En la preparación de este material se ha reutilizado parte de los cursos preparados por mis compañeros Pablo Gervás y Antonio Navarro, UCM

El Proceso del Software

Ingeniería del Software de Gestión 1
Facultad de Informática

Juan Pavón Mestras
Dep. Sistemas Informáticos y Programación
Universidad Complutense Madrid

http://www.fdi.ucm.es/profesor/jpavon

Objetivos

- Entender qué es el proceso de desarrollo de software
- Cuáles son los componentes que debe considerar un proceso de desarrollo de software
- Modelos de proceso de desarrollo de software
- Calidad del proceso de desarrollo de software

Conceptos importantes

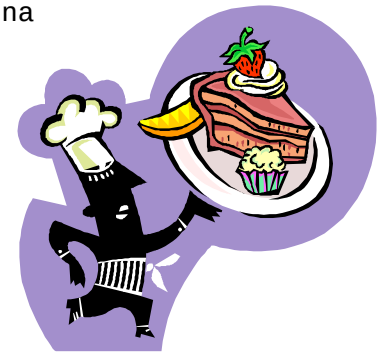
- *Personas*: los que trabajan
- *Producto*: lo que se obtiene
- *Proyecto*: la pauta a seguir para desarrollar un producto
- *Proceso*: la pauta a seguir para desarrollar un proyecto

Un traje

- *Personas*: El sastre
- *Producto*: El traje
- *Proyecto*: La secuencia de acciones para hacer un traje concreto
- *Proceso*: Lo que aprende un sastre cuando aprende a hacer trajes

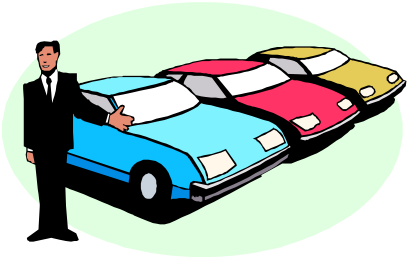
Una cena

- *Personas*: Empleados de una empresa de catering
- *Producto*: La cena que se sirve
- *Proyecto*: La secuencia de acciones de servir una cena concreta
- *Proceso*: Las instrucciones de la empresa sobre cómo se sirve 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



Software

- *Personas*: Vosotros
- *Producto*: La aplicación que elijáis
- *Proyecto*: IS1
- *Proceso*: ???

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: *Computer Aided Software Engineering*

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

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

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
- Cada una de estas fases se descompone en un conjunto de tareas

Visión general de la IS

- Fase de definición/especificación
  - 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
  - Tareas principales:
    - Planificación del proyecto software
    - Ingeniería de sistemas o de información
    - Análisis de requisitos

Visión general de la IS

- Fase de desarrollo
  - Centrada en el *CÓMO*
  - Se definen:
    - Cómo han de diseñarse las estructuras 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 validarse el producto (pruebas, verificación)
  - Tareas principales:
    - Diseño del software
    - Generación del código
    - Pruebas del software

Visión general de la IS

- Fase de mantenimiento
  - Centrada en cambios que se pueda necesitar realizar sobre un producto
  - 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

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

- 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 del software

- Conjunto estructurado de actividades y resultados asociados requeridos para desarrollar un sistema de software
  - Especificación: establecer los requisitos y restricciones del sistema
  - 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 y asegurar su operacionalidad
  - Evolución y mantenimiento: cambiar/adaptar el software según las demandas; reparar fallos en el sistema cuando sean descubiertos

  - Las actividades varían dependiendo de la organización y del tipo de sistema a desarrollar
  - Debe estar explícitamente modelado si va a ser bien administrado

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

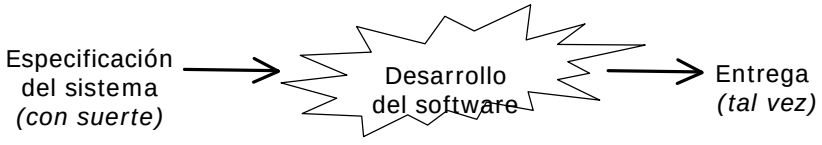
- Una organización podría variar su modelo de proceso para cada proyecto, según:
  - La naturaleza del proyecto
  - La naturaleza de la aplicación
  - Los métodos y herramientas a utilizar
  - Los controles y entregas requeridas

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 de proceso que no son...

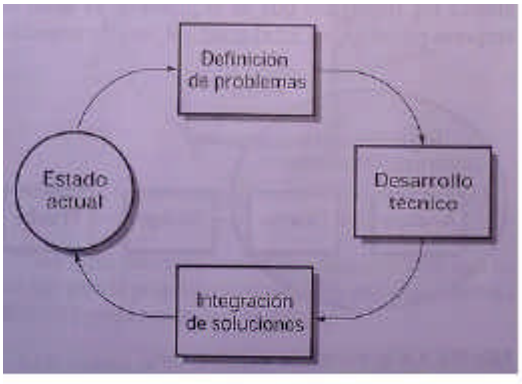
- Modelo de caja negra (*code and fix*)
  - Codificar y corregir no es un modelo de proceso
  - No se pierde el tiempo en la planificación, en la calidad, en los documentos que hay que realizar cuando se terminan etapas o en cualquier otra actividad que no sea la codificación. Por lo tanto este modelo no se necesita tener experiencia y una gran cantidad de conocimientos.
    - Al no seguir un modelo no tenemos ningún medio de ver si se cumplen las expectativas creadas, lo cual es un problema si encontramos un error casi al finalizar el proyecto ya que hay que empezar de nuevo. Por consiguiente tardamos más en ver los errores que en otro modelo que sigue un mínimo de planificación
  - No es lo mismo que *Extreme Programming (XP)*



Modelos de proceso que no son...

