2. El proceso

Índice

- Referencias
- IS: una tecnología estratificada
  - Capas en IS.
  - Una visión general de la IS
- El proceso de software
- Modelos de proceso
- Modelos de proceso del software clásicos
- Modelos de proceso del software evolutivos

Índice

- Modelos de proceso del software ágiles
- Madurez del proceso
- Conclusiones

Referencias

- Pressman, R.S. *Ingeniería del Software. Un Enfoque Práctico. Sexta Edición*. McGraw-Hill, 2005
- Sommerville, I. *Ingeniería del Software*. 7ª edición. Addison-Wesley, 2005
- Beck K. *Extreme Programming Explained. Embrace change*. Addison-Wesley, 1999

Referencias

- Jacobson I., Booch G., Rumbaugh J., *El proceso unificado de desarrollo de software*. Addison-Wesley 2001.
- López-Cortijo R., Amescua Antonio de, *Ingeniería del Software. Aspectos de Gestión. Tomo 1 conceptos básicos, teoría, ejercicios y herramientas*. Instituto Ibérico de la Industria del Software, 1998.

Referencias

- Schmidt, M. *Implementing the IEEE Software Engineering Standards*. Sams 2000

IS: tecnolología estratificada Introducción

- Definiciones más relevantes de IS:
  - IS trata del establecimiento de los *principios* y *métodos* de la *ingeniería* a fin de obtener software de modo *rentable* que sea *fiable* y trabaje en *máquinas reales*, Bauer 1972.

IS: Tecnología estratificada Introducción

– IEEE (1993) IS es:

1. La aplicación de un *enfoque sistemático*, *disciplinado* y *cuantificable* al *desarrollo*, *operación* (funcionamiento) y *mantenimiento* del *software*, es decir, la aplicación de ingeniería al software.
2. Es *estudio* de enfoques como en 1.

IS: tecnología estratificada Introducción

- Hay *mucho que hacer* en IS:
  - Principios y metodologías de ingeniería.
  - Software económico.
  - Software fiable.
  - Software eficiente.
  - Sistematizar, disciplinar y cuantificar.
  - Mantener el software.

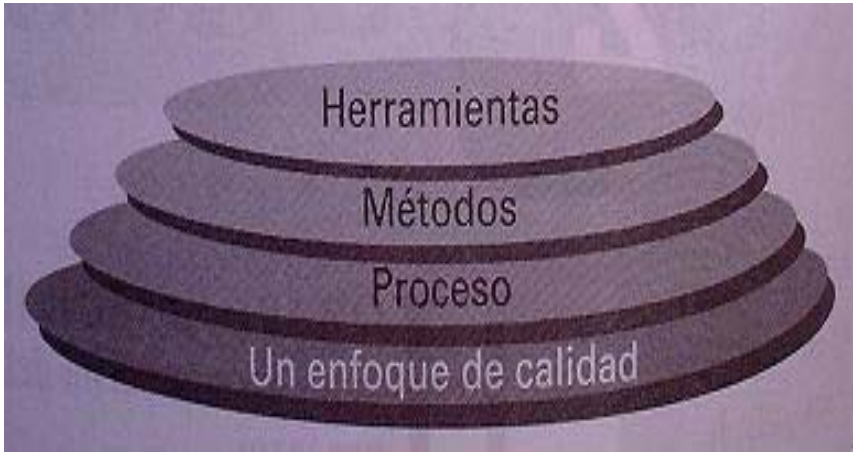
IS: tecnología estratificada Introducción

- ¿Cómo vamos a ser capaces de *hacer tanto*?
- Utilizando un *modelo de proceso* que imponga racionalidad en el *proceso* de desarrollo de software con el fin de satisfacer todas las demandas de la IS

IS: tecnología estratificada
Capas

- La IS es una tecnología *multicapa*

1. Capa de enfoque de calidad
2. Capa de proceso
3. Capa de métodos
4. Capa de herramientas



IS: tecnología estratificada
Capas

- 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.

IS: tecnología estratificada
Capas

- Capa de métodos
  - Un método incluye:
    - Análisis de requisitos.
    - Diseño.
    - Construcción de programas.
    - Prueba.
    - Mantenimiento.
  - Suelen estar bastante ligados al proceso.

IS: tecnología estratificada
Capas

- Capa de herramientas
  - Soporte automático o semiautomático para el proceso y los métodos.
  - Herramientas CASE: *Computer Aided Software Engineering*.
  - Fundamental para la correcta aplicación de IS.

IS: tecnología estratificada
Visión general de la IS

- *Ingeniería*: análisis, diseño, construcción y verificación de entidades técnicas (o sociales)
- La *entidad* caracteriza a la ingeniería:
  - Caminos, canales y puertos.
  - Aeronaves.
  - Buques.

IS: tecnología estratificada
Visión general de la IS

- Con independencia de la entidad debemos identificar y solucionar:
  - Problema a resolver.
  - Características de la entidad.
  - Forma de construir la entidad.
  - Enfoque para resolver los errores cometidos durante el diseño y construcción de la entidad.
  - Mantenimiento de la entidad frente a su evolución.

IS: tecnología estratificada
Visión general de la IS

- En IS, entidad = software.
- Soporte para el desarrollo de la entidad/soporte para la IS: modelo de proceso.
- Con independencia del modelo de proceso hay tres *fases* genéricas:
  - Fase de definición.
  - Fase de desarrollo.
  - Fase de mantenimiento

IS: tecnología estratificada
Visión general de la IS

- Fase de definición
  - Centrada en el *qué*.
  - Se identifican los *requisitos* del sistema y software:
    - Información a procesar.
    - Función y rendimiento deseados.
    - Comportamiento del sistema.
    - Interfaces establecidas.
    - Restricciones de diseño.

IS: tecnología estratificada
Visión general de la IS

  - Tareas principales:
    - Planificación del proyecto software.
    - Ingeniería de sistemas o de información.
    - Análisis de requisitos.
- Fase de desarrollo
  - Centrada en el *cómo*.

IS: tecnología estratificada
Visión general de la IS

- Se definen:
  - Cómo han de diseñarse las bases de datos.
  - Cómo han de implementarse las funciones.
  - Cómo han de caracterizarse las interfaces.
  - Cómo debe traducirse el diseño a un lenguaje de programación.
  - Cómo ha de realizarse la prueba.

IS: tecnología estratificada
Visión general de la IS

  - Tareas principales:
    - Diseño del software.
    - Generación del código.
    - Pruebas del software.
- Fase de mantenimiento
  - Centrada en el cambio asociado a:
    - Corrección de errores.
    - Adaptaciones requeridas por la evolución del entorno software.
    - Cambios en los requisitos del cliente.

IS: tecnología estratificada
Visión general de la IS

- En esta fase 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 en el entorno externo (CPU, SO, etc.).
  - *Mejora*. Ampliar los requisitos funcionales originales, a petición del cliente.
  - *Prevención*. Cambio para facilitar el cambio.

IS: tecnología estratificada
Visión general de la IS

- Estas fases se complementan con las *actividades protectoras*.
  - No crean software.
  - Mejoran su calidad.
  - Facilitan su desarrollo.

