Ingeniería de Software

Procesos de Software

Modelos de Proceso

Sommerville capítulo 2

Sección 2.1 Software process models

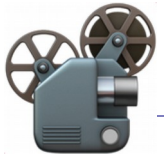
Material “Software Development Life Cycle Models and Methodologies”

Temario

1) Modelos de proceso de software
2) Actividades del proceso de software
3) Haciendo frente al cambio
4) Mejora de procesos

El proceso de software

- Para el desarrollo de un producto de software se necesita un conjunto estructurado de actividades.
- Si bien existen diferentes modelos de proceso, todos ellos incluyen actividades de:
  - ***Especificación*** – definir qué es lo que el sistema debe hacer.
  - ***Diseño e implementación*** – definir la organización del sistema e implementarlo.
  - ***Validación*** – comprobar que el sistema construido es lo que el cliente quiere/necesita.
  - ***Evolución*** – realizar cambios en el sistema como respuesta a los cambios en las necesidades del cliente.



Descripción de un proceso de software

- Cuando se describe un proceso, generalmente hablamos de las **actividades** de dicho proceso (“especificar el modelo de datos”, “diseñar la interfaz de usuario”) y el orden en el cual se deberían realizar.
- La descripción de un proceso además puede incluir:
  - Productos, los cuales son producidos por las actividades.
  - Roles, los cuales reflejan las responsabilidades de las personas involucradas en el proceso.
  - Pre- y Post-condiciones, las cuales deben cumplirse antes y después de que una actividad se haya realizado o un producto se haya desarrolado.

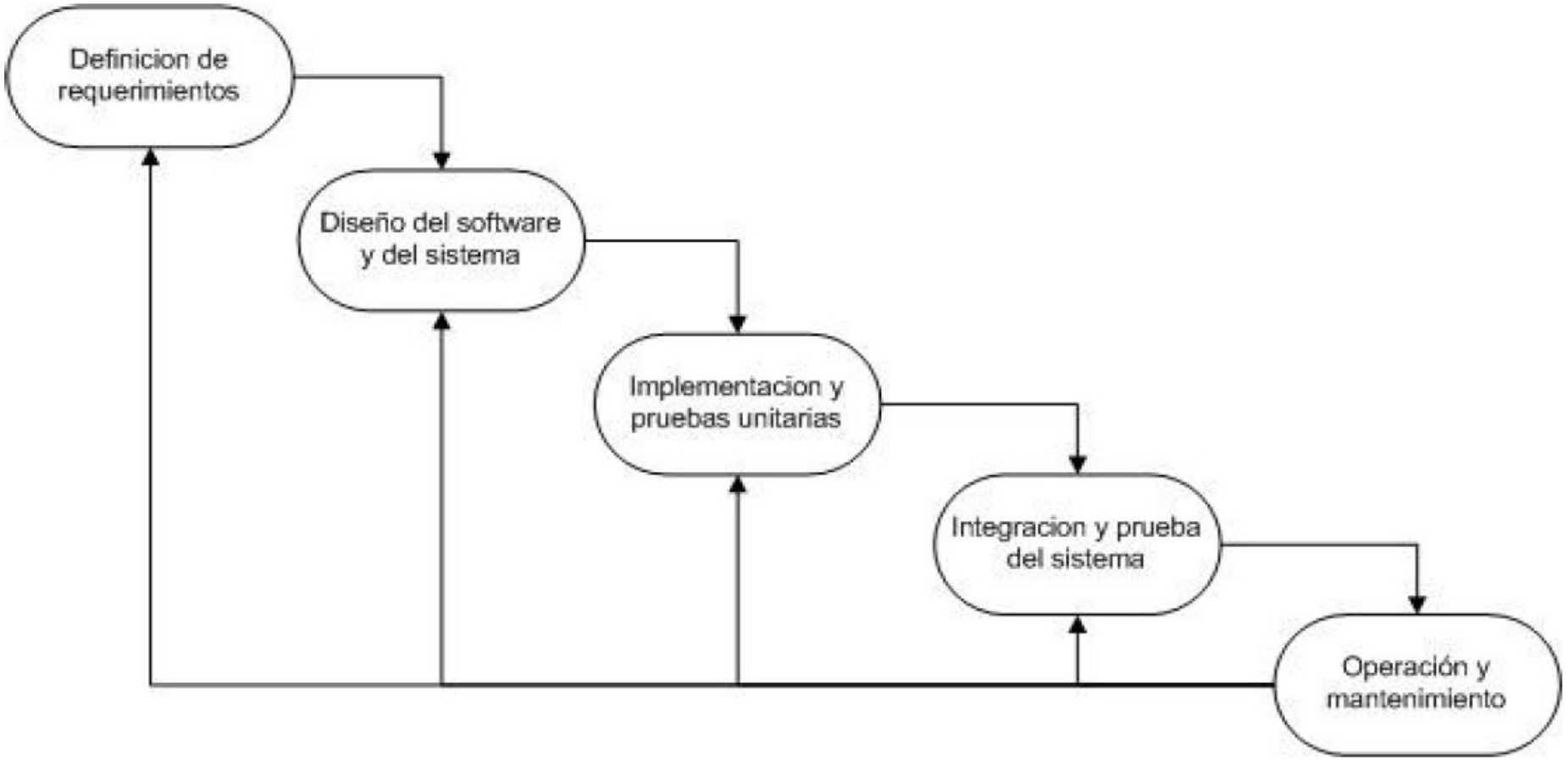
Procesos basados en planes y procesos ágiles

