Procesos de software

Modelos, actividades, productos y roles

¿Qué es un proceso de desarrollo de software?

Un proceso de desarrollo de software es una serie de actividades relacionadas que conduce a la elaboración de un producto de software.

Se clasifican como dirigidos por un plan o como procesos ágiles. Cada enfoque es adecuado para diferentes tipos de software. Por lo general, uno necesita encontrar un equilibrio.

Actividades fundamentales

1. **Especificación del software** (requerimientos funcionales y no funcionales).
2. **Diseño** e **implementación** del software
3. **Validación** del software
4. **Mantenimiento** del software (mantenimiento evolutivo y correctivo)

Los procesos describen **actividades**, el **orden** de dichas actividades y deben incluir:

Productos: resultados de una actividad del proceso.

Roles: Responsabilidades de la gente que interviene en el proceso.

Precondiciones y **postcondiciones**: Declaraciones válidas antes y después de que se realice una actividad del proceso.

Entonces una definición nuestra de proceso de desarrollo de software

Un proceso de desarrollo de software es una descripción de tareas, sus inputs y sus outputs (producto, artefacto), y cómo están ordenadas. Puedo tener varias tareas, pero cómo mínimo debe contar con 4, las fundamentales. Además de definir las tareas y su orden, también debemos describir los roles y las pre y post condiciones de cada tarea.

Modelos de procesos de desarrollo de software

Un modelo de proceso de software es una representación simplificada de este proceso (ofrece información parcial de dicho proceso)

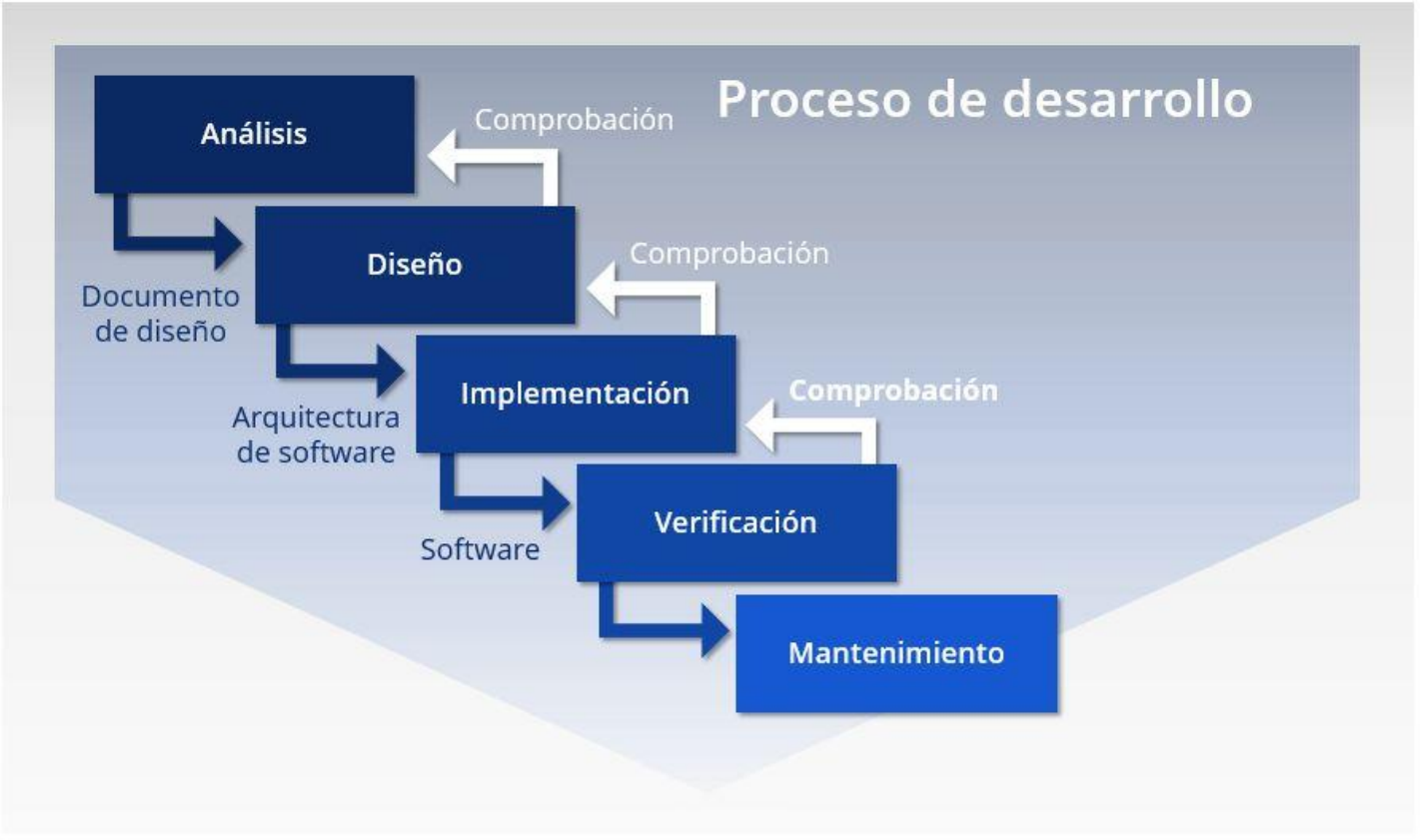
Los modelos de proceso muy generales (“paradigmas de proceso”) se muestran desde una perspectiva arquitectónica.

