CLASE 6

Procesos del Software

Definición

"Un proceso del software es un conjunto de actividades que conducen a la creación de un producto software. Estas actividades pueden consistir en el desarrollo de software desde cero en un lenguaje de programación estándar como Java o C. Sin embargo, cada vez más, se desarrolla nuevo software ampliando y modificando los sistemas existentes y configurando e integrando software comercial o componentes del sistema.

Los procesos del software son complejos y, como todos los procesos intelectuales y creativos, dependen de las personas que toman decisiones y juicios. Debido a la necesidad de juzgar y crear, los intentos para automatizar estos procesos han tenido un éxito limitado. Las herramientas de ingeniería del software asistida por computadora (CASE) pueden ayudar algunas actividades del proceso. Sin embargo, no existe posibilidad alguna, al menos en los próximos años, de una automatización mayor en el diseño creativo del software realizado por los ingenieros relacionados con el proceso del software. […]

Aunque existen muchos procesos diferentes de software, algunas actividades fundamentales son comunes para todos ellos:

1. *Especificación del software: Se debe definir la funcionalidad del software y las restricciones en su operación.*
2. *Diseño e implementación del software: Se debe producir software que cumpla su especificación.*
3. *Validación del software: Se debe validar el software para asegurar que hace lo que el cliente desea.*
4. *Evolución del software: El software debe evolucionar para cubrir las necesidades cambiantes del cliente."* [1]

La Proceso del Software complementa el concepto de ciclo de vida del software en el sentido de que éste último define el esqueleto y la filosofía para llevar a cabo un proceso de software, pero no es suficiente para guiar y controlar un proyecto de desarrollo y/o mantenimiento.

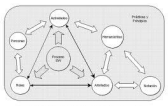
Un proceso del software es "un conjunto coherente de políticas, estructuras organizacionales, tecnologías, procedimientos y equipamientos que son necesarios para concebir, desarrollar, instalar y mantener un producto software".

[1] Ian Sommerville. (2006). 7ma Edición. Ingeniería del Software. PEARSON EDUCACION, SA. Madrid – España. Pág. 60.

La naturaleza especial de los procesos del software está determinada por las siguientes características:

- Son complejos.
- No son procesos de producción típicos, ya que están dirigidos por excepciones, se ven muy determinados por circunstancias impredecibles y cada uno tiene peculiaridades que lo distinguen de los demás.
- Tampoco son procesos de ingeniería 'pura', ya que se desconocen las abstracciones adecuadas (no existe una ciencia experimental en la que apoyarse), dependen demasiado de mucha gente, el diseño y la producción no están claramente diferenciados, y los presupuestos, calendarios y calidad no pueden ser planificados de forma suficientemente fiable.
- No son (completamente) procesos creativos, ya que algunas partes pueden ser descritas en detalle y algunos procedimientos son impuestos previamente.
- Están basados en descubrimientos que dependen de la comunicación, coordinación y cooperación dentro de marcos de trabajo predefinidos: los entregables generan nuevos requerimientos; los costes del cambio del software no suelen reconocerse; y el éxito depende de la implicación del usuario y de la coordinación de muchos roles (ventas, desarrollo técnico, cliente, etc.).

Modelos del proceso del software



"Un modelo del proceso del software es una representación abstracta de un proceso del software. Cada modelo de proceso representa un proceso desde una perspectiva particular, y así proporciona sólo información parcial sobre ese proceso. [...]

Estos modelos generales no son descripciones definitivas de los procesos del software. Más bien, son abstracciones de los procesos que se pueden utilizar para explicar diferentes enfoques para el desarrollo de software. Algunos modelos son:

1. *El modelo en cascada. Considera las actividades fundamentales del proceso de especificación, desarrollo, validación y evolución, y los representa como fases separadas del proceso, tales como la especificación de requerimientos, el diseño del software, la implementación, las pruebas, etc.*
2. *Desarrollo evolutivo. Este enfoque entrelaza las actividades de especificación, desarrollo y validación. Un sistema inicial se desarrolla rápidamente a partir de especificaciones abstractas. Éste se refina basándose en las peticiones del cliente para producir un sistema que satisfaga sus necesidades.*
3. *Ingeniería del software basada en componentes. Este enfoque se basa en la existencia de un número significativo de componentes reutilizables. El proceso de desarrollo del sistema se enfoca en integrar estos componentes en el sistema más que en desarrollarlos desde cero.*