- Los **procesos dirigidos por planes** (plan-driven processes) son aquellos en los que todas las actividades del proceso son planificadas en detalle y el avance y progreso se mide contra dicho plan.
- En los **procesos ágiles** (agile processes) la planificación es incremental y es fácil de cambiar, a modo de reflejar los cambios en los requerimientos del cliente.
- En la práctica, la mayoría de los procesos incluyen elementos de ambos enfoques (ágiles y dirigidos por planes).
- **No existen procesos correctos o incorrectos.**

Modelos de procesos

- Un **modelo de proceso** (a veces llamado ciclo de vida de desarrollo de software) es una representación abstracta de un proceso, bajo un cierto punto de vista o perspectiva.
- Los modelos de proceso que cubrimos en el curso son:
  - ***Modelo en cascada***: Tiene fases separadas de especificación y desarrollo.
  - ***Desarrollo incremental***: la especificación, el desarrollo y la validación están intercaladas. Puede aplicarse tanto ágil como dirigido por planes.
  - ***Integración y configuración***: El sistema es "ensamblado" configurando componentes pre-existentes. Puede aplicarse tanto al enfoque ágil, como al dirigido por planes.
- En la práctica, la gran mayoría de los sistemas son desarrollados **incorporando elementos de todos los modelos**.

