Tema 02 –
Proceso de desarrollo de software

Ingeniería del Software

Rubén Fuentes Fernández
Dep. Ingeniería del Software e Inteligencia Artificial
Facultad de Informática
Universidad Complutense Madrid

Trabajando con Antonio Navarro, Juan Pavón y Pablo Gervás

Contenidos

- El proceso de desarrollo de software
  - Introducción
  - Modelos de proceso
- Modelo de madurez de capacidades del software

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

INTRODUCCIÓN

El producto como base del proceso

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

Conceptos clave del proceso

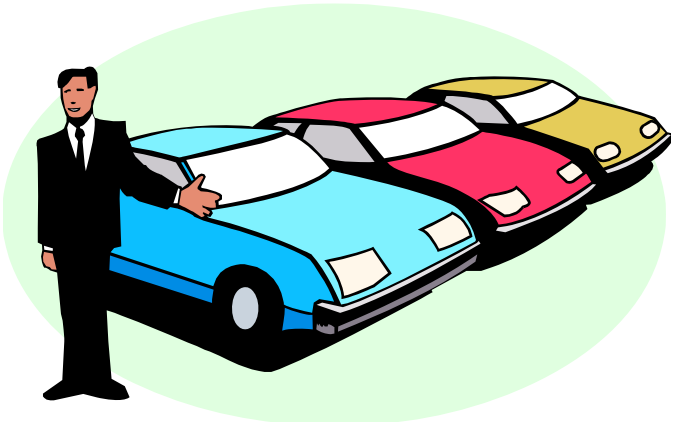
- *Personas*: los que trabajan
- *Producto*: lo que se obtiene
- *Proyecto*: la pauta a seguir para desarrollar un producto
- *Proceso*: la pauta a seguir para desarrollar un proyecto

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*: ???
- *Proyecto*: ???
- *Proceso*: Asignatura de IS

Aspectos a considerar en la Ingeniería

- 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

Tecnología estratificada

- La IS es una tecnología multicapa:
  - Capa de enfoque de calidad
    - Base de cualquier proceso de ingeniería
    - Mejores técnicas o buenas prácticas
  - Capa de proceso
    - Conjunto de actividades y resultados asociados que sirven para construir un producto software.
  - Capa de métodos
    - Un método ofrece pautas para: análisis de requisitos, diseño, construcción de programas, prueba, instalación y mantenimiento
  - Capa de herramientas

Objetivos de un proceso del software

- Desarrollar, operar y mantener software de forma que:
  - Se apliquen principios y metodologías de ingeniería
  - Se construya software que satisfaga las necesidades del cliente
    - No sólo funcionales
    - También coste, fiabilidad y eficiencia
  - Se pueda sistematizar, disciplinar y cuantificar
    - Establecer pautas sobre las que orientar, reflexionar, aprender y mejorar

Producto y fases genéricas

- 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

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
- Tareas principales:
  - Planificación del proyecto software
  - Ingeniería de sistemas o de información
  - Análisis de requisitos

Fase de desarrollo

- Centrada en el *cómo*
- Se define cómo han de plasmarse los requisitos en el sistema:
  - Diseño de las estructuras de datos
  - Implementación de las funciones
  - Caracterización de las interfaces
  - Traducción del diseño a un lenguaje de programación
  - Pruebas
- Tareas principales:
  - Diseño del software
  - Generación del código
  - Pruebas del software

Fase de mantenimiento

- Centrada en el cambio.
- Pueden producirse 4 tipos de cambio:
  - *Corrección*. Corregir los defectos
  - *Adaptación*. Modificaciones por cambio en el entorno externo
  - *Mejora*. Ampliar los requisitos originales, a petición del cliente
  - *Prevención*. Cambio para facilitar el cambio
- En esta fase se vuelven a aplicar las fases de definición y desarrollo, pero sobre software ya existente.

Fases, actividades y tareas

- Las fases de un proceso del software se realizan mediante una serie de *actividades*.
  - Las actividades se pueden extender a lo largo de varias fases.
  - Ej. comunicación con el cliente, planificación, diseño e implementación del software
- Las actividades a su vez se articulan mediante *acciones*.
  - Una *acción* es un conjunto de tareas relacionadas que produce un producto del trabajo o artefacto.
  - Ej. la actividad *diseño e implementación* puede incluir una acción de *análisis*, que puede descomponerse en las tareas *realizar diagramas casos de uso, realizar diagramas actividades, realizar diagramas clases análisis*…

Actividades de protección

- Las actividades de creación de artefactos se complementan con las actividades de *soporte* o *protección.*
  - 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 del sistema
  - *Implementación*: construcción del sistema de software
  - *Validación*: verificar 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
- Debe estar explícitamente modelado si va a ser bien administrado.

Modelos de proceso

- Un modelo de proceso del software (o paradigma de IS) es una representación abstracta de un proceso del software.
  - Plantilla, patrón o marco que define el proceso a través del cual se crea software
- Los modelos de proceso tienen en cuenta, en mayor o menor medida, las actividades estructurales comunes a todos los procesos de construcción de software.
- Un modelo de proceso ha de definir sus actividades estructurales así como su flujo de realización.

Adaptación de los modelos de proceso

- Los modelos de proceso no son fijos para una organización o tipo de proyecto.
- Una organización puede 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 requeridos
- Los modelos de proceso se ajustan a los proyectos concretos variando el conjunto de tareas en que se dividen las actividades estructurales.

Características para seleccionar el 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 con que el proceso proporciona un sistema a partir de una especificación