El proceso de software

- *Proceso*: conjunto de actividades y resultados asociados que sirven para construir un producto software.
- El proceso define un *marco de trabajo* compuesto por un conjunto de *actividades*:
  - *Actividades del marco de trabajo*.
  - ***Actividades estructurales, A.E.***

El proceso de software

- Como los proyectos no son siempre iguales, durante la planificación temporal el proceso se adapta al proyecto definiendo un *conjunto de tareas* concreto para cada actividad estructural

El proceso de software

- A un conjunto de tareas relacionadas que produce un producto del trabajo Pressman lo denomina *acción de ingeniería del software*
- Así, las acciones descomponen a las actividades estructurales y permiten adaptar un proceso a un proyecto concreto

El proceso de software

- Por ejemplo, para un proyecto concreto la actividad estructural *Ingeniería* puede descomponerse en las acciones *Análisis* y *Diseño*
- Para otro proyecto concreto, la actividad estructural *Ingeniería* puede descomponerse en las acciones *Reingeniería*, *Análisis* y *Diseño*

El proceso de software

- A su vez, la acción *Análisis* puede descomponerse en las tareas: *Realizar diagramas casos de uso*, *Realizar diagramas actividades*, *Realizar diagramas clases análisis*, etc.

El proceso de software

- Simultáneamente existen una serie de *actividades de protección*.

  - No crean software.

  - Mejoran su calidad.

  - Facilitan su desarrollo.

  - Es decir, evitan la *corrupción* del proceso.

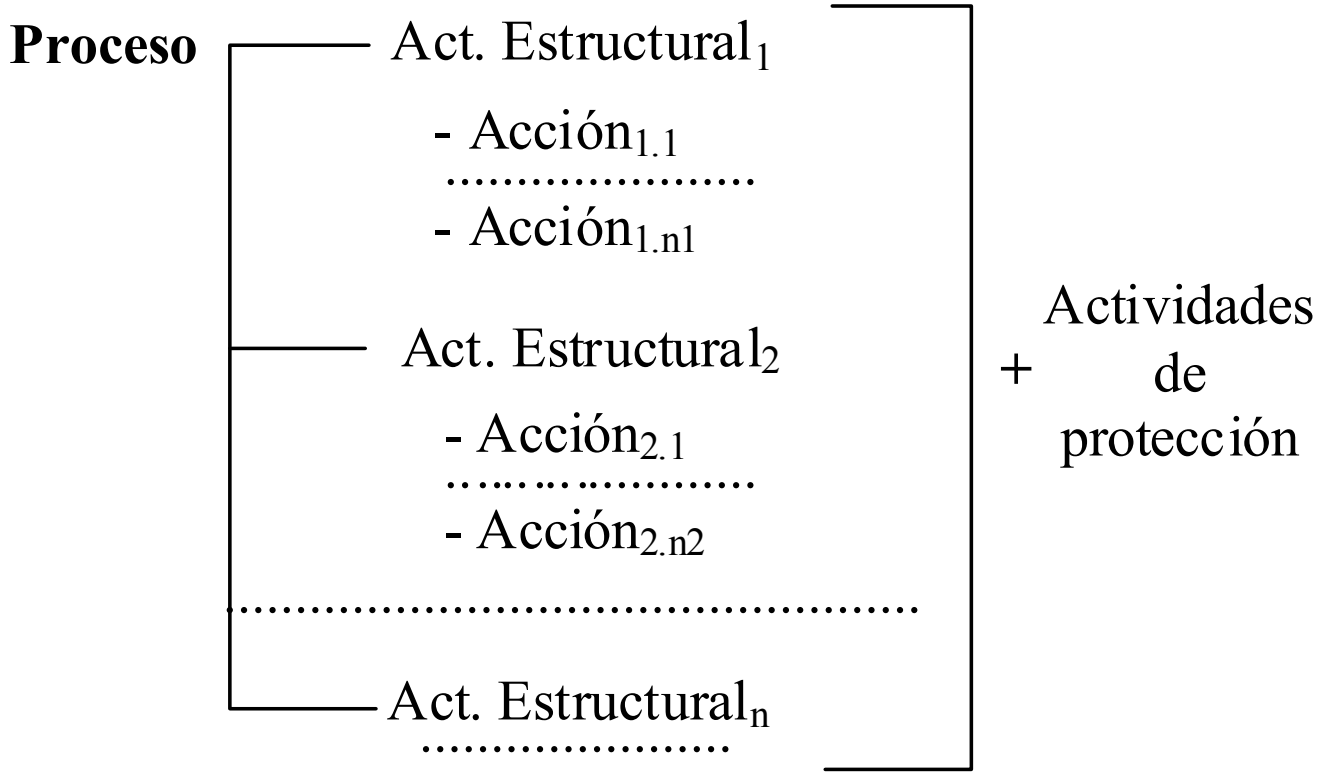
  - Están presentes en las AE, por eso no pueden constituir una AE por si mismas.

El proceso de software

- Estas actividades protectoras son:
  - Seguimiento y control del proyecto de software.
  - Revisiones técnicas formales.
  - Garantía de calidad del software.
  - Gestión de configuración del software.
  - Preparación y producción de documentos.
  - Gestión de reutilización.
  - Mediciones.
  - Gestión de riesgos.

El proceso de software

- Podemos representarlo:



El proceso de software

- Además, aunque hay distintos procesos, la mayor parte se ajusta a un *marco de trabajo genérico*
- Dicho marco está formado por una serie de *actividades estructurales comunes al proceso*

El proceso de software

- Según Pressman, estas actividades estructurales pueden ser:
  - Comunicación con el cliente.
  - Planificación.
  - Análisis de riesgos.
  - Ingeniería.
  - Construcción y adaptación.
  - Evaluación por el cliente.

El proceso de software

- Según Sommerville, estas actividades estructurales pueden ser:
  - Especificación del software.
  - Diseño e implementación del software.
  - Validación del software.
  - Evolución del software.

Modelos de proceso
Introducción

- 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
- Un modelo de proceso del software es una representación abstracta de un proceso del software

Modelos de proceso
Introducción

- Así, los modelos de proceso tienen en cuenta, en mayor o menor medida, las actividades estructurales comunes a todos los procesos de construcción de software
- Por tanto, un modelo de proceso define sus actividades estructurales así como su flujo de realización

Modelos de proceso
Introducción

- Como ya hemos comentado, los modelos de proceso se ajustan a los proyectos concretos variando el conjunto de tareas en el que se dividen las actividades estructurales

Modelos de proceso
Introducción

- Además, 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.

Modelos de proceso
Introducción

- Características de todo modelo de proceso:

  - *Entendibilidad*. Grado de definición y facilidad para entender el proceso.

  - *Visibilidad*. Grado en que las actividades del proceso proporcionan resultados.

  - *Soportabilidad*. Grado en que las actividades del proceso están soportadas por herramientas CASE.

  - *Aceptabilidad*. Grado en que los desarrolladores aceptan y usan el proceso.

Modelos de proceso
Introducción

- *Fiabilidad*. Grado en que el proceso permite evitar o detectar errores antes de que sean defectos.

- *Robustez*. Grado en que el proceso puede continuar a pesar de los problemas.

- *Mantenibilidad*. Grado en el que el proceso puede evolucionar para adaptarse.

