**TEMA 1-INGENIERÍA DEL SOFTWARE**

**¿Qué es el software?**

1. Instrucciones que cuando se ejecutan, proporcionan la función y comportamiento deseados
2. Estructuras de datos que facilitan a los programas manipular adecuadamente la información
3. Documentos que describen la operación y el uso de los programas

Los productos software se pueden dividir en dos tipos:

* Genéricos (mercado general): desarrollados para su venta a un grupo de clientes diferentes
* A medida (para un cliente): desarrollado para un cliente único de acuerdo con sus especificaciones

**¿Qué es la ingeniería del software?**

La ingeniería del software es la disciplina que incluye las técnicas, los métodos y las herramientas para el desarrollo profesional del software.

**¿Cuál es la diferencia entre ciencia de la computación e ingeniería del software?**

El enfoque de la ciencia de la computación se centra en la base científica con el fin de demostrar nuevas técnicas. En cambio, la ingeniería del software utiliza lo demostrado por los expertos, con el único objetivo de resolver un problema.

**¿Qué son las prácticas de desarrollo de software?**

Se tratan de unas actividades bien definidas que contribuyan al logro de las metas del proyecto.

**¿Qué es una buena práctica?**

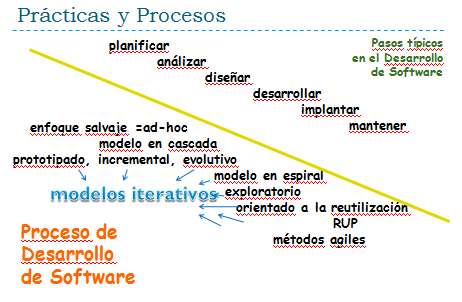
Aquella que a través de la experiencia y la investigación, ha demostrado conducir de forma segura a un resultado deseado. Además de que se considera prudente y recomendable para llevarla a cabo en múltiples contextos.

**¿Qué es un proceso de desarrollo de software?**

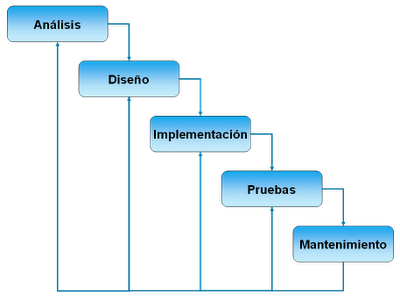
Es un proceso en el que las necesidades del usurio se transforman en una aplicación de software.

**¿Qué es un modelo de proceso de software?**

Es una definición simplificada y abstracta de un proceso de desarrollo de software.



**MODELO EN CASCADA**

Etapas muy definidas e inflexibles. Esto dificulta la reacción ante los cambios de los requisitos del cliente. Por eso este modelo es apropiado únicamente cuando los requisitos estén bien definidos.

Este modelo tiene una vista estática de los requisitos (no iterativo), ignora las necesidades de cambio y le falta la participación del usuario una vez que la especificación está escrita.

Muestra una separación por etapas del todo irreal.

Etapas

* Análisis: considerar todas las funciones o procesos del proyecto, incluyendo los objetivos, determinando las relaciones entre ellos y especificando qué aspectos serán incorporados en el sistema.
* Diseño: Una vez que los requisitos están definidos, es necesario identificar con detalle cómo se construirá el sistema para definir las tareas. ¿Qué información va a procesar? ¿Qué interfaz tendrá?
* Implementación: Programar o codificar.
* Pruebas: Comprobar que funciona correctamente y eficientemente. Enfocado en dos áreas: eficiencia interna(es óptimo) y efectividad externa (cumple con lo acordado).

**PROTOTIPADO**

Existen dos enfoques de este modelo:

De usar y tirar: Se basa en la creación de una herramienta no finalizada con la que el usuario experimenta con el fin de obtener más requisitos o mejorar los existentes.

Evolutivo: se generan prototipos enfocados a un producto final, pero se pueden mejorar aspectos (versiones betas).

Se utiliza para:

* Entender los requisitos para la interfaz de usuario
* Examinar la viabilidad de un diseño propuesto
* Explorar los problemas de rendimiento del sistema

El principal problema es que los usuarios tratan al prototipo como una solución final. Puede también conducir a falsas expectativas y sistemas pobres.

