Tema 2: El proceso de desarrollo del software. Paradigmas o modelos de desarrollo del Software

BLOQUE I: INTRODUCCIÓN Y PARADIGMAS DE DESARROLLO EN INGENIERÍA DEL SOFTWARE

Ingeniería del Software

Grado en Ingeniería Informática

Curso 2024/2025

Índice

Índice

El Proceso: Modelos de Desarrollo

Proceso: conjunto de actividades necesarias para transformar los requisitos de un usuario en un sistema funcional

- Características:
  - Tiene una serie de actividades principales
  - Utiliza recursos, está sujeto a restricciones y genera productos intermedios y finales
  - Compuesto por subprocesos que se encadenan de alguna forma
  - Cada actividad tiene sus criterios de entrada y salida, que permiten conocer cuando comienza y termina dicha actividad
  - Existen principios orientadores que explican las metas de cada actividad

El Proceso: Modelos de Desarrollo

- Según el estándar ISO/IEC 12207 de 1995:
  - **Ciclo de vida del software:** El periodo de tiempo comprendido desde la definición de los requisitos hasta el fin de su uso. Proporciona consistencia y estructura a las actividades, facilitando la correcta realización de las mismas de forma repetitiva.
  - **Procesos:** Conjunto de actividades y tareas interrelacionadas que, al ejecutarse de manera conjunta, transforman una entrada en una salida.

El Proceso: Modelos de Desarrollo

Procesos primarios del ciclo de vida del software

- Adquisición: Proceso global que sigue el adquiriente para obtener el producto
- Suministro: Proceso global que sigue el suministrador para proporcionar el producto
- Desarrollo: Proceso empleado por el suministrador para diseñar, construir y probar el producto
- Operación: Proceso seguido por el operador en el uso diario del producto
- Mantenimiento: Proceso empleado para mantener el producto, incluyendo cambios tanto en el producto como en su entorno de operación

El Proceso: Modelos de Desarrollo

Procesos de soporte del ciclo de vida del software

- Documentación: Actividades para registrar información específica utilizada por otros procesos
- Gestión de la configuración: Actividades para mantener un registro de los productos generados en la ejecución de los procesos
- Aseguramiento de la calidad: Actividades para garantizar objetivamente que el producto y los procesos cumplen con los requisitos documentados y planificados.
- Verificación: Actividades para comprobar el producto
- Validación: Actividades para confirmar que el producto cumple su propósito

El Proceso: Modelos de Desarrollo

Procesos de soporte del ciclo de vida del software

- Reuniones de revisión: Reuniones entre las dos partes para evaluar el estado del producto y de las actividades
- Auditorias: Actividades para comprobar que el proyecto cumple con los requisitos, planes y contratos
- Resolución de problemas: Actividades para analizar y resolver problemas relativas al proyecto, sea cual sea su fuente y naturaleza

El Proceso: Modelos de Desarrollo

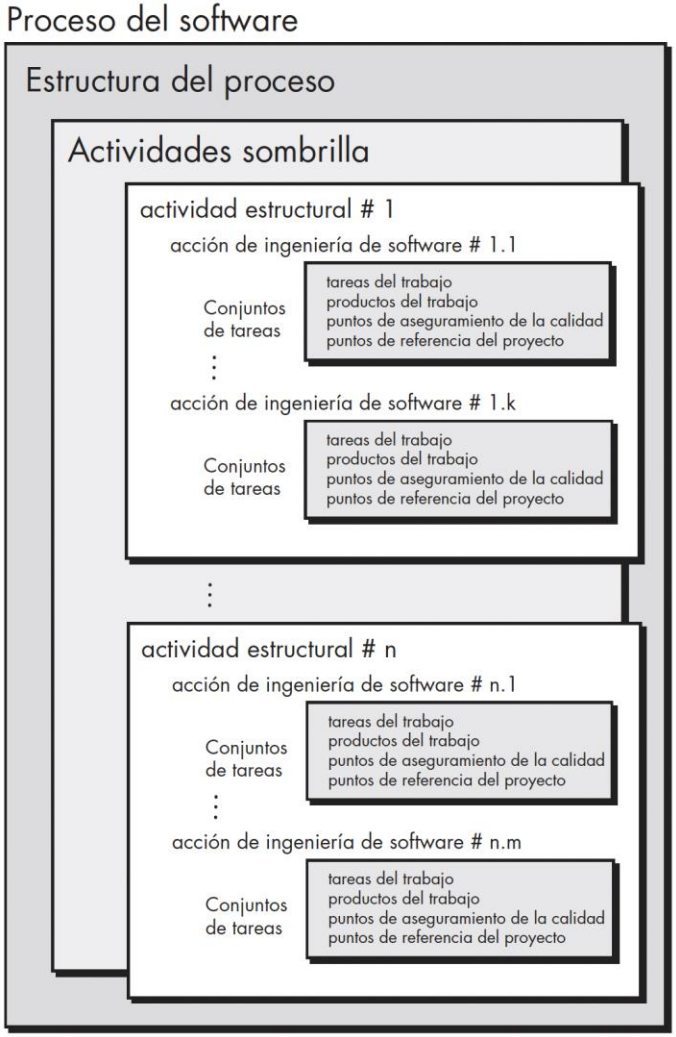
Procesos organizacionales

- Gestión: Actividades de gestión de la organización, incluyendo también la gestión de proyectos.
- Infraestructura: Actividades necesarias para que otros procesos del ciclo de vida puedan llevarse a cabo, incluyendo el capital y el personal
- Mejora: Actividades para mejorar la capacidad de los demás procesos
- Formación: Actividades para capacitar y desarrollar al personal

El Proceso: Modelos de Desarrollo

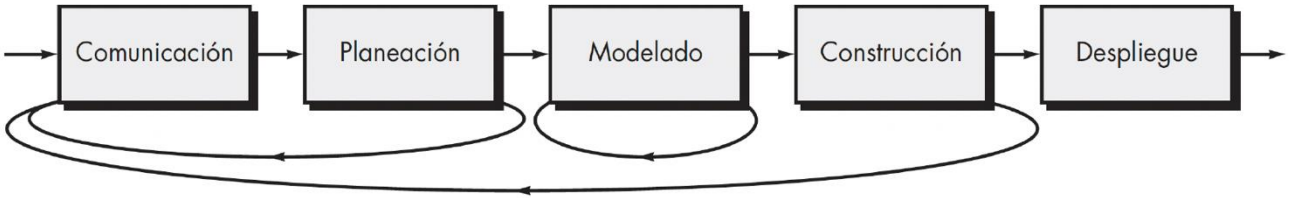
Proceso de desarrollo de software

- Dividido en actividades, acciones y tareas
- Cinco actividades principales: comunicación, planificación, modelado, construcción y despliegue
- Actividades “paraguas”: gestión y control del proyecto, gestión del riesgo, aseguramiento de la calidad, gestión de la configuración, entre otras.
- En las tareas intervienen:
  - Productos o artefactos utilizados (entrada) o construidos (salida)
  - Aspectos de control de la calidad
  - Hitos de proyecto

