



Proceso Software y Ciclo de Vida

Curso 2008-2009

Gonzalo Méndez
Dpto. de Ingeniería de Software e Inteligencia Artificial
Facultad de Informática
Universidad Complutense de Madrid

Conceptos importantes

- Personas
 - los que trabajan
- Producto
 - lo que se obtiene
- Proyecto
 - la pauta a seguir para desarrollar un producto
- Proceso
 - la pauta a seguir para desarrollar un proyecto

Un traje

- Personas
 - El sastre
- Producto
 - El traje
- Proyecto:
 - el sastre, el traje, el presupuesto del traje, el traje en sí, los pasos a dar para hacer el traje...
- Proceso
 - La secuencia de acciones para hacer un traje concreto

Una cena

- Personas
 - Empleados de una empresa de catering
- Producto
 - La cena que se sirve
- Proyecto
 - El menú, el presupuesto, lo que hay que hacer para conseguir el menú, ...
- Proceso
 - La secuencia de acciones de servir una cena

Una gama de automóviles

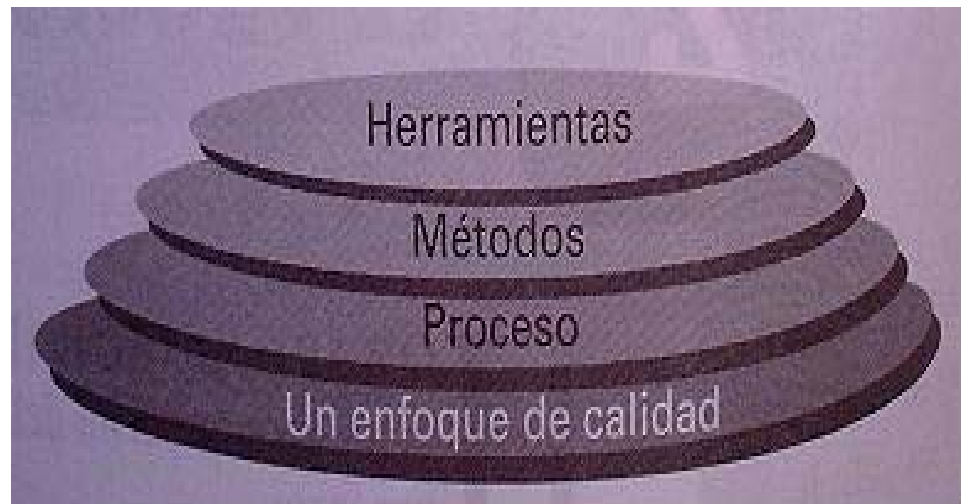
- Personas
 - Empleados de la marca
- Producto
 - Los automóviles
- Proyecto
 - Desarrollo de un modelo nuevo
- Proceso
 - Las instrucciones de la empresa sobre cómo desarrollar un modelo nuevo

Para vosotros

- Personas
 - vuestro grupo
- Producto
 - la aplicación elegida
- Proyecto
 - parte práctica IS
- Proceso
 - entregas mensuales + cómo vosotros decidáis organizaros

Capas de la IS

- Capa de enfoque de calidad
- Capa de proceso
- Capa de métodos
- Capa de herramientas



Capas de la IS

- Capa de calidad
 - Base de cualquier proceso de ingeniería
 - La IS se basa en calidad
 - Mejores técnicas de construcción de software
- Capa de proceso
 - Capa que une calidad y métodos
 - Desarrollo racional de la IS
 - Conjunto de actividades y resultados asociados que sirven para construir un producto software

Capas de la IS

- Capa de métodos
 - Un método incluye:
 - Análisis de requisitos
 - Diseño
 - Construcción de programas
 - Prueba
 - Mantenimiento
 - Suelen estar bastante ligados al proceso
- Capa de herramientas
 - Soporte automático o semiautomático para el proceso y los métodos
 - Herramientas CASE

Visión general de la IS

- Con independencia del modelo de proceso hay tres fases genéricas:
 - Fase de definición
 - Fase de desarrollo
 - Fase de mantenimiento
- Cada una de estas fases se descompone en un conjunto de tareas

Fase de definición/especificación

- Se identifican requisitos de sistema y software:
 - Información a procesar
 - Función y rendimiento deseados
 - Comportamiento del sistema
 - Interfaces establecidas
 - Restricciones de diseño
 - Tareas principales:
 - Planificación del proyecto software
 - Ingeniería de sistemas o de información
 - Análisis de requisitos

Fase de desarrollo

- Se define:
 - Cómo diseñar las estructuras de datos
 - Cómo implementar las funciones
 - Cómo caracterizar las interfaces
 - Cómo traducir el diseño a programación
 - Cómo validar el producto (pruebas, verificación)
 - Tareas principales:
 - Diseño del software
 - Generación del código
 - Pruebas del software

Fase de mantenimiento

- Centrada en cambios que se pueda necesitar realizar sobre un producto
- Se vuelven a aplicar las fases de definición y desarrollo, pero sobre software ya existente
- Pueden producirse cuatro tipos de cambio:
 - **Corrección:** Corregir los defectos
 - **Adaptación:** Modificaciones por cambio externo
 - **Mejora:** Ampliar los requisitos funcionales originales, a petición del cliente
 - **Prevención:** Cambio para facilitar el cambio

Visión general de la IS

- Estas fases se complementan con las actividades *de soporte*
 - No crean software
 - Mejoran su *calidad*
 - Facilitan su desarrollo
- Se aplican a lo largo de todo el proceso del software

Visión general de la IS

- Ejemplos de actividades *de soporte*
 - Documentación
 - Gestión de configuración
 - Seguimiento y control del proyecto de software
 - Revisiones técnicas formales
 - Garantía de la calidad del software
 - Gestión de reutilización
 - Mediciones
 - Gestión de riesgos

Proceso software

- Conjunto estructurado de actividades y resultados asociados requeridos para desarrollar un sistema de software
 - **Especificación:** establecer requisitos y restricciones
 - **Diseño:** Producir un modelo en papel del sistema
 - **Implementación:** construcción del sistema de software
 - **Validación:** verificar (por ejemplo mediante pruebas) que el sistema cumple con las especificaciones requeridas
 - **Instalación:** entregar el sistema al usuario
 - **Evolución y mantenimiento:** cambiar/adaptar el software según las demandas; reparar fallos en el sistema

Modelos de proceso

- Un modelo de proceso, o paradigma de IS, es una plantilla, patrón o marco que define el proceso a través del cual se crea software
- Dicho de otra forma, los procesos son instancias de un modelo de proceso
- En esta asignatura los términos proceso y modelo de proceso se utilizan indistintamente

Modelos de proceso

- Una organización podría variar su modelo de proceso para cada proyecto, según:
 - La naturaleza del proyecto
 - La naturaleza de la aplicación
 - Los métodos y herramientas a utilizar
 - Los controles y entregas requeridas

Características del proceso

- **Entendible**
- **Visibilidad**: Grado en que las actividades del proceso proporcionan resultados
- **Soportable** por herramientas CASE
- **Aceptabilidad**: Grado en que los desarrolladores aceptan y usan el proceso
- **Fiabilidad**: Capacidad de evitar o detectar errores antes de que sean defectos
- **Robustez**: Continuidad del proceso a pesar de los problemas
- **Mantenible**: Capacidad de evolución para adaptarse
- **Rapidez**: Velocidad en que el proceso puede proporcionar un sistema a partir de una especificación

