ACTIVIDADES DEL PROCESO DE SOFTWARE

MSc Marcos Raúl Córdova Bayas
2022

2.3 Actividades del proceso de software

- Los procesos de software real son secuencias entrelazadas de actividades técnicas, colaborativas y administrativas con la meta general de especificar, diseñar, implementar y probar un sistema de software.
- Los desarrolladores de software usan en su trabajo diferentes herramientas de software.
- Las herramientas son útiles particularmente para dar apoyo a la edición de distintos tipos de documentos y para manejar el inmenso volumen de información detallada que se reproduce en un gran proyecto de software.

2.3 Actividades del proceso de software

- Las cuatro actividades básicas del proceso de software son: especificación, desarrollo, validación y evolución, que se organizan de diversa manera en diferentes procesos de desarrollo.
- En el modelo en cascada se organizan en secuencia, mientras que se entrelazan en el desarrollo incremental.
- La forma en que se llevan a cabo estas actividades depende del tipo de software, del personal y de la inclusión de estructuras organizativas.

2.3.1 Especificación del software

- La especificación del software o la ingeniería de requerimientos consisten en el proceso de comprender y definir qué servicios se requieren del sistema, así como la identificación de las restricciones sobre la operación y el desarrollo del sistema.
- La ingeniería de requerimientos es una etapa particularmente crítica del proceso de software, ya que los errores en esta etapa conducen de manera inevitable a problemas posteriores tanto en el diseño como en la implementación del sistema.

2.3.1 Especificación del software

- El proceso de ingeniería de requerimientos (figura 2.4) se enfoca en producir un documento de requerimientos convenido que especifique los requerimientos de los interesados que cumplirá el sistema.
- Por lo general, los requerimientos se presentan en dos niveles de detalle:
  - Los usuarios finales y clientes necesitan un informe de requerimientos de alto nivel.
  - Los desarrolladores de sistemas precisan una descripción más detallada del sistema.

