarrollo de Software

- Proceso define
 - ◆ Quién está haciendo qué, cuándo y cómo para alcanzar un determinado objetivo
- **Objetivo** en la ingeniería del software
 - ◆ Construir un producto software
 - ◆ Mejorar uno existente

Proceso de desarrollo de Software



Requisitos de un usuario



Sistema software



Proceso de desarrollo de Software

■ Actividades fundamentales

- ◆ Especificación del software
 - Extracción de información (requisitos)
 - Analizar esos requisitos
 - Definir la funcionalidad y restricciones operacionales que debe cumplir el software
- ◆ Diseño e implementación del software
 - De acuerdo a la especificación
- ◆ Pruebas del software
 - Detectar errores de software
- ◆ Despliegue y mantenimiento
 - Adaptarse a las necesidades del cliente

Proceso de desarrollo de Software

■ Flujo de las actividades

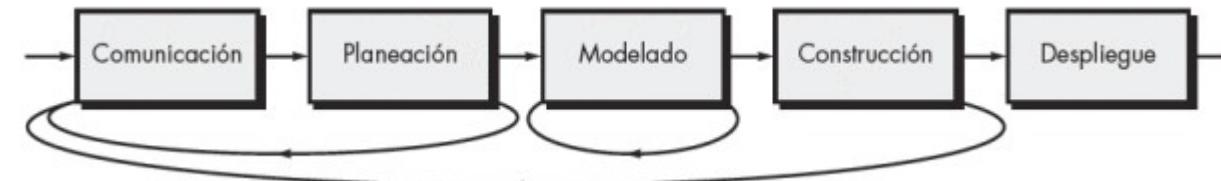
- ◆ Lineal

- Se ejecutan las actividades fundamentales secuencialmente



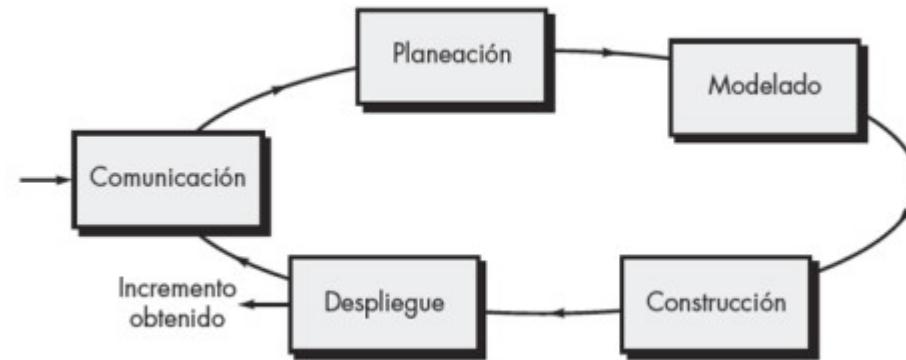
- ◆ Iterativo

- Se repite una o más de las actividades antes de pasar a la siguiente



Proceso de desarrollo de Software

- Flujo de las actividades
 - ◆ Evolutivo
 - Se realizan las actividades de forma circular