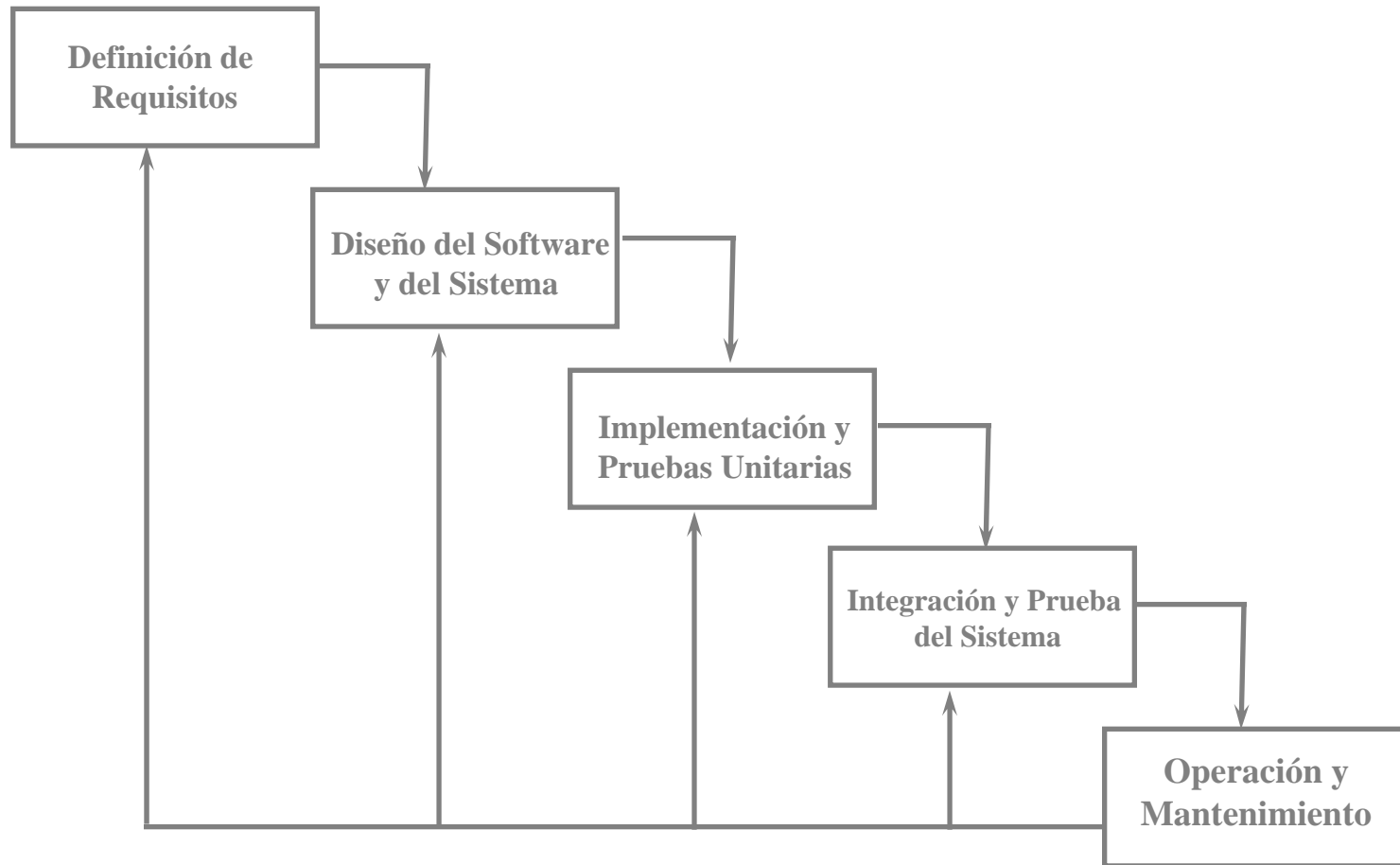
Modelos Genéricos de Desarrollo de Software

- Modelo de Cascada
- Prototipado
- Desarrollo Evolutivo
- En espiral
- Desarrollo basado en componentes
- Métodos Formales

Modelo en cascada (*waterfall*)

- Modelo de proceso clásico (desde los 70)
- Basado en la mentalidad de línea de ensamblaje (cartesiano)
- Es sencillo y fácil de entender
- El proyecto pasa a través de una serie de fases
 - Al final de cada fase se revisan las tareas de trabajo y productos
 - Para poder pasar a la siguiente fase se tiene que haber conseguido todos los objetivos de la fase anterior
 - No hay apenas comunicación entre las fases

Modelo en cascada (*waterfall*)



Modelo en cascada (*waterfall*)

- Fases:
 - **Conceptualización:** Se determina la arquitectura de la solución (división del sistema en subsistemas)
 - **Análisis de requisitos:** Básicamente se definen los requisitos funcionales y de rendimiento
 - **Diseño:** Representación de la aplicación que sirve de guía a la implementación
 - **Implementación:** Transforma el diseño en código
 - **Prueba:** Validación e integración de software y sistemas
 - **Instalación y comprobación:** Se instala el software al cliente, el cual comprueba la corrección de la aplicación
- Se supone que sólo se baja en la cascada... pero también se puede subir (aunque difícilmente)

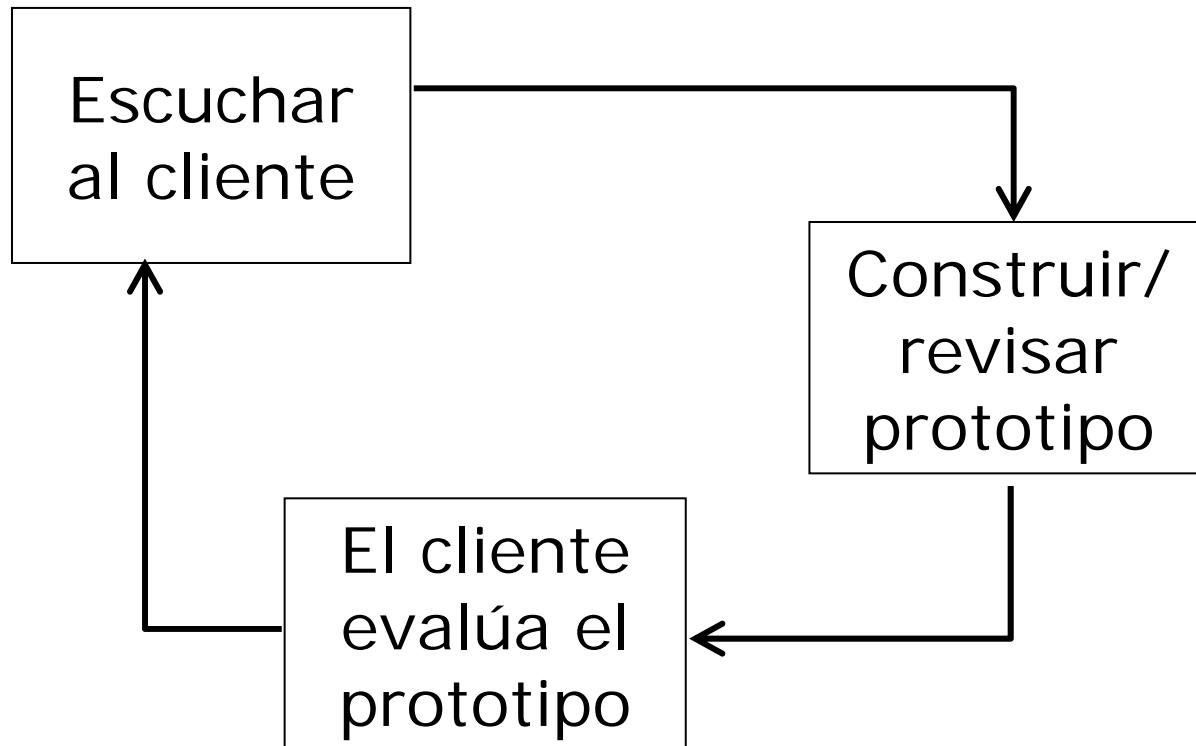
Modelo en cascada (*waterfall*)