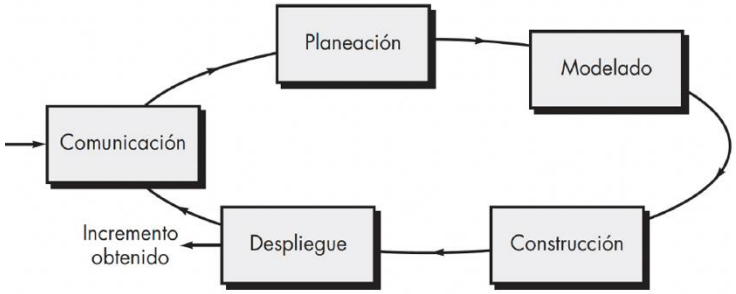


El Proceso: Modelos de Desarrollo

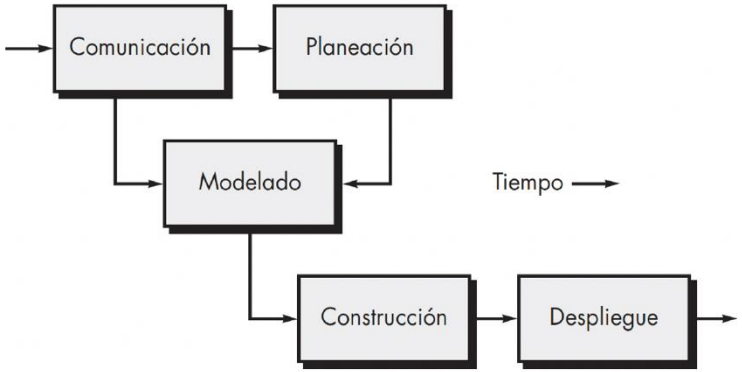
(a) Flujo del proceso lineal



(b) Flujo del proceso iterativo



(c) Flujo del proceso evolutivo



(d) Flujo del proceso paralelo

Índice

Paradigmas o Modelos de Desarrollo del Software

- Conjunto de técnicas y métodos organizativos que optimizan la creación de soluciones informáticas
- Estructuran equipos de trabajo para desarrollar funciones de forma eficiente
- Factores clave: costes, planificación, complejidad, recursos y lenguajes de programación
- Organizan el trabajo de manera ordenada y reducen la complejidad
- Sin una metodología clara, los proyectos pueden sufrir retrasos, errores y baja calidad
- Una metodología adecuada agiliza el proceso y mejora el resultado final

Paradigmas o Modelos de Desarrollo del Software

Metodologías Tradicionales:

- Definen requisitos de forma rígida al inicio
- Enfoque lineal y secuencial, sin retrocesos
- Dificultad para adaptarse a cambios

Metodologías Ágiles:

- Enfoque incremental, añadiendo funcionalidades en ciclos cortos
- Equipos autosuficientes y colaboración constante
- El cliente participa en el proceso, ajustando requisitos durante el desarrollo

Paradigmas o Modelos de Desarrollo del Software

Índice

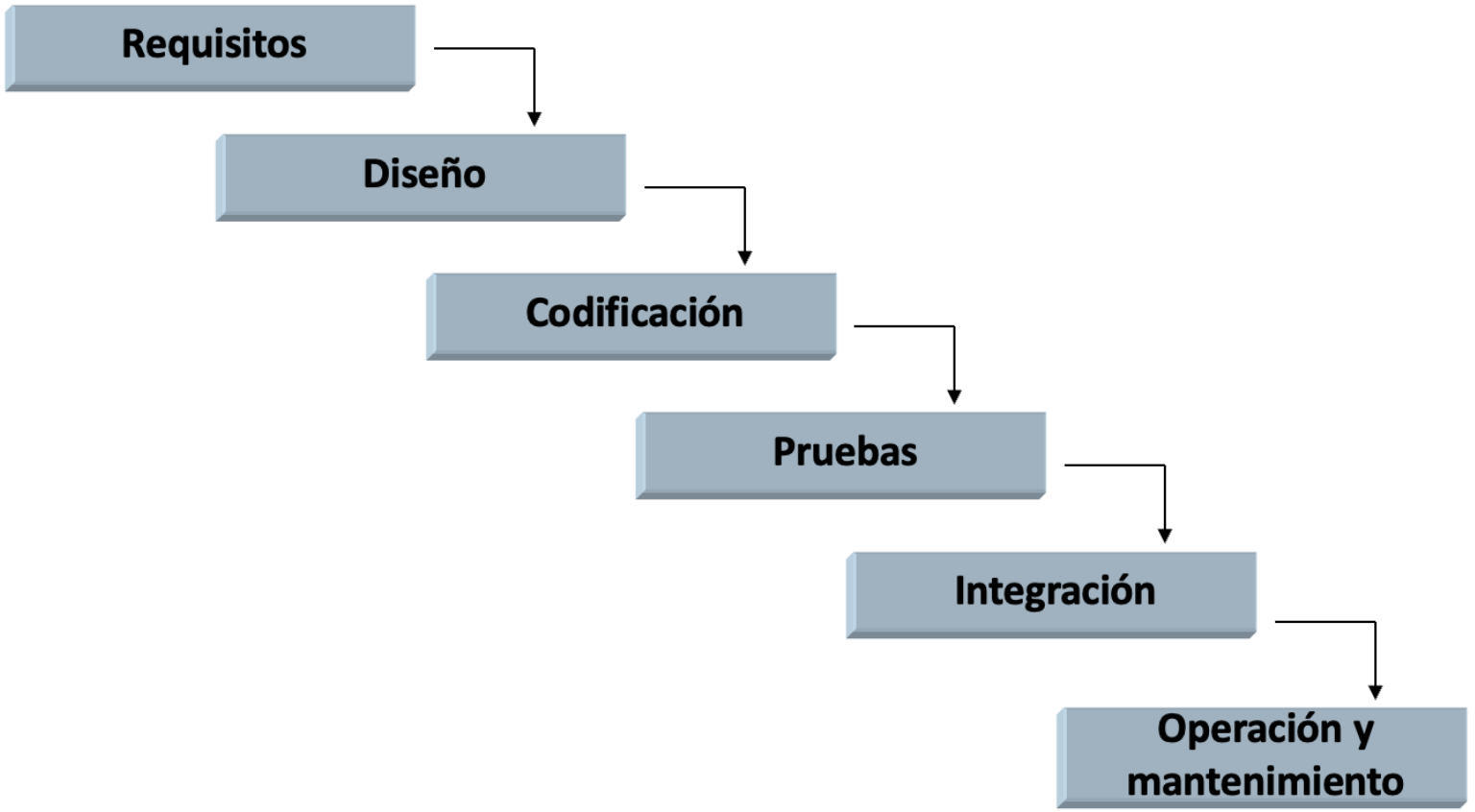
Modelos Tradicionales de Desarrollo de Software

Modelos tradicionales:

- Lineal o secuencial
- Cascada
- Espiral
- Incremental
- Evolutivo
- Prototipado
- Concurrencia