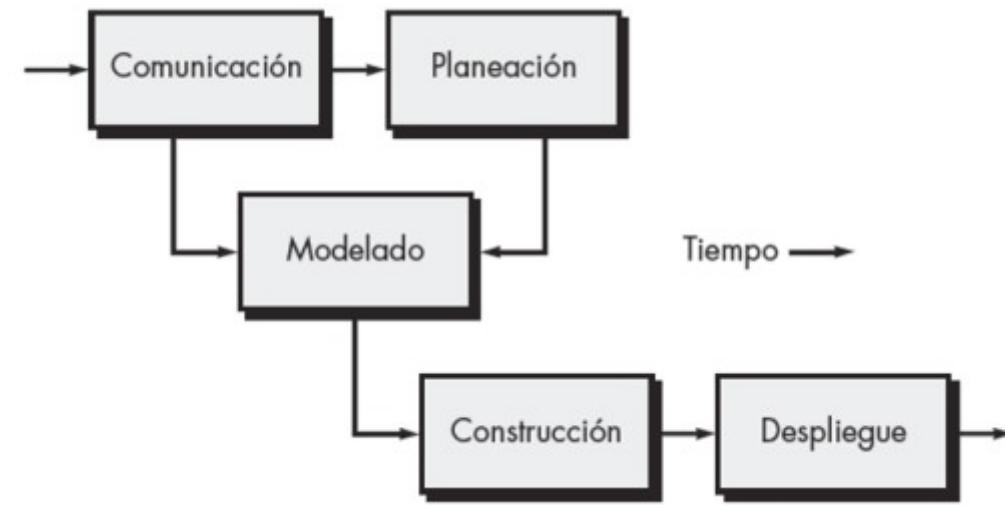


Proceso de desarrollo de Software

■ Flujo de las actividades

◆ Paralelo

- Se ejecutan una o más actividades en paralelo con otras



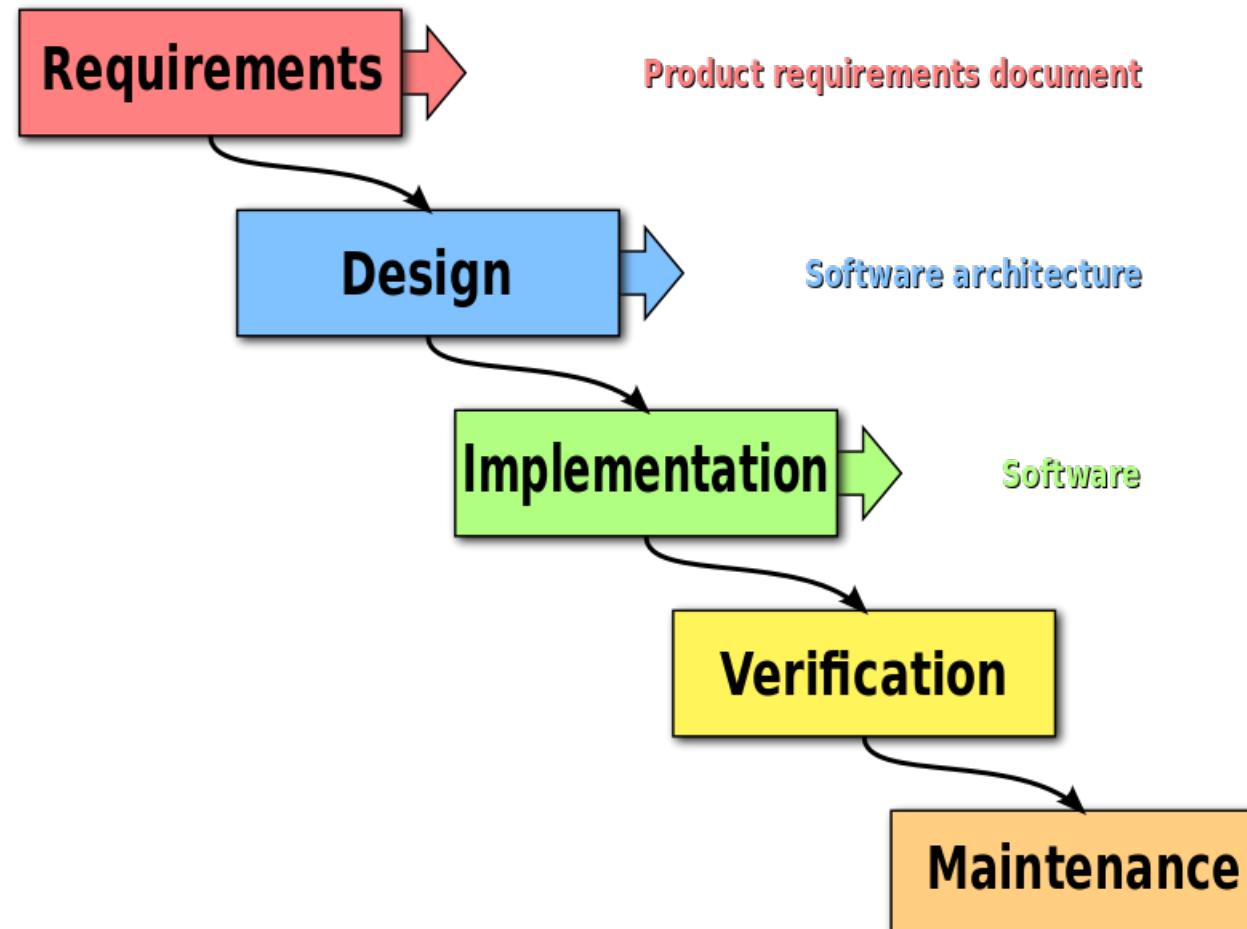
Proceso de desarrollo de Software

■ Paradigmas de desarrollo

- ◆ **Tradicional**
 - Modelos guiadas por una fuerte planificación inicial
 - El usuario interviene poco
- ◆ **Orientado a objetos**
 - Modelos basados en la POO
 - Concepto de clase y objeto en el análisis y diseño
- ◆ **Ágil**
 - Enfocado a las personas
 - Incremental, cooperativo, sencillo, adaptable