- El modelo del Caos
  - El desarrollo de software se caracteriza como un bucle de resolución de problemas con cuatro etapas:
    - Status Quo. Estado actual de procesos
    - Definición de problemas. Identifica el problema
    - Desarrollo técnico. Resuelve el problema
    - Integración de soluciones. Ofrece los resultados

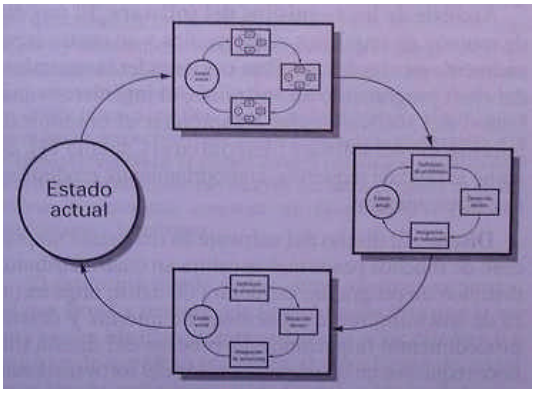
Bucle de resolución de problemas



Bucle de resolución de problemas

- Nótese que estas etapas son asimilables con las fases de IS presentadas
- Aplicación recursiva del bucle sobre sí mismo: modelo del caos
- El problema es que las fases de IS interactúan más que lo mostrado por esta aproximación

Modelo del caos



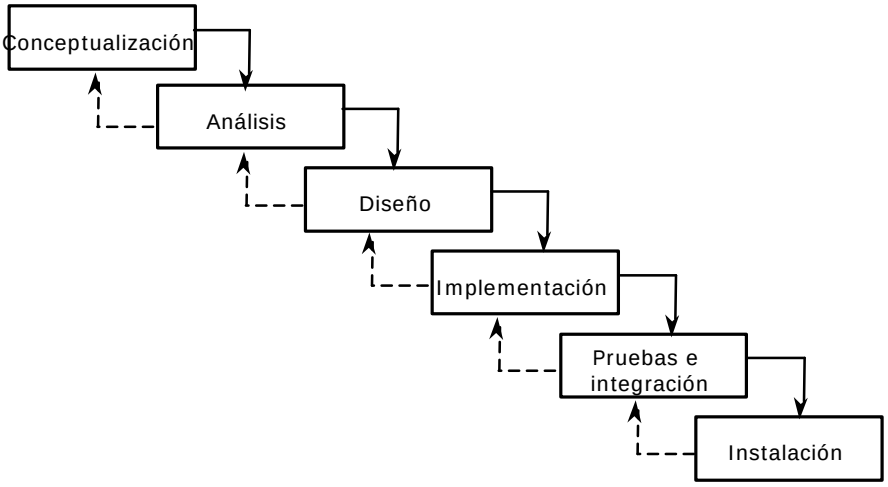
Modelos de Proceso del Software

- Modelo de Cascada
  - Separar en distintas fases de especificación y desarrollo que se realizan en secuencia lineal
- Desarrollo Evolutivo
  - La especificación y el desarrollo están intercalados
- Prototipado
  - Un modelo sirve de prototipo para la construcción del sistema final
- Transformación Formal
  - Un modelo matemático del sistema se transforma formalmente en la implementación
- Desarrollo basado en Reutilización
  - El sistema es ensamblado a partir de componentes existentes

Modelo en cascada (*waterfall*)

- Es el modelo de proceso clásico (desde los '70)
- Basado en la mentalidad de línea de ensamblaje
- 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 (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
  - Implementación: Transforma el diseño en código
  - Prueba: Validación 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 sólo se baja en la cascada...
  ... pero también se puede subir (aunque difícilmente)

Documentos del Modelo de Cascada

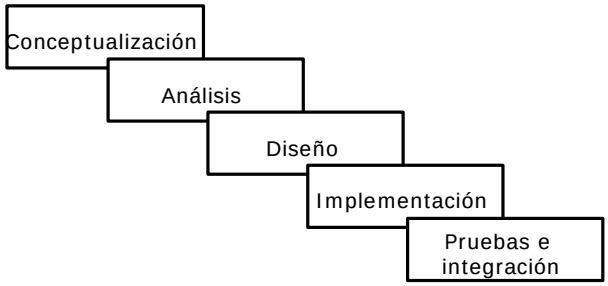
Actividad	Documentos Producidos
Análisis de Requisitos	Documento de Requisitos
Definición de Requisitos	Documento de Requisitos
Especificación del Sistema.	Especificación Funcional, Plan de Pruebas de Aceptación.
Diseño Arquitectural	Especificación de la Arquitectura, y Plan de Pruebas del Sistema
Diseño de Interfaces	Especificación de la Interfaces y Plan de pruebas de Integración
Diseño Detallado	Especificación del diseño y Plan de prueba de Unidades.
Codificación	Código de Programa
Prueba de Unidades	Informe de prueba de unidades
Prueba de Módulos	Informe de prueba de módulos
Prueba de Integración	Informe de prueba de integración y Manual de usuario final
Prueba del Sistema	Informe de prueba del sistema
Prueba de Aceptación	Sistema final y documentación

Modelo en cascada (*waterfall*)