Se ve el marco (framework) del proceso, pero no los detalles de las actividades específicas.

Paradigmas de procesos

1. Modelo en cascada
2. Desarrollo incremental
3. Orientado a la reutilización

01. Modelo en cascada



Características del modelo en cascada

Cuándo usar

Debe usarse cuando los requerimientos se entiendan bien y sea improbable el cambio radical durante el desarrollo del sistema. Ejemplo: Migración.

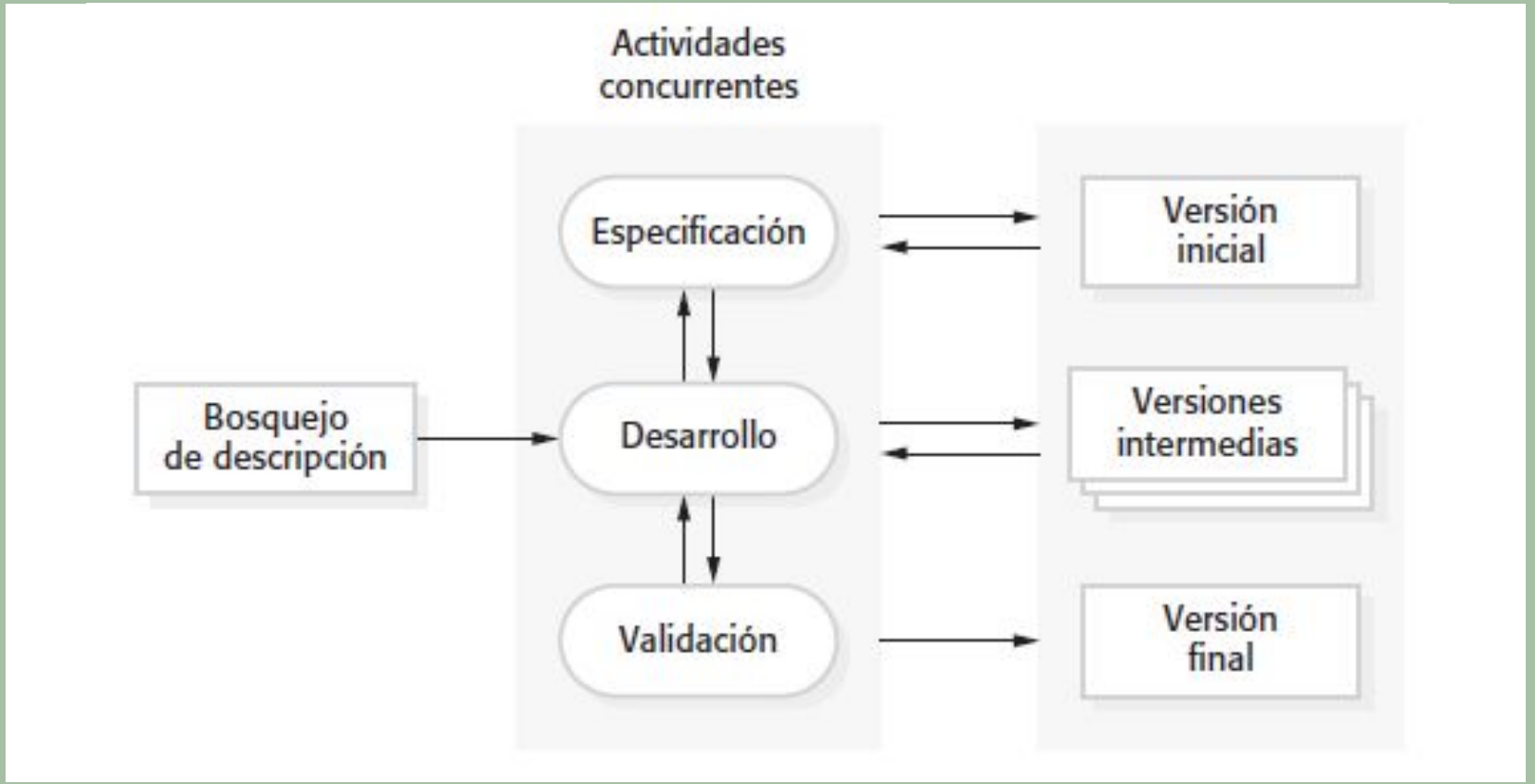
Actividades

1. Análisis
2. Diseño
3. Implementación
4. Verificación
5. Mantenimiento

Características

Proceso dirigido por un plan.
Planear todas las actividades antes de comenzar a trabajar.
La siguiente fase no comienza sino hasta que termine la previa.

02. Modelo incremental



Características del modelo incremental

Cuándo usar

Debe usarse cuando los requerimientos no se conocen al 100% y los cambios forman parte de la naturaleza del desarrollo.