- *Rapidez*. Grado de velocidad en que el proceso puede proporcionar un sistema a partir de una especificación

Modelos de proceso clásicos Introducción

- Los modelos de proceso clásicos surgieron en una primera etapa
- Muchos surgen como evolución frente a modelos de proceso anteriores
- Por imitación o negación, son la base de los modelos actuales

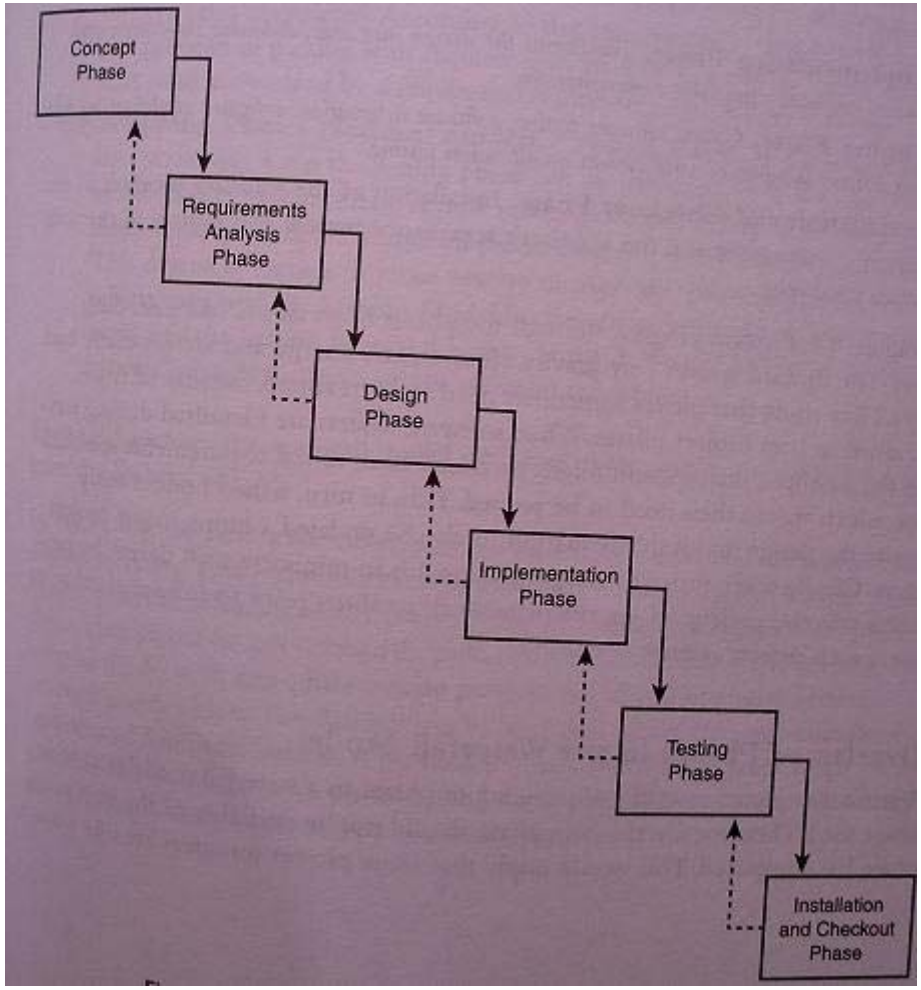
Modelos de proceso clásicos
Introducción

- Entre otros, caben destacar:
  - Modelo en cascada
  - Modelo de construcción de prototipos
  - Modelo de desarrollo rápido de aplicaciones

Modelos de proceso clásicos
Modelo en cascada

- También conocido como *waterfall*
- Es el modelo de proceso *clásico*.
- En general no es muy realista.
- Es sencillo y fácil de entender.
- El proyecto pasa a través de una serie de *fases* (AEs).
- Al final de cada fase se revisan las tareas de trabajo y productos.

Modelos de proceso clásicos
Modelo en cascada



Modelos de proceso clásicos
Modelo en cascada

- Fases:

  - *Conceptualización*. Se determina la *arquitectura* de la solución (i.e. división del sistema en subsistemas + comunicación).

  - *Análisis de requisitos*. Básicamente se definen los requisitos funcionales y de rendimiento.

  - *Diseño*. Representación de la aplicación que sirve de guía a la implementación.

Modelos de proceso clásicos
Modelo en cascada

- *Implementación*. Transforma el diseño en código.

- *Prueba*. Prueba e integración del software y de los sistemas.

- *Instalación y comprobación*. Se instala al cliente, el cual comprueba la corrección de la aplicación.

- Se supone que se solo se *baja* en la cascada...

  ... pero también se puede subir...

  ... por lo general malas noticias.

Modelos de proceso clásicos
Modelo en cascada

- Estrictamente no hay *solapamientos* entre las fases
- Se pueden producir *solapamientos de una fase*
- Se pueden producir *solapamientos de fases* (Modelo Sashimi)

Modelos de proceso clásicos
Modelo en cascada

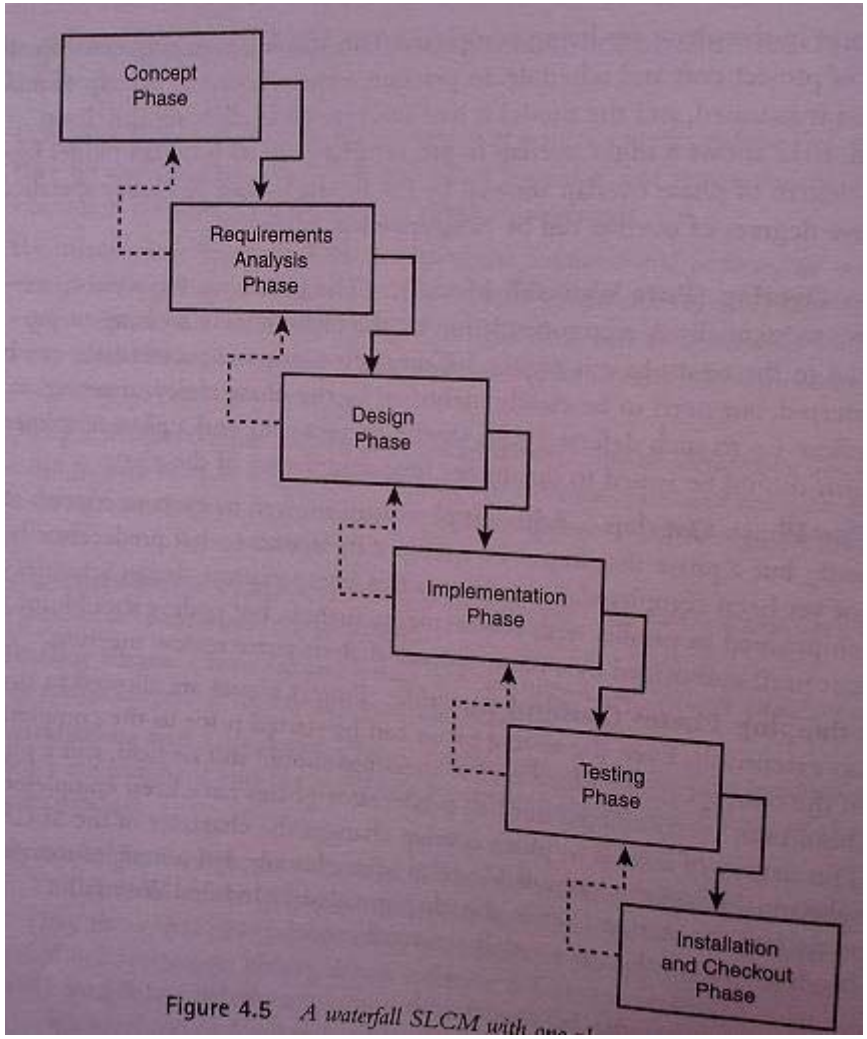
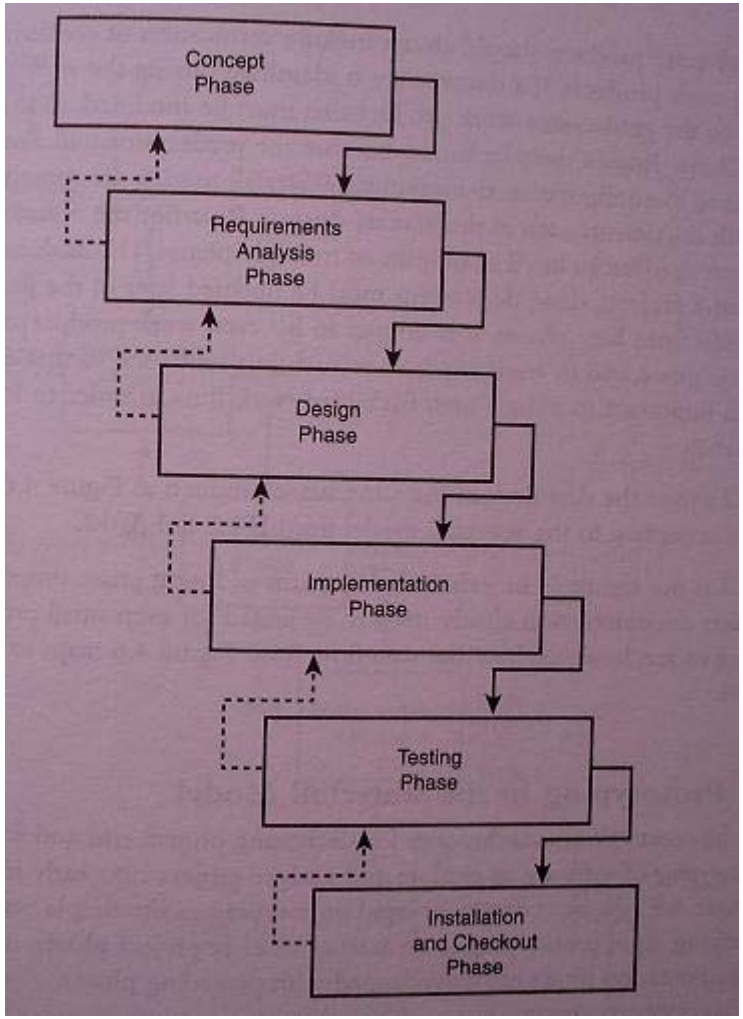