- Se tiene un seguimiento de todas las fases del proyecto y del cumplimiento de todos los objetivos marcados en cada etapa tanto de costes como fechas de entrega
- Al final de cada etapa se debe comprobar si el proyecto cumple todas las necesidades del usuario
  - Esto es un problema ya que si el cliente se da cuenta de que falta alguna función una vez pasada esta etapa, el trabajo que hay que realizar se retrasa en fechas de entrega y el coste es mayor
  - Por este motivo se puede modificar el modelo en cascada pudiendo pasar de una etapa a la anterior
    - Difícil ya que hay que rehacer la etapa anterior (modelo de ciclo de vida del salmón)
- El rendimiento puede mejorar notablemente variando el modelo de la cascada pura (cuando no hay solapamientos entre las fases):
  - Solapamientos de fases (Modelo Sashimi)

Modelo Sashimi (Cascada con fases solapadas)

- Viene del modelo de desarrollo de hardware de Fuji-Xerox
- Se solapan las fases (se pueden ejecutar varias a la vez)
  - Por ejemplo, se puede desarrollar parte de A&D antes de acabar con la conceptualización
  - Se sugiere que el mismo personal trabaje en varias fases
- Pero genera nuevos problemas
  - Debido al solapamiento los hitos resultan más ambiguos y es más difícil trazar el progreso del proceso correctamente

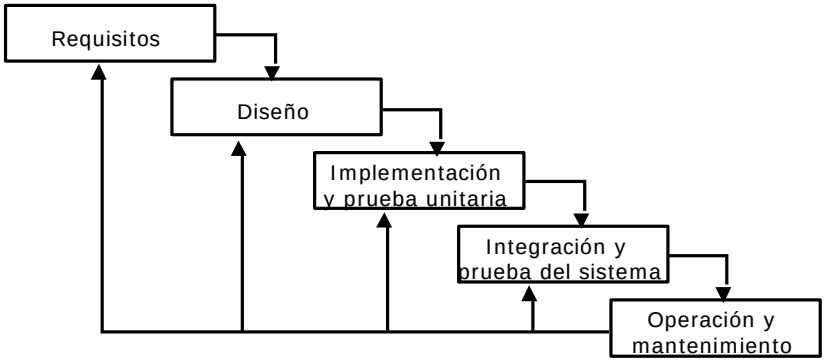


Modelo en cascada (*waterfall*)

- Otras variantes:
  - Sommerville
  - Pressman (lineal secuencial)
  - Modelo en V
  - Cascada con subproyectos

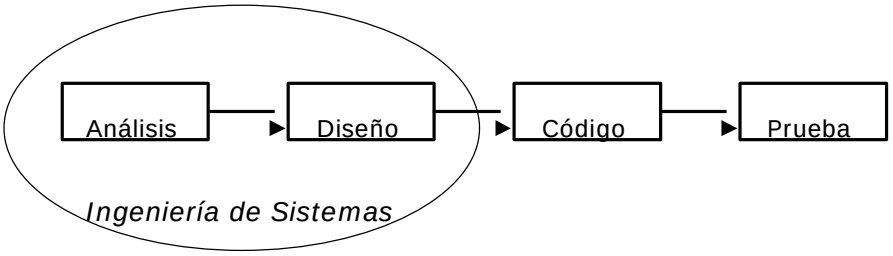
Modelo en cascada (Sommerville)

- El sistema se pone en funcionamiento durante la fase final del proyecto
  - Entonces se descubren errores en diseño y programación, y omisiones en los requisitos originales
  - Para adaptar los cambios requeridos hay que repetir algunas o todas las etapas del proceso



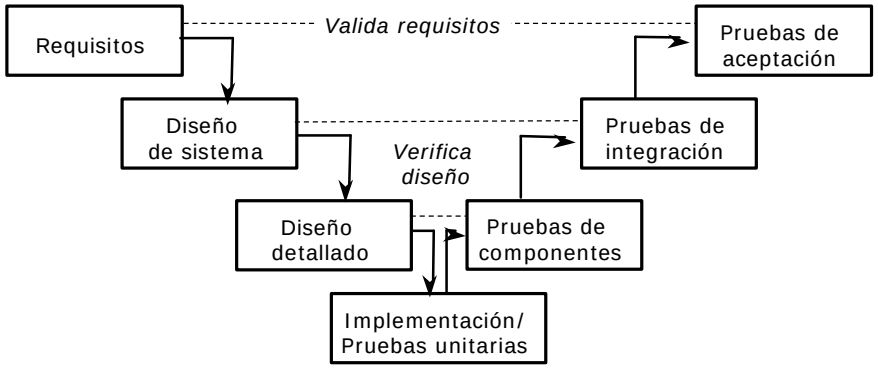
Modelo lineal secuencial (Pressman)

- Ingeniería de sistemas
  - Al ser el software parte de un sistema más grande el trabajo comienza estableciendo requisitos de todos los elementos del sistema y asignando al software una parte de estos requisitos



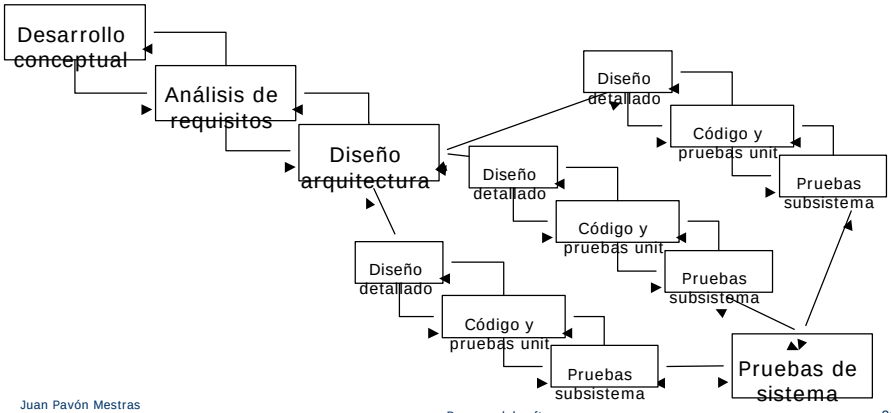
Modelo de cascada en V

- Muestra la relación de las actividades de prueba con el análisis y diseño



Modelo en cascada con subproyectos

- Después de requisitos y diseño arquitectural el proyecto se divide en subproyectos. Al final se integra y prueba el sistema completo
  - Desarrollo más rápido de tareas conocidas y mejor uso de recursos
  - Riesgo: Dependencias imprevistas entre subproyectos