- Posibles ventajas:
 - Sencillo: Sirve cuando el personal está poco cualificado
 - Aplicable cuando el problema es estable y cuando se trabaja con técnicas conocidas
- Críticas:
 - No se ve un producto hasta muy tarde en el proceso
 - Un error grave detectado en las últimas fases puede ser letal
 - Especificación de requisitos estable
 - Impone una estructura de gestión de proyectos
 - Fases muy rígidas
 - Las revisiones de proyectos de gran complejidad son muy difíciles

Prototipado

- Se usa un prototipo para dar al usuario una idea concreta de lo que va a hacer el sistema
- Se aplica cada vez más cuando la rapidez de desarrollo es esencial
- Prototipado evolutivo: el prototipo inicial se refina progresivamente hasta convertirse en versión final
- Prototipado desechable: de cada prototipo se extraen ideas buenas que se usan para hacer el siguiente, pero cada prototipo se tira **entero**

Prototipado



Prototipado

- Comienza con la recolección de requisitos
 - Cliente y desarrolladores definen los objetivos globales del software.
 - Además, identifican los requisitos conocidos y aquellos que deben ser más definidos.
- Aparece un diseño rápido centrado en los aspectos visibles para el cliente (e.g. información de E/S)
 - El diseño rápido lleva a la construcción de un prototipo
- El prototipo lo evalúa el cliente y lo utiliza para refinar los requisitos
- El proceso se itera... ¿desechando el prototipo?

Prototipado

■ Ventajas

- Permite identificar los requisitos incrementalmente
- Permite probar alternativas a los desarrolladores
- Tiene una alta visibilidad → tanto clientes como desarrolladores ven resultados rápidamente

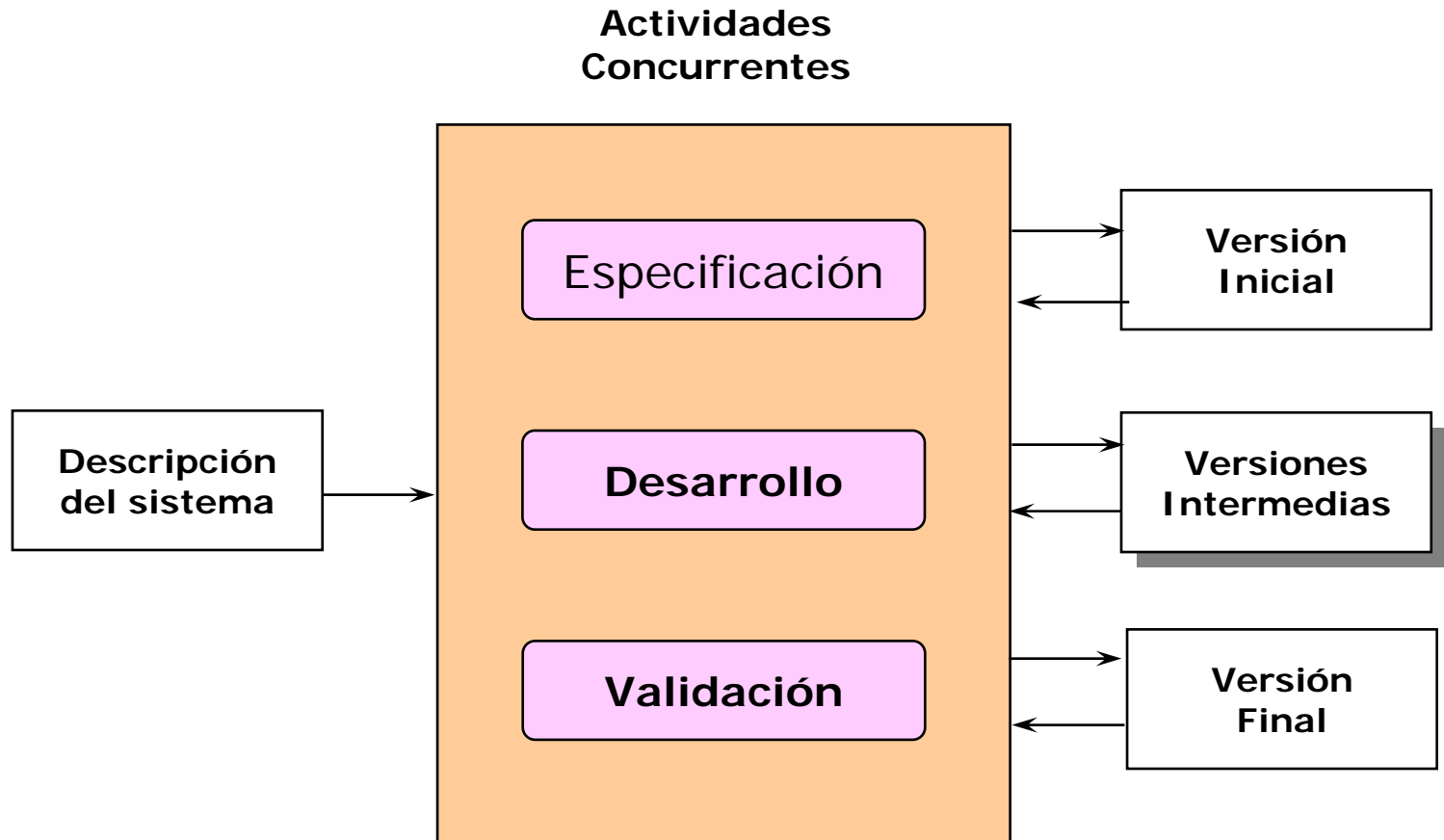
■ Inconvenientes

- El cliente no entiende por qué hay que desechar el prototipo
 - Si simplemente ha pedido unos ajustes...(¿?)
- Riesgo de software de baja calidad
 - Compromisos de implementación para que el prototipo funcione rápidamente y que al final son parte integral del sistema
 - Utilizar un SO o lenguaje de programación inadecuado pero conocido

Modelos evolutivos

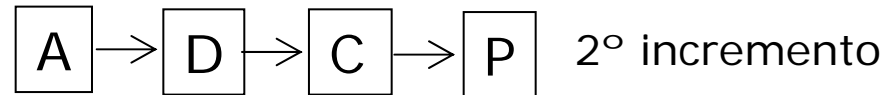
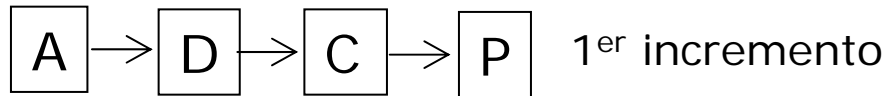
- Características:
 - Gestionan bien la naturaleza evolutiva del software
 - Son iterativos: construyen versiones de software cada vez más completas
- Se adaptan bien:
 - Los cambios de requisitos del producto
 - Fechas de entrega estrictas poco realistas
 - Especificaciones parciales del producto

Modelos evolutivos



Modelos evolutivos. Incremental

- Fusiona el modelo lineal secuencial con el de construcción de prototipos



..... N-ésimo incremento

Modelos evolutivos. Incremental

- Cada secuencia lineal secuencial produce un incremento del software
 - Incremento: producto operacional de una parte del sistema
- El primer incremento suele ser el núcleo
 - Requisitos básicos
 - Muchas funciones suplementarias se dejan para después
- Se evalúa (p.ej., por el cliente) el producto entregado
 - Como resultado se desarrolla un plan para el siguiente
- Se itera
 - Hasta elaborar el producto completo