Modelos Tradicionales de Desarrollo de Software

Lineal o secuencial



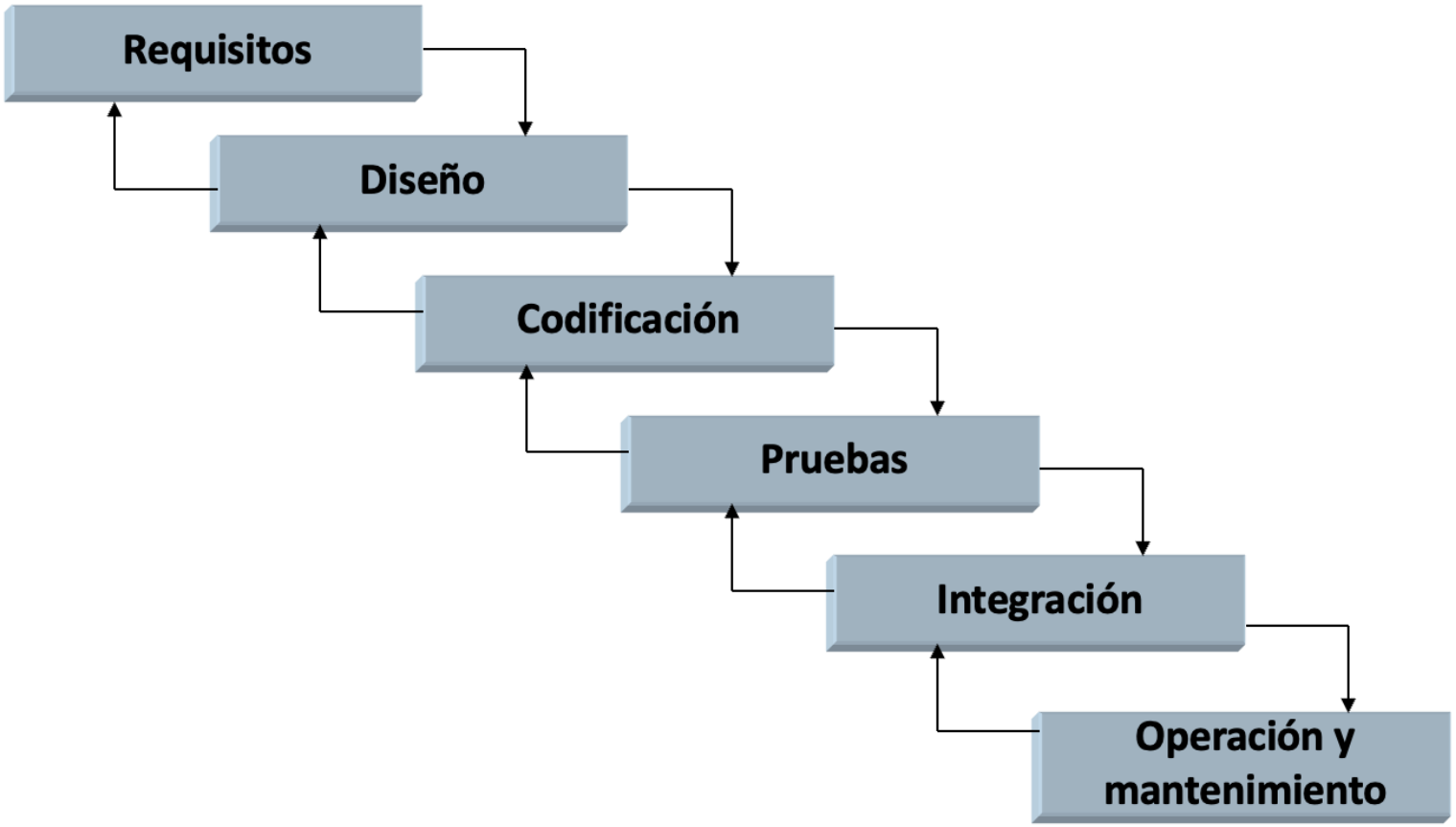
Modelos Tradicionales de Desarrollo de Software

Lineal o secuencial

- Desarrollo en etapas sucesivas: requisitos, diseño, codificación, pruebas e integración
- Se debe completar cada etapa antes de pasar a la siguiente
- Rígido: cada fase depende completamente del resultado anterior
- Problemas en la práctica: difícil obtener requisitos o diseños completos desde el inicio, lo que impide avanzar
- Apropiado para:
  - Nuevas versiones de sistemas con requisitos y entorno conocidos
  - Sistemas pequeños sin evolución a corto plazo

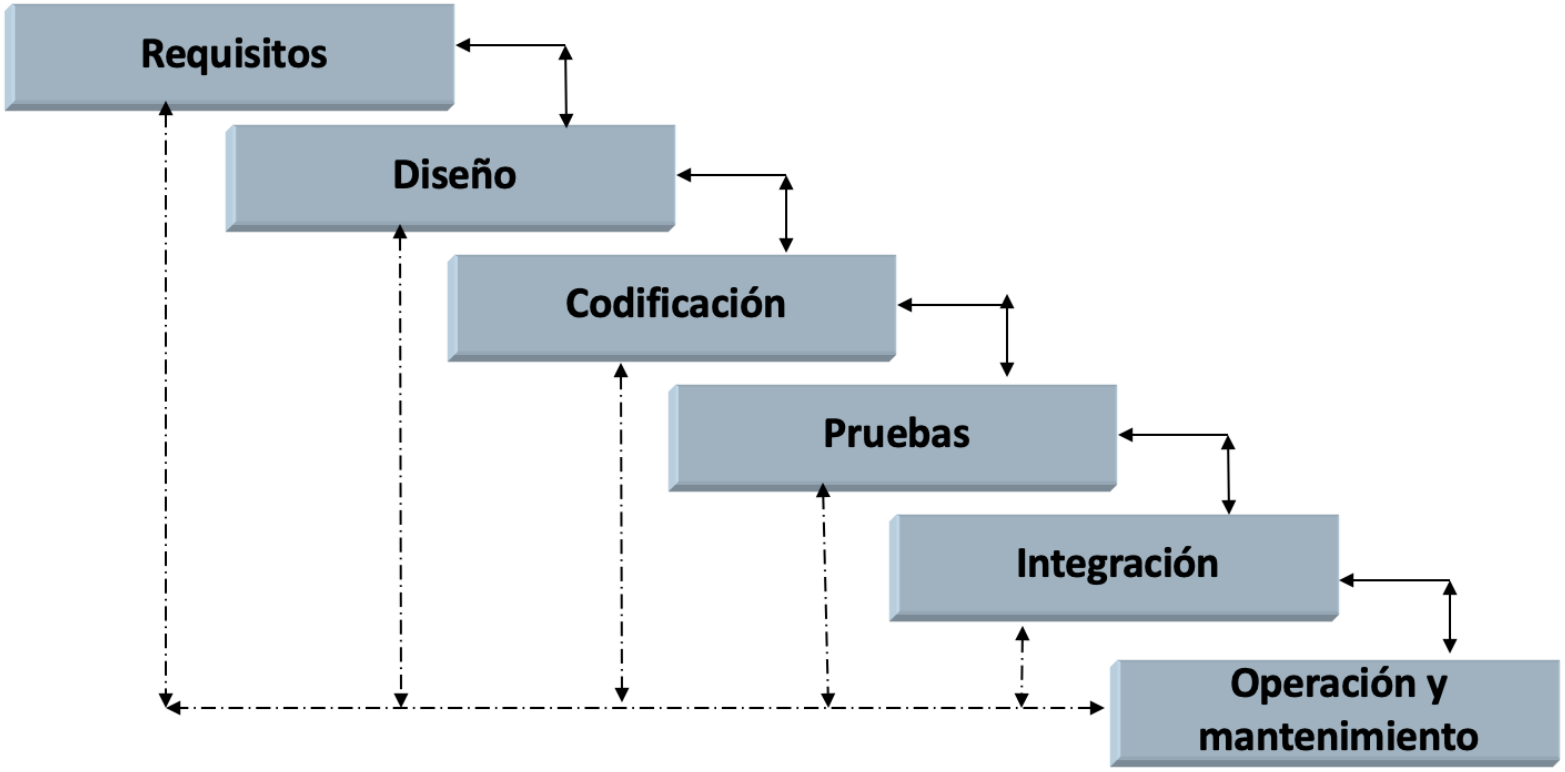
Modelos Tradicionales de Desarrollo de Software