Actividades

1. Descripción general
2. Especificación
3. Desarrollo
4. Validación

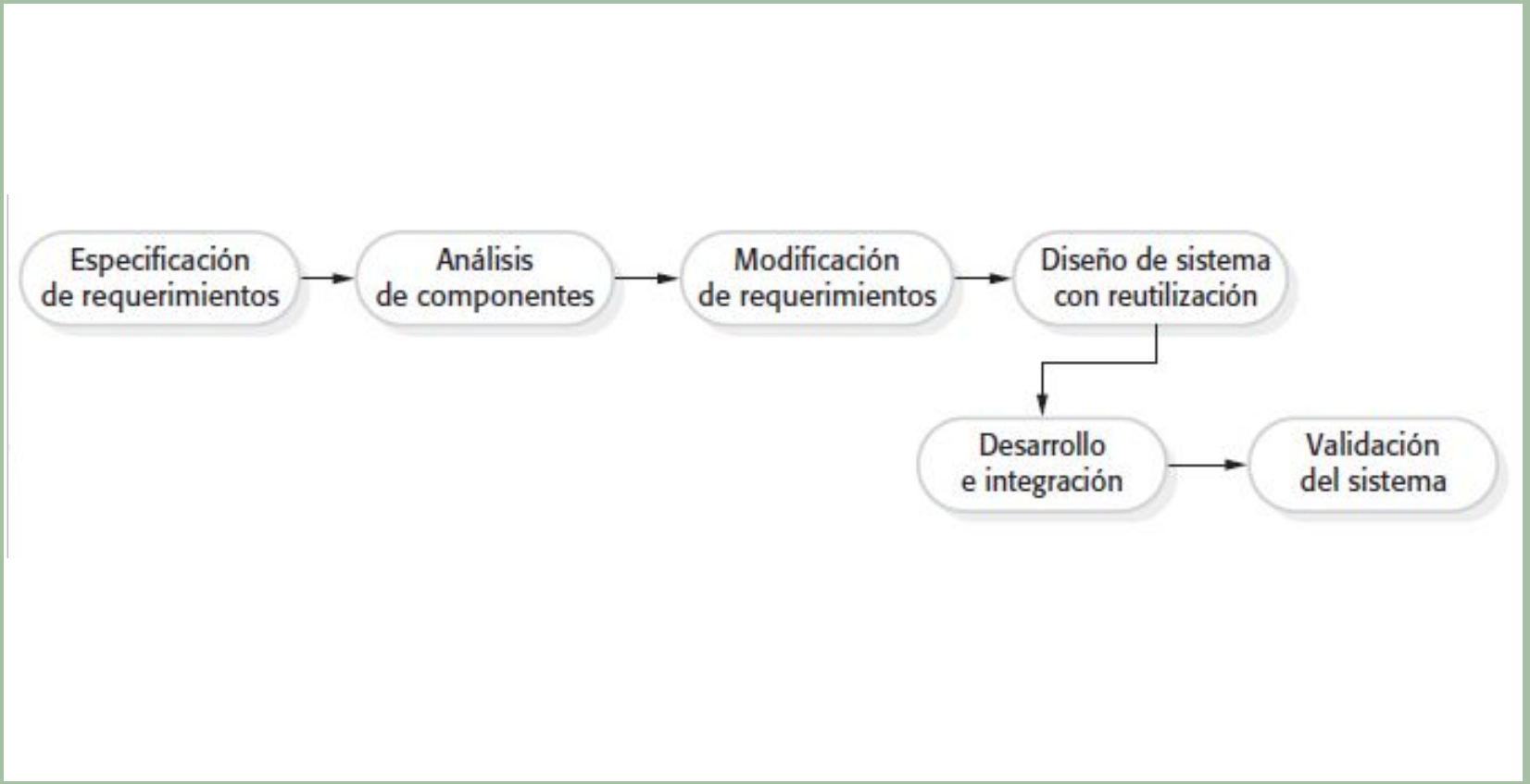
Características

Más barato modificar

Retroalimentación temprana

Entrega más rápida

02. Modelo orientado a la reutilización



Características del modelo orientado a la reutilización

Cuándo usar

Cuando los módulos del sistema estén bien claros y definidos.
Cuando son sistemas de integración definido por la propia naturaleza de los requerimientos.

Actividades

1. Especificación
2. Análisis de componentes