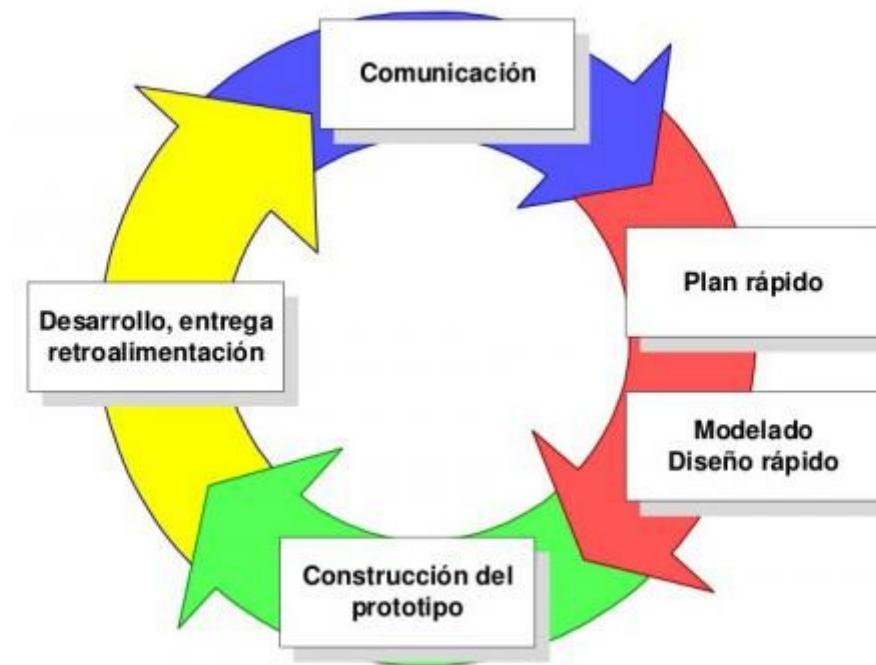
Proceso de desarrollo de Software

- Ejemplos de modelos: proceso en cascada



Proceso de desarrollo de Software

■ Ejemplos de modelos: prototipos



Proceso de desarrollo de Software

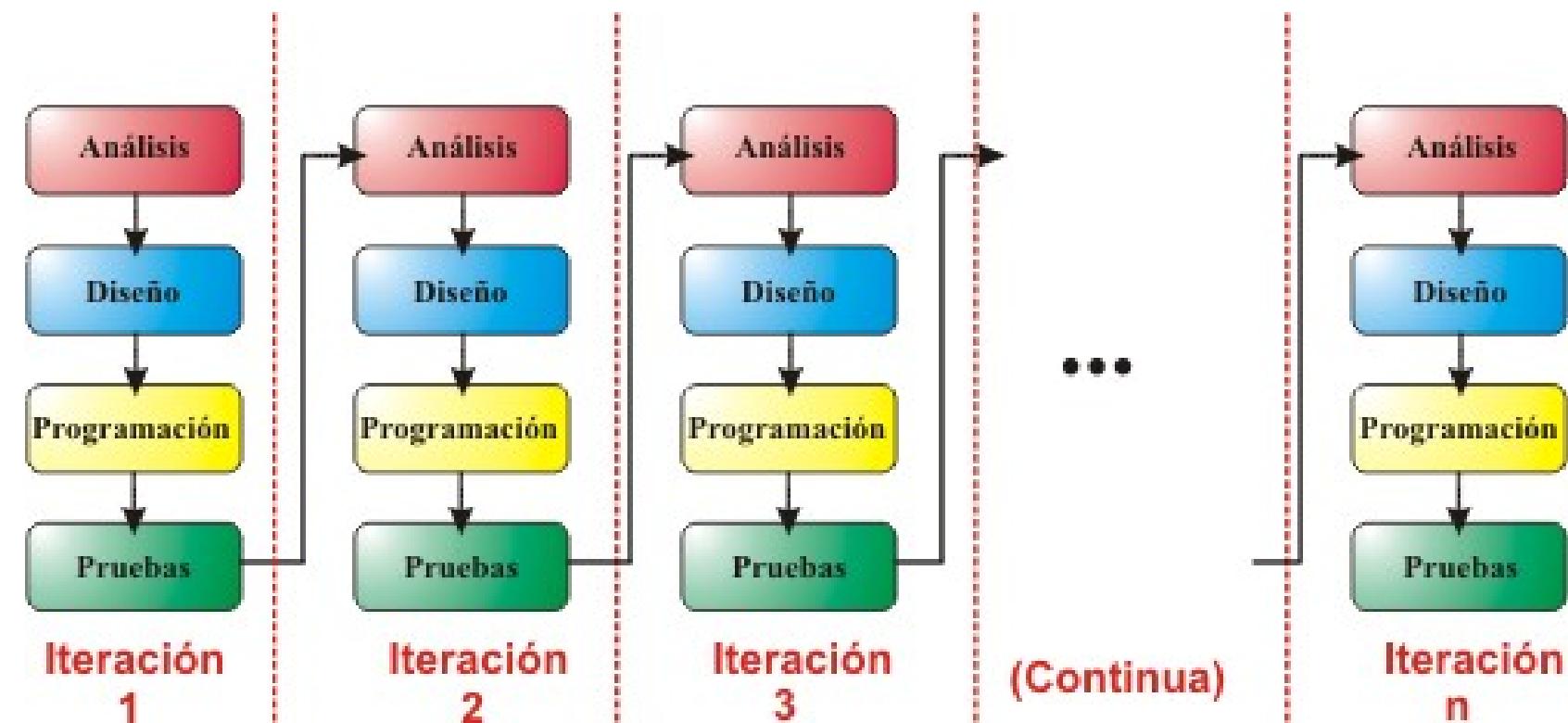
- Proceso complejo para proyectos medios o grandes
 - ◆ Dividir un proyecto en **mini-proyectos**
 - Más fáciles de completar y manejar
 - ◆ Cada mini-proyecto es una **iteración**
 - ◆ Cada iteración todos los elementos de un proyecto
 - Planificación
 - Análisis y Diseño
 - Construcción
 - Integración y pruebas
 - Versión del producto (interna o externa)

Proceso de desarrollo de Software

- Proceso complejo para proyectos medios o grandes
 - ◆ Cada iteración genera una línea base
 - Versión parcialmente completa del sistema final
 - Documentación asociada
 - ◆ Cada iteración se construye sobre las anteriores
 - Hasta alcanzar el sistema final
 - ◆ La diferencia entre dos líneas base se conoce como **incremento**

Proceso de desarrollo de Software

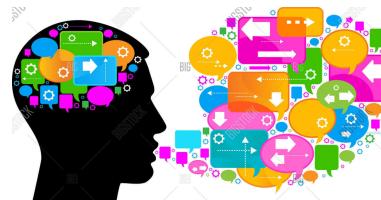
- Ejemplos de modelos: proceso incremental



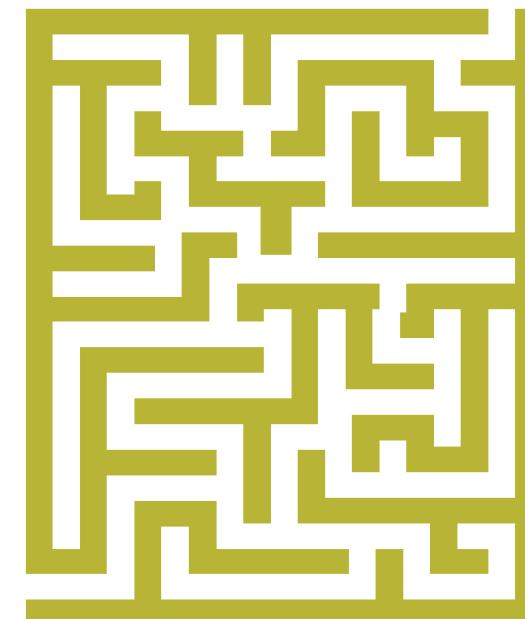
Proceso Unificado de Desarrollo de Software (USDP)

■ Objetivo

- ◆ Guiar a los desarrolladores en la implementación y distribución eficiente de sistemas que se ajusten a las necesidades de los clientes.