Cascada



Modelos Tradicionales de Desarrollo de Software

Cascada



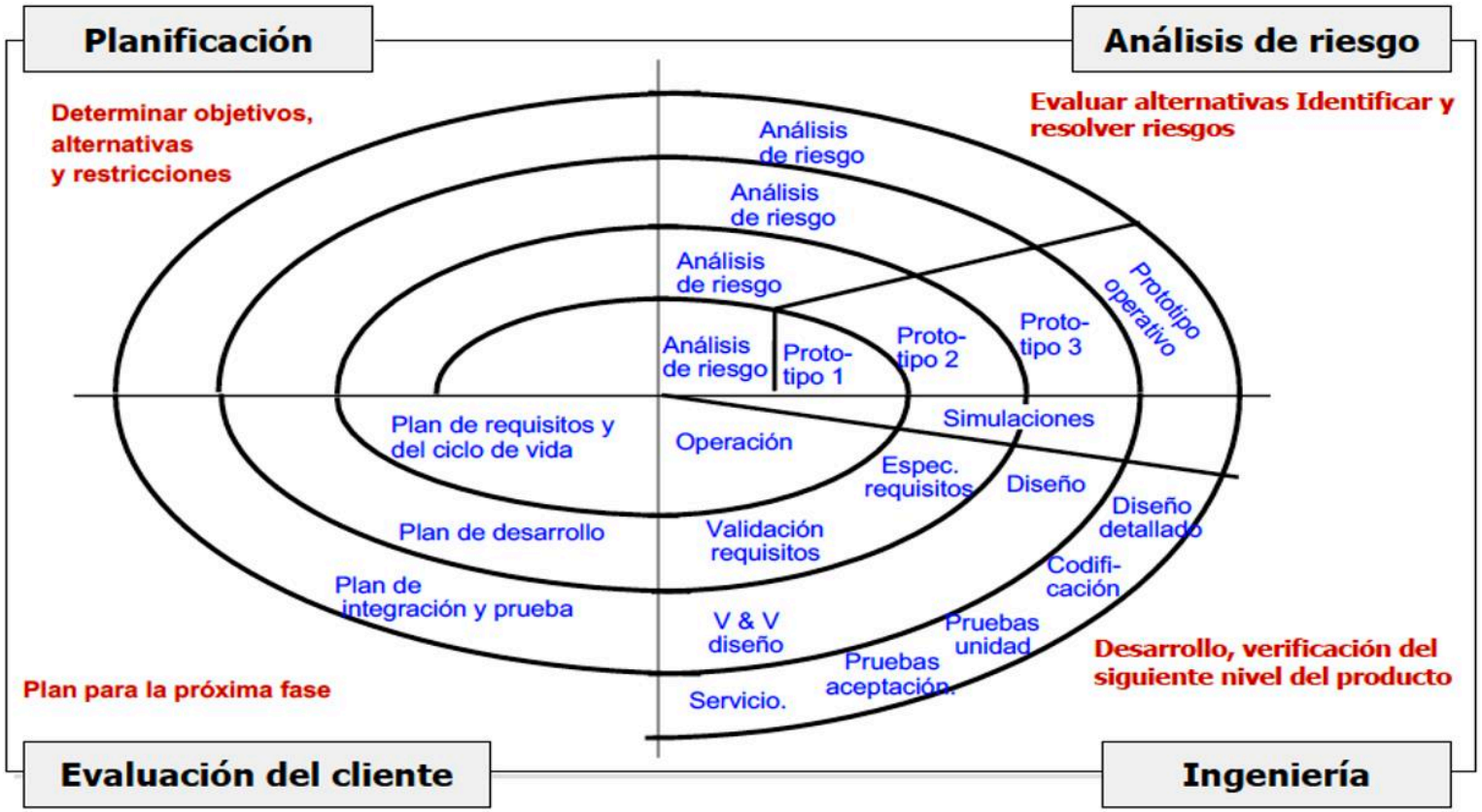
Modelos Tradicionales de Desarrollo de Software

Cascada

- Definido por Winston Royce en 1970, introduce flujos de retorno sobre el modelo secuencial
- Refleja la necesidad de regresar a fases anteriores con información nueva durante el desarrollo
- Representaciones comunes:
  - Una indica retorno solo a la fase anterior
  - Otra muestra que se puede retornar a cualquier fase
- Características:
  - Reconoce la importancia de requisitos y diseño previos antes de la codificación
  - La dificultad de obtener documentación completa puede bloquear el avance a la siguiente fase
- No muy popular debido a la tentación de comenzar diseño o codificación sin conocer bien los requisitos
- Apropiado para:
  - Nuevas versiones de sistemas con requisitos y entorno conocidos
  - Sistemas pequeños sin evolución a corto plazo

Modelos Tradicionales de Desarrollo de Software

Espiral



Modelos Tradicionales de Desarrollo de Software

Espiral

- Definido por Boehm en 1988, presenta un desarrollo evolutivo, en contraste a la linealidad de los anteriores.
- Introduce el "análisis de riego" para guiar la evolución del proceso de desarrollo
- Representación en espiral con cuatro cuadrantes
- Cada ciclo aborda una parte del desarrollo total, avanzando a través de las actividades de cada cuadrante
- Actividades clave:
  - **Planificación:** Establece el contexto y decide qué parte se abordará en el siguiente ciclo
  - **Análisis de riesgo:** Evalúa alternativas y selecciona la más ventajosa, previendo posibles riesgos
  - **Ingeniería:** Incluye análisis, diseño, codificación, etc., similar a modelos lineales
  - **Evaluación:** Analizan resultados de la fase de ingeniería y sirven como base para la siguiente fase