3. Modificación de requerimientos
4. Diseño
5. Desarrollo e integración
6. Validación

Características

Reducción de código a desarrollar.

Reducción de riesgos.

Módulos testeados.

Orientación a microservicios.

Procesos de desarrollo de software

1. Cascada
2. Proceso unificado de desarrollo
3. XP
4. SCRUM
5. KanBan

01. Cascada

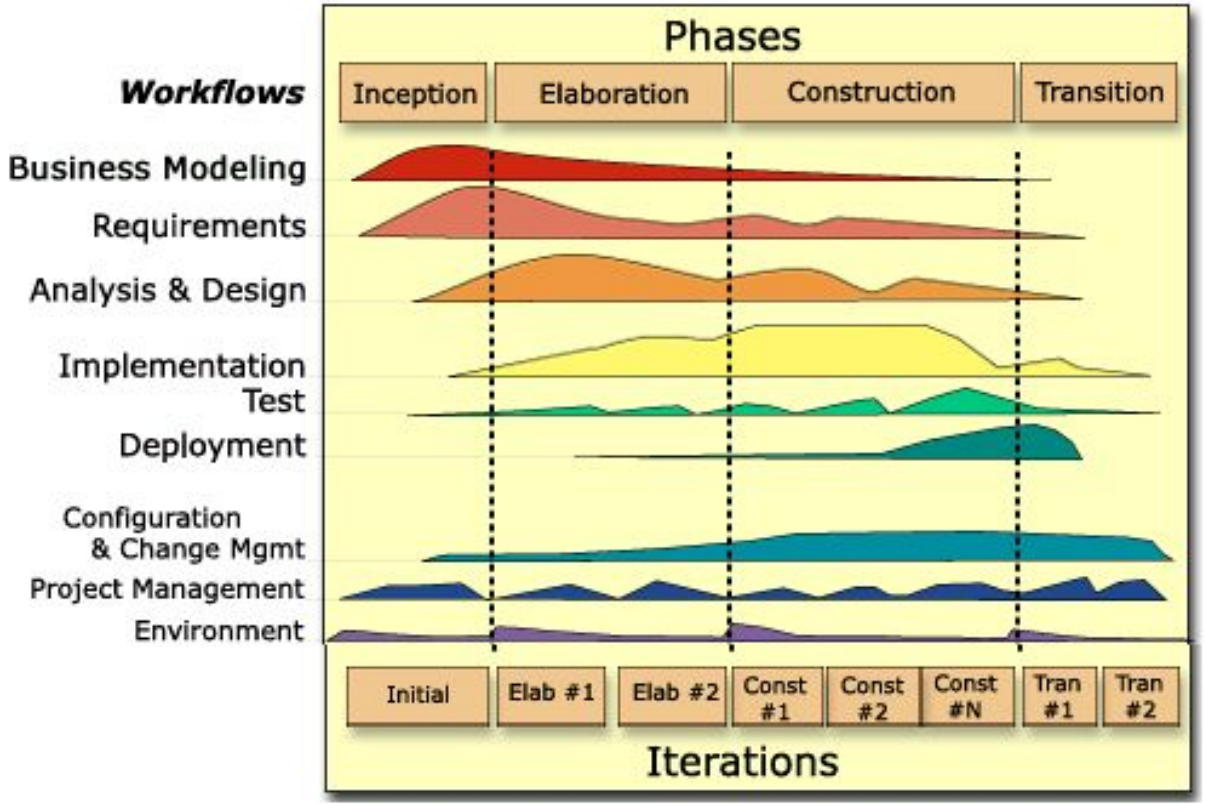
Proceso de desarrollo de software **guiado por un plan**, pertenece a la familia de las **metodologías de desarrollo tradicionales**.