MODELOS DE PROCESO QUE SÍ SON…

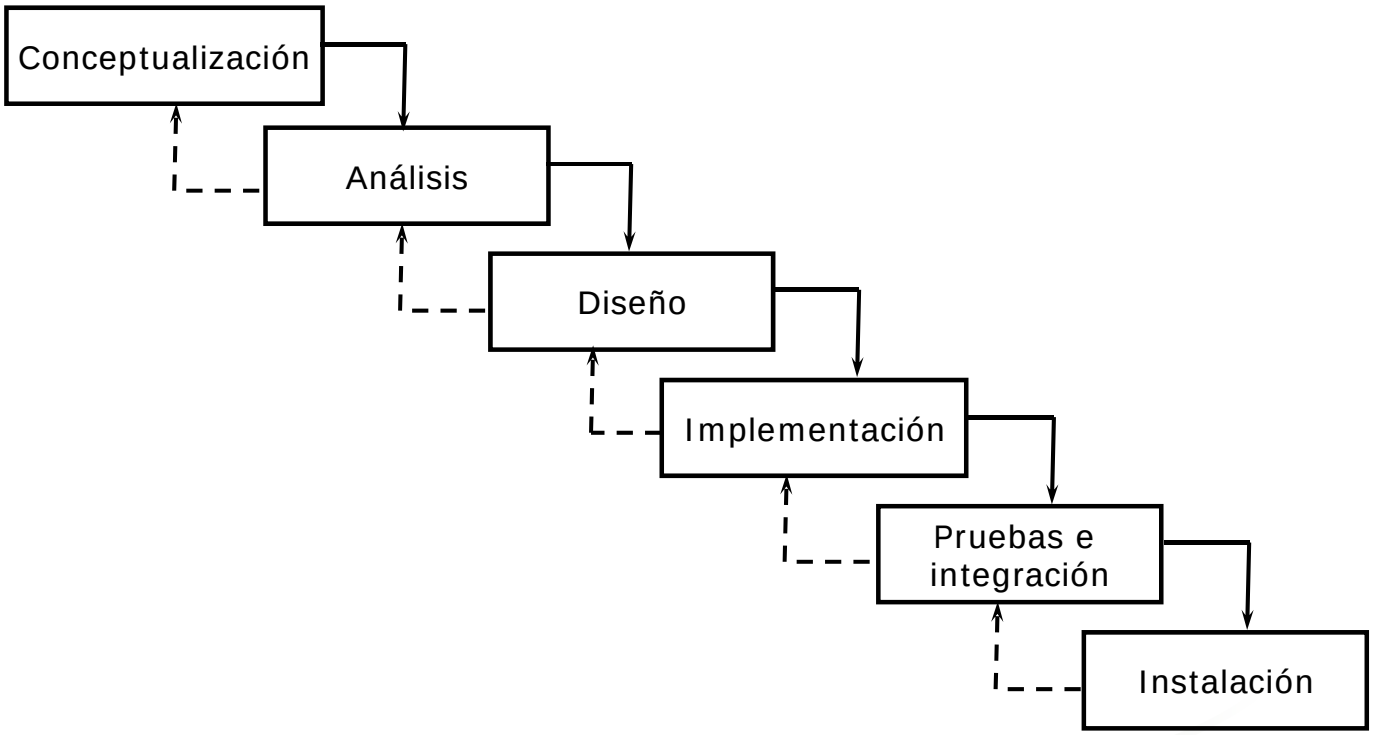
Principales modelos de proceso del software

- Modelo en cascada
  - Separa en distintas fases de especificación y desarrollo que se realizan en secuencia lineal.
- Prototipado
  - Un prototipo sirve de modelo para la construcción del sistema final.
- Desarrollo evolutivo
  - La especificación y el desarrollo están intercalados.
- 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.

Modelos de proceso que sí son…

MODELOS EN CASCADA

Modelo en cascada (*waterfall*) (1/2)



Modelo en cascada (2/2)

- Es el modelo de proceso clásico (desde los 1970s)
- Es sencillo y fácil de entender
  - Basado en la mentalidad de línea de ensamblaje
- 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 tienen que haber conseguido todos los objetivos de la fase anterior.
  - No hay apenas comunicación entre las fases
- Se supone que sólo se baja en la cascada...
  - ... pero también se puede subir (aunque difícilmente)

Modelo en cascada: fases

- Conceptualización
  - Se determina la arquitectura de la solución
  - Arquitectura = 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

Modelo en cascada: documentos

Actividad	Documentos producidos
Análisis de Requisitos	Documento de Requisitos
Definición de Requisitos	Documento de Requisitos
Especificación del Sistema	Especificación Funcional y Plan de Pruebas de Aceptación
Diseño Arquitectural	Especificación de la Arquitectura y Plan de Pruebas del Sistema
Diseño de Interfaces	Especificación de Interfaces y Plan de Pruebas de Integración
Diseño detallado	Especificación del Diseño y Plan de Pruebas 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

Variantes del modelo en cascada

- Modelo Sashimi
  - Solapamientos de fases
- Sommerville
- Pressman (lineal secuencial)
- Modelo en V
- Cascada con subproyectos

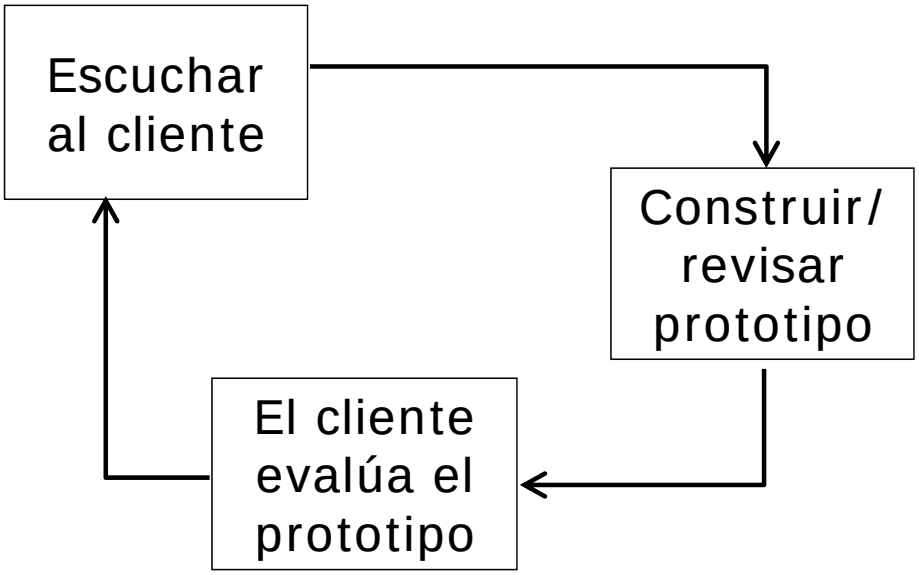
Modelo en cascada: evaluación

- Ventajas:
  - Muy probado
  - Sencillo
    - Sirve cuando el personal está poco cualificado
  - Aplicable cuando el problema es estable y cuando se trabaja con técnicas conocidas
- Inconvenientes:
  - Poco realista
  - Supone una especificación de requisitos estable
  - Baja visibilidad
    - No se ve un producto hasta muy tarde en el proceso
  - Un error grave en las últimas fases puede ser letal
  - Si no hay solapamientos entre fases puede haber bloqueos
  - Las revisiones de proyectos de gran complejidad son muy difíciles

Modelos de proceso que sí son…

PROTOTIPADO

Modelo de construcción de prototipos



Modelo de construcción de prototipos

- Comienza con la recolección de requisitos
  - Clientes 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 (ej. 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
    - Ej. utilizar un sistema operativo o lenguaje de programación inadecuado pero que ya es conocido