Estos tres modelos de procesos genéricos se utilizan ampliamente en la práctica actual de la ingeniería del software. No se excluyen mutuamente y a menudo se utilizan juntos, especialmente para el desarrollo de sistemas grandes." [2]

[2] Ian Sommerville. (2006). 7ma Edición. Ingeniería del Software. PEARSON EDUCACION, SA. Madrid – España. Pág. 60 - 61.

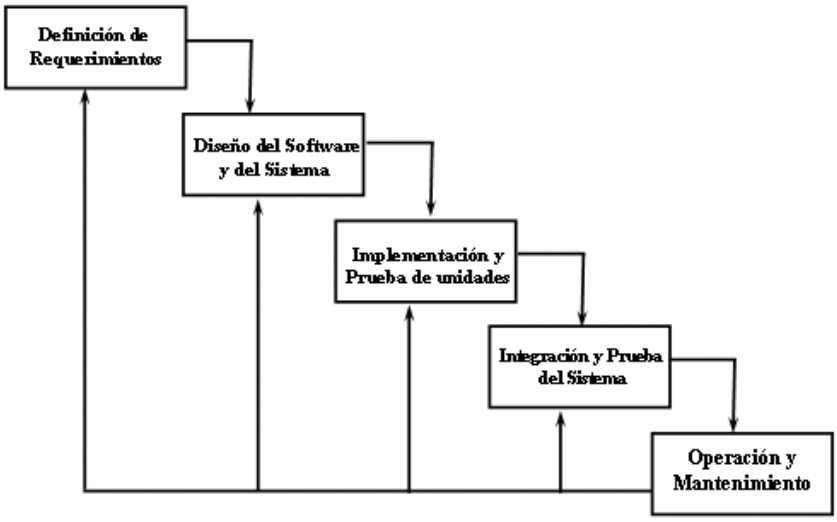
El modelo en cascada

"El primer modelo de proceso de desarrollo de software que se publicó se derivó de procesos de ingeniería de sistemas más generales (Royce, 1970). Debido a la cascada de una fase a otra, dicho modelo se conoce como modelo en cascada o como ciclo de vida del software. Las principales etapas de este modelo se transforman en actividades fundamentales de desarrollo:

1. *Análisis y definición de requerimientos. Los servicios, restricciones y metas del sistema se definen a partir de las consultas con los usuarios. Entonces, se definen en detalle y sirven como una especificación del sistema.*
2. *Diseño del sistema y del software. El proceso de diseño del sistema divide los requerimientos en sistemas hardware o software. Establece una arquitectura completa del sistema. El diseño del software identifica y describe las abstracciones fundamentales del sistema software y sus relaciones.*
3. *Implementación y prueba de unidades. Durante esta etapa, el diseño del software se lleva acabo como un conjunto o unidades de programas. La prueba de unidades implica verificar que cada una cumpla su especificación.*
4. *Integración y prueba del sistema. Los programas o las unidades individuales de programas se integran y prueban como un sistema completo para asegurar que se cumplan los requerimientos del software. Después de las pruebas, el sistema software se entrega al cliente.*
5. *Funcionamiento y mantenimiento. Por lo general (aunque no necesariamente), ésta es la fase más larga del ciclo de vida. El sistema se instala y se ponen en funcionamiento práctico. El mantenimiento implica corregir errores no descubiertos en las etapas anteriores del ciclo de vida, mejorar la implementación de las unidades del sistema y resaltar los servicios del sistema una vez que se descubren nuevos requerimientos.*

En principio, el resultado de cada fase es uno o más documentos aprobados (`firmados´). La siguiente fase no debe empezar hasta que la fase previa haya finalizado. En la práctica, estas etapas se superponen y proporcionan información a las otras. Durante el diseño se identifican los problemas con los requerimientos; durante el diseño del código se encuentran problemas, y así sucesivamente. El proceso del software no es un modelo lineal simple, sino que implica una serie de iteraciones de las actividades de desarrollo." [3]

[3] Ian Sommerville. (2006). 7ma Edición. Ingeniería del Software. PEARSON EDUCACION, SA. Madrid – España. Pág. 62 - 63.