Modelos evolutivos. Incremental

■ Ventajas

- Es interactivo

- Con cada incremento se entrega al cliente un producto operacional al cliente, que puede evaluarlo

- Personal

- Permite variar el personal asignado a cada iteración

- Gestión riesgos técnicos

- Por ejemplo, disponibilidad de hardware específico

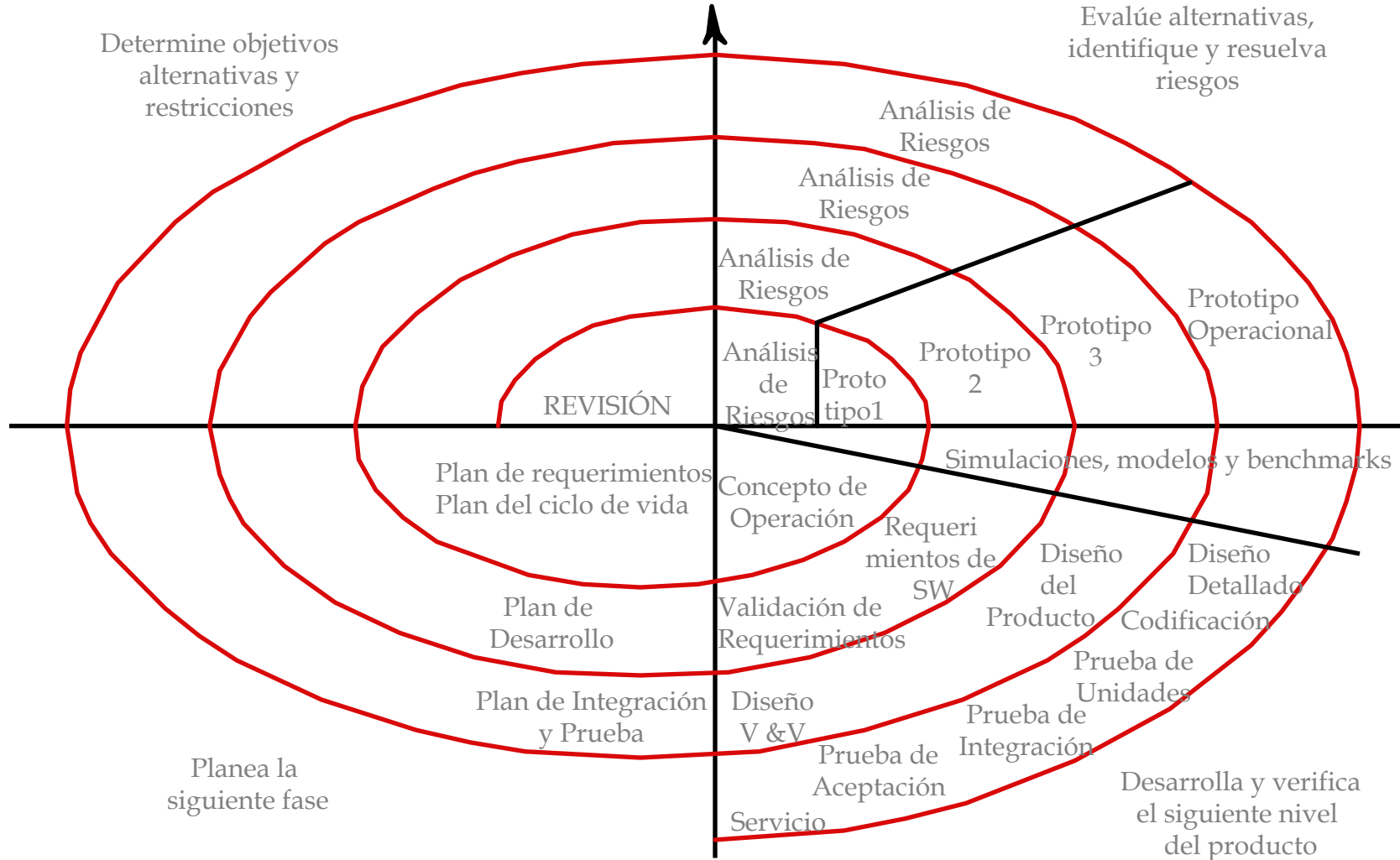
■ Inconvenientes

- La primera iteración puede plantear los mismos problemas que en un modelo lineal secuencial

Modelo de Proceso en Espiral

- Original: Boehm, 1988
- IEEE Std. 1490-1998
- Tratar primero las áreas de mayor riesgo
- Múltiples iteraciones sobre varias regiones de tareas
 - Vuelta a la espiral: ciclo
 - Número de iteraciones predeterminadas o calculadas dinámicamente
- Se pueden variar las actividades de desarrollo: familia de modelos de procesos

Modelo de Proceso de Espiral



Modelo de Proceso en Espiral

- El modelo en espiral es bastante adecuado para la gestión de riesgos
- Se puede añadir una actividad de gestión de riesgos
- De hecho, el modelo original de Boehm:
 - Fijar objetivos
 - Gestionar y reducir riesgos
 - Desarrollo y validación
 - Planificar siguiente ciclo

Modelos Evolutivos. Espiral Boehm

- Fijar objetivos
 - Definir objetivos del ciclo
 - Identificar restricciones del proceso y producto
 - Desarrollar plan de gestión
 - Identificar riesgos
 - Identificar estrategias alternativas
- Gestionar y reducir el riesgo
 - RSGR para cada riesgo identificado

Modelos Evolutivos. Espiral Boehm

- Desarrollo y validación
 - Elegir modelo de desarrollo
 - Algunos autores lo denominan metamodelo
 - Yo prefiero llamarlo modelo paramétrico
- Planificación
 - Revisión del proyecto
 - Decisión de una nueva vuelta

Modelos de Proceso en Espiral

■ Ventajas

- Enfoque realista
- Gestión explícita de riesgos
- Centra su atención en la reutilización de componentes y eliminación de errores en información descubierta en fases iniciales
- Los objetivos de calidad son el primer objetivo
- Integra desarrollo con mantenimiento