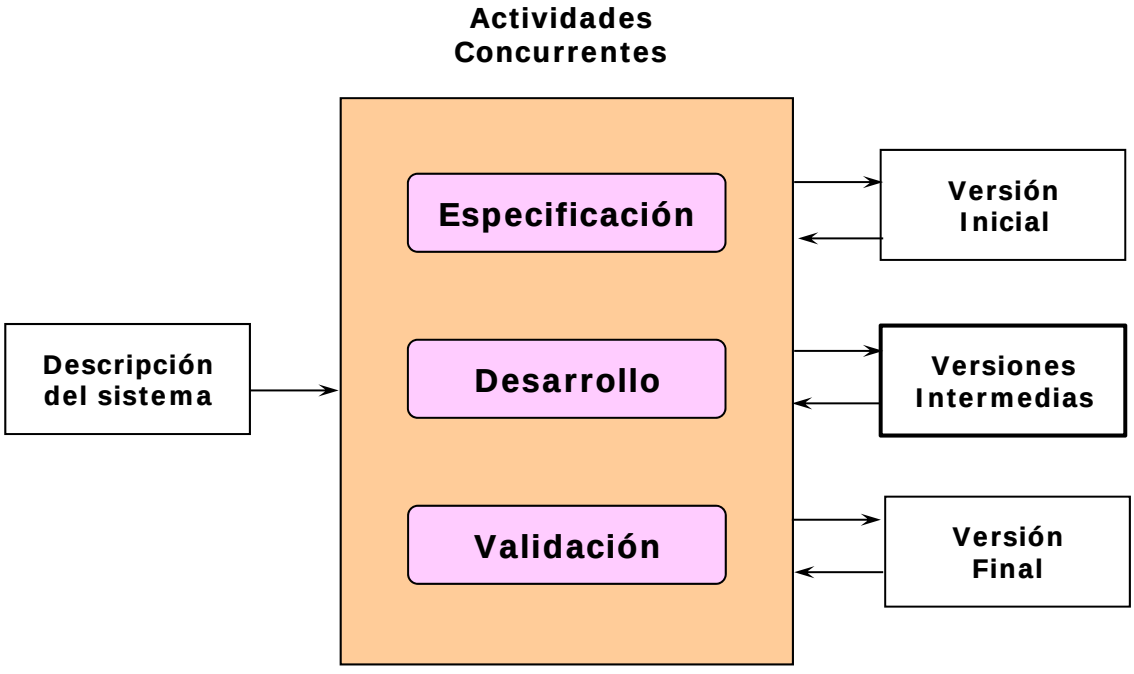
Modelos de proceso que sí son…

DESARROLLO EVOLUTIVO

Modelos evolutivos

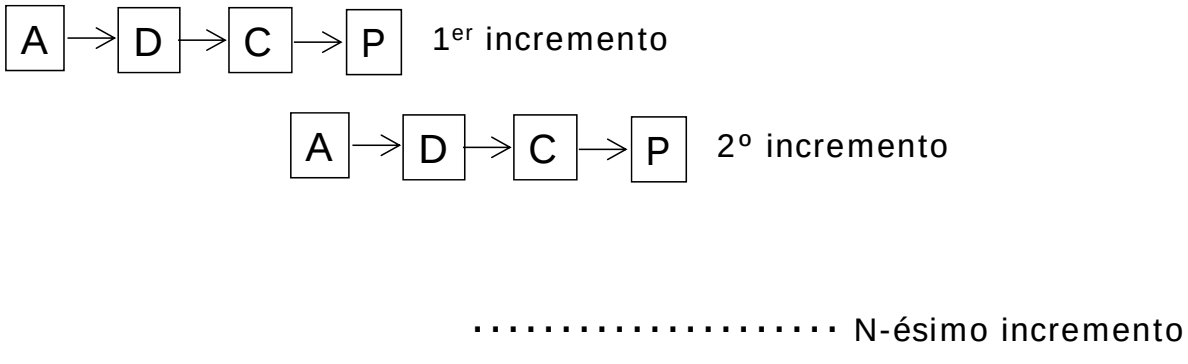
- Características:
  - Gestionan bien la naturaleza evolutiva del software
  - Son iterativos → construyen versiones del software cada vez más completas
- Se adaptan bien a:
  - Los cambios de requisitos del producto
  - Fechas de entrega estrictas poco realistas
  - Especificaciones parciales del producto

Modelos evolutivos



Modelos evolutivos – incremental

- Fusiona el modelo lineal secuencial con el de construcción de prototipos
  - El modelo lineal secuencial es una variante del modelo en cascada



Modelos evolutivos – incremental

- Cada secuencia lineal 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 (ej. por el cliente) el producto entregado.
  - Como resultado se desarrolla un plan para el incremento siguiente
- Se itera.
  - Hasta elaborar el producto completo

Modelos evolutivos – incremental

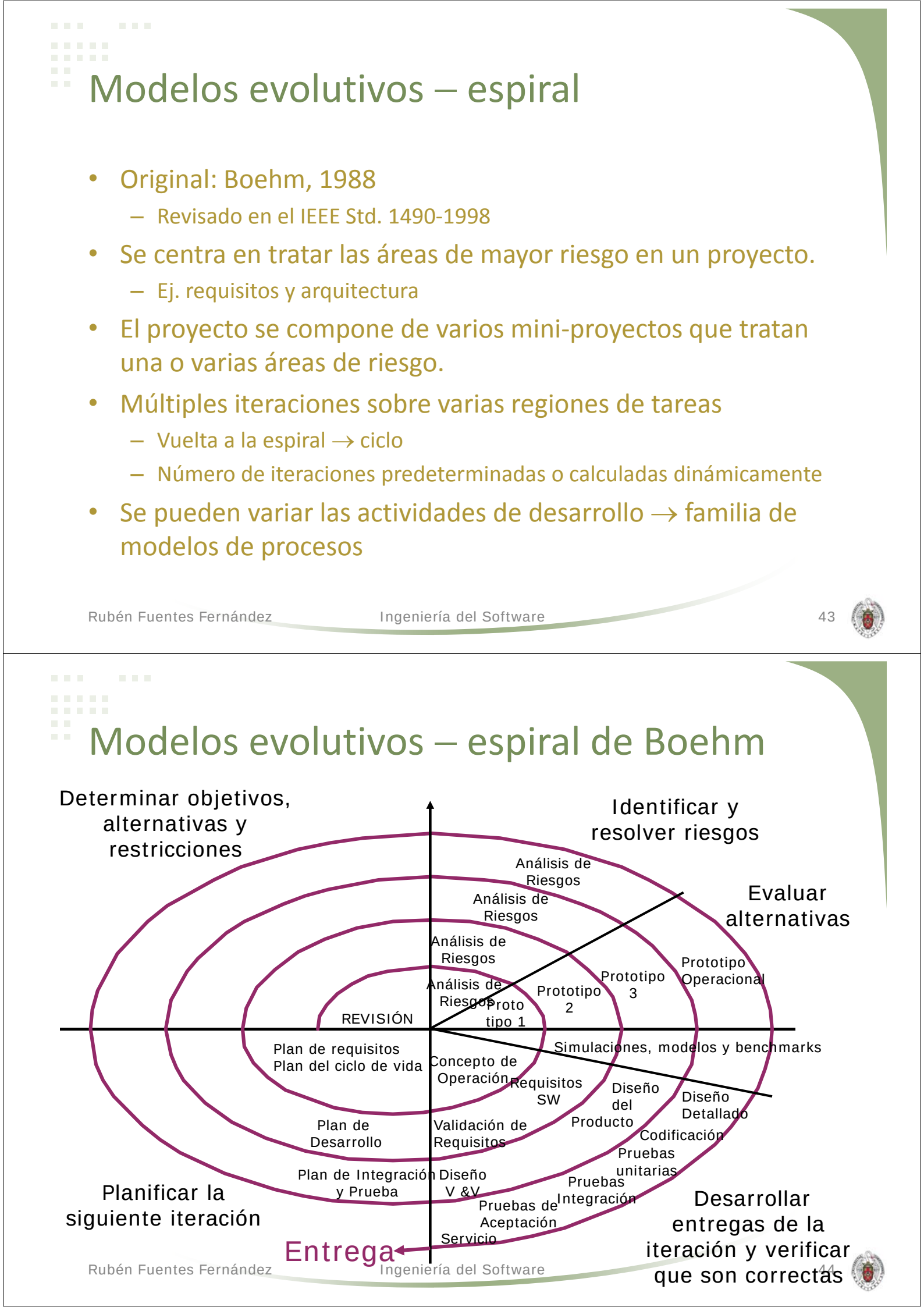
- Ventajas
  - Es interactivo
    - Con cada incremento se entrega al cliente un producto operacional que puede evaluar
  - Personal
    - Permite variar el personal asignado a cada iteración
  - Gestión riesgos técnicos
    - Ej. disponibilidad de hardware específico
- Inconvenientes
  - La primera iteración puede plantear los mismos problemas que en un modelo lineal secuencial