2.3.1 Especificación del software

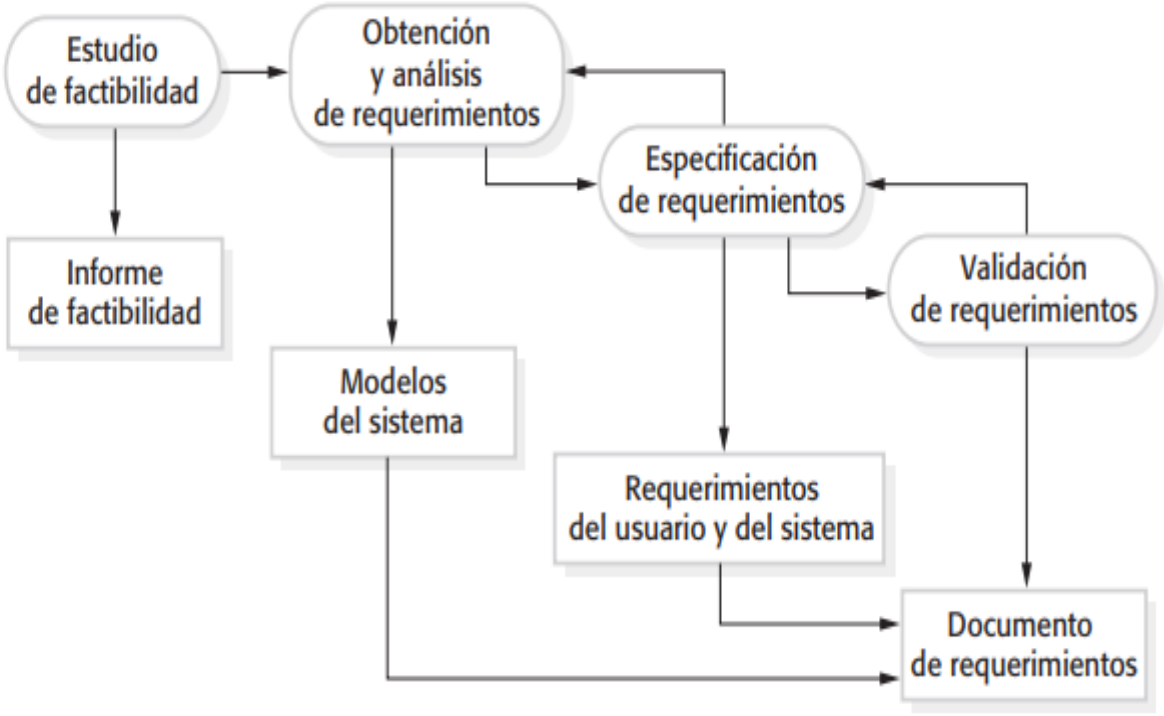

Figura 2.4 Proceso de ingeniería de requerimientos

2.3.1 Especificación del software

- Existen cuatro actividades principales en el proceso de ingeniería de requerimientos:

1. Estudio de factibilidad:

- Se realiza una estimación sobre si las necesidades identificadas del usuario se cubren con las actuales tecnologías de software y hardware.
- El estudio considera si el sistema propuesto tendrá un costo-beneficio desde un punto de vista empresarial, y si éste puede desarrollarse dentro las restricciones presupuestales existentes.
- Un estudio de factibilidad debe ser rápido y relativamente barato. El resultado debe informar la decisión respecto a si se continúa o no con un análisis más detallado.

2.3.1 Especificación del software

2. *Obtención y análisis de requerimientos:*

- En este proceso se derivan los requerimientos del sistema mediante observación de sistemas existentes, discusiones con los usuarios y proveedores potenciales, análisis de tareas, etc.
- Esto puede incluir el desarrollo de uno o más modelos del sistema y prototipos, lo que ayuda a entender el sistema que se va a especificar.

2.3.1 Especificación del software

3. *Especificación de requerimientos:*

- Consiste en transcribir la información recopilada durante el análisis en un documento que define los requerimientos del sistema.
- Se genera un documento que incluye dos tipos de requerimientos:
  - Los requerimientos del usuario: que son informes abstractos de requerimientos del sistema para el cliente y el usuario final del sistema; y
  - Los requerimientos de sistema: que son una descripción detallada de la funcionalidad a ofrecer, así como de las características de calidad que debe tener el sistema (requerimientos no funcionales)

2.3.1 Especificación del software

4. *Validación de requerimientos:*

- Esta actividad verifica que los requerimientos sean realistas, coherentes y completos.
- Permite descubrir errores en el documento de requerimientos.
- El documento debería modificarse con el fin de corregir dichos errores.

Las actividades en el proceso de requerimientos no se realizan simplemente en una secuencia estricta. El análisis de requerimientos continúa durante la definición y especificación, y a lo largo del proceso salen a la luz nuevos requerimientos; por lo tanto, las actividades de análisis, definición y especificación están vinculadas.

2.3.2 Diseño e implementación del software

- El diseño se entiende como una descripción de la estructura del software que se va a implementar, los modelos y las estructuras de datos utilizados por el sistema, las interfaces entre componentes del sistema y los algoritmos utilizados.
- El proceso es iterativo y tipo *backtracking* (vuelta atrás) constante para corregir diseños anteriores.
- La figura 2.5 es un modelo abstracto de este proceso que muestra las entradas al proceso de diseño, las actividades y los documentos generados como salidas del mismo.