Etapas

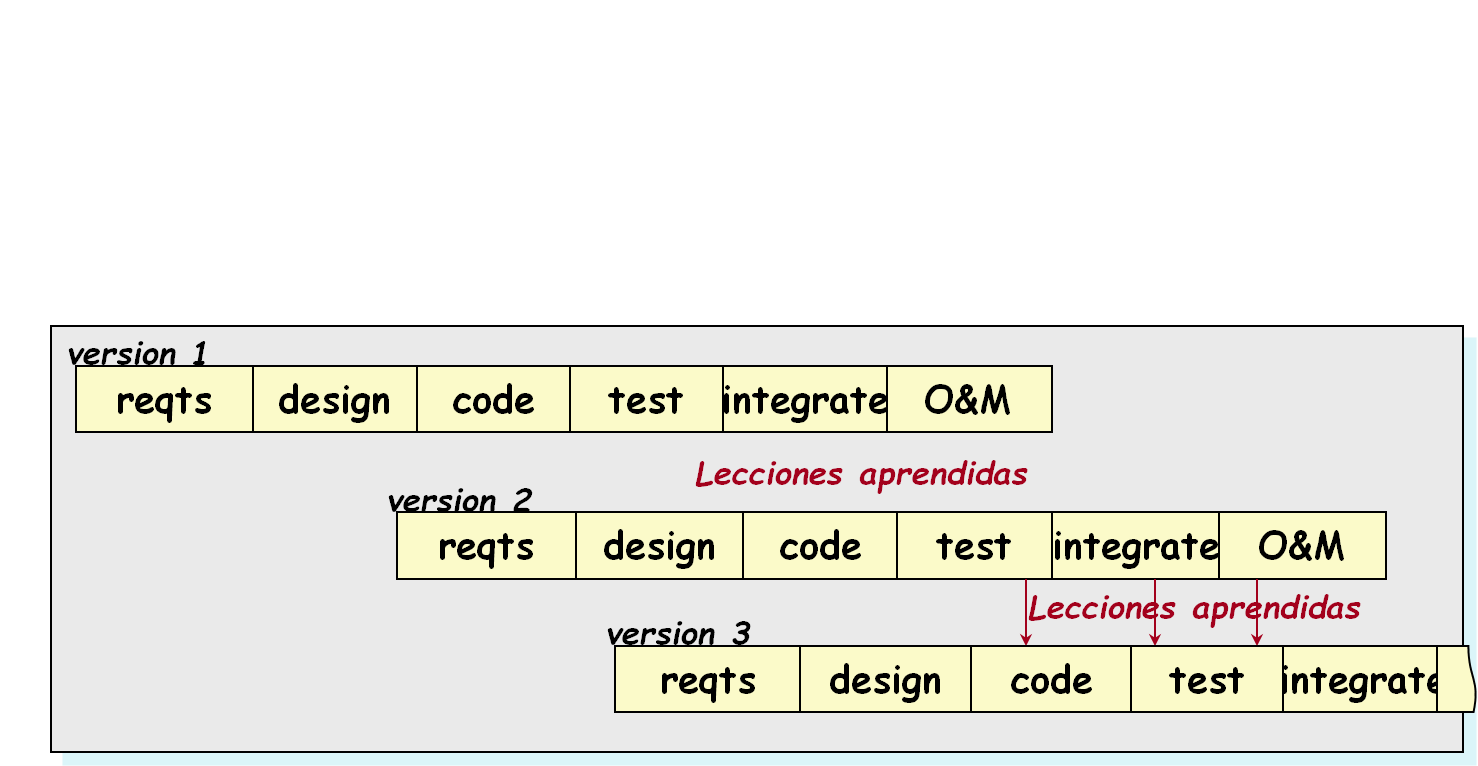
* Definición de requisitos: similar a la fase de conceptualización y análisis del modelo en cascada.
* Diseño: crear un diseño que será integrado en el prototipo.
* Creación del prototipo: generar un prototipo que incluya todo lo anteriormente mencionado.
* Evaluación: el prototipo es presentado al cliente y tras su experimentación, se recogen los comentarios y sugerencias.
* Modificación del prototipo: se revisa y mejora el prototipo.
* Implementación del sistema: se reescribe el sistema una vez que se tienen todos los requisitos bien especificados.