Figure 4.5 *A waterfall SLCM with*

Solapamiento de una fase

Modelos de proceso clásicos
Modelo en cascada



Modelo Sashimi

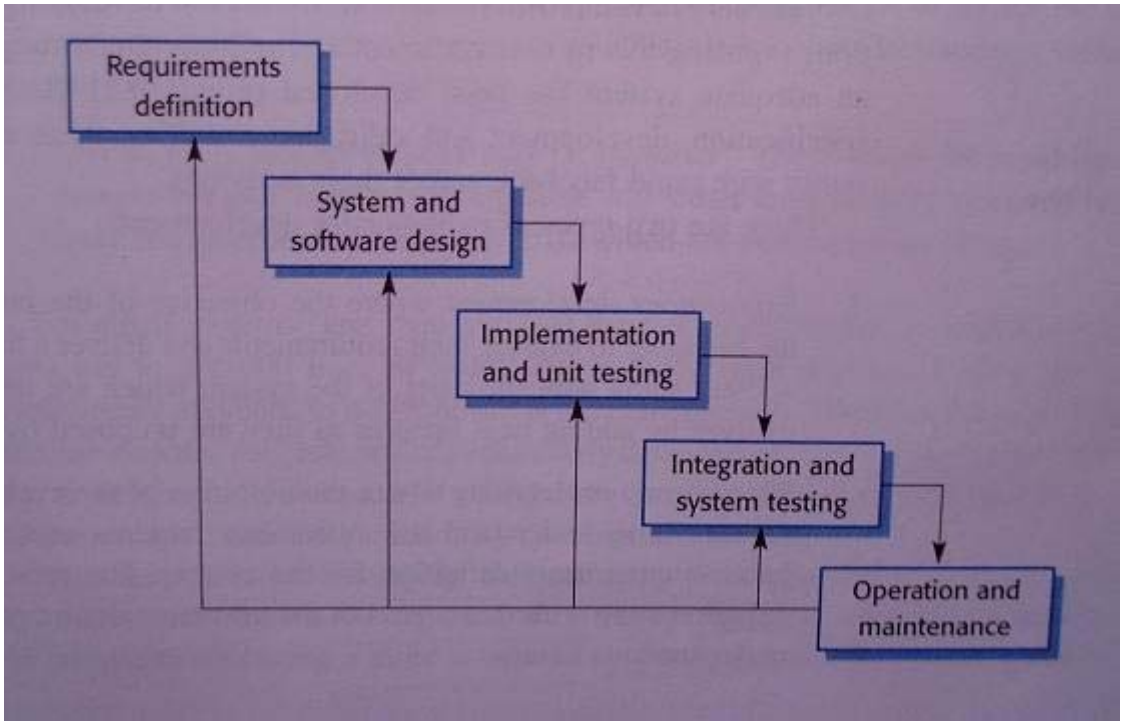
Modelos de proceso clásicos
Modelo en cascada

- Ventajas:
  - Muy probado.
  - Sencillo.
- Inconvenientes:
  - Poco realista.
  - Necesita especificación de requisitos estable.
  - Baja visibilidad.
  - Un error grave detectado en las últimas fases puede ser *letal*.
  - Si no hay solapamientos puede haber bloqueos.

Modelos de proceso clásicos
Modelo en cascada

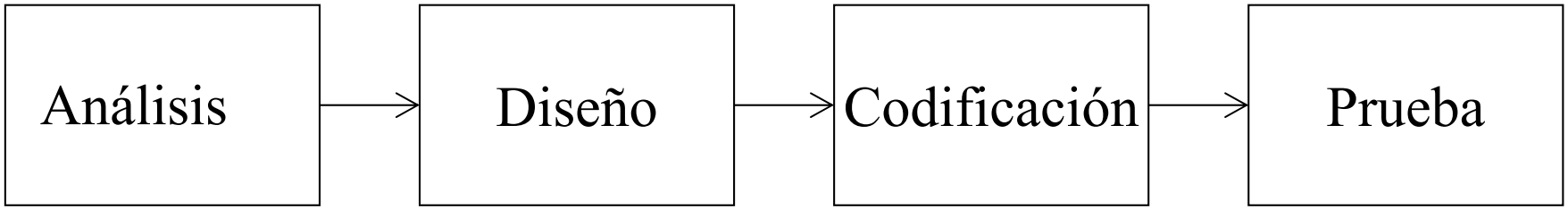
- Existen diversas variantes:
  - Sommerville.
  - Pressman (lineal secuencial).
  - Modelo en V.