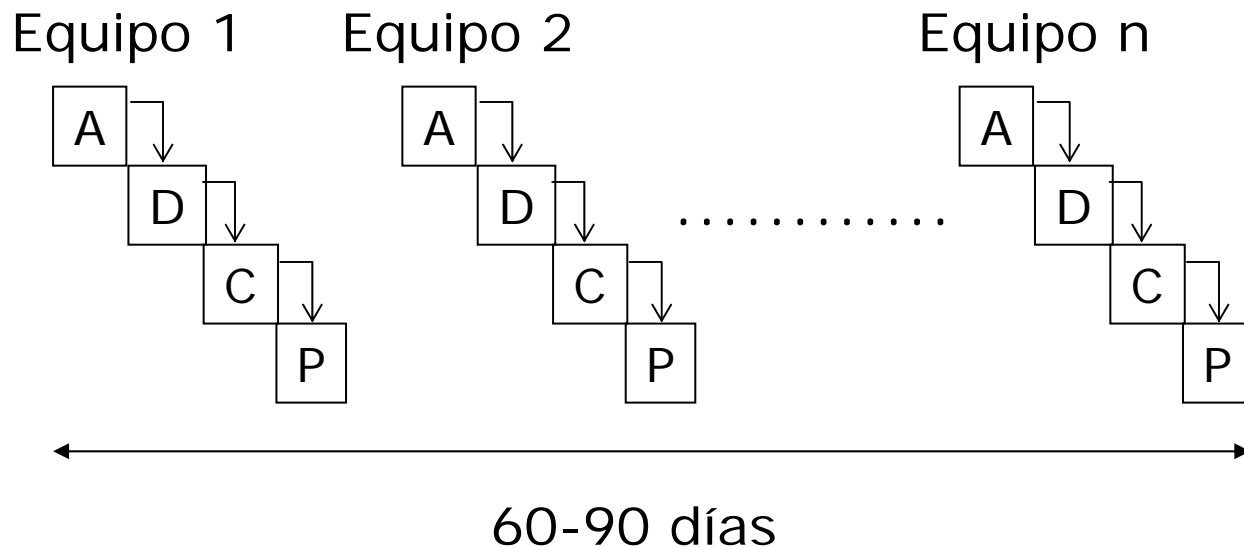
Modelos de Proceso en Espiral

■ Inconvenientes

- Convencer cliente enfoque controlable
- Requiere de experiencia en la identificación de riesgos
- Requiere refinamiento para uso generalizado

Desarrollo Basado en Componentes

- Desarrollo de sistemas en poco tiempo
- Adaptación a “alta velocidad” de la cascada
 - Equipos trabajando en paralelo
 - Aplicando tecnología de componentes



Desarrollo Basado en Componentes

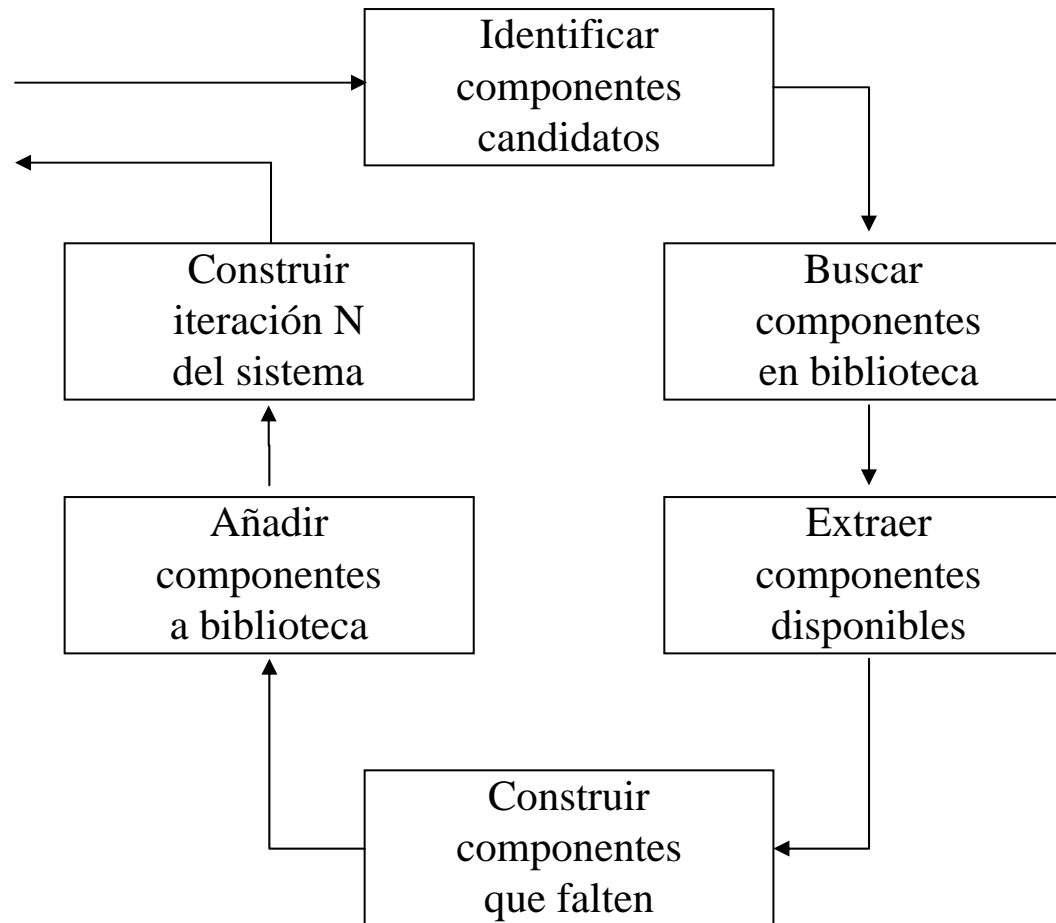
■ Ventajas

- Rapidez
- Válido para aplicaciones modularizables

■ Inconvenientes

- Exige conocer bien los requisitos y delimitar el ámbito del proyecto
- Número de personas
- Clientes y desarrolladores comprometidos
- Gestión riesgos técnicos altos
 - Uso de nueva tecnología
 - Alto grado de interoperabilidad con sistemas existentes