Ejemplo del Modelo en cascada

Desarrollo evolutivo

"El desarrollo evolutivo se basa en la idea de desarrollar una implementación inicial, exponiéndola a los comentarios del usuario y refinándola a través de las diferentes versiones hasta que se desarrolla una sistema adecuado. Las actividades de especificación, desarrollo y validación se entrelazan en vez de separarse, con una rápida retroalimentación entre éstas.

Existen dos tipos de desarrollo evolutivo:

1. *Desarrollo explorativo, donde el objetivo del proceso es trabajar con el cliente para explorar sus requerimientos y entregar un sistema final. El desarrollo empieza con las partes del sistema que se comprenden mejor. El sistema evoluciona agregando nuevos atributos propuestos por el cliente.*
2. *Prototipos desechables, donde el objetivo del proceso de desarrollo evolutivo es comprender los requerimientos del cliente y entonces desarrollar una definición mejorada de los requerimientos para el sistema. El prototipo se centra en experimentar con los requerimientos del cliente que no se comprende del todo.*

En la producción de sistemas, un enfoque evolutivo para el desarrollo de software suele ser más efectivo que el enfoque en cascada, ya que satisface las necesidades inmediatas de los clientes. La ventaja de un proceso del software que se basa en un enfoque evolutivo es que la especificación se puede desarrollar de forma creciente. Tan pronto como los usuarios desarrollen un mejor entendimiento de su problema, éste se puede reflejar en el sistema software. Sin embargo, desde una perspectiva de ingeniería y de gestión, el enfoque evolutivo tiene dos problemas:

1. *El proceso no es visible. Los administradores tienen que hacer entregas regulares para medir el proceso. Si los sistemas se desarrollan rápidamente, no es rentable producir documentos que reflejen cada versión del sistema.*
2. *A menudo los sistemas tienen una estructura deficiente. Los cambios continuos tienden a corromper la estructura del software. Incorporar cambios en él se convierte cada vez más en una tarea difícil y costosa.*

Para sistemas pequeños y de tamaño medio, el enfoque evolutivo de desarrollo es el mejor. Los problemas del desarrollo evolutivo se hacen particularmente agudos para sistemas grandes y complejos con un periodo de vida largo, donde diferentes equipos desarrollan distintas partes del sistema. Es difícil establecer una arquitectura del sistema estable usando este enfoque, el cual hace difícil integrar las contribuciones de los equipos." [4]

Ejemplo, Desarrollo evolutivo

Ingeniería del software basada en componentes

"En la mayoría de los proyectos de software existe algo de reutilización de software. Por lo general, esto sucede informalmente cuando las personas que trabajan en el proyecto conocen diseños o códigos similares al requerido. Los buscan, los modifican según lo creen necesario y los incorporan en el sistema. [...]

Esta reutilización informal es independiente del proceso de desarrollo que se utilice. Sin embargo, en los últimos años, ha surgido un enfoque de desarrollo de software denominado ingeniería del software basada en componentes (CBSE) que se basa en la reutilización, el cual se está utilizando de forma amplia. [...]

Este enfoque basado en la reutilización se compone de una gran base de componentes software reutilizables y de algunos marcos de trabajo de integración para éstos. Algunas veces estos componentes son sistemas por sí mismos (COTS o sistemas comerciales) que se pueden utilizar para proporcionar una funcionalidad específica, como dar formato al texto o efectuar cálculos numéricos.