Está basado en el modelo en cascada (en sí porque el proceso de desarrollo en cascada es el mismo modelo).

La versión original fue propuesta por Winston W. Royce en 1970 (recuerden que la crisis del software fue en 1969, recién ahí se empieza a hablar de Ing. de Software).

Actividades: Relevamiento => Análisis => Diseño => Codificación => Pruebas => Despliegue => Mantenimiento

02. Proceso unificado de desarrollo

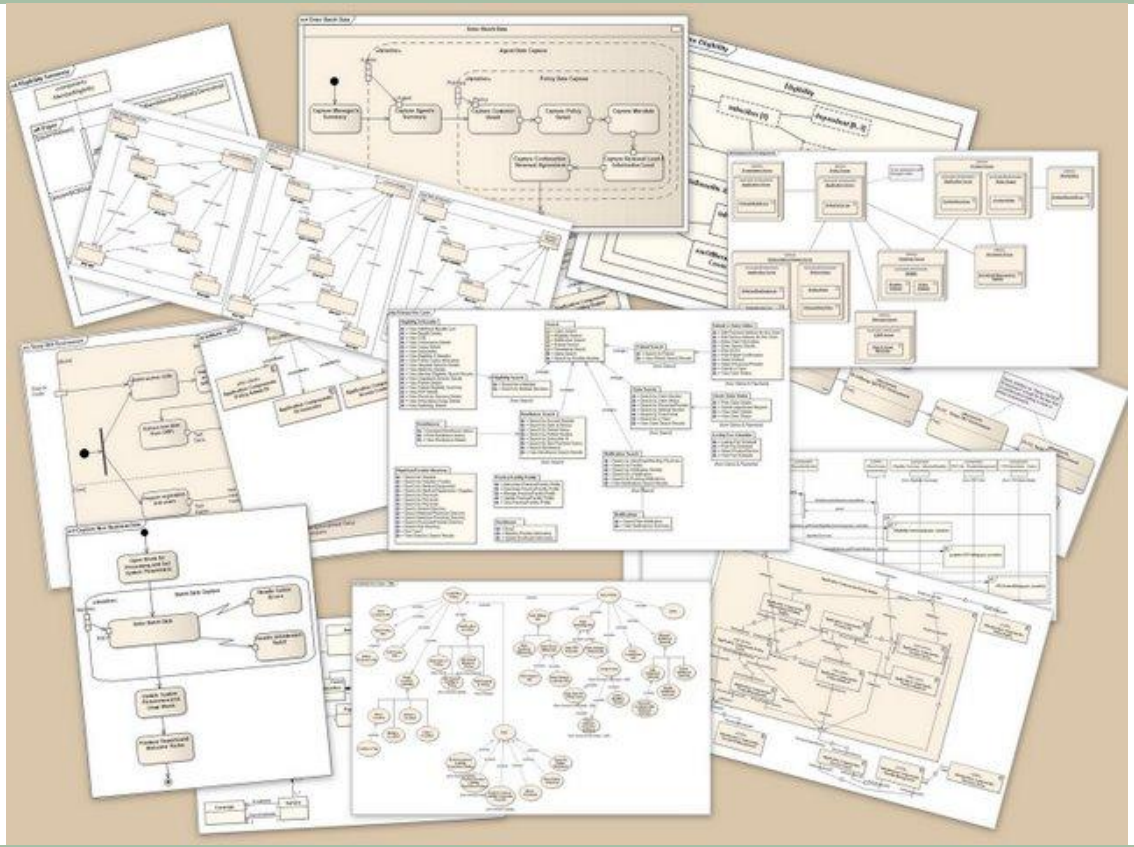


02. Proceso unificado de desarrollo

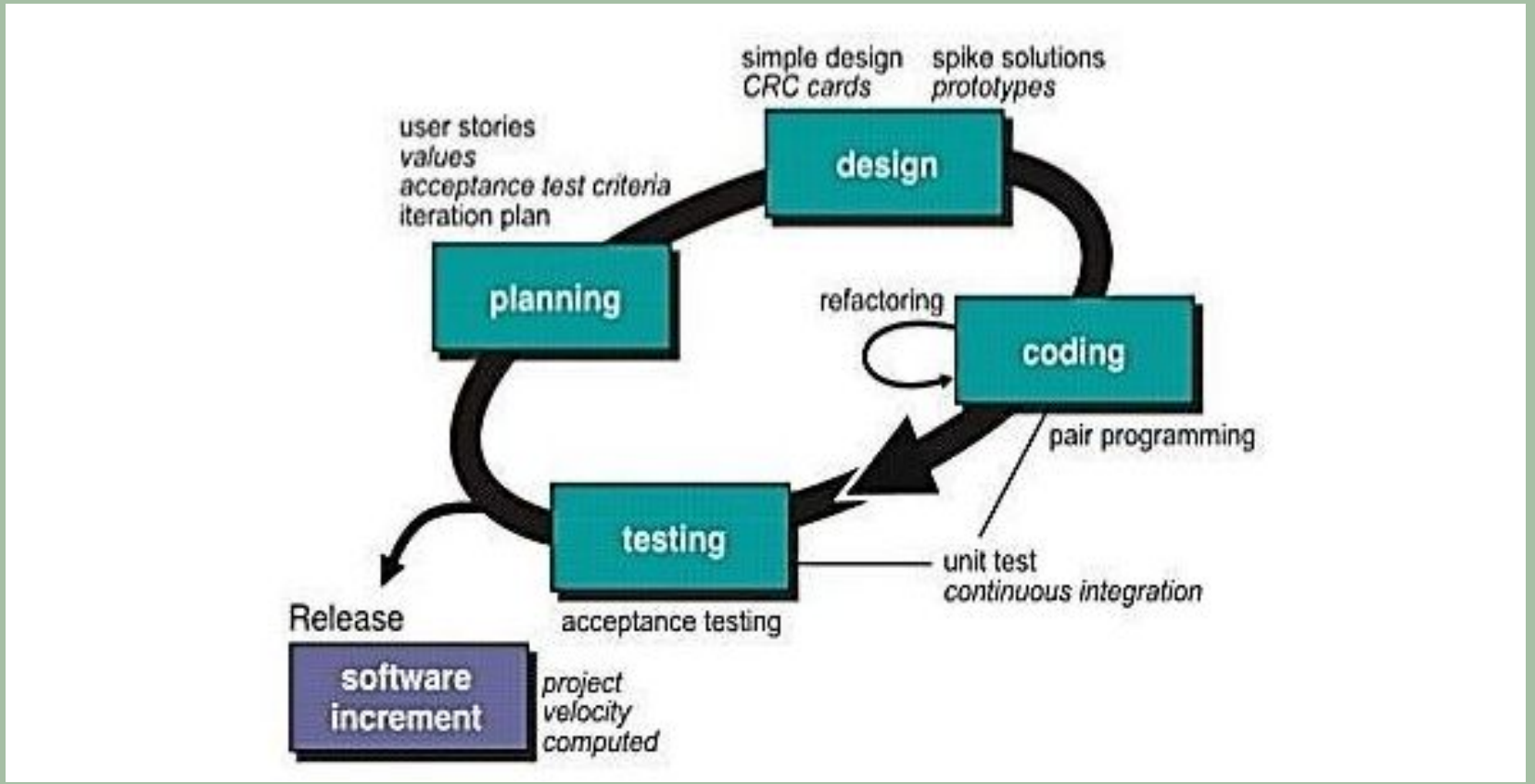
El Proceso Unificado de Desarrollo Software es un marco de desarrollo de software que se caracteriza por:

- Dirigido por casos de uso
- Centrado en la arquitectura
- Iterativo e incremental
- Utiliza el UML como lenguaje de representación visual.