**PROCESO INCREMENTAL**

Se tienen los requisitos bien definidos y se dividen en incrementos con el fin de realizarlos en función de la prioridad, aunque se pueden dar casos en los que se lleven a cabo simultáneamente.

Cada incremento añade funcionalidad al sistema que se ha visto como un “todo” inicial.

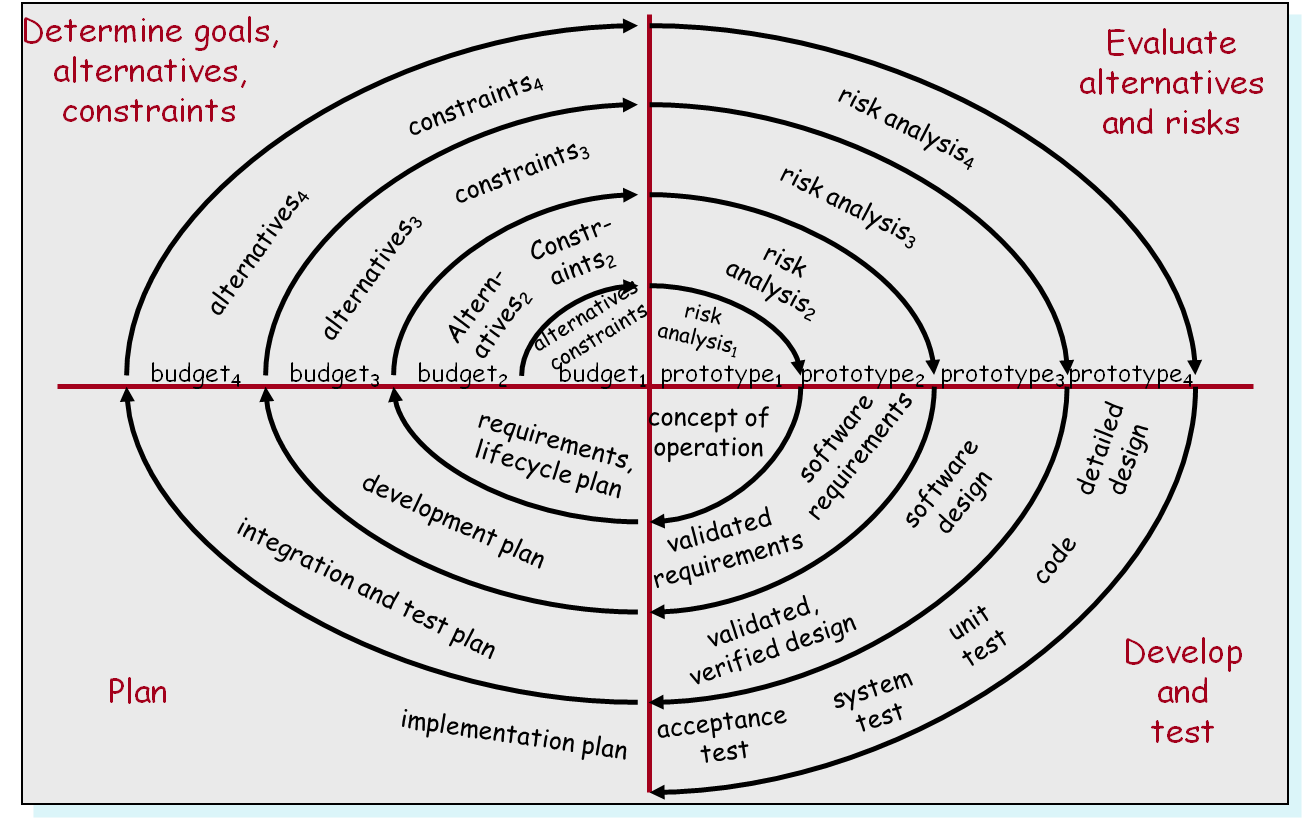
**PROCESO EVOLUTIVO**



Las primeras versiones “actúan” como prototipos que ayudan a obtener requisitos para las nuevas versiones. Cada versión incorpora nuevos requisitos.