Modelo general del proceso de diseño

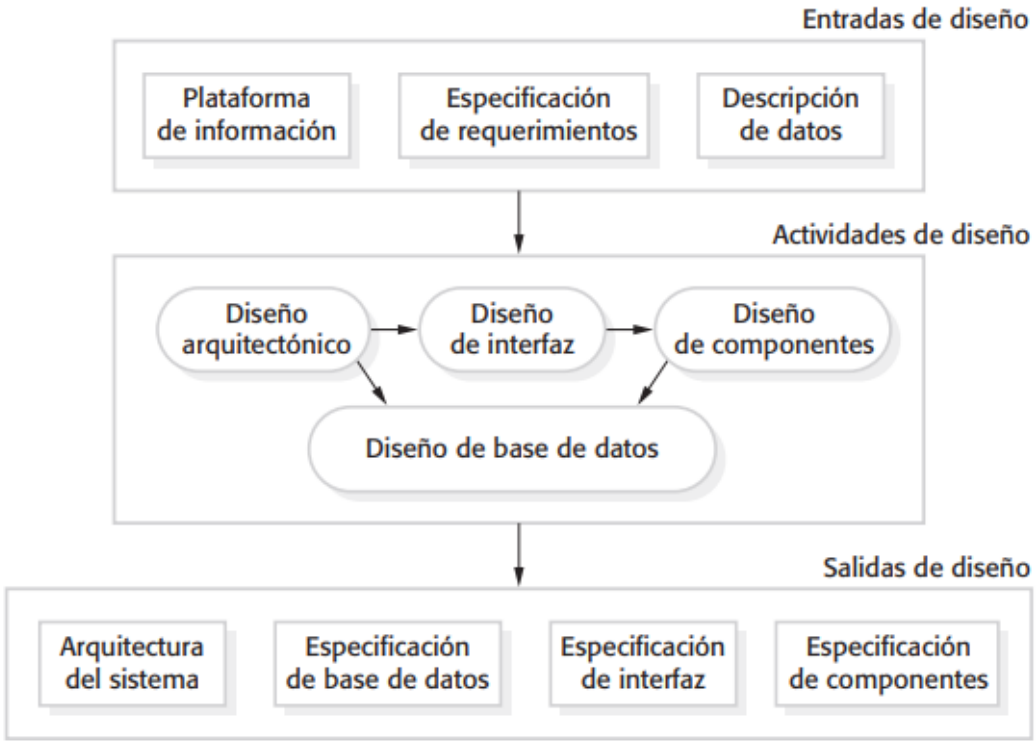

Figura 2.5 Modelo general del proceso de diseño

2.3.2 Diseño e implementación del software

- El diagrama sugiere que las etapas del proceso de diseño son secuenciales.
- De hecho, las actividades de proceso de diseño están vinculadas.
- En todos los procesos de diseño es inevitable la retroalimentación de una etapa a otra y la consecuente reelaboración del diseño.
- En el diseño también se definen las interfaces de software: sistema operativo, base de datos, *middleware* y otros sistemas de aplicación.
- Estos constituyen la "plataforma del software" que es el entorno donde se ejecutará el software.

2.3.2 Diseño e implementación del software

- La información sobre esta plataforma es una entrada esencial al proceso de diseño, así que se debe decidir sobre ella y sobre la mejor forma de integrarla con el entorno de software.
- Las actividades del diseño dependen del tipo de sistema a desarrollar.
- Para sistemas de tiempo real se precisa del diseño de temporización, pero sin incluir una base de datos.
- La figura 2.5 muestra cuatro actividades que podrían formar parte del proceso de diseño para sistemas de información:

2.3.2 Diseño e implementación del software

1. *Diseño arquitectónico*: se identifica la estructura global del sistema, los principales componentes (subsistemas o módulos), sus relaciones y cómo se distribuyen.
2. *Diseño de interfaz:* se definen las interfaces entre los componentes del sistema. Las interfaces deben ser no ambiguas, de tal manera que un componente puede ser usado sin que otros componentes tengan que conocer cómo está implementado. Una vez acordadas las especificaciones de las interfaces, los componentes pueden ser diseñados y desarrollados concurrentemente.

2.3.2 Diseño e implementación del software

3. *Diseño de componentes*: se toma cada componente y se diseña cómo funcionará. Puede ser una simple sentencia de la funcionalidad que se espera implementar, dejando al programador el diseño específico. También se puede especificar una lista de cambios a realizar sobre un componente que se reutiliza o sobre un modelo de diseño detallado. El modelo de diseño sirve para generar automáticamente el código a ser implementado.

4. *Diseño de base de datos:* se diseñan las estructuras del sistema de datos y cómo se representarán en una base de datos. Otra vez, el trabajo depende de si una base de datos se reutilizará o se creará una nueva.