Necesidades del cliente

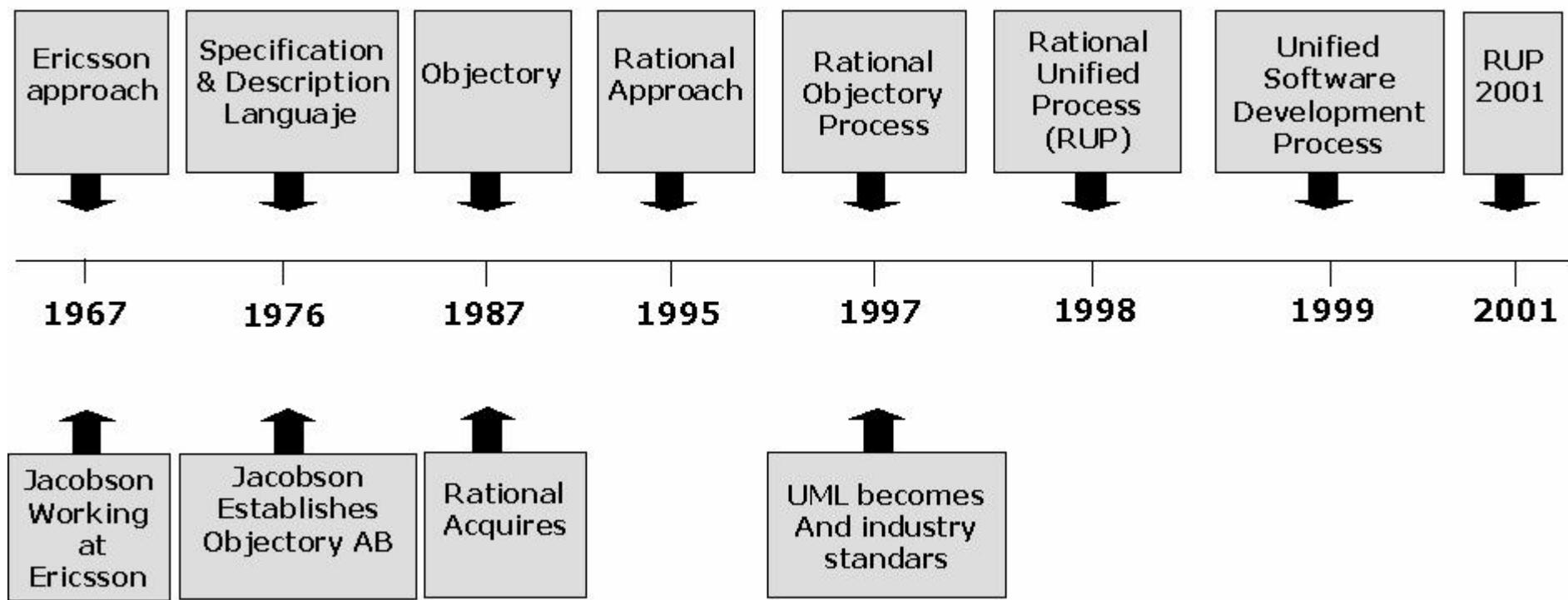


Implementación



Proceso Unificado de Desarrollo de Software (USDP)

■ Historia



Proceso Unificado

- Proceso genérico de desarrollo de software
 - ◆ Adaptarlo a una empresa y a cada proyecto en particular
 - Todos los proyectos no son iguales
 - ◆ Modela el “quién”
 - Trabajador (*worker*) 
 - ◆ Modela el “qué”
 - Actividades y artefactos  
 - ◆ Modela el “cómo”
 - Flujos de trabajo (*workflow*)    →

Proceso Unificado

- La adaptación debe definir e incluir
 - ◆ Estándares internos
 - ◆ Plantillas para documentos
 - ◆ Herramientas
 - Compiladores, gestión de configuración
 - ◆ Bases de datos
 - Seguimiento de errores y proyectos
 - ◆ Modificaciones del ciclo de vida

Proceso Unificado

■ Características

- ◆ **Está basado en componentes**
 - El sistema software en construcción está formado por componentes software interconectados a través de interfaces bien definidas
- ◆ **Utiliza el Lenguaje Unificado de Modelado (UML)**
 - Preparar todos los esquemas de un sistema software

■ Axiomas básicos

- ◆ **Dirigido por casos de uso y riesgos**
- ◆ **Centrado en la arquitectura**
- ◆ **Iterativo e incremental**

USDP: Axiomas (Dirigido por casos de uso)

■ ¿Qué significa?

- ◆ El proceso de desarrollo sigue un hilo que parte de los casos de uso.
 - Los casos de uso se especifican
 - Los casos de uso se diseñan
 - Los casos de uso son la fuente de los casos de prueba
- ◆ Se desarrollan a la vez que la arquitectura del sistema, no aisladamente
 - Los casos de uso guían la arquitectura del sistema
 - La arquitectura del sistema influye en la selección de los casos de uso
 - Maduran juntos.

USDP: Axiomas (Dirigido por casos de uso)

- Sistema software terminado
 - ◆ Dará cierto servicio a los usuarios
 - ◆ Sea éxito
 - Conocer lo que los **usuarios** necesitan y desean
- Usuario
 - ◆ Alguien o algo que **interactúa** con el sistema
- Una interacción es un **caso de uso**
 - ◆ Fragmento de funcionalidad del sistema que proporciona una funcionalidad importante
 - ◆ Representan los requisitos funcionales

USDP: Axiomas (Dirigido por casos de uso)

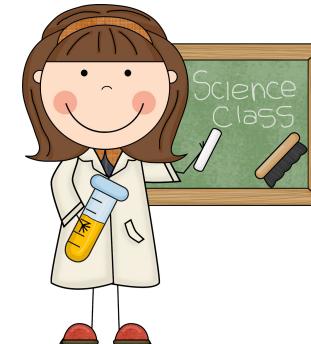
■ Especificación funcional

- ◆ ¿Qué debe hacer el sistema?
 - El sistema debe almacenar información sobre los grupos: tamaño máximo, actual, ..
 - El sistema debe almacenar información sobre los miembros de los grupos
 - El sistema no permitirá que un alumno se inscriba en un curso lleno
 - El sistema debe permitir consultar la composición de un grupo
 - El sistema debe permitir añadir nuevos usuarios a los grupos
 - El sistema debe permitir eliminar usuarios de los grupos