Modelo evolutivo incremental vs Prototipos

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

Modelos evolutivos – espiral

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



Modelos evolutivos – espiral de Boehm

Modelos evolutivos – espiral

- El IEEE Std. 1490 especifica cuatro ciclos prototípicos:
  - Prueba de concepto
    - Requisitos generales
  - Primer desarrollo
    - Requisitos del sistema
  - Segundo desarrollo
    - Requisitos de los subsistemas
  - Ciclo final
    - Requisitos de unidades
    - Realización de pruebas de aceptación
- Las regiones de tareas son asimilables a las fases del modelo en cascada.

Modelos evolutivos – espiral IEEE Std. 1490: regiones (1/2)

- 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 IEEE Std. 1490: regiones (2/2)

- 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: evaluación

- Ventajas
  - Enfoque realista
  - Gestión explícita de riesgos
  - Centra su atención en la reutilización de componentes y la eliminación de errores en información descubiertos en las 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

Ejemplos de procesos evolutivos modernos

- 3 modelos de proceso concretos
  - Modelos pesados
    - Proceso Unificado
  - Modelos ágiles
    - *eXtreme Programming*
    - *Scrum*

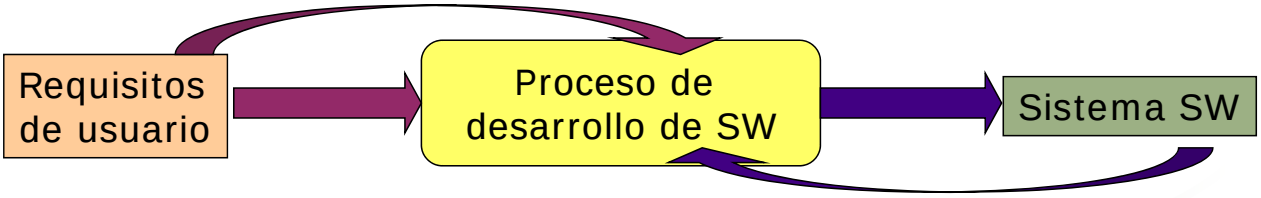
PROCESO UNIFICADO DE DESARROLLO

Proceso unificado de desarrollo

- El Proceso Unificado (*Unified Process*, UP) es creado por los autores de UML.
  - Booch: método Booch
  - Rumbaugh: OMT
  - Jacobson: proceso Objectory
- También es conocido como RUP (*Rational Unified Process*) en su versión comercial.