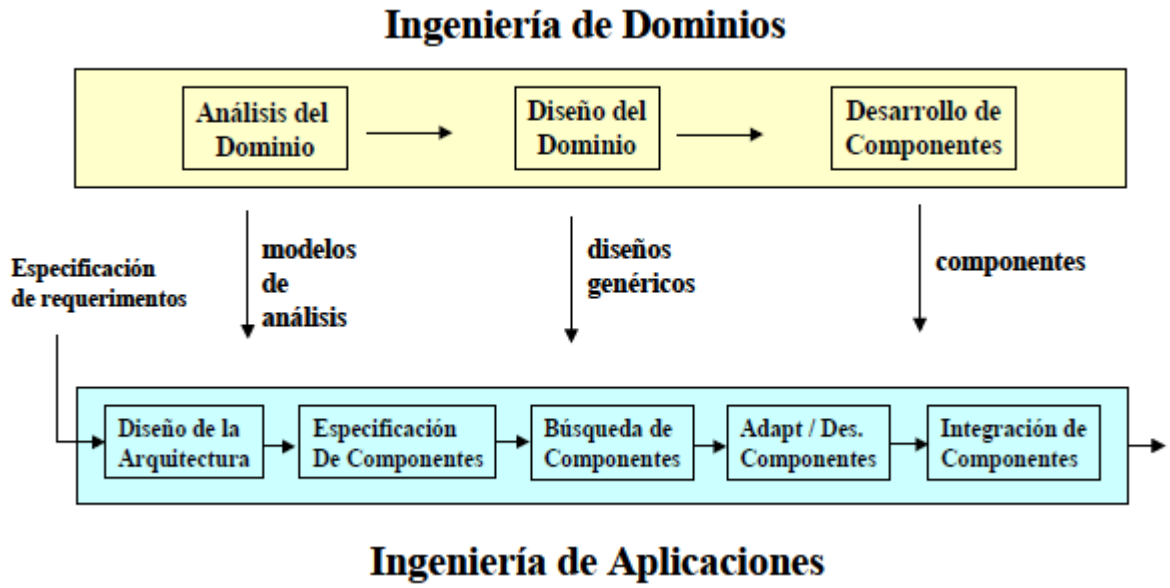
Aunque la etapa de especificación de requerimientos y la de validación son comparables con otros procesos, las etapas intermedias en el proceso orientado a la reutilización son diferentes. Estas etapas son:

1. *Análisis de componentes. Dada la especificación de requerimientos, se buscan los componentes para implantar esta especificación. Por lo general, no existe una concordancia exacta y los componentes que se utilizan sólo proporcionan parte de la funcionalidad requerida.*

[4] Ian Sommerville. (2006). 7ma Edición. Ingeniería del Software. PEARSON EDUCACION, SA. Madrid – España. Pág. 63 - 64.

2. *Modificación de requerimientos. En esta etapa, los requerimientos se analizan utilizando información acerca de los componentes que se han descubierto. Entonces, esos componentes se modifican para reflejar los componentes disponibles. Si las modificaciones no son posibles, la actividad de análisis de componentes se pude llevar a cabo nuevamente para buscar soluciones alternativas.*
3. *Diseño del sistema con reutilización. En esta fase se diseña o se reutiliza un marco de trabajo para el sistema. Los diseñadores tienen en cuenta los componentes que se reutilizan y organizan el marco de trabajo para que los satisfaga. Si los componentes reutilizables no están disponibles, se puede tener que diseñar nuevo software.*
4. *Desarrollo e integración. Para crear el sistema, el software que no se puede adquirir externamente se desarrolla, y los componentes y los sistemas COTS se integran. En este modelo, la integración de sistemas es parte del proceso de desarrollo, más que una actividad separada.*

La ingeniería del software basada en componentes tiene la ventaja obvia de reducir la cantidad de software a desarrollarse y así reduce los costos y los riesgos. Por lo general, también permite una entrega más rápida del software. Sin embargo, los compromisos en los requerimientos son inevitables, y esto puede dar lugar a un sistema que no cumpla las necesidades reales de los usuarios." [5]



Ejemplo, Ingeniería del software basada en componentes

[5] Ian Sommerville. (2006). 7ma Edición. Ingeniería del Software. PEARSON EDUCACION, SA. Madrid – España. Pág. 64 - 66.