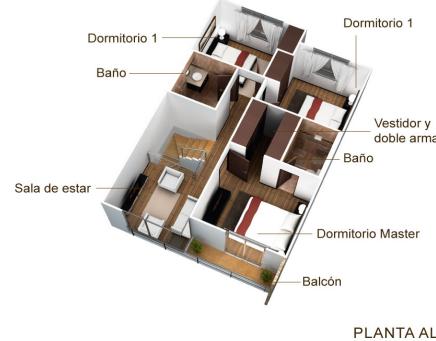
USDP: Axiomas (Dirigido por casos de uso)

■ Casos de uso

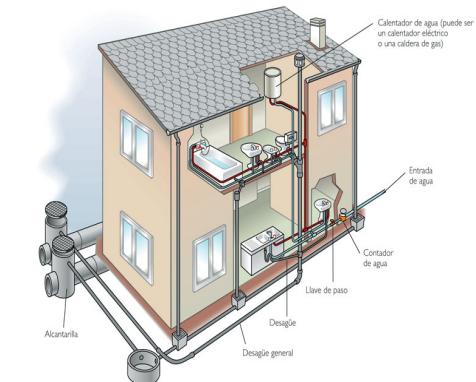
- ◆ ¿Que debe hacer el sistema **para cada usuario?**
 - Profesor
 - Crear un grupo nuevo
 - Consultar la composición de un grupo
 - Cerrar un grupo
 - Alumno
 - Consultar la composición de un grupo
 - Inscribirse en un grupo
 - Darse de baja de un grupo



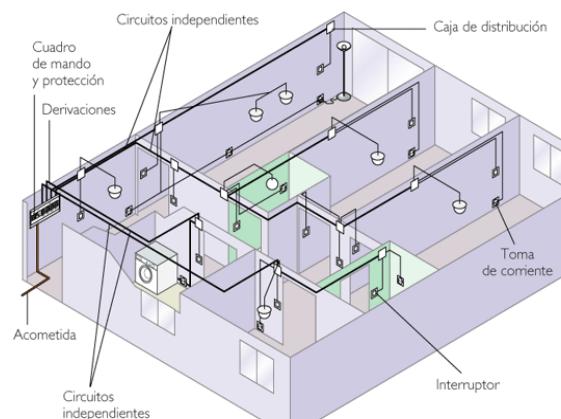
USDP: Axiomas (Centrado en la arquitectura)



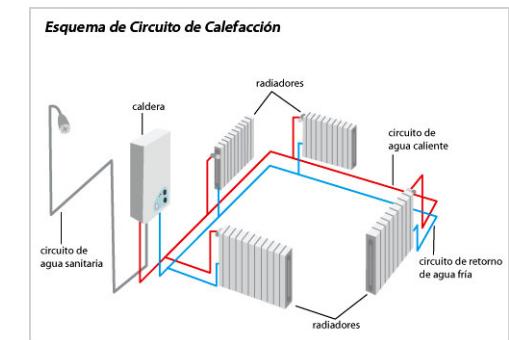
Estructura



Fontanería



Electricidad



Calefacción

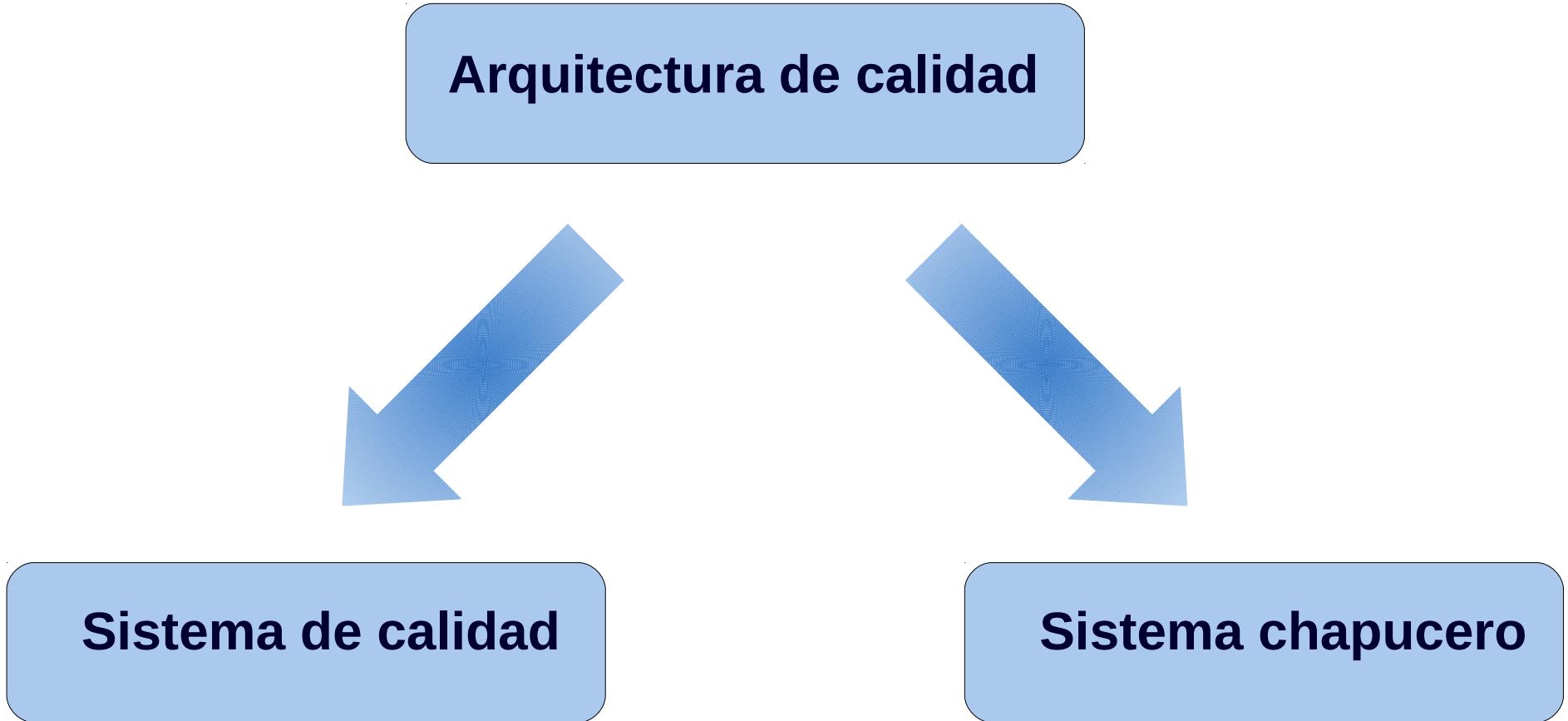
USDP: Axiomas (Centrado en la arquitectura)