Proceso unificado de desarrollo: claves

- 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

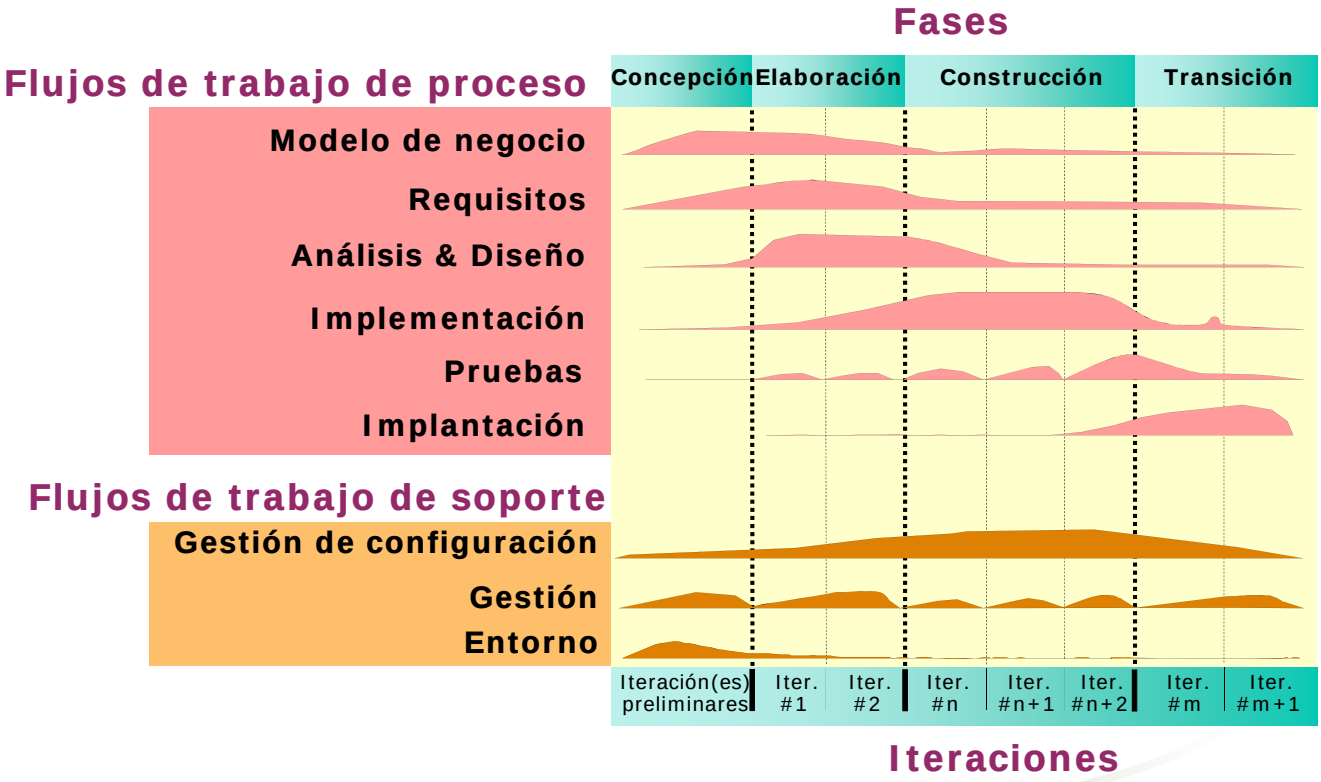


Los requisitos cambian y el sistema SW evoluciona

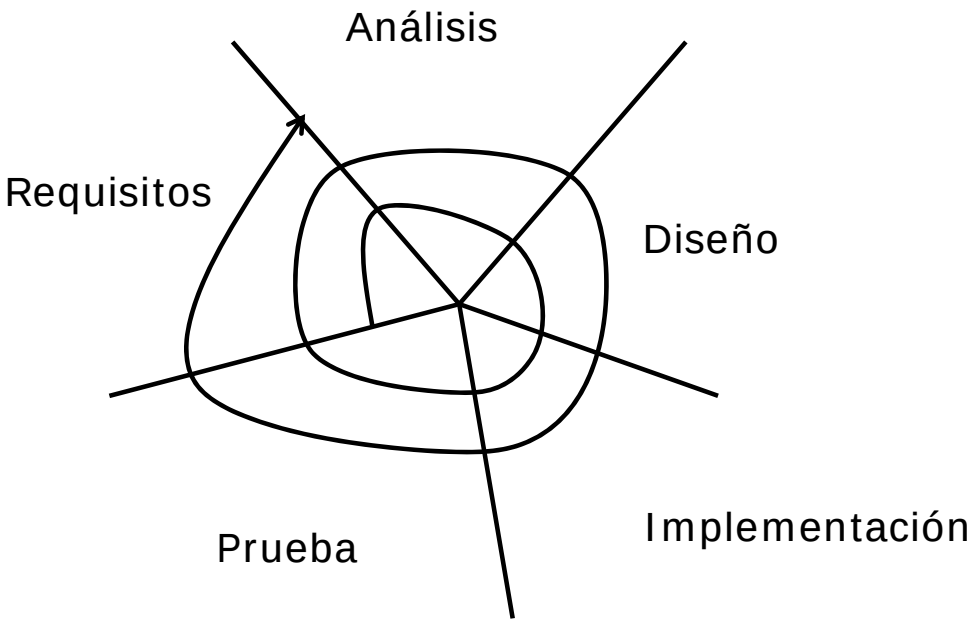
Proceso unificado de desarrollo

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

Proceso unificado de desarrollo: mapa



Proceso unificado de desarrollo: iteración



Proceso unificado de desarrollo: fase

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

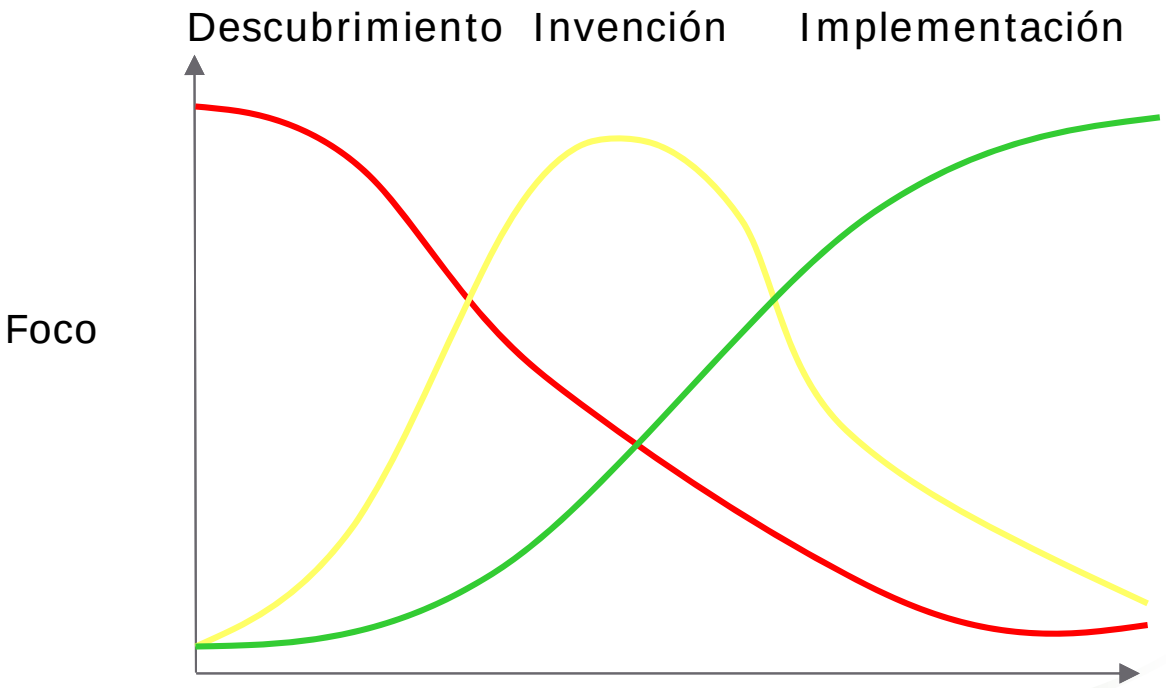
Proceso unificado de desarrollo: fases

- 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: ciclo

- Las agrupaciones de fases se denominan ciclo.
- Cada ciclo concluye con una versión del producto.

- Discusión:
  ¿es lo mismo un ciclo UP que un ciclo del modelo en espiral?

Proceso unificado de desarrollo: evaluación

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

DESARROLLO ÁGIL

Modelos de proceso ágil: supuestos

- Los procesos ágiles parten de 3 supuestos clave:
  - Es difícil predecir qué requisitos software persistirán y cuáles cambiarán.
    - También es difícil predecir las prioridades del cliente.
  - El diseño y el desarrollo de software están intercalados.
    - Por tanto, se deben realizar de manera conjunta, de forma que el diseño se pruebe según se crea.
    - Es difícil predecir cuanto diseño es necesario antes de construir el código que lo implementa.
  - El análisis, el diseño y la construcción no son predecibles desde el punto de vista de la planificación, lo que sería deseable.