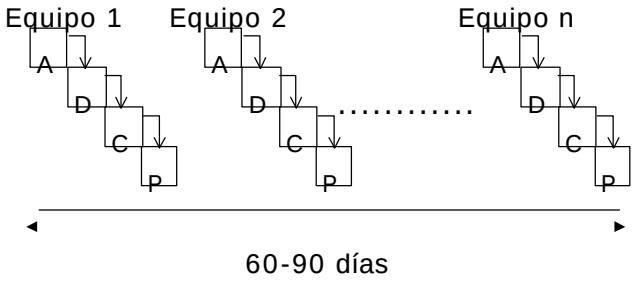


Modelo en cascada (*waterfall*)

- Posibles ventajas:
  - Sencillo
    - Sirve cuando el personal está poco cualificado
  - Aplicable cuando el problema es estable y cuando se trabaja con técnicas conocidas
    - Por ejemplo, será apropiado para la migración de una aplicación o para generar una nueva versión de mantenimiento bien definida
- Críticas:
  - No se ve un producto hasta muy tarde en el proceso
    - Un error grave 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

Desarrollo Rápido de Aplicaciones (DRA)

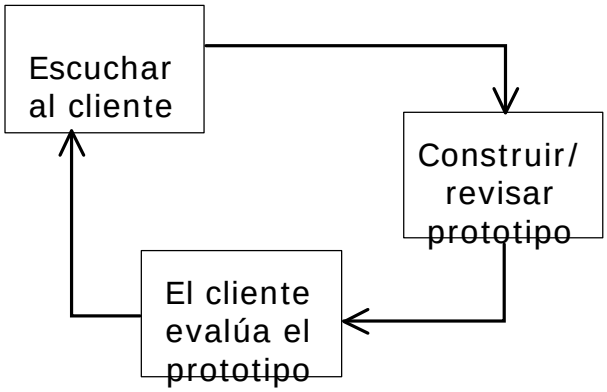
- Objetivo: Desarrollo de sistemas en poco tiempo
- Adaptación a “alta velocidad” del lineal secuencial
  - Equipos trabajando en paralelo
  - Aplicando tecnología de componentes



Desarrollo Rápido de Aplicaciones (DRA)

- 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

Modelo de construcción de prototipos



Modelo de construcción de prototipos

- 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 primer prototipo

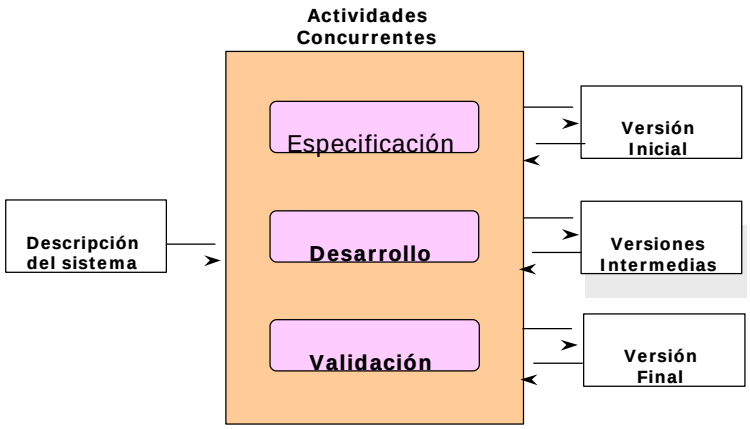
Modelo de construcción de prototipos

- 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 porque hay que desechar el primer 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
      - Por ejemplo, utilizar un sistema operativo o lenguaje de programación inadecuado pero que ya es conocido

Modelos evolutivos

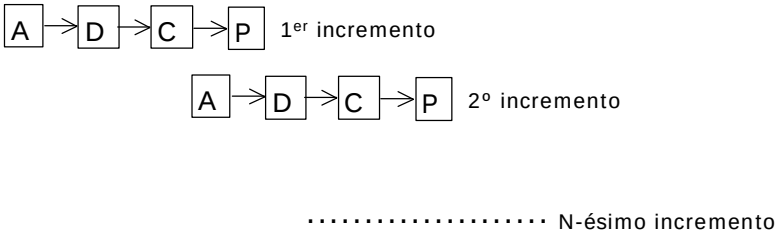
- Características:
  - Gestionan bien la naturaleza evolutiva del software
  - Son iterativos: construyen versiones de software cada vez más completas
- Se adaptan bien:
  - Los cambios de requisitos del producto
  - Fechas de entrega estrictas poco realistas
  - Especificaciones parciales del producto

Modelos evolutivos



Modelos evolutivos. Incremental

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



Modelos evolutivos. Incremental

- Cada secuencia lineal secuencial produce un incremento del software
  - Un incremento es un producto operacional de una parte del sistema
- El primer incremento suele ser un producto esencial o 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 incremento siguente
- Se itera
  - Hasta elaborar el producto completo

Modelos evolutivos. Incremental

- Discusión: aparte de las fases de desarrollo concretas, ¿qué diferencia esencial hay entre el modelo de construcción de prototipos y el incremental?