Modelo en cascada

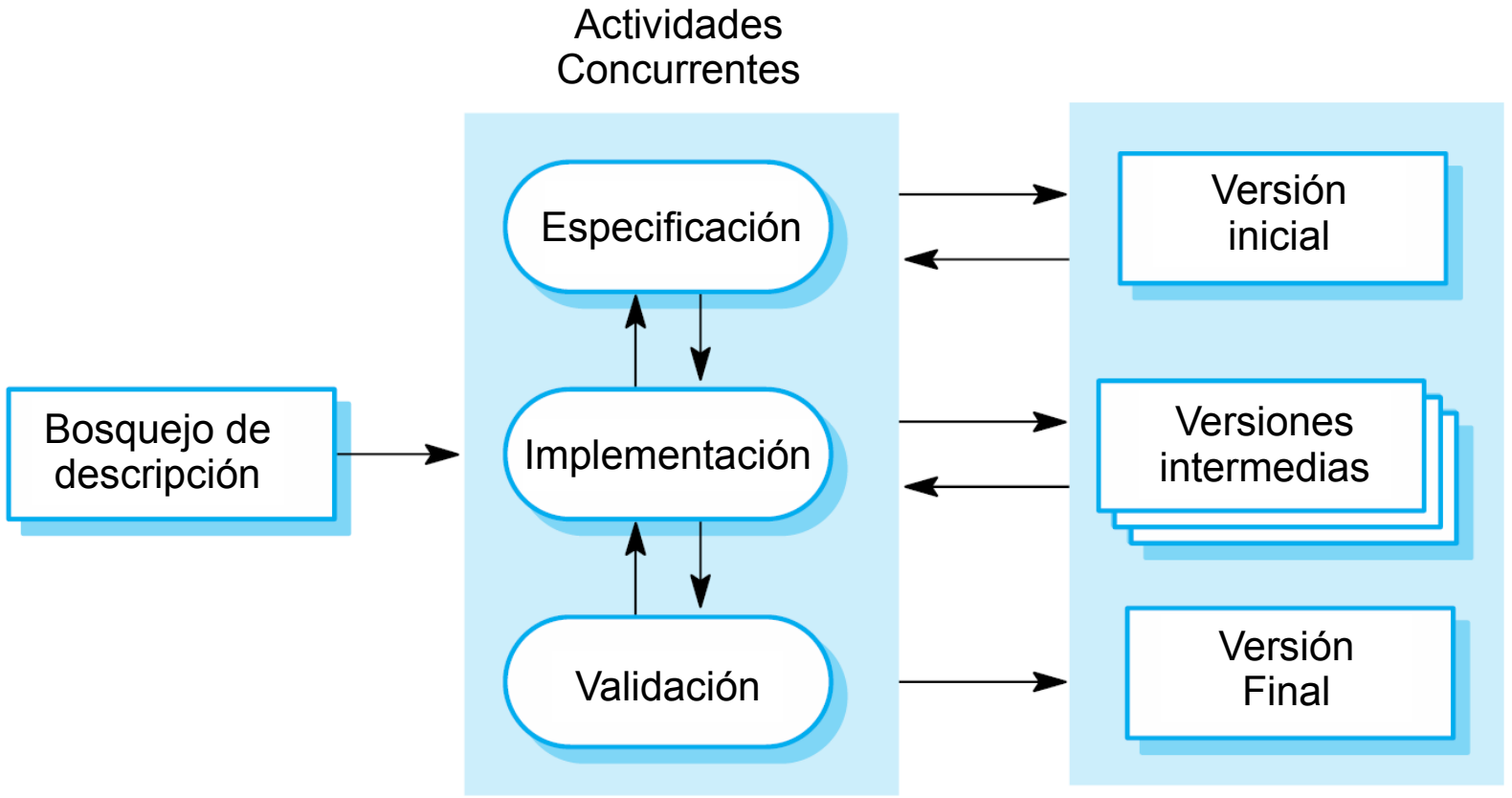


Problemas del modelo en cascada

- Dificultad de responder a los cambios de requerimientos del cliente.
  - Este modelo sería apropiado únicamente si los requerimientos son bien claros y estables desde el inicio (con muy baja probabilidad de que ocurran).
- Entrega tardía de valor para el cliente.
  - Con alto riesgo de que el producto no cumpla sus expectativas o no se adapte a sus necesidades.

Desarrollo incremental

- Se divide en “incrementos” o “fases”, también llamado “Desarrollo en Fases”.



Beneficios del Desarrollo Incremental

- Se reduce el **costo de acomodar los cambios** en los requisitos.
  - El re-trabajo es mucho menor que en el modelo en cascada
- Es más fácil obtener **retroalimentación del cliente** sobre la parte del desarrollo terminada
  - El cliente puede ver el avance del proyecto y realizar comentarios en las demostraciones
- Se pueden realizar entregas más rápidas de software que puede ser útil para el cliente
  - El cliente puede **usar y ganar valor con el software** de forma más temprana que con el modelo en cascada

Problemas con el desarrollo incremental

- Si los sistemas son desarrollados de forma rápida, es difícil mantener la **trazabilidad** de documentos entre una versión y otra. Además, no es eficiente producir documentación que refleje cada versión del sistema
- La estructura del sistema tiende a **degradarse** con cada incremento
  - Los cambios regulares tienden a corromper la estructura del sistema
  - Tiempo y dinero son invertidos en la refactorización del sistema
  - A medida que pasa el tiempo, incorporar los cambios se torna difícil y costoso

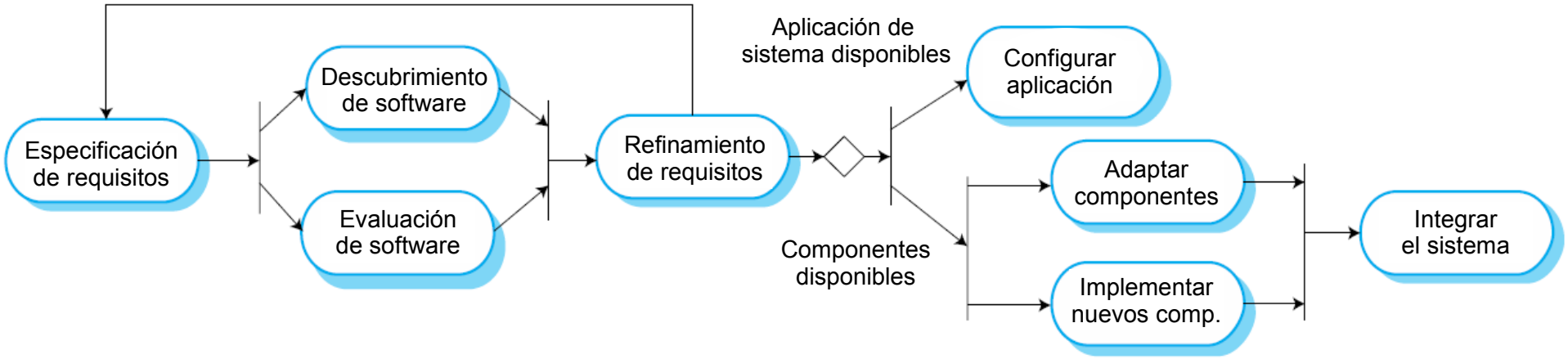
Integración y Configuración

- Está basado en la **reutilización de componentes de software o COTS** (Commercial-off-theshelf), los cuales son integrados en el sistema a construir
  - Los componentes en general tienen que ser configurados para adaptar su comportamiento y funcionalidad a los requerimientos del cliente
  - La reutilización de componentes es hoy en día un enfoque estándar para la construcción de muchos tipos de sistemas de negocio

Algunos tipos de software reusable

- Aplicaciones del tipo *stand-alone* (llamadas COTS), las cuales son configuradas para uso en algún ambiente en particular (por ejemplo ERP's).
- Paquetes, librerías o colecciones de objetos que son desarrollados para ser integrados con un framework (como ser .NET or J2EE).
- Servicios web que son desarrollados de acuerdo a estándares de servicios, los cuales se encuentran disponibles mediante invocación remota.

IS orientada a la reutilización



Ventajas y desventajas de Integración y Configuración

- Se reducen costos y riesgos ya me que menos software se desarrolla desde cero.
- Desarrollo y entrega del producto más rápida.
- Puede que el sistema no cumpla con las necesidades reales del cliente (o las cumpla parcialmente).
- Se pierde control sobre la evolución de los elementos reutilizados en el sistema.