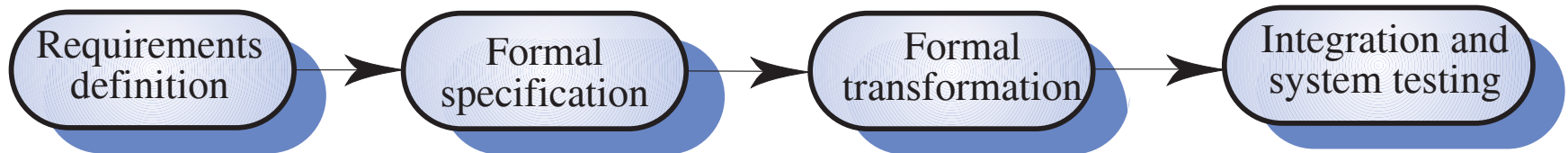
Desarrollo Basado en Componentes



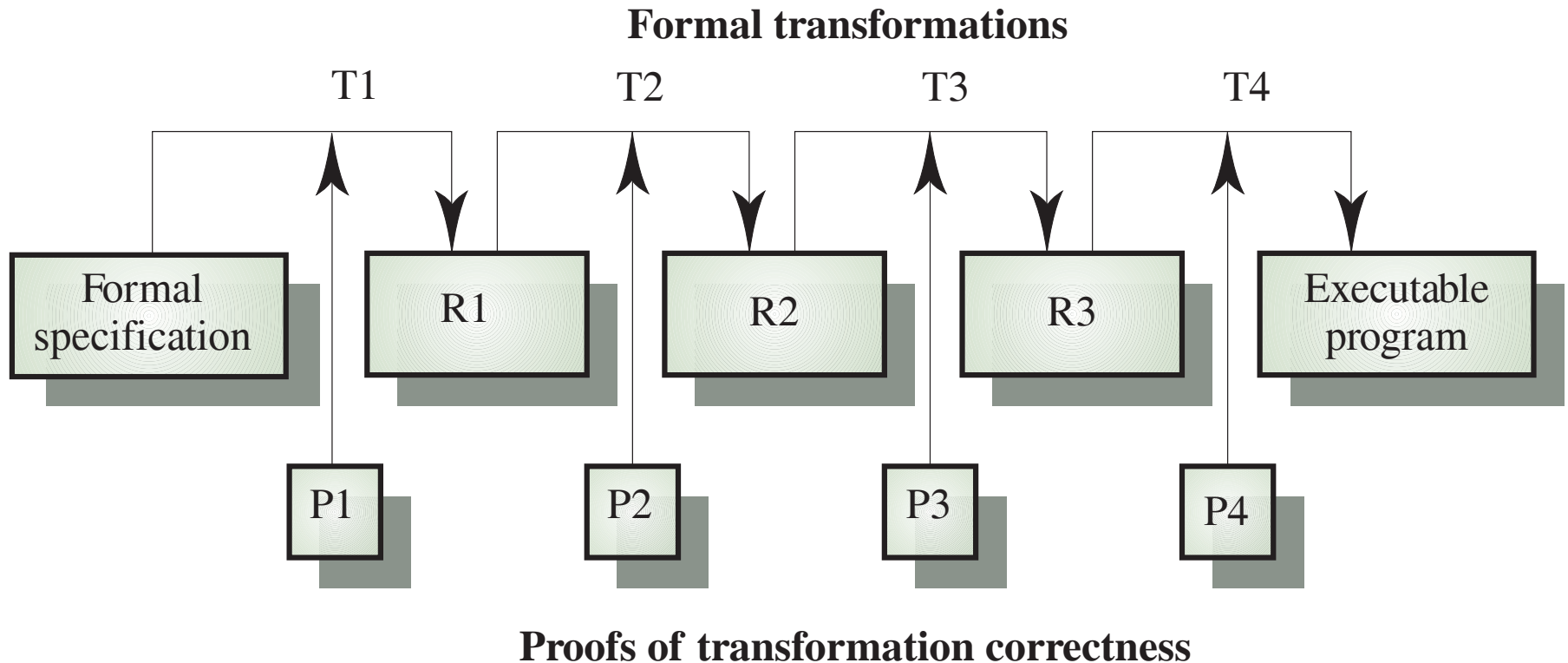
Desarrollo con métodos formales

- Se basa en la transformación de una especificación formal a lo largo de varias representaciones hasta llegar a un programa ejecutable
- Las transformaciones preservan la corrección
 - Permiten comprobar fácilmente que el programa es conforme a la especificación

Desarrollo con métodos formales



Transformaciones formales



Desarrollo formal de sistemas

■ Problemas

- Hace falta una formación especializada para aplicar la técnica
- Muchos aspectos de los sistemas reales son difíciles de especificar formalmente
 - Interfaz de usuario
 - Requisitos no funcionales

■ Aplicabilidad

- Sistemas críticos en los que la seguridad y fiabilidad debe poder asegurarse antes de poner el sistema en operación

Visibilidad de Procesos

- Los sistemas de software son intangibles por lo que los administradores necesitan documentación para identificar el progreso en el desarrollo
- Esto puede causar problemas
 - El tiempo planeado para entregar resultados puede no coincidir con el necesario para completar una actividad
 - La necesidad de producir documentos restringe la iteración entre procesos
 - Tiempo para revisar y aprobar documentos significativo
- El modelo de cascada es aún el modelo basado en resultados mas utilizado

Visibilidad de Procesos

Modelo de Proceso	Visibilidad del Proceso
Modelo de Cascada	Buena visibilidad, cada actividad produce un documento o resultado
Desarrollo Evolutivo	Visibilidad pobre, muy caro al producir documentos en cada iteración
Modelos Formales	Buena visibilidad, en cada fase deben producirse documentos
Desarrollo orientado a la reutilización	Visibilidad moderada. Importante contar con documentación de componentes reutilizables
Modelo de Espiral	Buena visibilidad, cada segmento y cada anillo del espiral debe producir un documento

¿Qué modelo utilizar?

- Para sistemas bien conocidos se puede utilizar el Modelo de Cascada. La fase de análisis de riesgos es relativamente fácil
- Con requisitos estables y sistemas de seguridad críticos, se recomienda utilizar modelos formales
- Con especificaciones incompletas, el modelo de prototipado ayuda a identificarlos y va produciendo un sistema funcional
- Pueden utilizarse modelos híbridos en distintas partes del desarrollo

Ejemplos

- Dos modelos de proceso concretos:
 - Proceso Unificado de Rational (pesado)
 - Extreme Programming (ágil)

Proceso Unificado

- Los autores de UML
 - Booch: método Booch
 - Rumbaugh: OMT
 - Jacobson: proceso Objectory
- También conocido como RUP: Rational Unified Process

Proceso Unificado

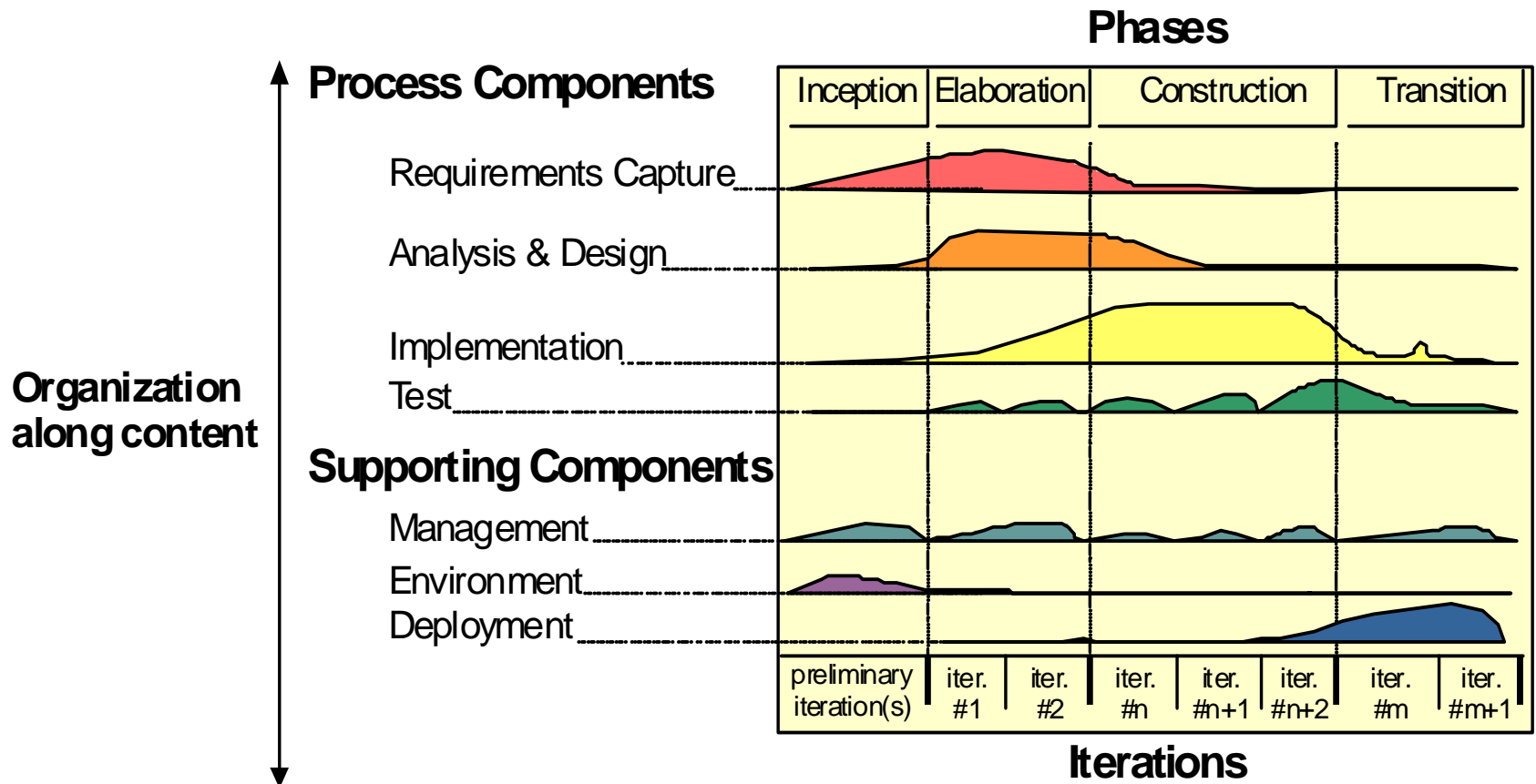
- Es un proceso de desarrollo de software
 - Dirigido por casos de uso
 - Centrado en la arquitectura
 - Iterativo e incremental
- Utiliza UML para definir los modelos del sistema software
- El sistema software en construcción está formado por
 - *componentes* software
 - interconectados a través de *interfaces*