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

Modelos evolutivos. Espiral

- Original: Boehm, 1988
- IEEE Std. 1490-1998
- Se centra en tratar las áreas de mayor riesgo en un proyecto (requisitos, arquitectura, etc.)
- El proyecto se compone de varios mini-proyectos, que tratan una o varias áreas de riesgo
- Multiples 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

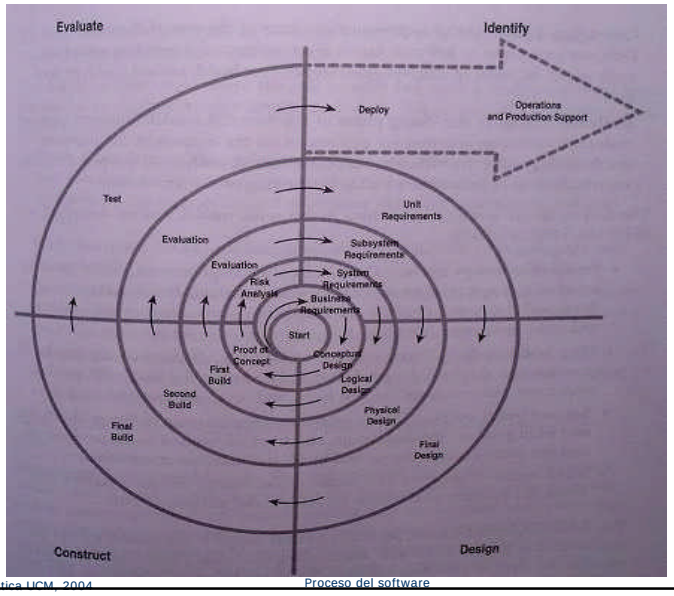
Modelos evolutivos. Espiral

- Las regiones de tareas son asimilables a las fases del modelo en cascada
  - Identificar requisitos
    - En función del ciclo:
      - Requisitos de negocio
      - Requisitos del sistema
      - Requisitos de subsistemas
      - Requisitos de unidad
  - Diseñar
    - En función del ciclo:
      - Diseño conceptual
      - Diseño lógico
      - Diseño físico
      - Diseño final

Modelos evolutivos. Espiral

- Construir
  - En función del ciclo:
    - Prueba de concepto (prototipo)
    - Primer desarrollo
    - Segundo desarrollo
    - Desarrollo final
- Evaluar
  - En función del ciclo se hace:
    - Análisis de riesgo
    - Prueba del concepto
    - Evaluación de los primeros desarrollos
    - Prueba del desarrollo final

Modelos evolutivos. Espiral (IEEE Std. 1490)



Modelos evolutivos. Espiral

- El IEEE Std. 1490 especifica cuatro ciclos prototípicos
  - Prueba de concepto
  - Primer desarrollo
  - Segundo desarrollo
  - Ciclo final

Modelos evolutivos. Espiral

- Prueba del concepto:
  - Captura de requisitos
  - Definir objetivos
  - Diseño conceptual del sistema
  - Diseño y construcción del prototipo
  - Planes de prueba
  - Análisis del riesgo
  - Hacer recomendaciones

Modelos evolutivos. Espiral

- Primer desarrollo
  - Requisitos del sistema
  - Definir objetivos del primer desarrollo
  - Diseño lógico del sistema
  - Diseñar y construir el primer desarrollo
  - Planes de prueba
  - Evaluar el primer desarrollo
  - Hacer recomendaciones

Modelos evolutivos. Espiral

- Segundo desarrollo
  - Requisitos de los subsistemas
  - Definir objetivos del segundo desarrollo
  - Diseño físico
  - Construir el segundo desarrollo
  - Planes de prueba
  - Evaluar el segundo desarrollo
  - Hacer recomendaciones

Modelos evolutivos. Espiral

- Ciclo final
  - Completar los requisitos de unidad
  - Finalizar diseño
  - Construir el desarrollo final
  - Prueba de rendimiento de unidad, subsistemas, sistemas y pruebas de aceptación

Modelos evolutivos. Espiral

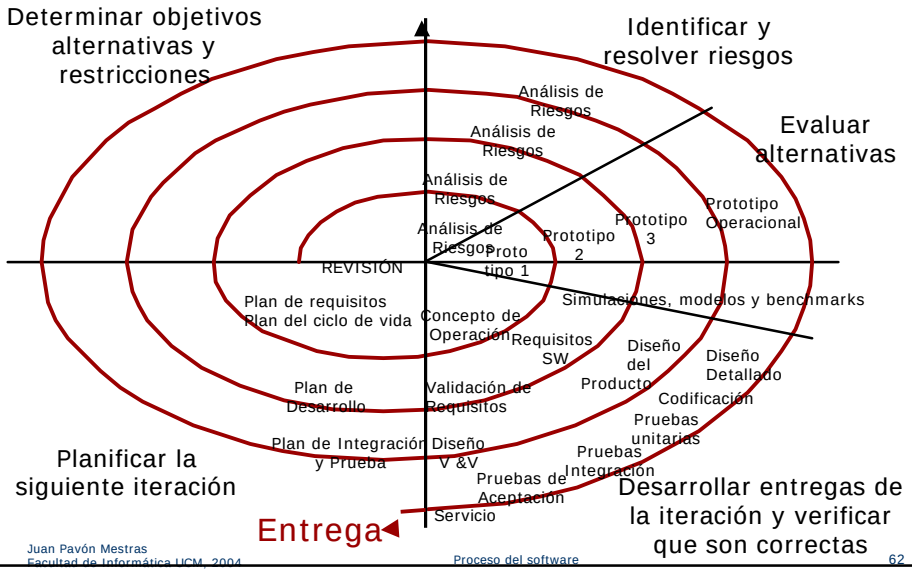
- El Std. 1490 no describe los términos: diseño conceptual, lógico, físico o final...