Modelos de proceso ágil: características

- Un proceso ágil tiene 3 características clave:
  - Es adaptable de forma incremental.
  - Necesita retroalimentación del cliente.
  - Se basa en la entrega continua de incrementos.
- Teóricamente, la agilidad se puede aplicar a cualquier proceso de software.
  - En cualquier caso, surgen modelos de proceso propios del desarrollo ágil.

eXtreme Programming

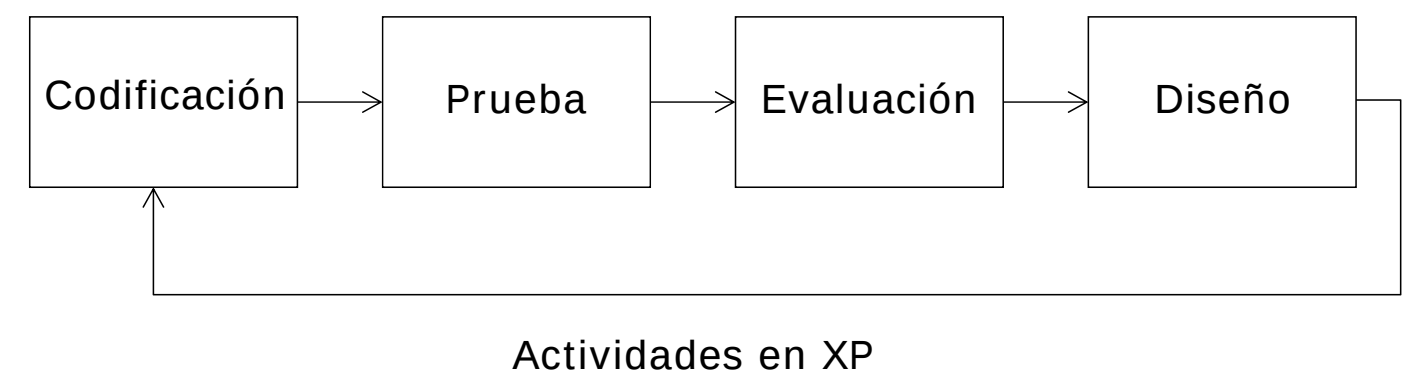
- El *eXtreme Programming* (XP) es un modelo de proceso propuesto por K. Beck:

  "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: claves

- 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: estructura

- La estructura del proceso, si se puede considerar que la hay, es un poco atípica

Actividades en XP

eXtreme Programming: prácticas

- Las 4 actividades de XP están soportadas por 12 prácticas:
  - El juego de planificación
  - Pequeñas entregas
  - Metáfora
  - Diseño simple
  - Prueba
  - Refactorización
  - Programación en pareja
  - Propiedad colectiva
  - Integración continua
  - Semana de cuarenta horas
  - Cliente en el lugar de desarrollo
  - Codificación estándar

eXtreme Programming: evaluación

- Ventajas:
  - Bueno para especificaciones cambiantes
  - Fundamentación práctica
- Inconvenientes:
  - Poco probado
  - Poco compatible con especificaciones/diseños totales
  - Solo funciona con equipos pequeños (hasta 10 personas)

- Esta evaluación es extensible al resto de procesos ágiles.

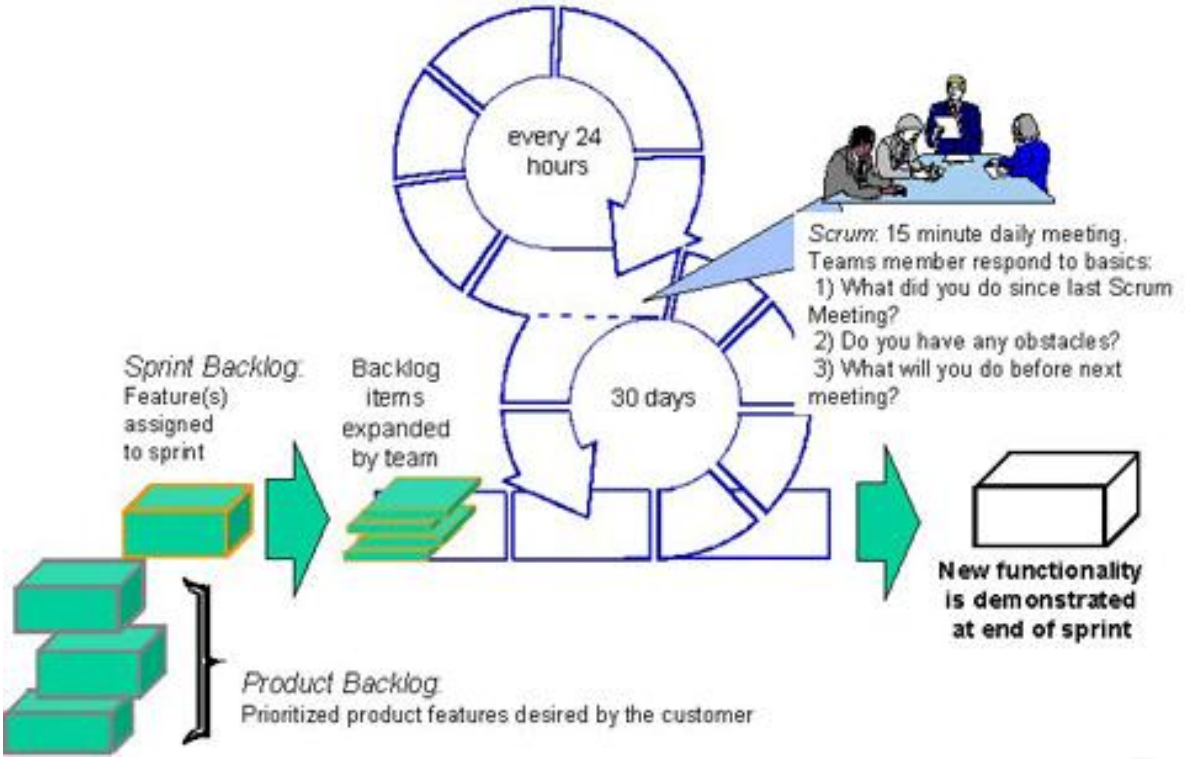
SCRUM

- SCRUM (melé) es un modelo de proceso desarrollado por J. Sutherland a principios de los 90, y revisado posteriormente por Schwaber y Beedle.
- Se basa en:
  - Equipos de trabajo organizados para maximizar la comunicación y minimizar los gastos
  - Proceso adaptable a cambios técnicos y de negocio para asegurar el mejor producto
  - Incrementos frecuentes de software
  - Trabajo y equipo divididos en paquetes de bajo acoplamiento
  - Capacidad de construir un producto cuando se requiera

SCRUM: actividades

- El proceso incluye actividades clásicas:
  - Requisitos
  - Análisis
  - Diseño
  - Evolución
  - Entrega
- Dentro de cada actividad las tareas se ajustan a patrones de proceso.
  - Un patrón de proceso es un conjunto de tareas.
- En general, SCRUM se basa en cuatro patrones de proceso para organizar las actividades.

SCRUM: patrones de proceso



SCRUM: patrones de proceso (1/2)

- *Product backlog* (trabajo acumulado)
  - Lista de requisitos priorizados que puede actualizarse en cualquier momento, salvo en el propio *sprint*
- *Sprint*
  - Trabajo requerido para satisfacer un requisito definido
  - 30 días normalmente

SCRUM: patrones de proceso (2/2)

- *SCRUM*
  - Reunión diaria corta (15”) centrada en:
    - ¿Qué hiciste desde la última reunión?
    - ¿Tienes algún obstáculo?
    - ¿Qué harás antes de la próxima reunión?
  - Un *maestro de SCRUM* preside la reunión y evalúa las respuestas de cada persona.
- Demostración
  - Se entrega el incremento al cliente.