Iteración de procesos

Definición

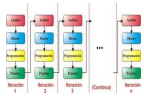
"Los cambios son inevitables en todos los proyectos de software grandes. Los requerimientos del sistema cambian cuando el negocio que procura el sistema responde a las presiones externas. Las prioridades de gestión cambian. Cuando se dispone de nuevas tecnologías, cambian los diseños y la implementación. Esto significa que el proceso del software no es un proceso único; más bien, las actividades del proceso se repiten regularmente conforme el sistema se rehace en respuesta a peticiones de cambios.

Se describen dos modelos de procesos que han sido diseñados explícitamente para apoyar la iteración de procesos:

1. *Entrega incremental. La especificación, el diseño y la implementación del software se dividen en una serie de incrementos, los cuales se desarrollan por turnos;*
2. *Desarrollo en espiral. El desarrollo del sistema gira en espiral hacia fuera, empezando con un esbozo inicial y terminando con el desarrollo final del mismo.*

La esencia de los procesos iterativos es que la especificación se desarrolla junto con el software. Sin embargo, esto crea conflictos con el modelo de obtención de muchas organizaciones donde la especificación completa del sistema es parte del contrato de desarrollo del mismo. En el enfoque incremental, no existe una especificación completa del sistema hasta que el incremento final se especifica.

Entrega incremental



El modelo de desarrollo en cascada requiere que los clientes de un sistema cumplan un conjunto de requerimientos antes de que se inicie el diseño y que el diseñador cumpla estrategias particulares de diseño antes de la implementación. Los cambios de requerimientos implican rehacer el trabajo de captura de éstos, de diseño e implementación. Sin embargo, la separación en el diseño y la implementación deben dar lugar a sistemas bien documentados susceptibles de cambio. En contraste, un enfoque de desarrollo evolutivo permite que los requerimientos y las decisiones de diseño se retasen, pero también origina un software que puede estar débilmente estructurado y difícil de comprender y mantener.

La entrega incremental es un enfoque intermedio que combina las ventajas de estos modelos. Es un proceso de desarrollo incremental, los clientes identifican, a grandes rasgos, los servicios que proporcionará el sistema. Identifican qué servicios son más importantes y cuáles menos. Entonces, se definen varios incrementos en donde cada uno proporciona un subconjunto de la funcionalidad del sistema. La asignación de servicios a los incrementos depende de la prioridad del servicio con los servicios de prioridad más alta entregados primero.

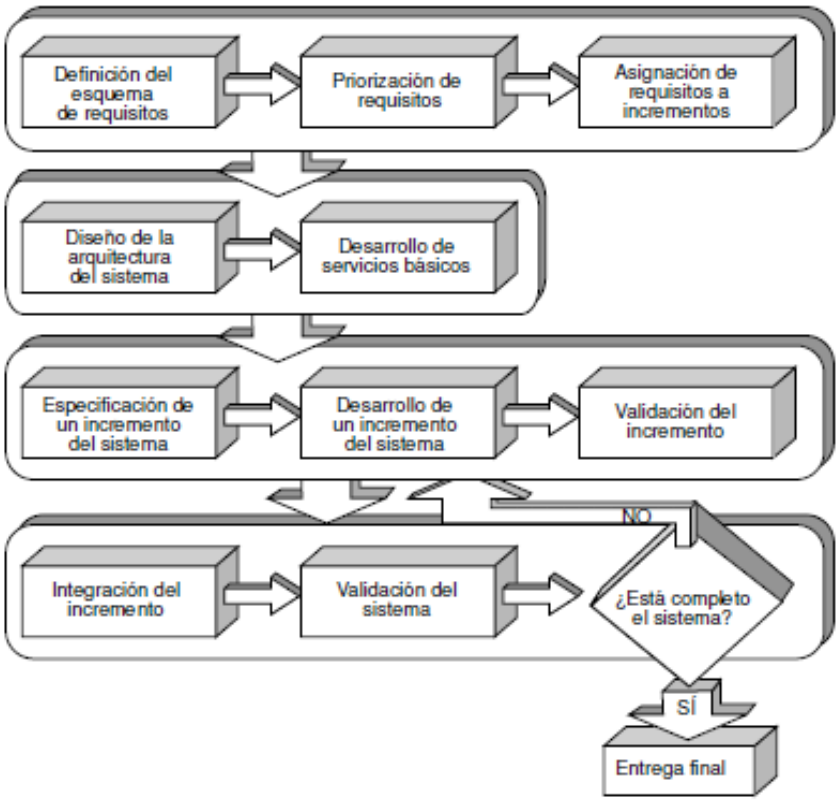
Una vez que los incrementos del sistema se han identificado, los requerimientos para los servicios que se van a entregar en el primer incremento se definen en detalle, y éste se desarrolla. Durante el desarrollo, se puede llevar a cabo un análisis adicional de requerimientos para los requerimientos posteriores, pero no se aceptan cambios en los requerimientos para el incremento actual.