  ... se entienden como guías de diseño para implementar el prototipo o el desarrollo correspondiente
- Las cuatro actividades son orientativas, y se pueden adecuar a las necesidades de cada organización

Modelos evolutivos. Espiral Boehm (1988)

- 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 de Boehm

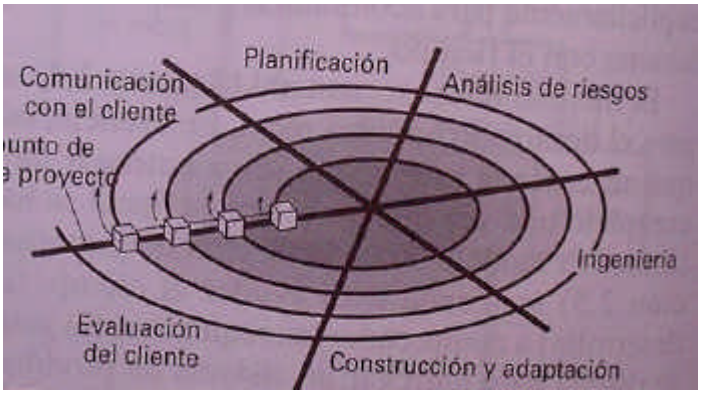


Modelos evolutivos. Espiral Boehm

- 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
- 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 evolutivos. Espiral Boston

- Espiral de Boehm no paramétrico.



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

Modelos evolutivos. Espiral Boston

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

Modelos evolutivos. Espiral Boston

- Vida de un proyecto

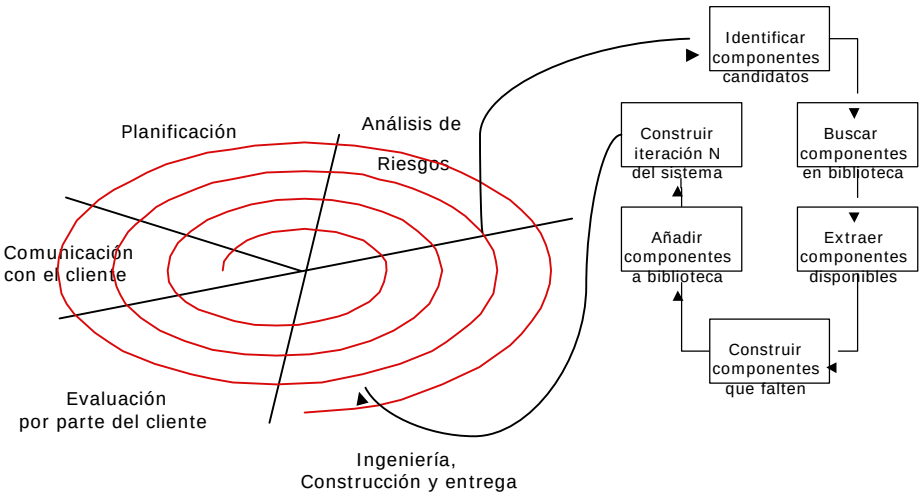
Modelos evolutivos. Espiral

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

Modelos evolutivos. Ensamblaje de componentes

- El modelo de ensamblaje de componentes es un espiral de Boston adaptado al uso de componentes software reutilizables
- Ventajas
  - Componentes
- Inconvenientes
  - Tamaño biblioteca
    - Cómo encontrar los componentes adecuados

Modelos evolutivos. Ensamblaje de componentes



Ejemplos de proceso

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

Proceso unificado de desarrollo

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

Proceso unificado de desarrollo

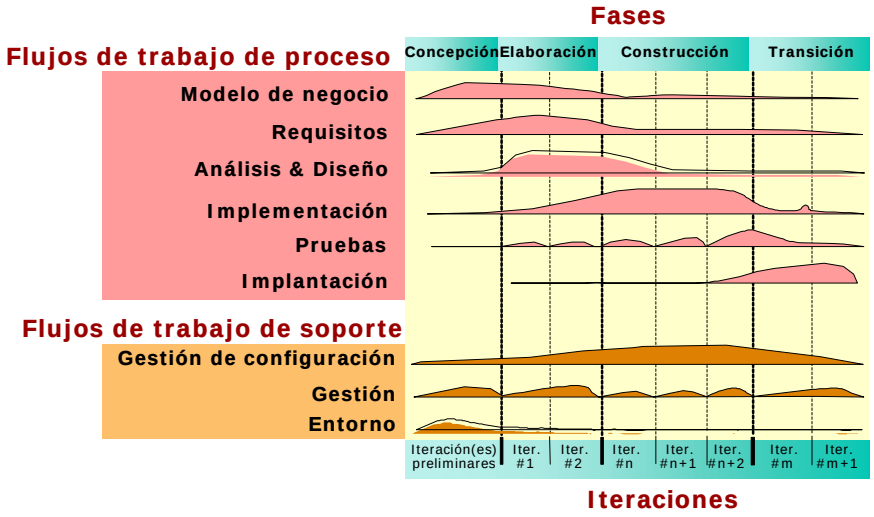
- Es un proceso de desarrollo de software
  - Dirigido por casos de uso
  - Centrado en la arquitectura
  - Interativo e incremental
- Utiliza UML para definir los modelos del sistema sofware
- El sistema software en construcción está formado por
  - *componentes* software
  - interconectados a través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