Modelos de proceso que sí son…

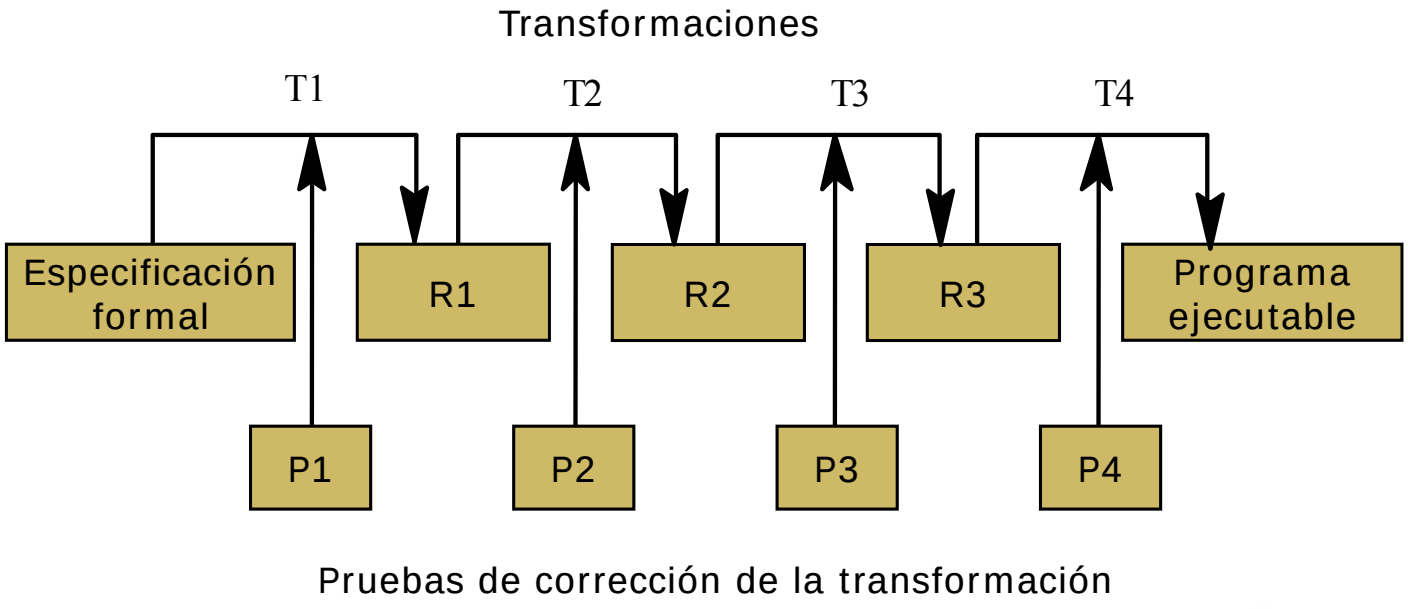
TRANSFORMACIONES FORMALES

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 transformaciones preservan la corrección.
  - Permiten comprobar fácilmente que el programa es conforme a la especificación.

Desarrollo formal de sistemas

- Transformación formal



Desarrollo formal de sistemas: evaluación

- 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 deben poder asegurarse antes de poner el sistema en operación.

Modelos de proceso que sí son…

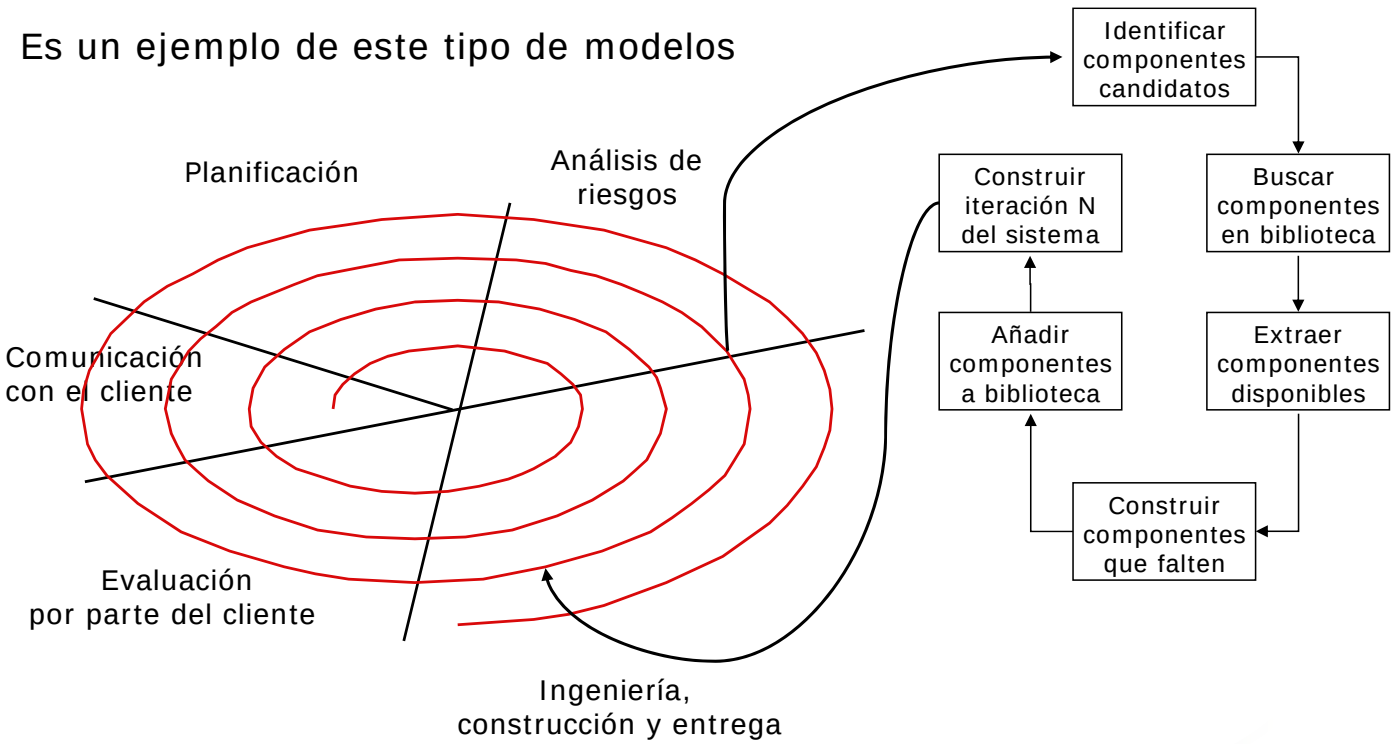
DESARROLLO BASADO EN REUTILIZACIÓN

Componentes

- Una *interfaz* es una colección de operaciones que son utilizadas para especificar un servicio de una clase o de un componente.
- Un *componente* es una parte física y reemplazable de un sistema que se ajusta a, y proporciona la realización de, un conjunto de interfaces.
- Un sistema software basado en componentes se modela y construye en base a *componentes* software interconectados a través de *interfaces* bien definidas.

Modelos evolutivos – ensamblaje de componentes

Es un ejemplo de este tipo de modelos



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

- Realmente todos los procesos vistos pueden hacer uso de componentes.