- Arquitectura software define :
 - ◆ Visión estática
 - Componentes que llevan a cabo alguna tarea
 - ◆ Visión funcional
 - Sus interfaces (qué hace cada componente)
 - ◆ Visión dinámica
 - Comunicación entre ellos y comportamiento en el tiempo
- Ensamblar los diversos elementos de manera adecuada
 - ◆ Satisfacer funcionalidad y requerimientos del sistema

USDP: Axiomas (Centrado en la arquitectura)

- Influenciada por
 - ◆ Plataforma en la que debe funcionar el software
 - Arquitectura hardware
 - Sistema operativo
 - Sistema de gestión de bases de datos
 - Protocolos de comunicación
 - ◆ Bloques de construcción reutilizables
 - Marco de trabajo para interfaces gráficas de usuario
 - ◆ Sistemas heredados
 - ◆ Requisitos no funcionales
 - Rendimiento, fiabilidad

USDP: Axiomas (Centrado en la arquitectura)



USDP: Axiomas (Iterativo e incremental)

■ ¿Qué significa?

- ◆ Construimos el software en un proceso de mejora, paso a paso
 - Un proyecto se desglosa en pequeños subproyectos
 - Iteraciones
 - Los subproyectos distribuyen la funcionalidad en bloques
 - Incrementos

■ Cada iteración (mini-proyecto)

- ◆ Trata un conjunto de casos de uso
- ◆ Análisis, diseño, implementación y prueba
- ◆ Código ejecutable

USDP: Flujos de trabajo

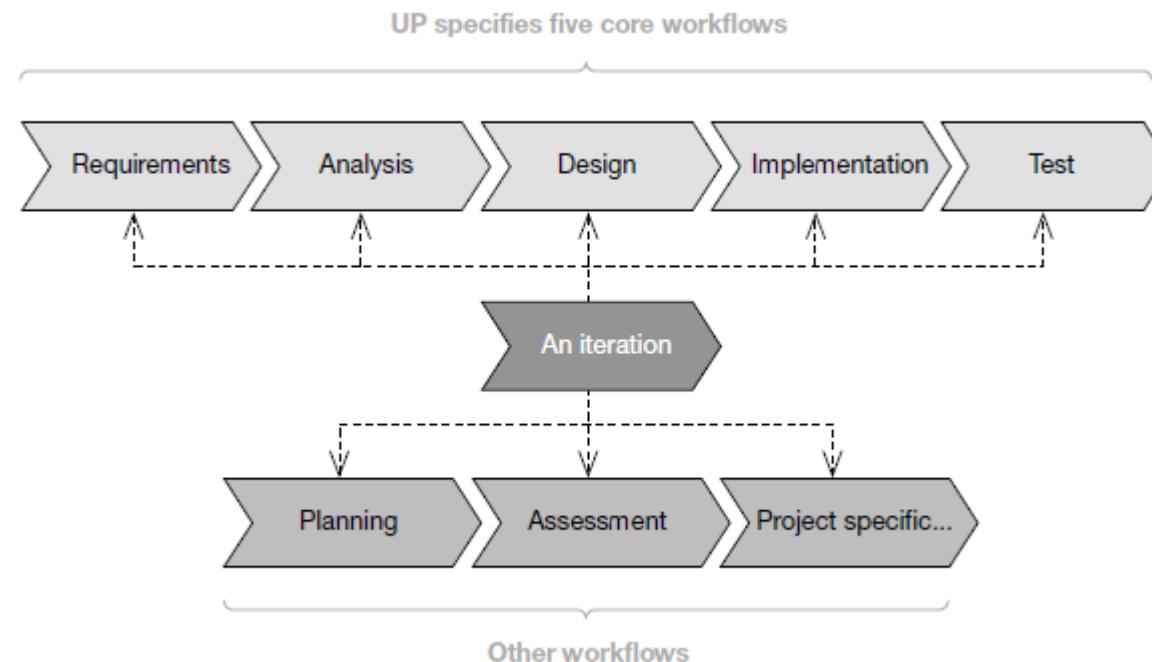
■ Cada iteración

- ◆ 5 *workflows* centrales
 - Requisitos
 - Capturar lo que el sistema debería hacer
 - Análisis
 - Refinar y estructurar los requisitos
 - Diseño
 - Realizar los requisitos en la arquitectura del sistema
 - Implementación
 - Construir el software
 - Prueba
 - Verificar que la implementación funciona según se desea