02. Proceso unificado de desarrollo



03. XP (eXtreme Programming)



03. eXtreme Programming (XP)

Formulada por Kent Beck en 1996 y publicada formalmente en 1999 (Extreme Programming Explained: Embrace Change)

Valores

- ➢ Simplicidad => Refactoring/Clean Code
- ➢ Comunicación => Entre desarrolladores/Con clientes
- ➢ Retroalimentación
- ➢ Coraje o valentía => Agregar/Quitar/Modificar
- ➢ Respeto

03. eXtreme Programming (XP)

Características fundamentales

- Desarrollo iterativo e incremental
- Pruebas unitarias continuas
- Programación en parejas
- Integración del equipo de programación con el cliente
- Corrección de los errores antes de añadir funcionalidad
- Refactorización del código
- Propiedad del código compartida
- Simplicidad en el código

03. eXtreme Programming (XP)

Roles

- Programador
- Test developer
- Cliente
- Tester
- Tracker
- Entrenador (coach)
- Consultor
- Gestor

04. Scrum


SCRUM PROCESS
SCRUM MASTER
24
HOURS
REVIEW
IMPLEMENTATION
RETROSPECT
2-4 WEEKS
PRODUCT OWNER
SPRINT BACKLOG
PLANNING
DEFINITION OF DONE
USER STORIES

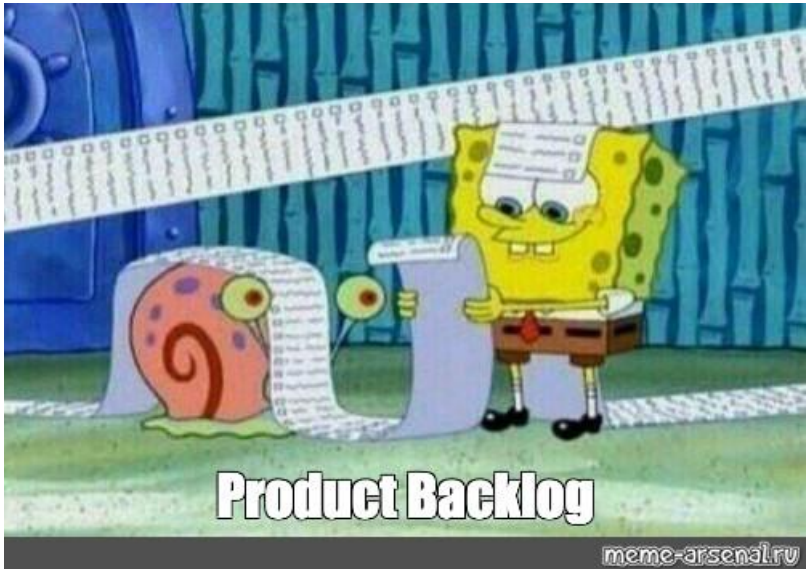
04. Scrum

Scrum es un marco que permite el trabajo **colaborativo** entre equipos.

Al igual que un equipo de rugby (de donde proviene su nombre) cuando entrena para un gran partido, scrum anima a los equipos a **aprender a través de las experiencias**, a **autoorganizarse** mientras aborda un problema y a **reflexionar** sobre sus **victorias** y **derrotas** para mejorar continuamente.

04. Scrum – Conceptos claves

Product Backlog: El backlog de un producto es una lista de trabajo ordenado por prioridades para el equipo de desarrollo que se obtiene de sus requisitos.



04. Scrum – Conceptos claves

Sprint Backlog: Registro de los requisitos desde el punto de vista de los desarrolladores. Es la lista de tareas que se deben realizar durante un sprint para lograr el incremento previsto.