Modelos de proceso clásicos
Modelo en cascada



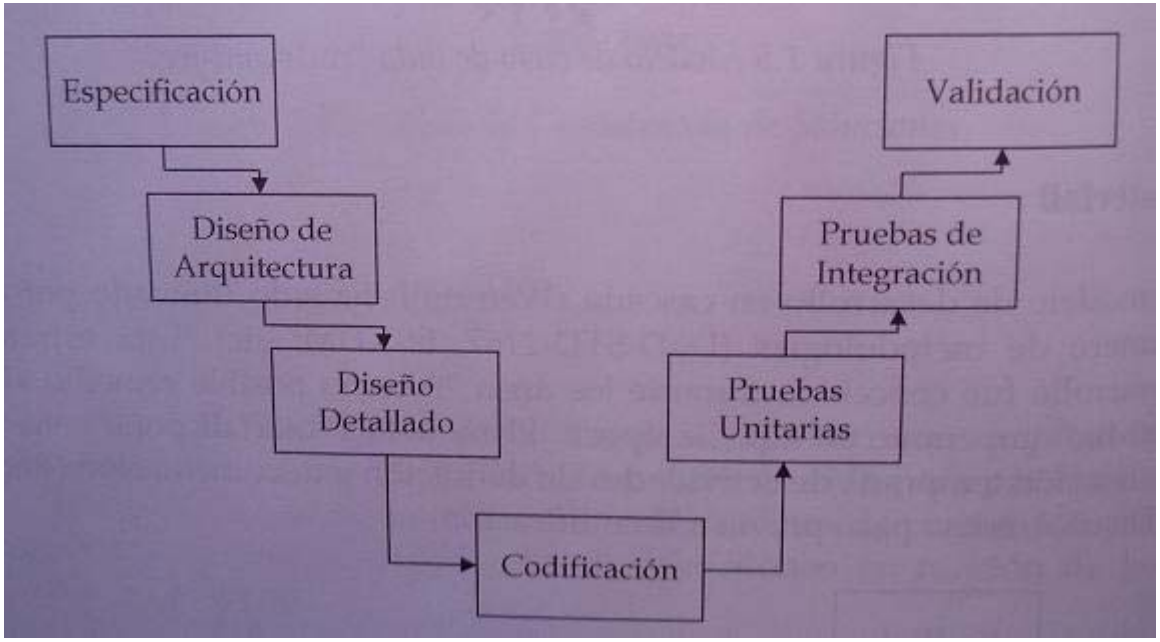
Modelo Waterfall (Sommervile)

Modelos de proceso clásicos
Modelo en cascada



Modelo lineal-secuencial (Pressman)

Modelos de proceso clásicos
Modelo en cascada



Modelo en V

Modelos de proceso evolutivos Introducción

- Características:
  - Gestionan bien la naturaleza *evolutiva* del software
  - Son *iterativos*: construyen software repitiendo una secuencia de pasos
  - Suelen construir el sistema por *incrementos*: productos operacionales de una parte del sistema
- Incremento vs. *prototipo*

Modelos de proceso evolutivos Introducción

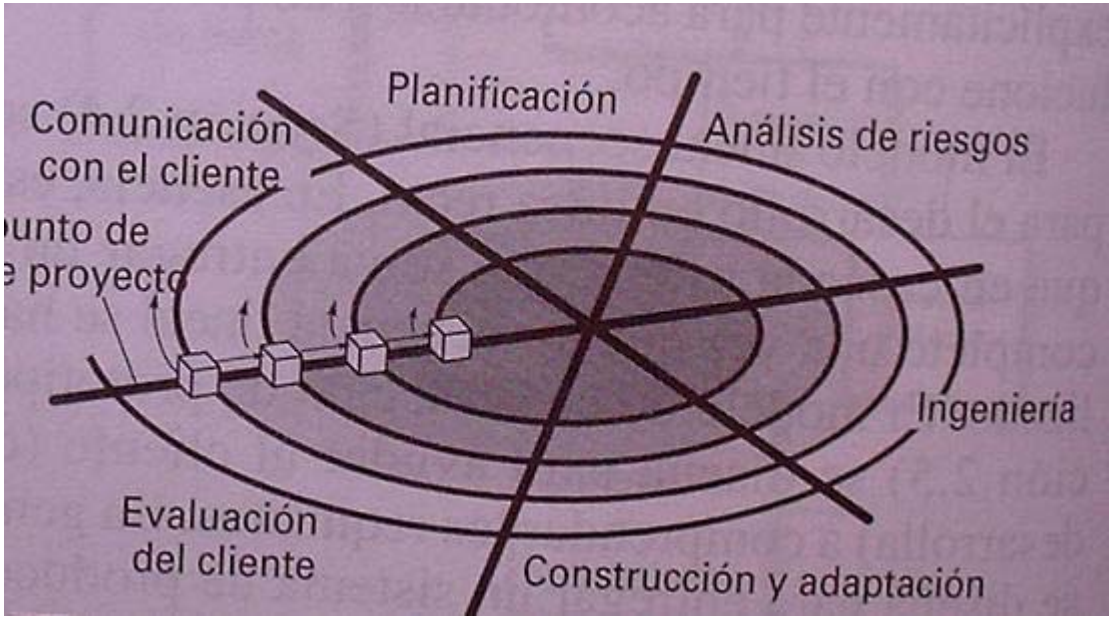
- Se adaptan bien:
  - Los cambios de requisitos del producto.
  - Fechas de entrega estrictas poco realistas.
  - Especificaciones parciales del producto.

Modelos de proceso evolutivos Introducción

- Entre otros cabe destacar:
  - Modelo de desarrollo incremental
  - Modelo en espiral de Boehm
  - Modelo en espiral de Boston
  - Proceso unificado de desarrollo de software

Modelos de proceso evolutivos
Espiral Boston

- Espiral de Boston:



Modelo espiral Boston

Modelos de proceso evolutivos
Espiral Boston

- Comunicación con el cliente
  - Tareas requeridas para establecer comunicación entre el desarrollador y el cliente.
- Planificación
  - Tareas requeridas para definir recursos, tiempo y otras informaciones relacionadas con el proyecto.

Modelos de proceso evolutivos
Espiral Boston

- Análisis de riesgos
  - Tareas requeridas para evaluar riesgos técnicos y de gestión.
- Ingeniería
  - Tareas requeridas para construir una o más representaciones de la aplicación.

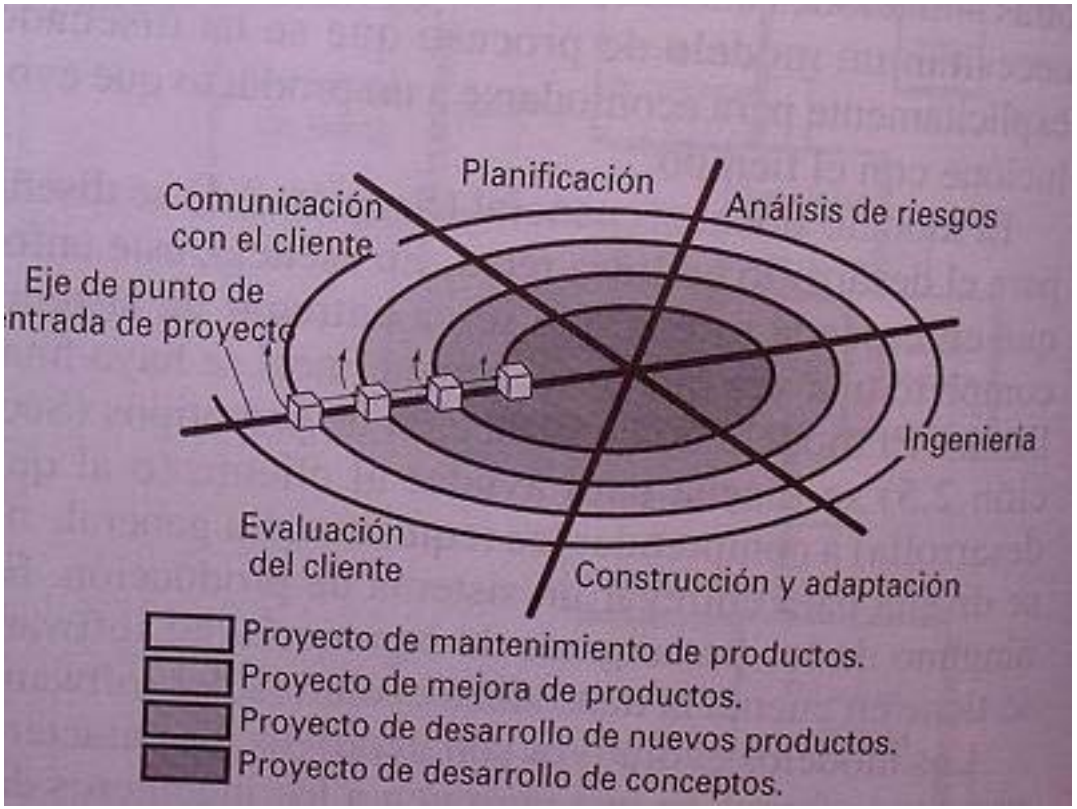
Modelos de proceso evolutivos
Espiral Boston