Modelos Tradicionales de Desarrollo de Software

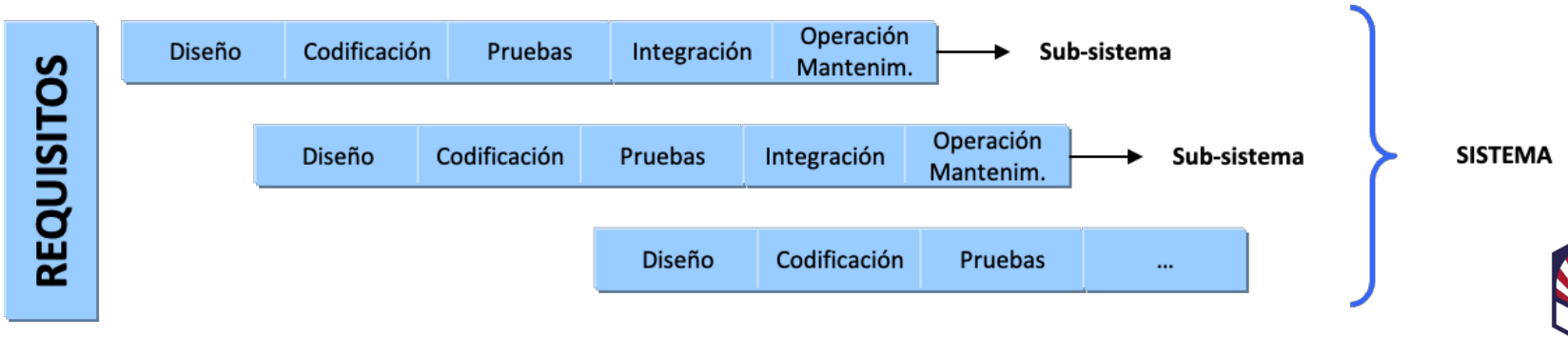
Espiral

- Flexibilidad del Modelo:
  - Permite múltiples combinaciones en cada ciclo, eligiendo el avance a ejecutar (requisitos, diseño, codificación, subsistemas completos)
  - Dependiendo de las combinaciones, el desarrollo en espiral puede parecerse a otros modelos:
    - Modelo en cascada: Si cada ciclo sigue una fase secuencialmente
    - Modelo incremental: Si se desarrollan partes del sistema global
- Distinción Clave:
  - Si se decide seguir fases de cascada de forma secuencial, se está en un modelo en cascada
  - Si se desarrollan partes del sistema, se opta por un ciclo de vida incremental
  - Si se da un pequeño paso, se evalúa, se planifica el siguiente y se analizan riesgos, se sigue un modelo en espiral

Modelos Tradicionales de Desarrollo de Software

Incremental

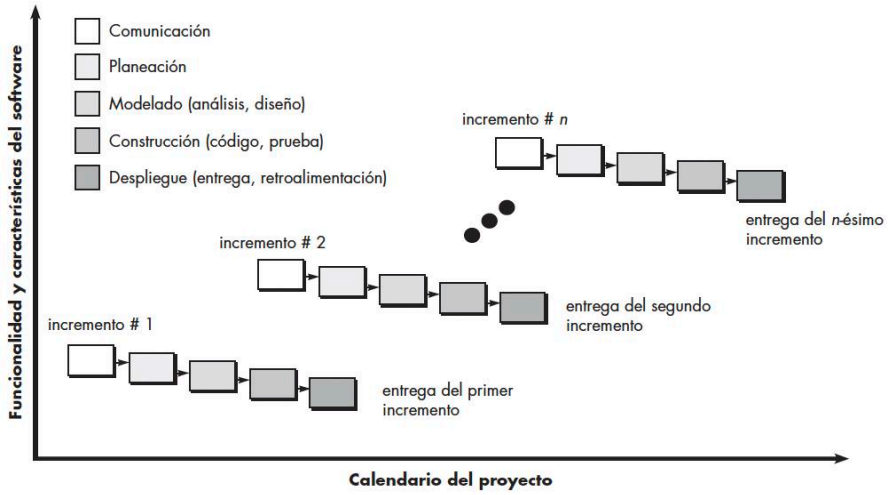
- Mitiga la rigidez del modelo en cascada al descomponer el desarrollo en partes
- Cada parte se desarrolla mediante un ciclo de desarrollo (representado en cascada)
- Ventajas:
  - Subsistemas operativos: Permiten al usuario perfilar mejor las necesidades del sistema completo
  - Entregas parciales: Se producen en periodos cortos, facilitando la incorporación de nuevos requisitos que pueden no estar disponibles al inicio del desarrollo



Modelos Tradicionales de Desarrollo de Software

Incremental

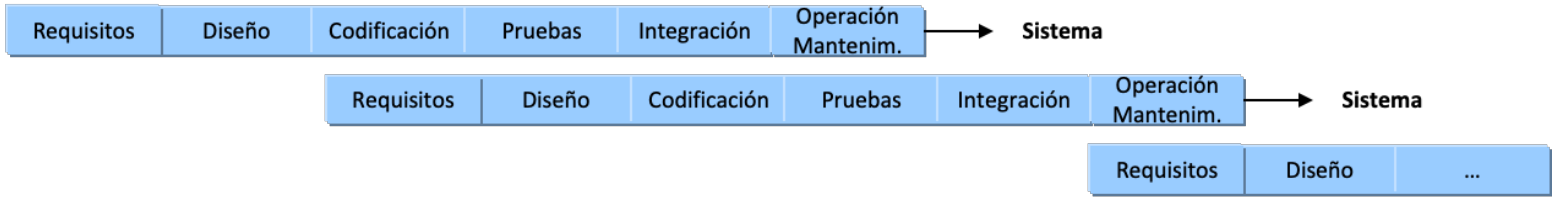
- Los desarrollos de cada subsistema o versión pueden solaparse en el tiempo
- La funcionalidad del primer incremento se denomina producto “núcleo” (*core product*)
- Apropiado para:
  - Sistemas donde el cliente necesita parte de la funcionalidad antes completar el sistema
  - Contextos donde es beneficioso obtener requisitos de forma escalonada a través de subsistemas



Modelos Tradicionales de Desarrollo de Software

Evolutivo

- Compuesto por varios ciclos de desarrollo, cada uno produce un sistema completo para operar en el entorno real.
- La información acumulada mejora o amplía los requisitos y el diseño del siguiente ciclo
- Representa un ciclo de vida común a todos los sistemas, mejorando a través de versiones sucesivas
- Apropiado para:
  - Desconocimiento inicial de todas las necesidades operativas, especialmente en entornos nuevos
  - Necesidad de operar el sistema en plazos más cortos que los requeridos para un diseño exhaustivo
  - Desarrollo en entornos cambiantes (normas legislativas, mejora continua frente a la competencia)
  - Puede incluir desarrollos internos en cascada o en espiral



Modelos Tradicionales de Desarrollo de Software

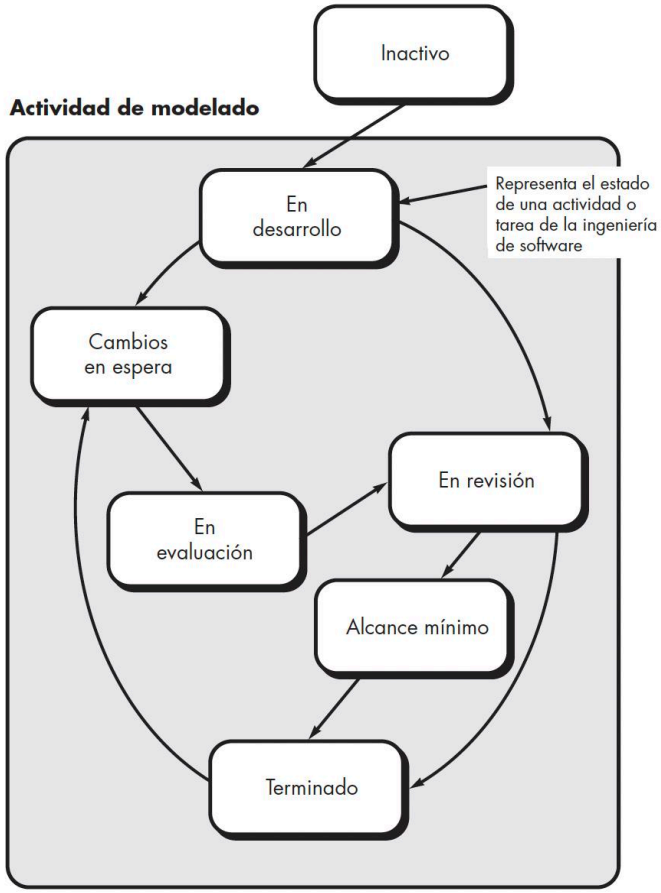
Prototipado

- Construcción de modelos de prueba para simular el funcionamiento del sistema
- Tipos de Prototipos:
  - Ligeros: Simulaciones visuales (dibujos de pantallas con enlaces)
  - Operativos: Módulos de software funcionales en entornos RAD (*Rapid Application Development*), sin cubrir todo el sistema
- Experimentar con un entorno similar al final para obtener retroalimentación del usuario o cliente y definir mejor los requisitos
- Riesgos:
  - El cliente puede pensar que gran parte del trabajo está hecho debido a un interfaz sofisticado
  - Los prototipos operativos pueden crecer fuera de la planificación, consumiendo más recursos
  - Prototipos ligeros desarrollados fuera del equipo pueden mostrar funcionalidades no implementables
  - El prototipo puede parecer más funcional que el sistema final debido a diferencias en entornos o funcionalidades incompletas