Los requisitos cambian
y el sistema software evoluciona

Proceso Unificado

Organization along time



Extreme Programming (XP)

- Modelo de proceso de Kent Beck

Un modo ligero, eficiente, de bajo riesgo, flexible, predecible, científico y divertido de producir software

- Características

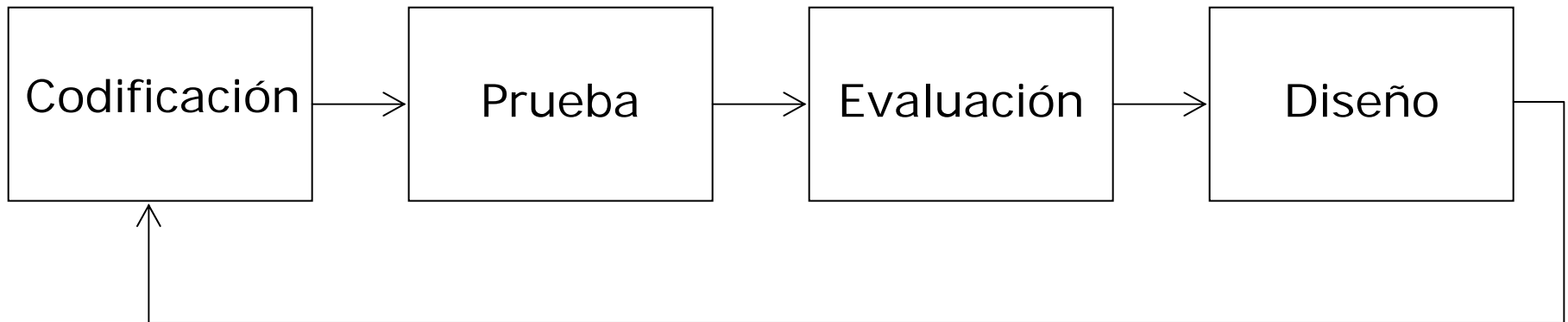
- Alta visibilidad debido a ciclos muy cortos
- Planificación incremental
- Se adapta a cambios de negocio

Extreme Programming (XP)

- Basado en pruebas automatizadas escritas por desarrolladores y clientes (*test-driven*)
- Alta comunicación
- Diseño evolutivo
- Colaboración entre programadores
- Busca equilibrio entre las necesidades a corto plazo de los programadores y las de largo plazo del proyecto

Extreme Programming (XP)

- La estructura del proceso, si la hay, es un poco atípica



Actividades en XP

Extreme Programming (XP)

- Las cuatro actividades están soportadas por doce prácticas:
 - El juego de planificación
 - Pequeñas entregas
 - Metáfora
 - Diseño simple
 - Prueba
 - Refactoring
 - Programación en pareja
 - Propiedad colectiva
 - Integración continua
 - Semana de cuarenta horas
 - Cliente en el lugar de desarrollo
 - Codificación estándar

Extreme Programming (XP)

■ Ventajas:

- Bueno para especificaciones cambiantes
- Fundamentación práctica

■ Inconvenientes:

- Poco probado
- Poco compatible con especificaciones/diseños totales
- Sólo funciona con equipos pequeños (hasta diez personas)

Extreme Programming (XP)

- Diferencias fundamentales (hay más que ya se verán)
 - No hay requisitos explícitos sino que el cliente participa en el desarrollo
 - Se empieza por automatizar las pruebas
 - Se desarrolla siempre la versión más simple posible que resuelva el problema
 - Se ejecutan todas las pruebas todos los días
 - Se cambia el diseño (aunque sea radicalmente) siempre que haga falta

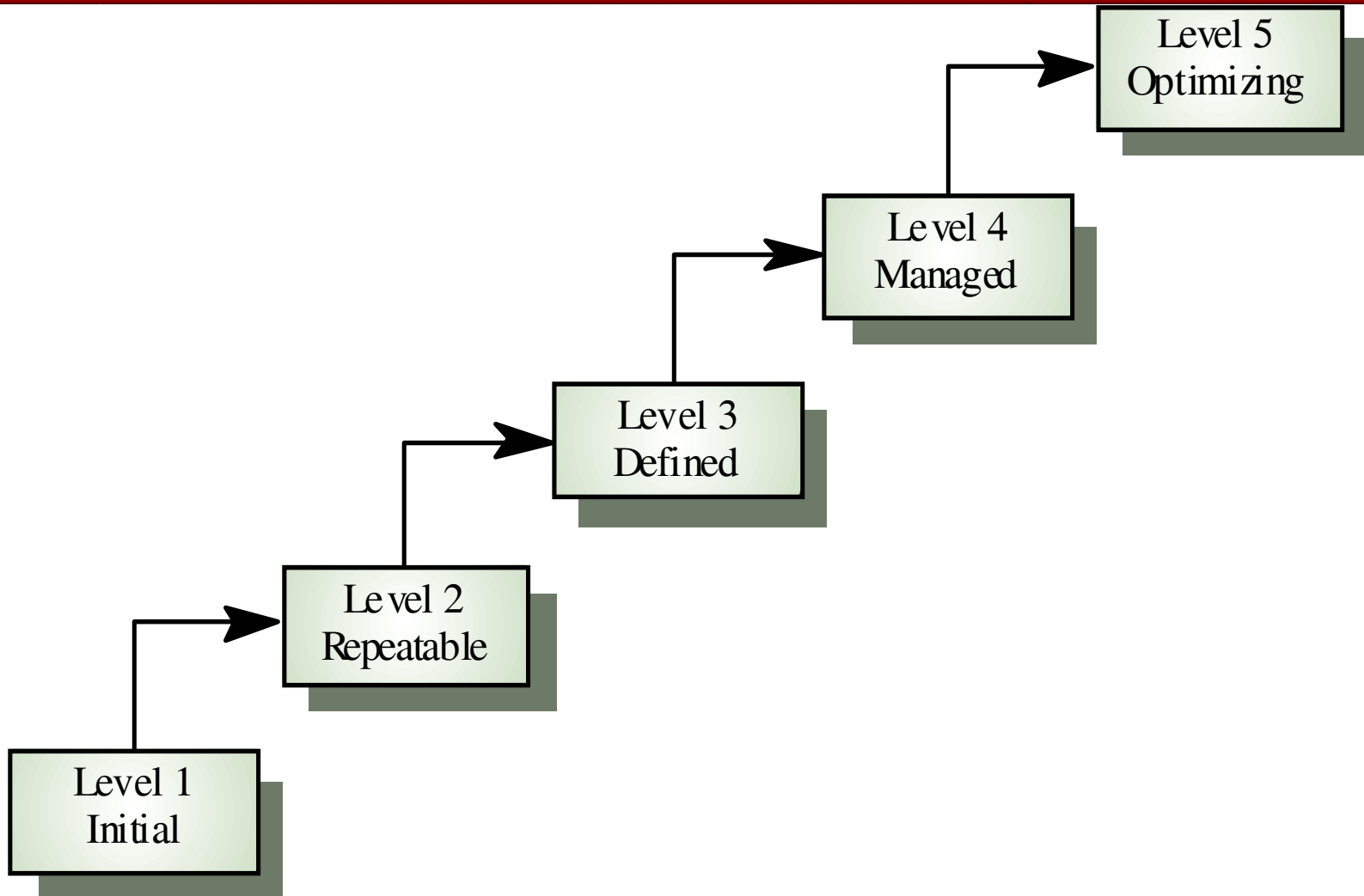
Calidad del proceso de software

¿ Cómo medir la calidad del
proceso de software de una
empresa ?