Una vez que un incremento se completa y entrega, los clientes pueden ponerlo en servicio.

Este proceso de desarrollo incremental tiene varias ventajas:

1. *Los clientes no tienen que esperar hasta que el sistema completo se entregue para sacar provecho de él. El primer incremento satisface los requerimientos más críticos de tal forma que pueden utilizar el software inmediatamente.*
2. *Los clientes pueden utilizar los incrementos iniciales como prototipos y obtener experiencia sobre los requerimientos de los incrementos posteriores del sistema.*
3. *Existe un bajo riesgo de un fallo total del proyecto. Aunque se pueden encontrar problemas en algunos incrementos, lo normal es que el sistema se entregue de forma satisfactoria al cliente.*
4. *Puesto que los servicios de más alta prioridad se entregan primero, y los incrementos posteriores se integran en ellos, es inevitable que los servicios más importantes del sistema sean a los que se les hagan más pruebas. Esto significa que es menos probable que los clientes encuentren fallos de funcionamiento del software en las partes más importantes del sistema.*

Sin embargo, existen algunos problemas en el desarrollo incremental. Los incrementos deben ser relativamente pequeños y cada uno debe entregar alguna funcionalidad del sistema.



Ejemplo, Entrega incremental

Desarrollo en espiral

El modelo en espiral del proceso del software fue originalmente propuesto por Boehm (Boehm, 1988). Más que representar el proceso del software como una secuencia de actividades con retrospectiva de una actividad a otra, se representa como una espiral. Cada ciclo en la especial representa una fase del proceso del software. Así, el ciclo más interno podría referirse a la viabilidad del sistema, el siguiente ciclo a la definición de requerimientos, el siguiente ciclo al diseño del sistema, y así sucesivamente.

Cada ciclo de la espiral se divide en cuatro sectores:

1. *Definición de objetivos. Para esta fase del proyecto se definen los objetivos específicos. Se identifican las restricciones del proceso y el proyecto, y se traza un plan detallando de gestión. Se identifican los riesgos del proyecto. Dependiendo de estos riesgos, se planean estrategias alternativas.*
2. *Evaluación y reducción de riesgos. Se lleva a cabo un análisis detallado para cada uno de los riegos del proyecto identificados. Se definen los pasos para reducir dichos riesgos. Por ejemplo, si existe el riesgo de tener requerimientos inapropiados, se puede desarrollar un prototipo del sistema.*
3. *Desarrollo y validación. Después de la evaluación de riesgos, se elige un modelo para el desarrollo del sistema. Por ejemplo, si los riesgos en la interfaz de usuario son dominantes, un modelo de desarrollo apropiado podría ser la construcción de prototipos evolutivos. Si los riesgos de seguridad son la principal consideración, un desarrollo basado en transformaciones formales podría ser el más apropiado, y así sucesivamente. El modelo en cascada puede ser el más apropiado para el desarrollo si el mayor riesgo identificado es la integración de los subsistemas.*
4. *Planificación. El proyecto se revisa y se toma la decisión de si se debe continuar con un ciclo posterior de la especial. Si se decide continuar, se desarrollan los planes para la siguiente fase del proyecto.*

La diferencia principal entre el modelo en espiral y los otros modelos del proceso del software es la consideración explícita del riesgo en el modelo en espiral. [...]