Modelos Tradicionales de Desarrollo de Software

Concurrencia

- Consiste en el solapamiento de actividades, cada una en un estado concreto
- Puede aportar beneficios o ser origen de problemas en la planificación del proyecto
- Factores a considerar:
  - Índice de concurrencia: Puede ser reducido (escaso flujo de modificaciones) o intensivo (problemas en planificación y distribución del trabajo)
  - Gestión de la concurrencia: Puede ser planificada o inducida por las circunstancias. La gestión adecuada maximiza los beneficios o minimiza los perjuicios en los planes y la calidad del proyecto



Índice

Proceso Unificado de Desarrollo

- Marco de desarrollo de software que se caracteriza por estar **dirigido por casos de uso**, centrado en la **arquitectura** y por ser **iterativo e incremental**
- Propuesto por la empresa *Rational Software* (ahora parte de IBM)
- Integra enfoques tradicionales y ágiles, ubicándose en un punto intermedio entre ambos
- Características:
  - Configurabilidad: Adaptable a diferentes contextos (intranet empresarial, software militar).
  - Ciclos cortos: Permite adaptación continua.
  - Más rígido que metodologías ágiles puras (Scrum, XP), pero acepta retroalimentación y cambios durante el desarrollo.

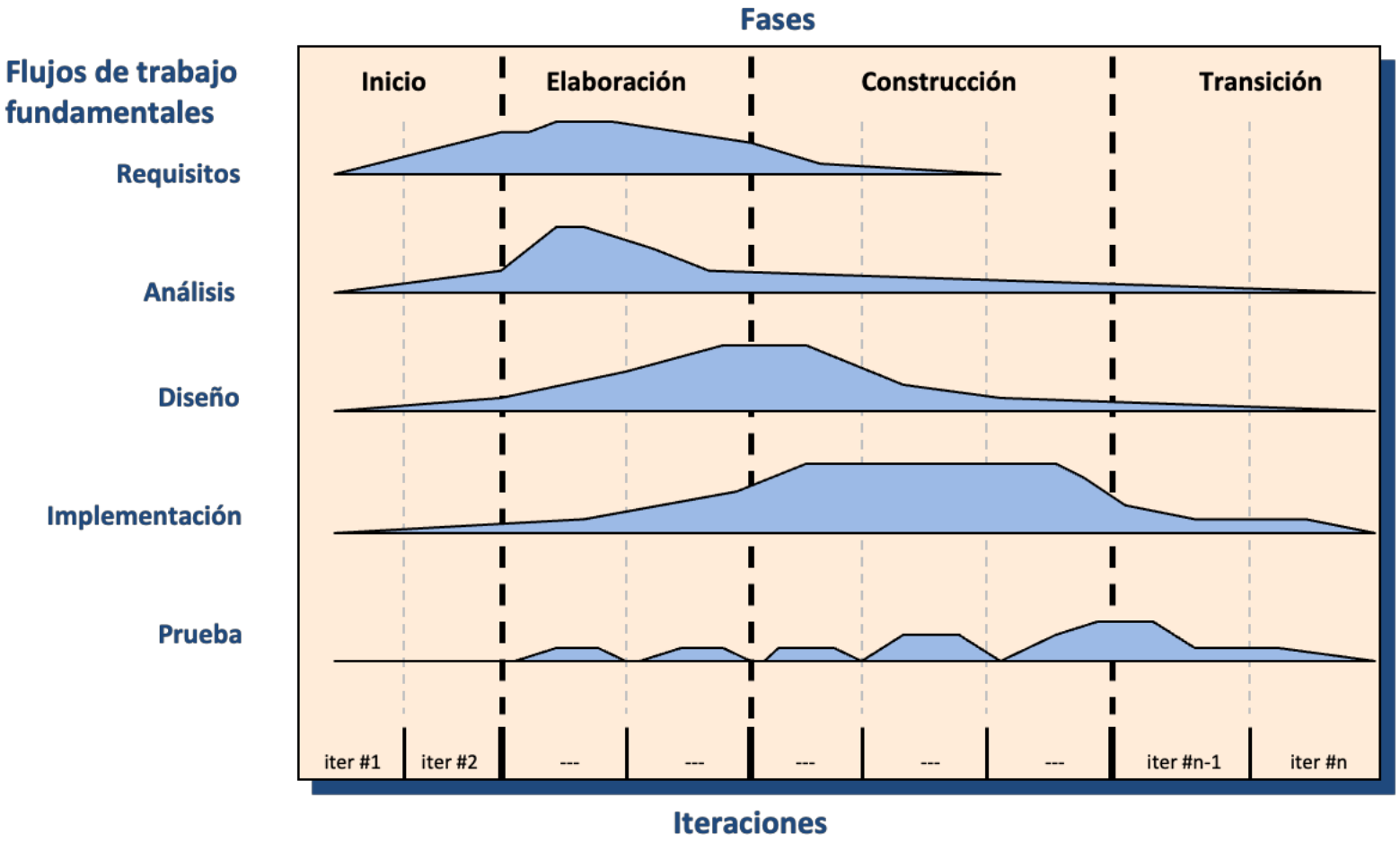
Proceso Unificado de Desarrollo

- Prácticas Fundamentas del Proceso Unificado:
  1. Desarrollo iterativo e incremental
  2. Gestión de los requisitos (mediante casos de uso)
  3. Arquitecturas basadas en componentes (subsistemas con funciones claras)
  4. Uso de modelos visuales (UML)
  5. Verificación de la calidad del software en todas las etapas
  6. Control de los cambios al software
- Estructura del Proceso:
  - **Descomposición temporal:** Organizado en fases
  - **Descomposición de actividades:** Desglosado en tareas específicas
  - Define roles y tareas para cada persona involucrada en el proyecto

Proceso Unificado de Desarrollo

- Ciclos del Proceso Unificado (UP), se componen en cuatro fases:
  1. **Inicio:** Definición del producto y análisis de negocio. Identificación de funciones principales, usuarios, arquitectura y plan del proyecto.
  2. **Elaboración:** Detalle de casos de uso y diseño de la arquitectura. Se planifican actividades y estiman recursos.
  3. **Construcción:** Desarrollo del producto integrando software en la arquitectura. Se completan los casos de uso, aunque pueden existir defectos.
  4. **Transición:** Conversión a versión beta, pruebas de usuario, corrección de defectos, formación y soporte al usuario.
- Cada fase se divide a su vez en iteraciones