Los requisitos cambian
y el sistema software evoluciona

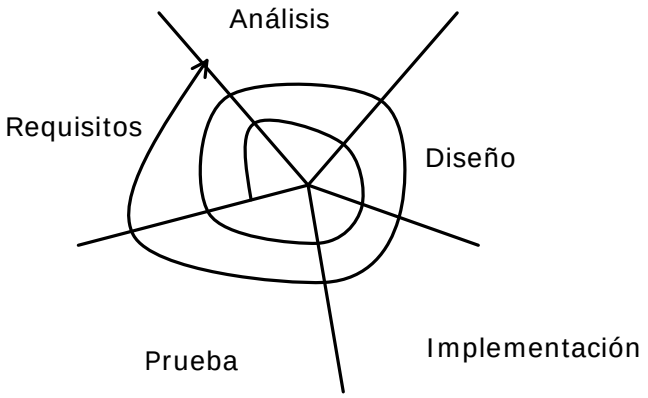
Proceso unificado de desarrollo

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

Proceso unificado de desarrollo



Proceso unificado de desarrollo



Proceso unificado de desarrollo

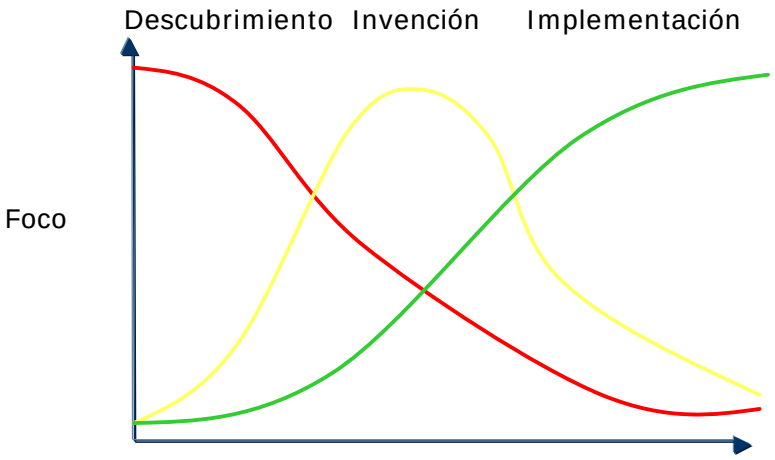
- Cada vuelta en la espiral se denomina iteración
- La agrupación de iteraciones se denomina fase
  - Inicio
  - Elaboración
  - Construcción
  - Transición

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
- Fase de construcción
  - Se crea el producto
- Fase de transición
  - Periodo durante el cual el producto se entrega a clientes
- No todos los flujos de trabajo tienen el mismo peso dentro de cada fase

Proceso unificado de desarrollo

- Actividad en el tiempo



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?

Proceso unificado de desarrollo

- Ventajas
  - Modelo de proceso racional
  - Tecnologías de componentes
- Inconvenientes
  - Muy ligado al método

Extreme Programming (XP)

- Modelo de proceso de K. Beck

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

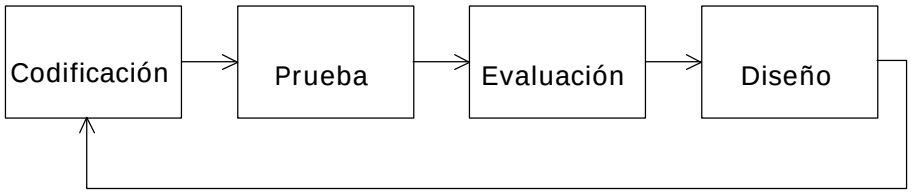
- Características
  - Alta visibilidad debido a ciclos muy cortos
  - Planificación incremental
  - Se adapta a cambios de negocio

Extreme Programming (XP)

- Basado en pruebas automatizadas escritas por desarrolladores y clientes
- 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
  - Solo funciona con equipos pequeños (hasta diez personas)

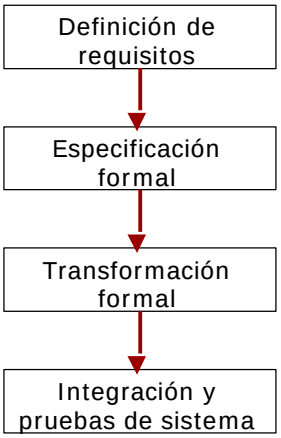
Otros modelos de proceso

- El modelo de desarrollo concurrente
- El modelo de métodos formales
- Técnicas de cuarta generación
- El sentido común

Desarrollo formal de sistemas

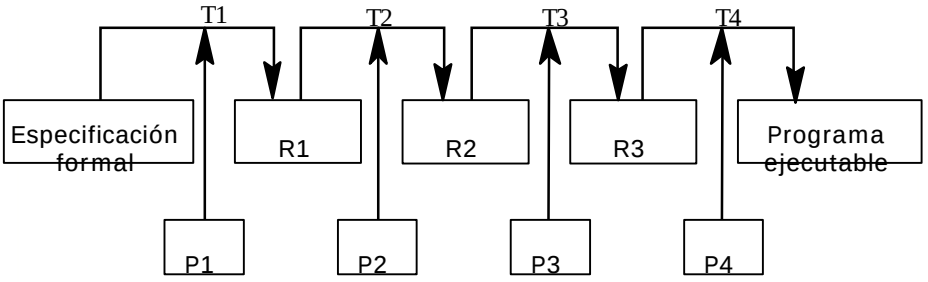
- Se basa en la transformación de una especificación formal a lo largo de varias representaciones hasta llegar a un programa ejecutable
- Las trasnformaciones preservan la corrección
  - Permiten comprobar fácilmente que el programa es conforme a la especificación