Permite aprender de cada versión pero resulta difícil planificar para las versiones más allá de la primera y las lecciones se pueden aprender demasiado tarde.

**MODELO EN ESPIRAL**

Las palabras clave son riesgo y prototipos.

Incorpora la creación y análisis de riesgo pero no puede hacer frente a cambios imprevistos (nuevos objetivos de negocio).

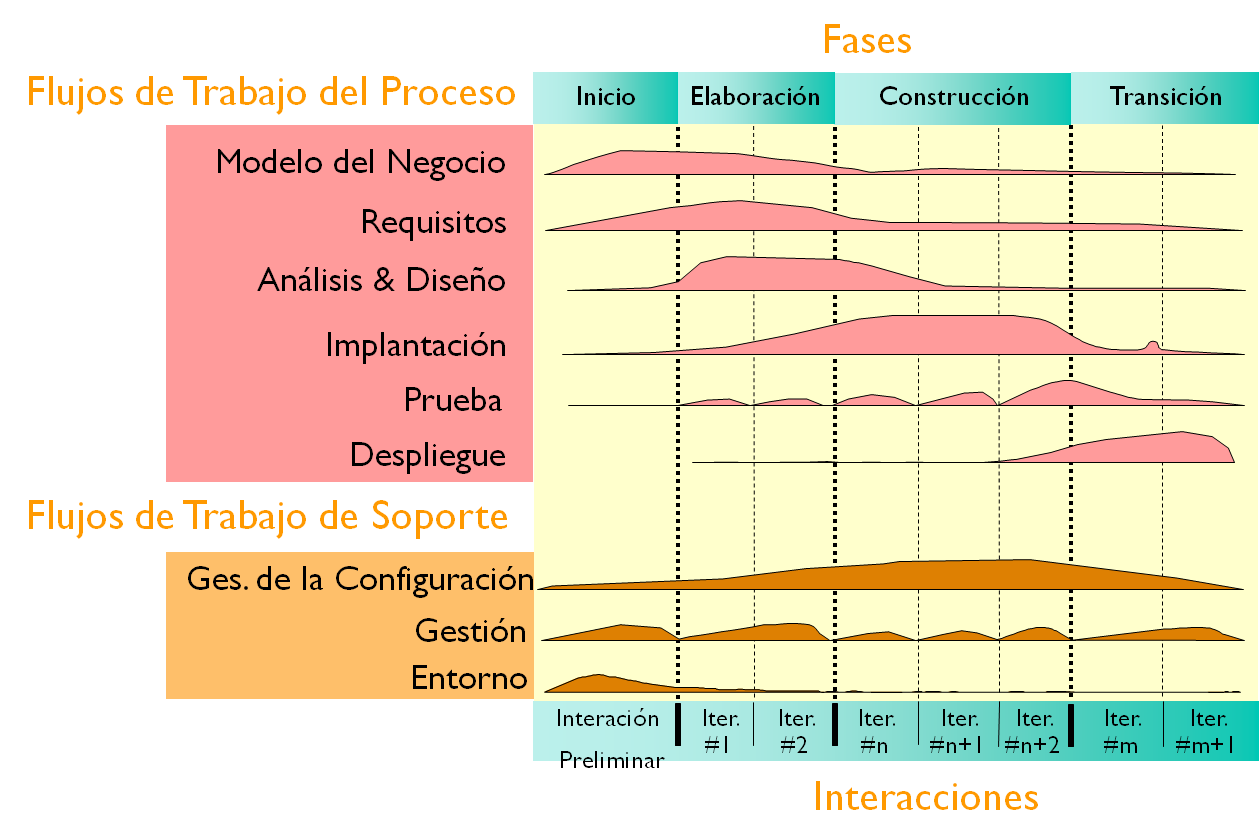
**MODELO DE LA REUSABILIDAD**



Reutilización sistemática en la que los sistemas se construyen integrando componentes ya existentes o sistemas COTS (Commercial Off-The-Self [software de estantería]: soluciones comerciales listas para su uso).

Apropiado para la programación orientada a objetos.