Proceso Unificado de Desarrollo



Índice

Modelos Ágiles de Desarrollo de Software

- Basados en el "Manifiesto Ágil de Software" (2001)
- Firmado por 16 notables desarrolladores, escritores y consultores
- Enfocado en 4 valores principales:

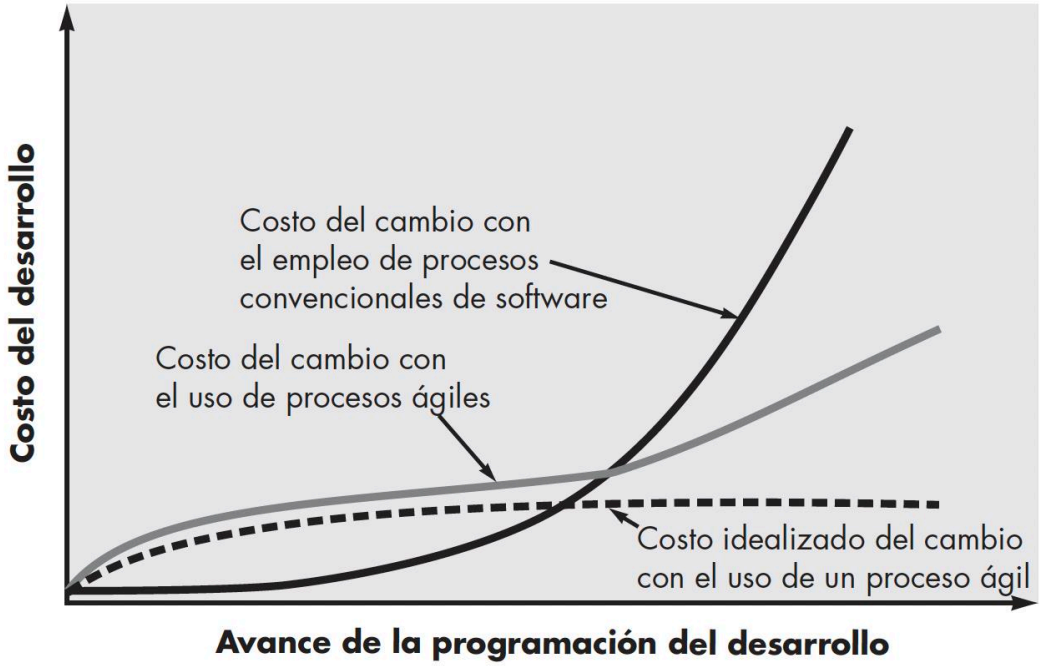
Individuos e Interacciones	**sobre**	**Procesos y Herramientas**
Software que funciona	**sobre**	**Documentación exhaustiva**
Colaboración con el Cliente	**sobre**	**Negociación de Contratos**
Responder ante el cambio	**sobre**	**Seguimiento de un Plan**

Modelos Ágiles de Desarrollo de Software

- El manifiesto Ágil está basado en 12 principios, que son:
  1. Satisfacer al cliente a través de la entrega de valor
  2. Aceptamos que los requisitos cambien
  3. Entregamos software funcional frecuente
  4. La gente del negocio y los desarrolladores trabajamos juntos diariamente
  5. Los proyectos se hacen en entornos de individuos motivados
  6. Las comunicaciones cara a cara
  7. El software funcionando es la medida principal de progreso
  8. Promover un paso sostenido
  9. Excelencia técnica y buen diseño
  10. La simplicidad es esencial
  11. Equipos auto-organizados
  12. Inspeccionar y adaptar

Modelos Ágiles de Desarrollo de Software

- Ideal para aplicarse en situaciones donde:
  - **Requisitos del software** son difíciles de predecir o cambian con frecuencia (cambios en las prioridades del cliente)
  - Se necesita **intercalar diseño y construcción** de software
  - **Análisis, diseño y construcción** no siguen una planificación predecible



Modelos Ágiles de Desarrollo de Software

- La Ingeniería del Software Ágil combina una filosofía y directrices de desarrollo:
  - Satisfacción del cliente y la entrega temprana del software incremental
  - Equipos pequeños y con alta motivación
  - Métodos informales
  - Mismos productos de trabajo
  - Simplicidad general del desarrollo
- Relevante hoy por la aceleración y cambio en los sistemas basados en computadoras
- Alternativa eficaz a la ingeniería tradicional para ciertos tipos de proyectos
- Diferentes metodologías Ágiles:
  - Scrum
  - Programación extrema XP
  - Kanban

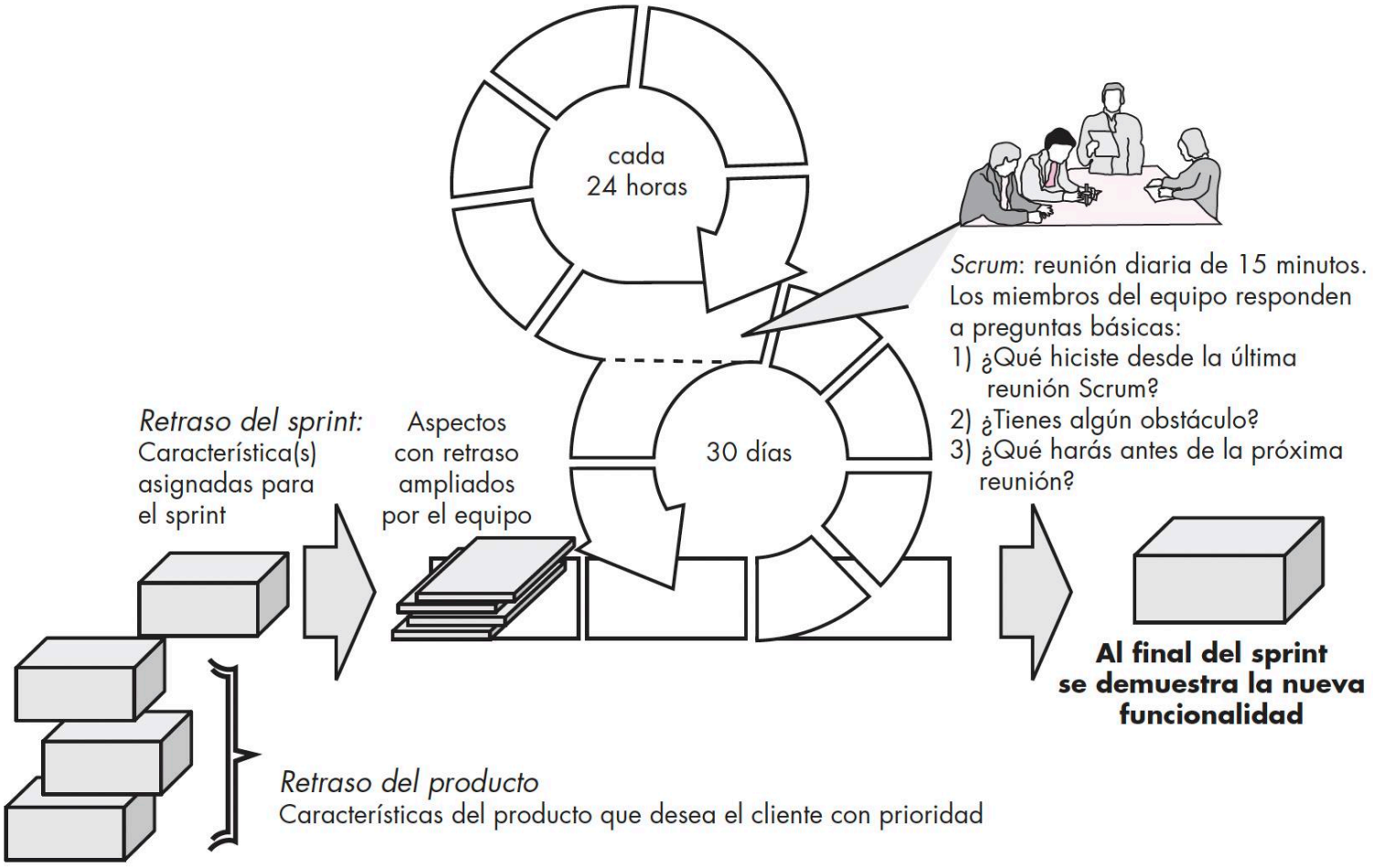
Modelos Ágiles de Desarrollo de Software