USDP: Flujos de trabajo

■ Cada iteración

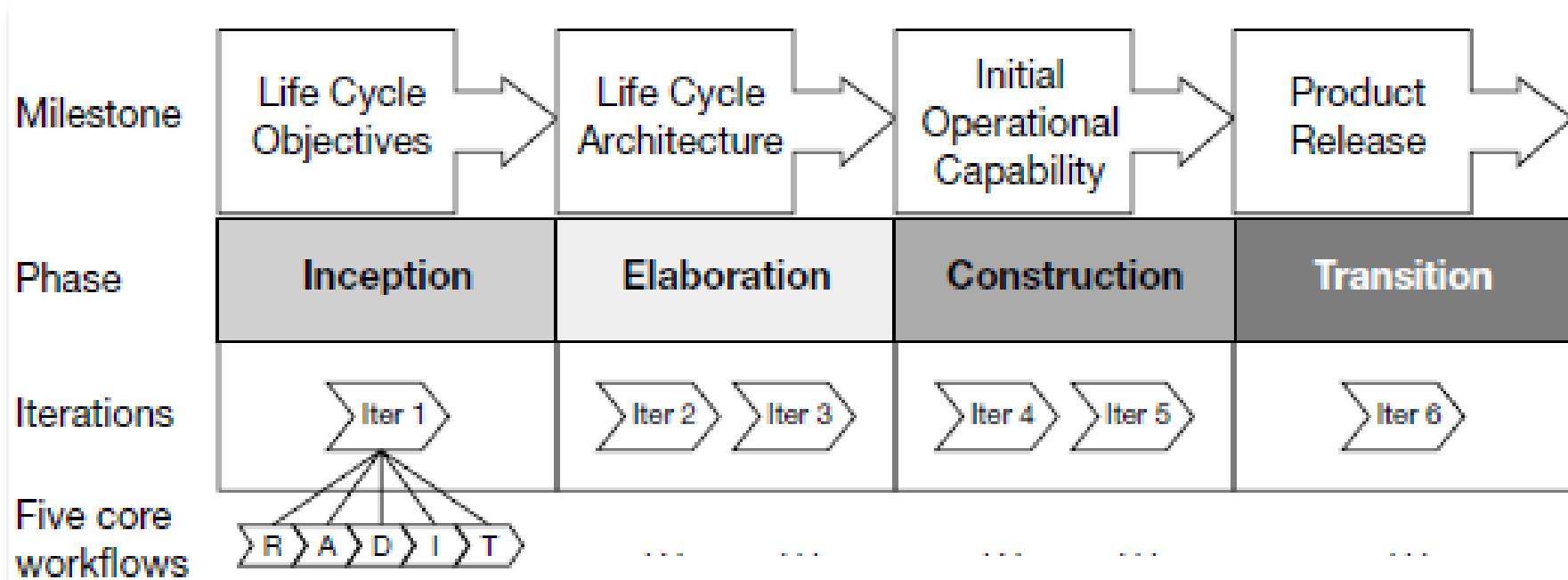
- ◆ Otros (no cubiertos por USDP)
 - Planificación, valoración, específicos del proyecto



USDP: Fases

- Ciclo de vida de un proyecto
 - ◆ 4 fases
 - **Comienzo**: objetivos del proyecto
 - **Elaboración**: arquitectura del sistema
 - **Construcción**: capacidad operativa inicial
 - **Transición**: entrega del producto
 - ◆ Cada fase una o más iteraciones
 - En cada iteración se ejecutan los 5 *workflows* y los otros que sean necesarios
 - El número de iteraciones/fase depende del proyecto
 - ◆ Cada fase termina con un **hito** importante
 - Proceso basado en el objetivo

USDP: Fases



USDP: Fases

En cada fase, la cantidad de trabajo que se realiza en cada *workflow* cambia

Flujos de trabajo fundamentales

Requisitos

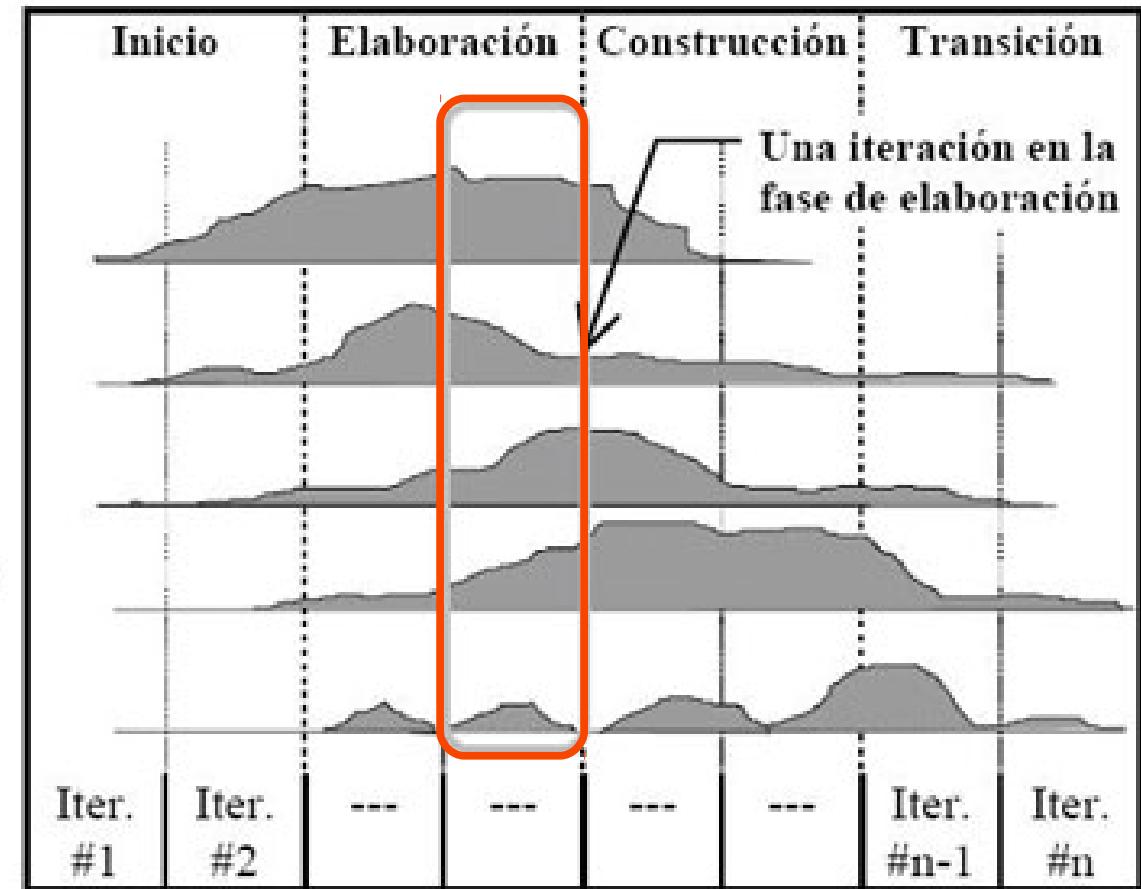
Análisis

Diseño

Implementación

Prueba

Fases



USDP: Fases (Comienzo)