- Construcción y adaptación
  - Tareas requeridas para construir, probar, instalar y proporcionar soporte al usuario.
- Evaluación por el cliente
  - Tareas requeridas para obtener la reacción del cliente tras su evaluación.
- Actividades de protección continuas.
- Discusión: ¿por qué decimos *tareas*?

Modelos de proceso evolutivos
Espiral Boston

- Este modelo de proceso caracteriza la *vida* de un sistema.
- Eje de entrada en el proyecto.
- Tipos de proyecto:
  - Desarrollo del concepto.
  - Desarrollo de nuevos productos.
  - Mejora de productos.
  - Mantenimiento de productos.

Modelos de proceso evolutivos
Espiral Boston



Vida de un proyecto

Modelos de proceso evolutivos
Espiral Boston

- Ventajas
  - Enfoque realista.
  - Gestión explícita de riesgos.
  - Adecuado para desarrollos OO.
- Inconvenientes
  - Convencer cliente enfoque controlable.
  - Menos probado que el modelo en cascada o prototipos.

Modelos de proceso evolutivos
Proceso unificado de desarrollo

- 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*

Modelos de proceso evolutivos
Proceso unificado de desarrollo

- Modelo *basado en componentes*

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

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

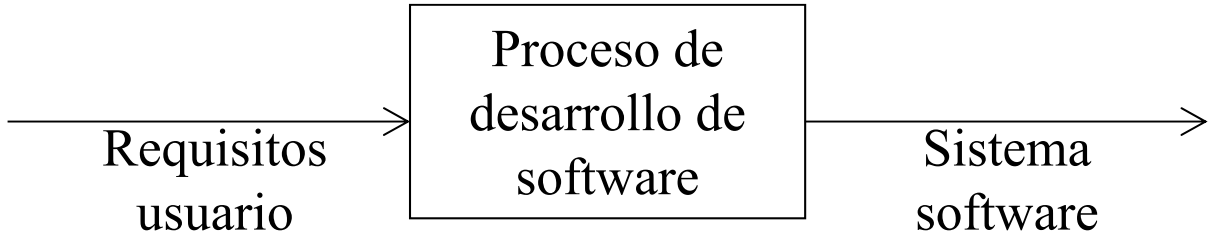
Modelos de proceso evolutivos
Proceso unificado de desarrollo

- *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?

Modelos de proceso evolutivos
Proceso unificado de desarrollo

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



Un proceso de desarrollo de software

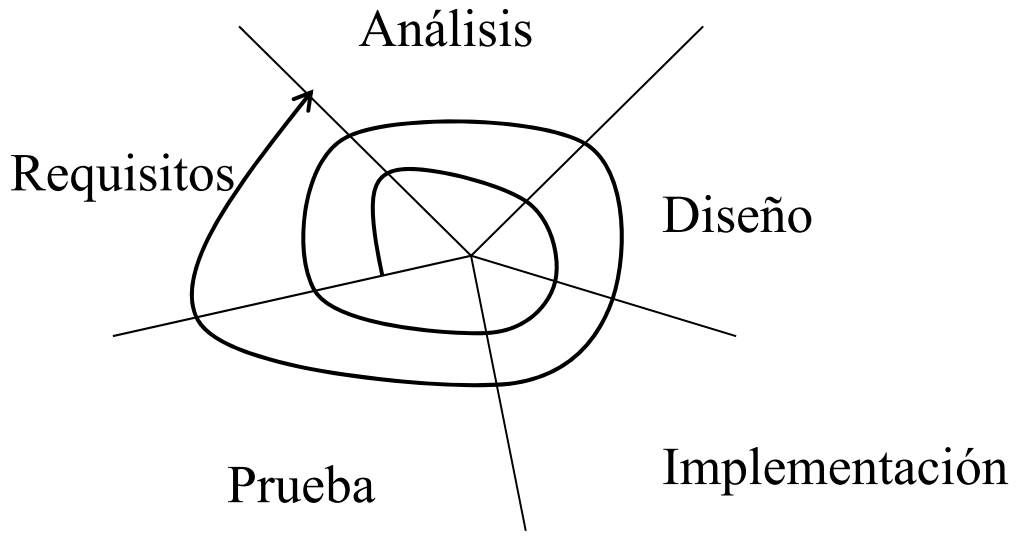
Modelos de proceso evolutivos
Proceso unificado de desarrollo

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

Modelos de proceso evolutivos
Proceso unificado de desarrollo

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

Modelos de proceso evolutivos
Proceso unificado de desarrollo



El proceso unificado de desarrollo

Modelos de proceso evolutivos
Proceso unificado de desarrollo

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

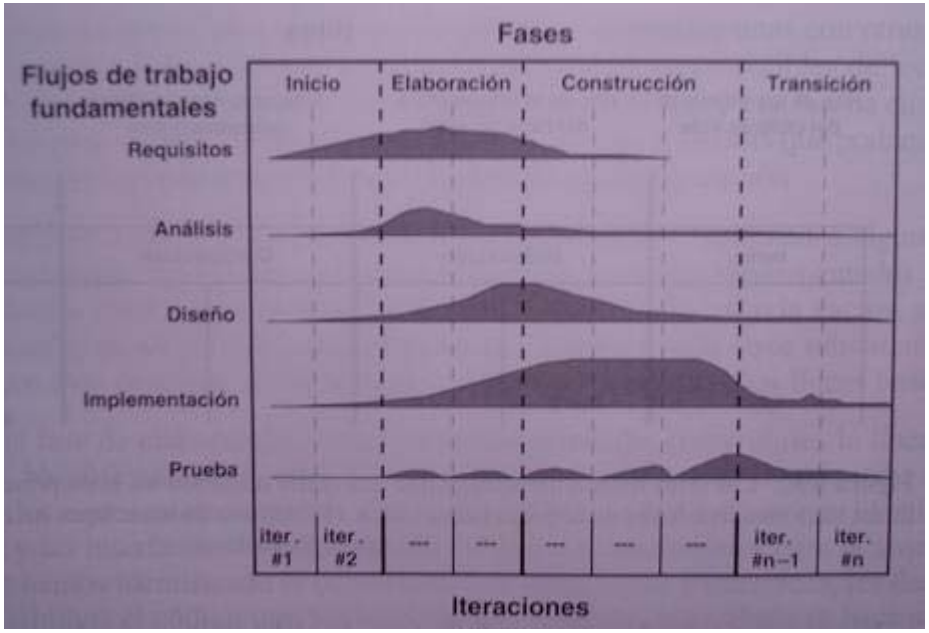
Modelos de proceso evolutivos
Proceso unificado de desarrollo

- 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.