**RUP**

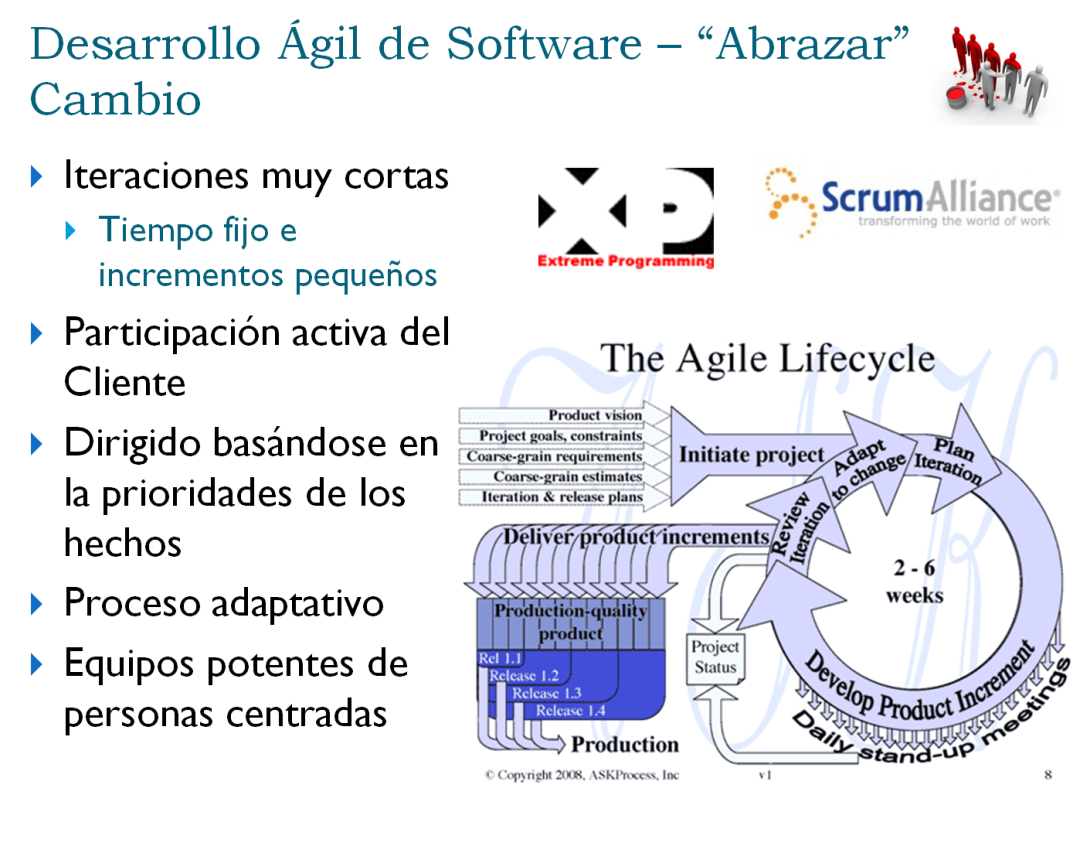
  
  
RUP no es un sistema con pasos firmemente establecidos, sino un conjunto de metodologías adaptables al contexto y necesidades de cada organización (modelo híbrido). Además, se caracteriza por ser iterativo e incremental, estar centrado en la arquitectura y guiado por los casos de uso.