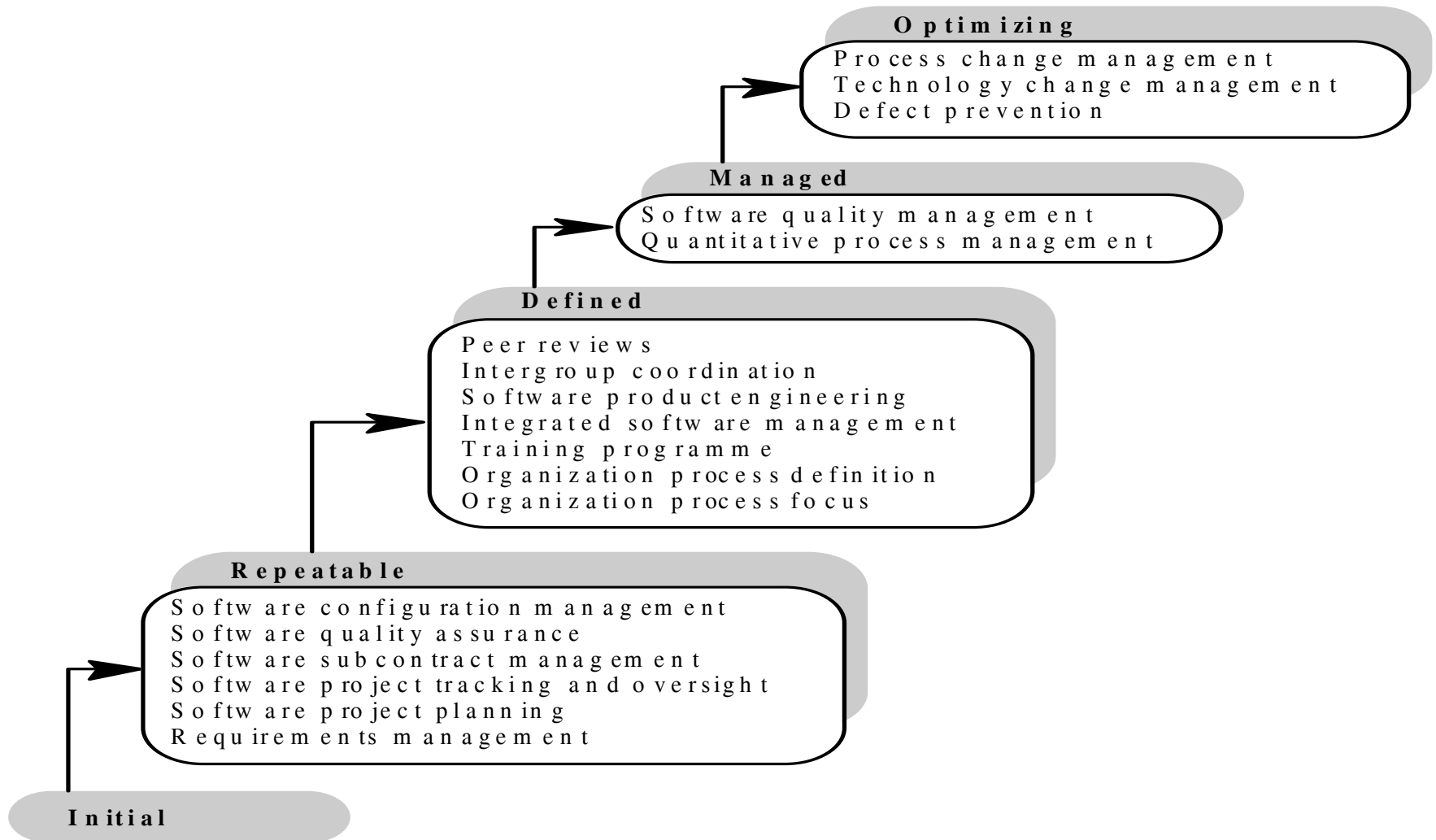
La empresa ideal

- El Dpto. de la Defensa de los US fundó el Software Engineering Institute (SEI) asociado con la Universidad de Carnegie Mellon (CMU).
- Desarrollan el Software Capability Maturity Model (SW CMM) a mediados de 1980s, refinado en los inicios de 1990s.
- Este modelo define una serie de áreas clave de proceso (ACP)
 - Un área clave de proceso es, básicamente, una actividad de IS

Software Capability Maturity Model



Áreas clave del proceso



CMM

- Los niveles son acumulativos
- Nivel 1: Inicial
 - El proceso de software se caracteriza según el caso
 - Se definen poco procesos
 - El éxito depende del esfuerzo individual

CMM

■ Nivel 2: Repetible

- Se incluye seguimiento del coste, la planificación y la funcionalidad
- Se repiten técnicas de proyectos anteriores con buenos resultados
- Las ACP son:
 - Planificación del proyecto de software
 - Seguimiento y supervisión del proyecto de software
 - Gestión de requisitos
 - Gestión de la configuración software (GCS)
 - Garantía de calidad del software (SQA)
 - Gestión de la subcontratación

CMM

■ Nivel 3: Definido

- Nivel 2
- Las actividades se documentan, estandarizan e integran en un proceso a nivel organización
- Existe un proceso documentado
- Las ACP son:
 - Definición y enfoque del proceso de la organización
 - Programa de formación
 - Revisiones periódicas
 - Coordinación entre grupos
 - Ingeniería de productos software
 - Gestión de integración del software

CMM

■ Nivel 4: Gestionado

- Nivel 3
- Se recopilan medidas del proceso del software y de la calidad del producto
- Estas medidas sirven para gestionar el proceso
- Las ACP son:
 - Gestión de la calidad del software
 - Gestión cuantitativa del software

CMM

- Nivel 5: Optimización
 - Nivel 4
 - En base a la experiencia y métricas se optimiza el proceso
 - Las ACP son:
 - Gestión de cambios del proceso
 - Gestión de cambios de tecnología
 - Prevención de defectos

CMM

- Un nivel razonable es el definido (nivel 3)
- Un nivel deseable es optimización (nivel 5)
- Con independencia del CMM, ACPs mínimas:
 - Planificación del proyecto
 - Seguimiento y supervisión del proyecto
 - Gestión de requisitos
 - GCS
 - SQA
 - Definición del proceso
 - Revisiones periódicas
 - Coordinación entre grupos

Datos Agosto 2000

- Inicial 34,9%
 - Repetible 38,2%
 - Definido 18,5%
 - Gestionado 5,5%
 - Optimizado 2,9%
-
- Resultados de 901 empresas desde 1996.

Referencias

- Modelos de proceso
 - Pressman 17-46, Sommerville 42-67
- Proceso Unificado
 - Jacobson, Krutchen
- SW CMM
 - Pressman Cap. 2, Sommerville Cap. 25
 - <http://www.sei.cmu.edu/cmm/obtain.cmm.html>
- Material
 - Pablo Gervás
 - Juan Pavón y Jorge Gómez
 - Natalia Juristo