■ Objetivos

Get the project off the ground

- ◆ Establecer la viabilidad del proyecto
 - Plan de trabajo
 - Cuánto costaría desarrollarlo
- ◆ Capturar los requisitos esenciales
 - ¿Cuáles son las principales funciones del sistema?
 - ¿Cómo podría ser la arquitectura del sistema?
- ◆ Identificar riesgos críticos



Gestor del proyecto



Arquitecto del sistema

USDP: Fases (Comienzo)

- Foco de actividad
 - ◆ Requisitos y Análisis
 - Énfasis principal
 - ◆ Diseño e Implementación
 - Si se decide crear algún prototipo técnico o prueba de concepto
 - ◆ Prueba
 - No es aplicable
 - Cualquier artefacto, sería un prototipo desecharable.

USDP: Fases (Comienzo)

■ Hito: Tener claro los objetivos

Condiciones de satisfacción	Entregable
Los grupos de decisión se ha puesto de acuerdo en los objetivos	Documento con los requisitos principales, características y restricciones
Se ha definido el ámbito de aplicación	Modelo inicial de casos de uso (10-20%)
Se han capturado los requisitos claves	Glosario del proyecto
Se ha acordado el coste y la estimación del calendario	Plan de proyecto inicial
Se ha lanzado un business case	Business case
Se ha realizado un análisis de riesgo	Documento de análisis de riesgos
Se ha confirmado la viabilidad por medio de estudios técnicos y/o prototipos	Uno o más prototipos desecharables
Se ha esbozado la arquitectura	Documento inicial de arquitectura

USDP: Fases (Elaboración)

■ Objetivos

- ◆ Crear una línea base ejecutable de la arquitectura
 - Sistema real, ejecutable que se crea de acuerdo a la arquitectura especificada
 - No es prototipo, sino el primer modelo del sistema
- ◆ Mejorar el análisis de riesgos
- ◆ Definir atributos de calidad
- ◆ Capturar casos de uso (80% requisitos funcionales)
- ◆ Crear un plan detallado de la fase de construcción
- ◆ Formular una oferta que incluya recursos, tiempo, personal y coste

USDP: Fases (Elaboración)

■ Foco de actividad

◆ Requisitos

- Mejora el ámbito de aplicación del sistema y requisitos

◆ Análisis

- Establece lo que se va a crear

◆ Diseño

- Crea una arquitectura estable

◆ Implementación

- Crea la línea base de la arquitectura

◆ Prueba

- Prueba la línea base de la arquitectura

USDP: Fases (Elaboración)

■ Hito: Arquitectura del sistema

Condiciones de satisfacción	Entregable
Se ha creado una línea base ejecutable de la arquitectura	La línea base ejecutable
Se han identificado y resueltos riesgos importantes	Modelo UML estático, dinámica y de casos de uso
Se ha estabilizado la visión del producto	Documento de visión
Se ha revisado el análisis de riesgos	Ánalysis de riesgos actualizado
Se ha revisado el business case	Business case actualizado
Se ha creado un plan de proyecto con suficiente detalle	Plan de proyecto actualizado
El business case se ha verificado con el plan del proyecto	Business case
Se llega a un acuerdo para continuar el proyecto	Documento firmado

USDP: Fases (Construcción)

■ Objetivos

- ◆ Crear el producto
 - Completar todos los requisitos, análisis y diseño
 - Producto para ser entregado
- ◆ Evolucionar la linea base de la arquitectura generada en la elaboración hacia el sistema final
 - Mantener la integridad de la arquitectura

USDP: Fases (Construcción)

■ Foco de actividad

- ◆ Requisitos
 - Descubrir cualquier requisito que hubiera pasado
- ◆ Análisis
 - Finalizar el modelo de análisis
- ◆ Diseño
 - Finalizar el modelo de diseño
- ◆ Implementación
 - Crear la capacidad operativa inicial
- ◆ Prueba
 - Probar la capacidad operativa inicial

USDP: Fases (Construcción)

■ Hito: Capacidad operativa inicial

Condiciones de satisfacción	Entregable
El producto software es suficientemente estable y de suficiente calidad para desplegarse a la comunidad de usuarios	El producto software, el modelo UML, la suite de prueba
Los grupos de decisión se han puesto de acuerdo y están preparados para la transición del software a su entorno	Manuales de usuario, descripción de esta versión
Los gastos reales vs los gastos planificados son aceptables	Plan de proyecto

USDP: Fases (Transición)

■ Objetivos

- ◆ Corregir defectos
 - Sobre la versión beta del sistema desplegado
 - Usuarios con experiencia prueban el producto
 - Desarrolladores corrigen los problemas e incorporan mejoras sugeridas
- ◆ Preparar los sitios del usuario para el nuevo software y adaptar el software para que funcione en ellos
- ◆ Crear manuales de usuario y otra documentación
- ◆ Proporcionar consultoría al usuario
- ◆ Realizar una revisión después del proyecto

USDP: Fases (Transición)

■ Foco de actividad

- ◆ Requisitos y Análisis
 - No aplicable o poco trabajo
- ◆ Diseño
 - Modificar el diseño si aparecen problemas
- ◆ Implementación
 - Adaptar el software para el sitio del usuario
 - Corregir los problemas descubiertos
- ◆ Prueba
 - Prueba beta
 - Prueba de aceptación en el sitio del usuario.

USDP: Fases (Transición)

■ Hito: Versión del producto

Condiciones de satisfacción	Entregable
Se ha completado la prueba beta, se han realizado los cambios necesarios y los usuarios están de acuerdo en que el sistema se ha desplegado con éxito	El producto software
La comunidad de usuarios está utilizando activamente el producto	
Se han acordado estrategias de soporte del producto con los usuarios y se han implementado	Plan de soporte de usuarios Manuales de usuarios actualizados