SCRUM

- Método ágil creado por Jeff Sutherland en los años 90
- Desarrollado y ampliado por Schwaber y Beedle
- Principios alineados con el Manifiesto Ágil
- Proceso de análisis estructurado en fases:
  - Requerimientos
  - Análisis
  - Diseño
  - Evolución
  - Entrega
- Las tareas en cada fase se organizan en ciclos de trabajo llamados ***sprints***
- Cada ***sprint*** se ajusta y modifica en tiempo real, según la complejidad del proyecto y las necesidades del equipo Scrum

Modelos Ágiles de Desarrollo de Software

SCRUM



Modelos Ágiles de Desarrollo de Software

SCRUM

ROLES	• Scrum Master • Product owner • Team
ARTEFACTOS	• Product backlog • Sprint backlog • Release burndown chart • Sprint burndown
REUNIONES	• Sprint planning meeting • Daily scrum • Sprint review meeting • Sprint retrospective

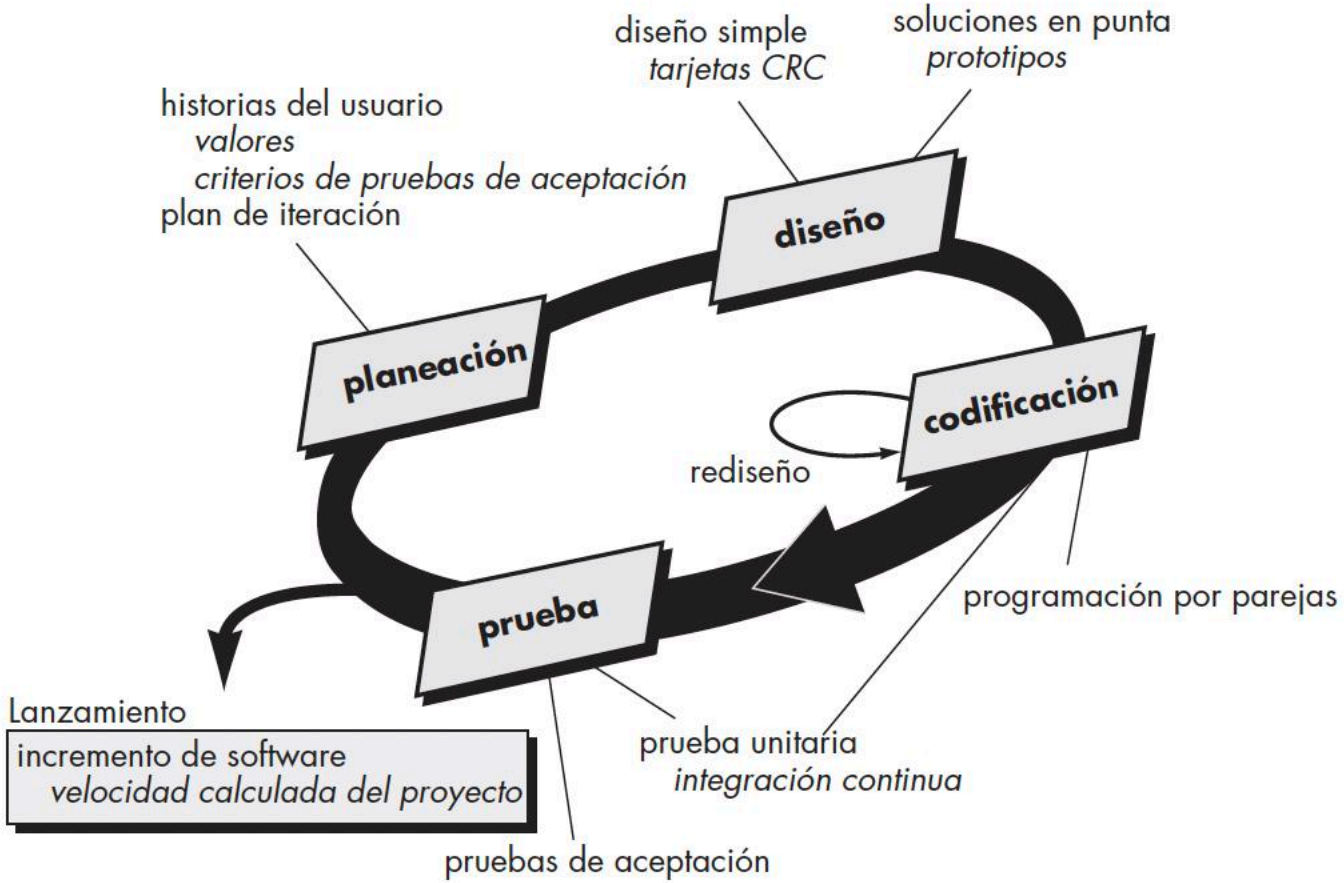
Modelos Ágiles de Desarrollo de Software

Programación Extrema XP

- Metodología ágil que mejora la calidad del software y responde rápidamente a cambios en los requisitos del cliente
- Fomenta la colaboración
- Enfoque en objetos y estructurada en cuatro actividades clave:
  - **Planeación:** Comunicación continua para comprender y ajustar los requisitos del cliente
  - **Diseño:** Diseño evolutivo, flexible y sencillo, evitando complejidad innecesaria
  - **Codificación:** Código claro con programación en parejas para asegurar calidad
  - **Pruebas:** Pruebas automatizadas y desarrollo basado en pruebas (TDD) para validar cada funcionalidad

Modelos Ágiles de Desarrollo de Software

Programación Extrema XP



Modelos Ágiles de Desarrollo de Software

Kanban

- Método ágil de gestión de proyectos y de flujo de trabajo, originado en Toyota
- Objetivo: Mejorar la eficiencia y la productividad visualizando tareas y gestionando el flujo de trabajo de manera continua
- A diferencia de Scrum, Kanban es más flexible y no requiere iteraciones o *sprints* predefinidos, enfocándose en una entrega continua
- Principios básicos:
    1. Visualizar el flujo de trabajo con un tablero dividido en columnas ("Por hacer" → "Haciendo" → "Hecho")
    2. Limitar la cantidad de trabajo en progreso (WIP) en cada etapa, completando tareas antes de iniciar nuevas
    3. Gestionar el flujo de trabajo para mantener una entrega continua
    4. Hacer explicitas las políticas del proceso
    5. Mejora continua mediante retroalimentación
    6. Cambios colaborativos en el proceso, involucrando a todo el equipo y partes interesadas

Modelos Ágiles de Desarrollo de Software

Tablero de Kanban