Modelos de proceso evolutivos
Proceso unificado de desarrollo

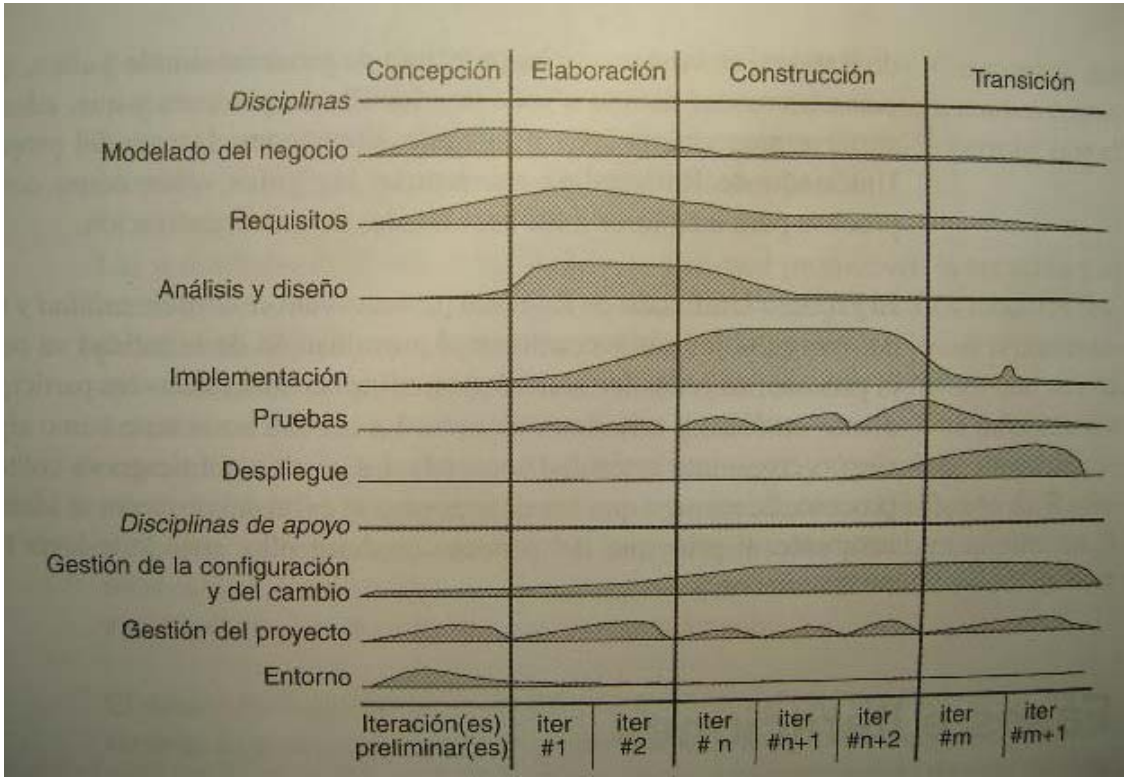
- Fase de construcción
  - Se crea el producto.
- Fase de transición
  - Periodo durante el cual el producto se convierte en versión beta.
- No todos los flujos de trabajo tienen el mismo peso dentro de cada fase

Modelos de proceso evolutivos
Proceso unificado de desarrollo



Relación entre flujos de trabajo y fases en RUP

Modelos de proceso evolutivos
Proceso unificado de desarrollo

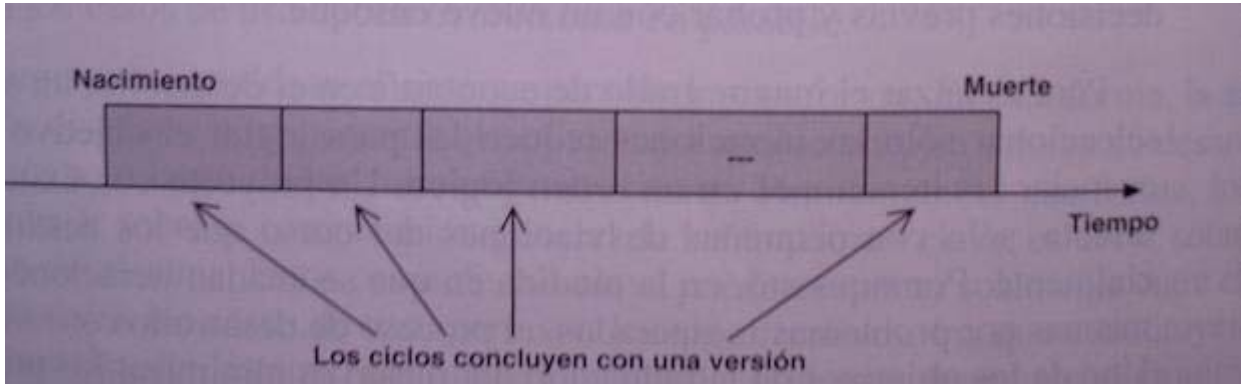


Versión más actual del proceso unificado de desarrollo

Modelos de proceso evolutivos
Proceso unificado de desarrollo

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

Modelos de proceso evolutivos
Proceso unificado de desarrollo



Ciclos en RUP

Modelos de proceso evolutivos
Proceso unificado de desarrollo

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

Modelos de proceso ágiles
Introducción

- El desarrollo ágil surge a principios del 2001 en respuesta a los modelos de proceso clásicos
- Valora:
  - A los *individuos* sobre los *procesos*
  - Al *software funcionando* sobre la *documentación*
  - La *colaboración del cliente* sobre el *negocio*
  - La *respuesta al cambio* sobre *seguir un plan*

Modelos de proceso ágiles
Introducción

- La *Alianza Ágil* define 12 principios:
  1. Satisfacer al cliente con entrega temprana y continua de software valioso
  2. Bienvenidos los requisitos cambiantes, incluso en fases tardías del desarrollo
  3. Entregar con frecuencia (de dos semana a dos meses) software funcionando, cuanto antes mejor

Modelos de proceso ágiles
Introducción

4. El cliente y los desarrolladores deben trabajar juntos a diario a lo largo del proyecto
5. Individuos motivados. Darles el ambiente y el soporte que necesitan, y confiar en ellos para obtener el trabajo realizado
6. El método más eficiente y efectivo de transmitir información hacia y dentro del equipo es la conversación cara a cara

Modelos de proceso ágiles
Introducción

7. El software en funcionamiento es la medida principal del progreso
8. Desarrollo sostenible. Los participantes deben ser capaces de mantener un paso constante de manera indefinida
9. Atención continua a la excelencia técnica y a un buen diseño
10. La simplicidad (arte de maximizar el trabajo no realizado) es esencial

Modelos de proceso ágiles
Introducción

11. Las mejores arquitecturas, los mejores requisitos y los mejores diseños emergen de equipos auto organizados
12. A intervalos regulares el equipo refleja la forma en que se puede volver más efectivo, entonces su comportamiento se ajusta y adecúa en concordancia