Desarrollo formal de sistemas



Desarrollo formal de sistemas

- Transformación formal



Pruebas de corrección de la transformación

Desarrollo formal de sistemas

- Problemas
  - Hace falta una formación especializadda 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 entrega de resultados puede no coincidir con el tiempo necesario para completar una actividad
  - La necesidad de producir documentos restringe la iteración entre procesos
  - El tiempo para revisar y aprobar documentos es 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

Que 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

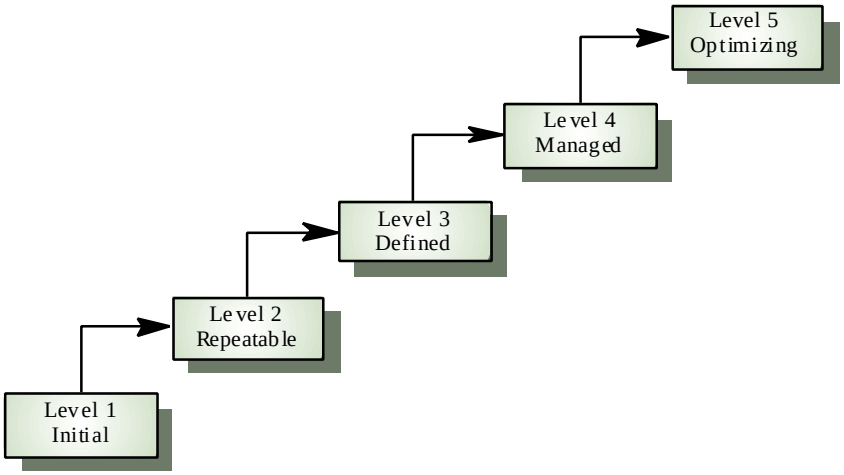
Calidad del proceso de software

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

Madurez del proceso software

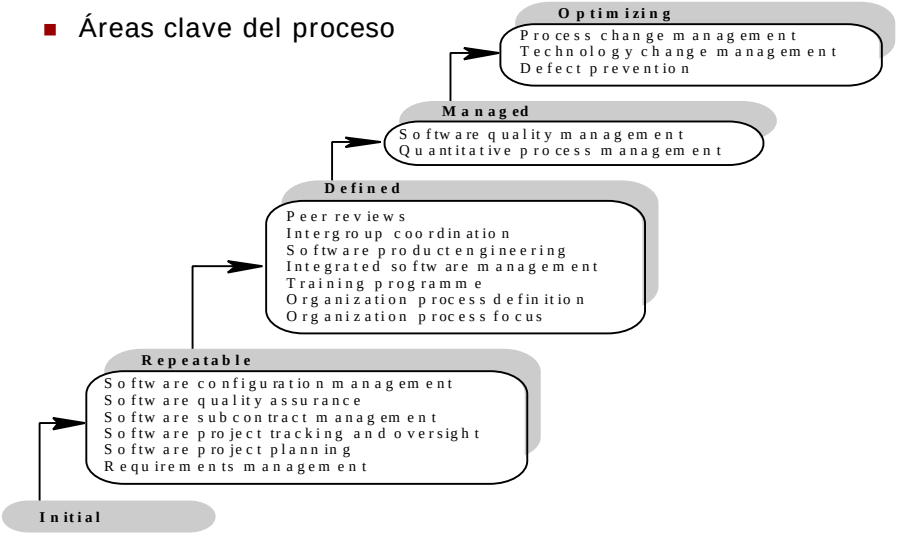
- Madurez del proceso: corrección en la aplicación de un proceso de software
- El Software Engineering Institute (SEI) [http://www.sei.cmu.edu] ha identificado una serie de funciones que deberían estar presentes en el proceso
- El grado de madurez del proceso se determina con un cuestionario y un esquema de cinco niveles
  - CMM: Capability Maturity Model
    - Desarrollado a mediados de los ‘80
    - Refinado a principios de los ’90
  - Este modelo define una serie de áreas clave de proceso (ACP)
    - Un área clave de proceso es, básicamente, una actividad de IS

CMM



CMM

- 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, 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.
    - 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 minimas:
  - Planificación del proyecto.
  - Seguimiento y supervisión del proyecto.
  - Gestión de requisitos.
  - GCS.
  - SQA.
  - Definición del proceso.
  - Revisiones periodicas.
  - Coordinación entre grupos.

CMM

- Datos Agosto 2000
  - Inicial 34,9%
  - Repetible 38,2%
  - Definido 18,5%
  - Gestionado 5,5%
  - Optimizado 2,9%

    - Resultados de 901 empresas desde 1996

Conclusiones

- En IS hay mucho que hacer...
  ... el modelo de proceso es nuestro soporte
- La IS es una tecnología multicapas
- La IS es una ingeniería
- En IS hay tres fases genéricas
  - Estas tres fases están presentes en todos los modelos de proceso

Conclusiones

- Se puede medir la madurez
- Disponemos de múltiples modelos de proceso
- También hay modelos que no son de proceso
- Al final TODOS, de una forma u otra, resuelven las fases de IS
- Tensión entre el producto y el proceso

Bibliografía

- Pressman, capítulo 2
- Sommerville, capítulo 3
- CMM
  - Pressman, cap. 2
  - Sommerville, cap. 25
  - http://calisto.sip.ucm.es/people/pablo/teaching/tp/areasClaveSWCMM.pdf
  - http://www.sei.cmu.edu/cmm/obtain.cmm.html