04. Scrum – Conceptos claves

Sprint: Un sprint es un período breve de tiempo fijo en el que un equipo de scrum trabaja para completar una cantidad de trabajo establecida.

04. Scrum – Ceremonias

Sprint planning: Define qué se puede entregar en el sprint próximo y cómo se va a conseguir ese trabajo.

04. Scrum – Ceremonias

Daily: 15 minutos, todos los días, 3 preguntas.

04. Scrum – Ceremonias

Sprint review: Demostrar el duro trabajo de todo un equipo.

04. Scrum – Ceremonias

Retrospectiva del sprint: Al finalizar el sprint. Todos los miembros del equipo dejan sus impresiones sobre el sprint recién superado.

04. Scrum – Roles

Product Owner: Escribe las historias de usuario, las prioriza, y las coloca en el Product Backlog.

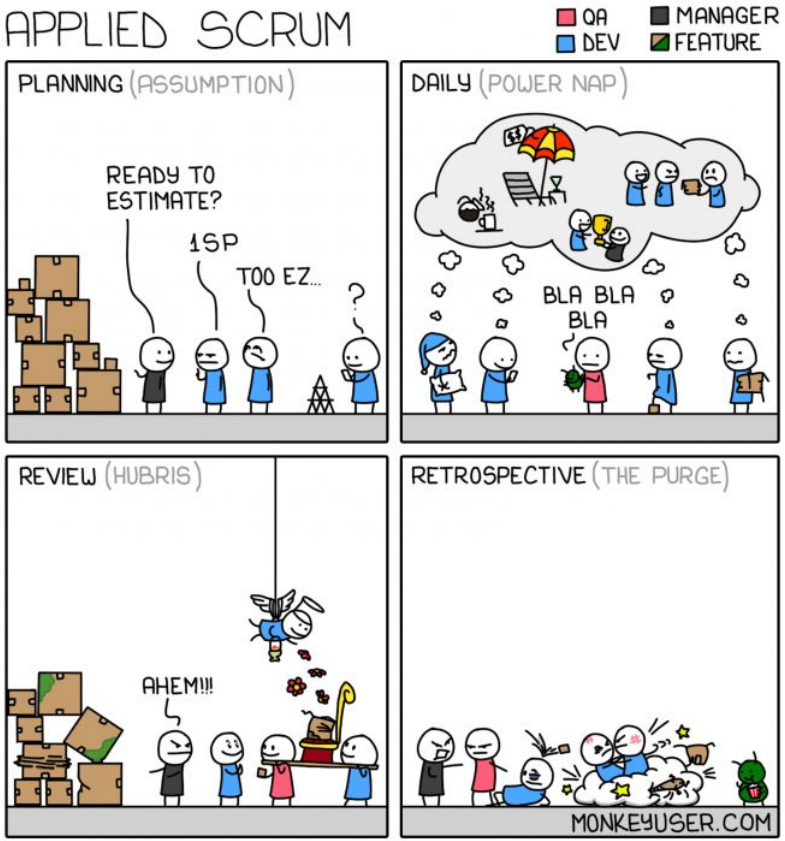
04. Scrum – Roles

Scrum Master: Facilitador. Asesora y da la formación necesaria al propietario del producto y al equipo de desarrolladores..

04. Scrum – Roles

Desarrollador: Cada uno de los profesionales que realizan la entrega del incremento de producto generado en cada sprint.

04. Scrum - Resumen en una imagen



05. KanBan

05. KanBan

Kanban es un método para gestionar el trabajo intelectual, con énfasis en la **entrega justo a tiempo**, mientras no se sobrecarguen los miembros del equipo.

En este enfoque, el proceso, desde la definición de una tarea hasta su entrega al cliente, **se muestra para que los participantes lo vean** y los miembros del equipo **tomen el trabajo de una cola**.

05. KanBan

Principios

1. Comience por lo que va a hacer ahora
2. Se acuerda perseguir el cambio incremental y evolutivo
3. Respetar el proceso actual, los roles, las responsabilidades y los cargos
4. Liderazgo en todos los niveles

05. KanBan

Prácticas

1. Visualizar
2. Limitar el trabajo en curso
3. Dirigir y gestionar el flujo
4. Hacer las Políticas de Proceso Explícitas
5. Utilizar modelos para reconocer oportunidades de mejora

¡Gracias!