SELECCIÓN DE PROCESOS

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 más 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

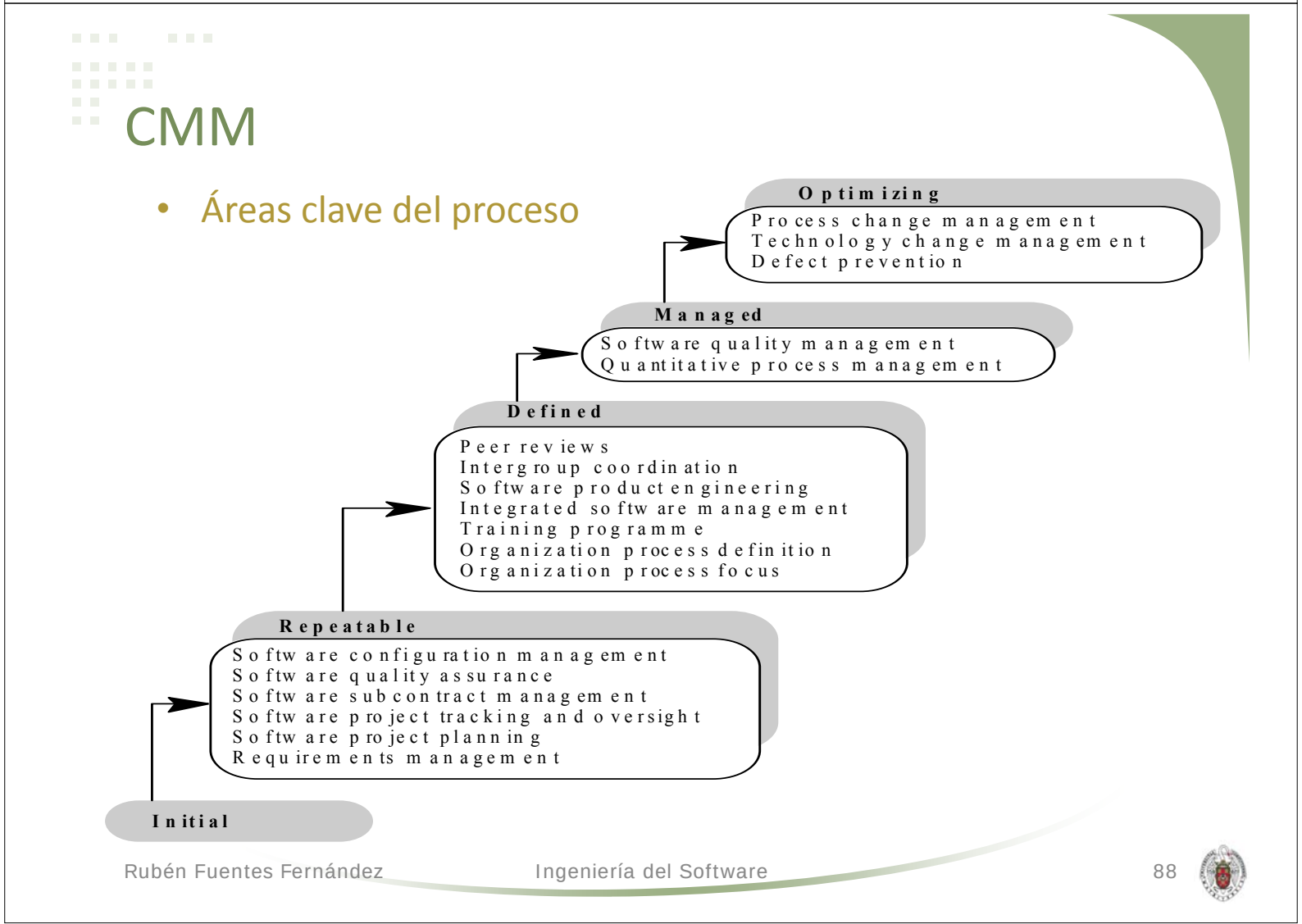
¿Qué modelo utilizar?

- Para sistemas bien conocidos se puede utilizar el modelo en cascada.
  - La fase de análisis de riesgos debe ser 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.

MADUREZ DEL PROCESO SOFTWARE DE UNA ORGANIZACIÓN

Madurez del proceso software

- La madurez del proceso software es la corrección en la aplicación de un proceso de software.
- El Instituto de Ingeniería del Software (SEI) ha desarrollado el Modelo de Madurez del Software (SW CMM) para identificar este nivel.
- El CMM identifica una serie de Áreas Clave del Proceso (ACPs).
  - Una ACP es básicamente una *actividad* de IS.
  - Las ACPs han de aparecer en la aplicación del proceso.
- Dependiendo de la ejecución del proceso, las ACPs se encuentran en distintos niveles.
- El nivel de las ACPs determina el nivel de madurez de la organización.



CMM

- Áreas clave del proceso

CMM: niveles (1/3)

- Nivel 1: Inicial
  - El proceso de software se caracteriza según el caso.
  - Se definen poco los procesos.
  - El éxito depende del esfuerzo individual.
- Nivel 2: Repetible
  - Nivel 1
  - Se incluye seguimiento del coste, de la planificación y de la funcionalidad.
  - Se repiten técnicas de proyectos anteriores con buenos resultados.
  - Las ACPs 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: niveles (2/3)

- Nivel 3: Definido
  - Nivel 2
  - Las actividades se documentan, estandarizan e integran en un proceso a nivel organización.
  - Existe un proceso documentado.
  - Las ACPs 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: niveles (3/3)

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

CMM: niveles

- 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

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

Conclusiones

- La IS es una tecnología multicapa.
- La IS es una ingeniería.
- El proceso del software es un conjunto estructurado de actividades requeridas para desarrollar un sistema de software.
  - Varían dependiendo de la organización y el tipo de sistema a desarrollar.
- El proceso debe estar explícitamente modelado si va a ser bien administrado.
- En IS hay tres fases genéricas.
  - Estas tres fases están presentes en todos los modelos de proceso.

Glosario

- ACP = Área Clave de Proceso
- CASE = *Computer-Aided Software Engineering*
- GCS = Gestión de Configuración Software
- IS = Ingeniería del Software
- RUP = *Rational Unified Process*
- SEI = *Software Engineering Institute*
- SQA = *Software Quality Assurance*
- SW CMM = *Software Capability Maturity Model*
- UML = *Unified Modelling Language*
- UP = *Unified Process*
- XP = *eXtreme Programming*

Referencias

- R. Pressman: Ingeniería del Software. Un enfoque práctico, 7ª edición. McGraw-Hill, 2010.
  - Modelos de proceso → pp. 17-46
  - SW CMM → pp. 21-25
- I. Sommerville: Ingeniería del Software, 7ª edición. Addison Wesley, 2007.
  - Modelos de proceso → pp. 42-67
  - SW CMM → pp. 557-575
- Proceso unificado de Rational
  - Jacobson, Krutchen
- SW CMM
  - http://www.sei.cmu.edu/cmm/obtain.cmm.html