Modelos de proceso ágiles
Introducción

- Teóricamente, la agilidad se puede aplicar a cualquier proceso de software
- En cualquier caso, surgen modelos de proceso propios del desarrollo ágil

Modelos de proceso ágiles Introducción

- El *proceso ágil* parte de tres supuestos clave:
  1. Es difícil predecir qué requisitos software persistirán y cuáles cambiarán. También es difícil predecir las prioridades del cliente

Modelos de proceso ágiles
Introducción

2. El diseño y el desarrollo de software están intercalados. Por tanto, se deben realizar de manera conjunta, de forma que el diseño se prueba según se crea. Es difícil predecir cuanto diseño es necesario antes de construir el código que lo implemente

Modelos de proceso ágiles Introducción

3. El análisis, el diseño y la construcción no son predecibles desde el punto de vista de la planificación, lo que sería deseable

- Para responder a estos supuestos, el modelo de proceso ágil:
  - Es adaptable de forma incremental
  - Necesita retroalimentación del cliente
  - Se basa en la entrega continua de incrementos

Modelos de proceso ágiles
Introducción

- El proceso ágil requiere una serie de características sobre el equipo de desarrollo:
  - Competencia técnica.
  - Enfoque común: entregar al cliente un incremento dentro del plazo.
  - Colaboración entre todos los participantes.
  - Autonomía para la toma de decisiones.
  - Capacidad de resolución de problemas confusos.

Modelos de proceso ágiles
Introducción

- Confianza y respeto mutuo en el equipo.
- Organización propia

Modelos de proceso ágiles
Introducción

- Entre otros cabe destacar:
  - Extreme Programming (XP)
  - Scrum
  - Desarrollo adaptativo de software

Modelos de proceso ágiles
XP

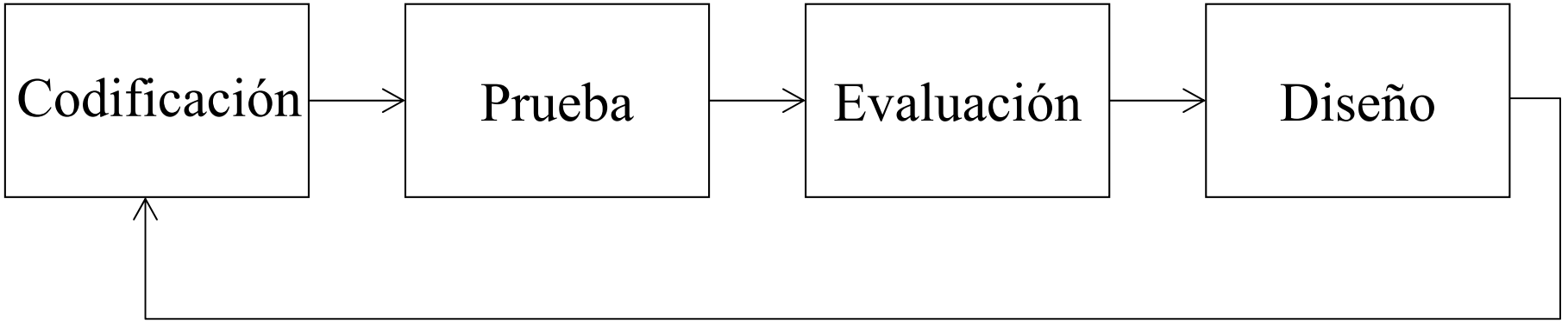
- Modelo de proceso de K. Beck
- XP es un “modo ligero, eficiente, de bajo riesgo, flexible, predecible, científico y divertido de producir software”.
- Características
  - Alta visibilidad debido a ciclos cortos.
  - Planificación incremental.
  - Se adapta a cambios de negocio.

Modelos de proceso ágiles
XP

- Basado en test automatizados escritos por desarrolladores y clientes.
- 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.

Modelos de proceso ágiles
XP

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



Actividades en XP

Modelos de proceso ágiles
XP

- Las cuatro actividades están soportadas por doce *prácticas*:
  - El juego de planificación.
  - Pequeñas entregas.
  - Metáfora.
  - Diseño simple.
  - Prueba.
  - *Refactoring*.

Modelos de proceso ágiles
XP

- Programación en pareja.
- Propiedad colectiva.
- Integración continua.
- Semana de cuarenta horas.
- Cliente en el lugar de desarrollo.
- Codificación estándar.

Modelos de proceso ágiles
XP

- Ventajas (extensibles al resto):
  - Bueno para especificaciones cambiantes.
  - Fundamentación práctica.
- Inconvenientes (extensibles al resto):
  - Poco probado.
  - Poco compatible con especificaciones/diseños *totales*.
  - Solo funciona con equipos *pequeños* (hasta diez personas).

Madurez del proceso

- *Madurez del proceso*: corrección en la aplicación de un proceso de software
- El *Software Engineering Institute (SEI)** ha identificado una serie de funciones que deberían estar presentes en el proceso
- El grado de madurez del proceso se determina con un cuestionario y un esquema de cinco grados

*http://www.sei.cmu.edu

Madurez del proceso

- El modelo de cinco grados determina la conformidad con un *modelo de capacidad de madurez*:*
  - Inicial
  - Repetible
  - Definido
  - Gestionado
  - Optimización

*CMM: *Capability Maturity Model*

Madurez del proceso

- Este modelo define una serie de *áreas clave de proceso* (*ACP*)
- Un *área clave de proceso* es, basicamente, una *actividad* de IS
- Los niveles son acumulativos

Madurez del proceso

- Nivel 1: *Inicial*
  - El proceso de software se caracteriza según el caso.
  - Se definen poco procesos.
  - El éxito depende del esfuerzo individual.

Madurez del proceso

- Nivel 2: *Repetible*
  - Se incluye seguimiento del coste, de la planificación y de 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.

Madurez del proceso

- Gestión de la configuración software (GCS).
- Garantía de calidad del software (SQA).
- Gestión de la subcontratación.

- Nivel 3: *Definido*
  - Nivel 2
  - Las actividades se documentan, estandarizan e integran en un proceso a nivel organización.
  - Existe un proceso documentado.

Madurez del proceso

- 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.

Madurez del proceso

- 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.

Madurez del proceso

- 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.

Madurez del proceso

- 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 periodicas.
  - Coordinación entre grupos.

Conclusiones

- En IS *hay mucho que hacer*...

  ... el modelo de proceso es nuestro *soporte*
- La IS es una disciplina de varias capas
- La IS concibe la informática como una *ingeniería*
- En IS hay tres fases
- Estas tres fases están presentes en todos los modelos de proceso

Conclusiones

- El proceso va a encaminado a desarrollar software
- Presenta unas actividades comunes
- Los modelos de proceso son representaciones del proceso de desarrollo
- Así, las actividades comunes están presentes en mayor o menor medida en todos los modelos de proceso

Conclusiones

- El modelo de proceso determina y queda determinado por sus AEs
- Las AEs son fijas para cada modelo de proceso
- El modelo de proceso se ajusta al proyecto a través de las acciones y el conjunto de tareas de IS
- Disponemos de múltiples modelos de proceso

Conclusiones

- También hay modelos que no son de proceso
- Modelos *clásicos* vs *ágiles*
- Al final TODOS, de una forma u otra, resuelven las fases de IS
- Tensión entre el producto y el proceso
- No todos las organizaciones aplican de la misma forma el mismo proceso
- Se puede medir la *madurez* del proceso