2.3.2 Diseño e implementación del software

- La etapa de implementación corresponde al proceso de convertir una especificación del sistema en un sistema ejecutable.
- Incluye procesos de diseño y programación de software, pero puede también involucrar la corrección en la especificación del software, si se utiliza un enfoque incremental de desarrollo.
- El diseño de un programa para implementar el sistema se sigue naturalmente de los procesos de elaboración del sistema.
- Las herramientas de desarrollo de software se usan para generar un programa de "esqueleto" a partir de un diseño.
- Esto incluye un código para definir e implementar interfaces y, en muchos casos, el desarrollador solo necesita agregar detalles de la operación de cada componente del programa.

2.3.2 Diseño e implementación del software

- La programación es una actividad personal y no hay un proceso que se siga de manera general.
- Algunos programadores comienzan con componentes que entienden, los desarrollan y, luego, cambian hacia componentes que entienden menos.
- Otros toman el enfoque opuesto, y dejan hasta el último los componentes familiares, porque saben cómo diseñarlos.
- Algunos desarrolladores definen con anticipación datos en el proceso, para usarlos luego para el desarrollo del programa; otros dejan datos sin especificar tanto como sea posible.

2.3.2 Diseño e implementación del software

- Los programadores desarrollan pruebas del código que desarrollaron.
- Esto revela con frecuencia defectos del programa que deben eliminarse, lo que se llama depuración (*debugging*).
- La prueba de defectos y la depuración son procesos diferentes.
- La primera establece la existencia de defectos, en tanto que la segunda se dedica a localizar y corregir dichos defectos.
- Cuando se depura, uno debe elaborar una hipótesis sobre el comportamiento observable del programa y, luego, poner a prueba dichas hipótesis con la esperanza de encontrar la falla que causó la salida anómala.

2.3.2 Diseño e implementación del software

- Poner a prueba las hipótesis quizá requiera rastrear manualmente el código del programa; o bien, tal vez se necesiten nuevos casos de prueba para localizar el problema.
- Con la finalidad de apoyar el proceso de depuración, se deben utilizar herramientas interactivas que muestren valores intermedios de las variables del programa, así como el rastro de las instrucciones ejecutadas.

2.3.3 Validación del software

- La validación de software o, más generalmente, su verificación y validación (V&V), se crea para mostrar que un sistema cumple tanto con sus especificaciones como con las expectativas del cliente.
- Las pruebas del programa, donde el sistema se ejecuta a través de datos de prueba simulados, son la principal técnica de validación.
- Esta última también puede incluir procesos de comprobación, como inspecciones y revisiones en cada etapa del proceso de software, desde la definición de requerimientos del usuario hasta el desarrollo del programa.
- Dada a la predominancia de las pruebas, la mayoría de los costos de validación se incurren durante y después de la implementación.

2.3.3 Validación del software

- Excepto para programas pequeños, los sistemas no deberían ser probados como una unidad monolítica.
- La figura 2.6 muestra un proceso de prueba en tres etapas:
  - Prueba de componentes
  - Prueba del sistema
  - Prueba de aceptación
- Idealmente, los defectos de los componentes se detectan en las pruebas de unidad, mientras que los problemas de interfaz se localizan cuando se hacen las pruebas de integración.
- Conforme se descubren defectos el programa debe depurarse, lo cual requiera la repetición de otras etapas en el proceso de pruebas.
- El proceso es iterativo, con información retroalimentada desde etapas posteriores hasta las iniciales del proceso.

Etapas de pruebas

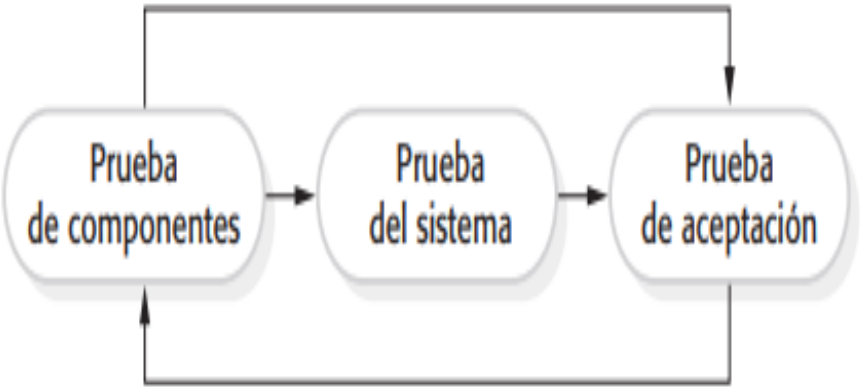

Figura 2.6 Etapas de pruebas

2.3.3 Validación del software

Etapas en el proceso de pruebas:

1. *Prueba de desarrollo*: Las personas que desarrollan el sistema ponen a prueba los componentes que constituyen el sistema.

- Cada componente se prueba de manera independiente, es decir, sin otros componentes del sistema.
- Éstos pueden ser simples entidades, como funciones o clases de objeto, o agrupamientos coherentes de dichas entidades.
- Por lo general, se usan herramientas de automatización de pruebas, como JUnit (Massol y Husted, 2003), que pueden volver a correr pruebas de componentes cuando se crean nuevas versiones del componente.

2.3.3 Validación del software

2. *Pruebas del sistema*: aquí se prueban los componentes del sistema que se integran para crear el sistema completo.

- Este proceso tiene la finalidad de descubrir errores que resulten de interacciones no anticipadas entre componentes y problemas de interfaz de componente, así como de mostrar que el sistema cubre sus requerimientos funcionales y no funcionales, y poner a prueba las propiedades emergentes del sistema.

- Para sistemas grandes, esto puede ser un proceso de múltiples etapas, donde los componentes se conjuntan para formar subsistemas que se ponen a prueba de manera individual, antes de que dichos subsistemas se integren para establecer el sistema final.

2.3.3 Validación del software

3. *Pruebas de aceptación*: el sistema se pone a prueba con datos suministrados por el cliente del sistema, en vez de datos de prueba simulados.

- Las pruebas de aceptación revelan los errores y las omisiones en la definición de requerimientos del sistema, ya que los datos reales ejercitan el sistema en diferentes formas a partir de los datos de prueba.

- Asimismo, las pruebas de aceptación revelan problemas de requerimientos, donde las instalaciones o infraestructura del sistema en realidad no cumplan las necesidades del usuario o cuando sea inaceptable el rendimiento del sistema.

2.3.3 Validación del software

- Por lo general, los procesos de desarrollo y de pruebas de componentes están entrelazados. Los programadores construyen sus propios datos de prueba y experimentan el código de manera incremental conforme lo desarrollan.
- Éste es un enfoque económicamente sensible, ya que el programador conoce el componente y, por lo tanto, es el más indicado para generar casos de prueba.
- Si se usa un enfoque incremental para el desarrollo, cada incremento debe ponerse a prueba conforme se diseña, y tales pruebas se basan en los requerimientos para dicho incremento.

2.3.3 Validación del software

- A las pruebas de aceptación se las identifica como “pruebas alfa”. Se las realiza en el entorno de desarrollo y son realizadas por el cliente (o usuarios) del sistema, hasta que estén de acuerdo en que tienen un implementación aceptable de los requerimientos.
- Cuando un sistema se marca como producto de software, se realizan las denominadas “prueba beta”. Se las realiza por los usuarios del sistema en el entorno de producción.
- Al encontrar errores los reportan a los desarrolladores para que los corrijan y, después de esta retroalimentación, el sistema se modifica y libera, ya sea para más pruebas beta o para su venta o entrega final.

2.3.4 Evolución del software

- La flexibilidad de los sistemas software es una de las razones principales por las que cada vez más software se incorpora en sistemas grandes y complejos.
- Si se construye un sistema en base al hardware, resulta muy costoso hacer cambios a su diseño.
- Por el contrario, en cualquier momento durante o después del desarrollo del sistema, pueden hacerse cambios al software.
- Incluso los cambios mayores son todavía más baratos que los correspondientes cambios al hardware del sistema.
- La ingeniería de software es un proceso evolutivo, donde el software cambia continuamente a lo largo de su ciclo de vida, en función de los requerimientos y las necesidades del cliente.