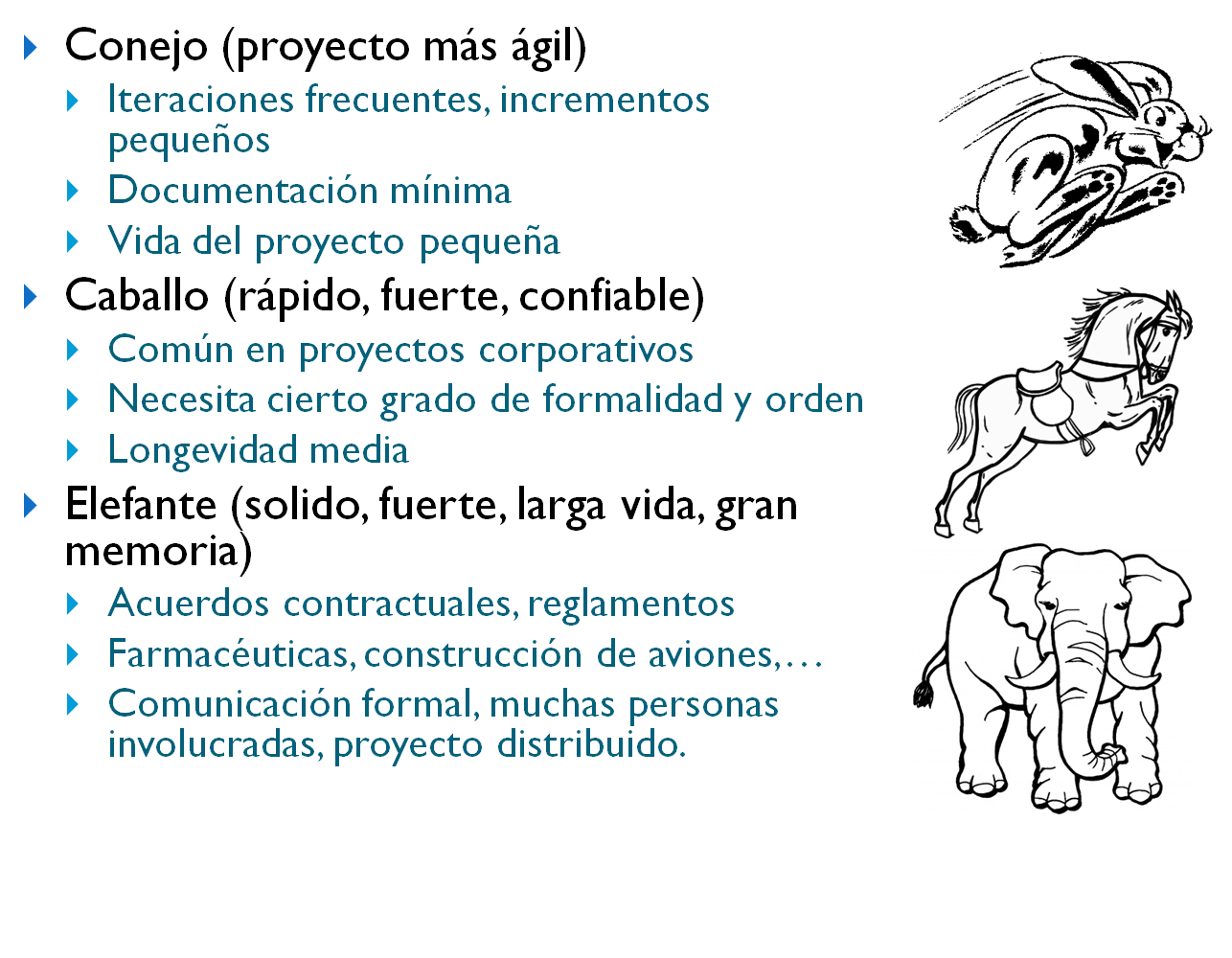
\*Este pico se refiere a los arreglos que se realizan a medida que se hacen pruebas.

Se basa en seis principios:

* Adaptar el proceso: se deberá adaptar a las necesidades del cliente influyendo en su diseño.
* Equilibrar prioridades: debe encontrarse un equilibrio que satisfaga los deseos de todos.
* Demostrar valor iterativamente: existen iteraciones en cada etapa.
* colaboración entre equipos: comunicación fluida entre los equipos participantes.
* Elevar el nivel de abstracción: evita que los ingenieros vayan directamente de los requisitos a la codificación.
* Enfocarse en la calidad: verificar continuamente la calidad.

**DESARROLLO ÁGIL DE SOFTWARE**



  
**CASE: Computer Aided Software Engineering**

Sistemas software creados para proporcionar soporte automatizado a las actividades de los procesos de Software, aumentando la productividad.

Upper-CASE (CASE de alto nivel)

Herramientas para el soporte de las actividades iniciales del proceso de requisitos y diseño.

Lower-CASE (CASE de bajo nivel)

Herramientas que dan soporte a las actividades finales como programación, prueba y depuración (Eclipse or Netbeans).

**RESPONSABILIDAD ETICA Y PROFESIONAL**