Un ciclo de la espiral empieza con la elaboración de objetivos, como el rendimiento y la funcionalidad. Entonces se enumeran formas alternativas de alcanzar estos objetivos y las restricciones impuestas en cada una de ellas. Cada alternativa se evalúa contra cada objetivo y se identifican las fuentes de riesgo del proyecto. El siguiente paso es resolver estos riesgos mediante actividades de recopilación de información con la de detallar más el análisis, la construcción de prototipos y la simulación. Una vez que se han evaluado los riesgos, se lleva a cabo cierto desarrollo, seguido de una actividad de planificación para la siguiente fase del proceso. [6]

[6] Ian Sommerville. (2006). 7ma Edición. Ingeniería del Software. PEARSON EDUCACION, SA. Madrid – España. Pág. 66 - 69.

Actividades del proceso

"Las cuatro actividades básicas del proceso de especificación, desarrollo, validación y evolución se organizan de forma distinta en diferentes procesos del desarrollo. En el enfoque en cascada, están organizadas en secuencia, mientras que en el desarrollo evolutivo se entrelazan. Cómo se llevan a cabo estas actividades depende del tipo de software, de las personas y de la estructura organizacional implicada. No hay una forma correcta o incorrecta de organizar estas actividades.

Especificación del software

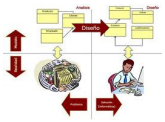
La especificación del software o ingeniería de requerimientos es el proceso de compresión y definición de qué servicios se requieren del sistema y de identificación de las restricciones de funcionamiento y desarrollo del mismo. La ingeniería de requerimientos es una etapa particularmente crítica en el proceso del software ya que los errores en esta etapa originan inevitablemente problemas posteriores en el diseño o implementación del sistema.[...]

Existen cuatro fases principales en el proceso de ingeniería de requerimientos:

1. *Estudio de viabilidad. Se estima si las necesidades del usuario se pueden satisfacer con las tecnologías actuales de software y hardware. El estudio analiza si el sistema propuesto será rentable desde un punto de vista de negocios y si se puede desarrollar dentro de las restricciones de presupuesto existentes. El resultado debe informar si se va a continuar con un análisis más detallado.*
2. *Obtención y análisis de requerimientos. Es el proceso de obtener los requerimientos del sistema por medio de la observación de los sistemas existentes, discusiones con los usuarios potenciales y proveedores, el análisis de tareas, etc. Esto puede implicar el desarrollo de uno o más modelos y prototipos del sistema que ayudan al analista a comprender el sistema a especificar.*
3. *Especificación de requerimientos. Es la actividad de traducir la información recopilada durante la actividad de análisis en un documento que define un conjunto de requerimientos. En este documento se pueden incluir dos tipos de requerimientos: los requerimientos del usuario, que son declaraciones abstractas de los requerimientos del cliente y del usuario final del sistema, y los requerimientos del sistema, que son una descripción más detallada de la funcionalidad a proporcionar.*
4. *Validación de requerimientos. Esta actividad comprueba la veracidad, consistencia y completitud de los requerimientos. Durante este proceso, inevitablemente se descubren errores en el documento de requerimientos. Se debe modificar entonces para corregir estos problemas.*

Por supuesto, las actividades en el proceso de requerimientos no se llevan a cabo de forma estrictamente secuencial. El análisis de requerimientos continúa durante la definición y especificación, y a lo largo del proceso surgen nuevos requerimientos. Por lo tanto, las actividades de análisis, definición y especificación se entrelazan.

Diseño e implementación del software



La etapa de implementación del desarrollo de software es el proceso de convertir una especificación del sistema en un sistema ejecutable. Siempre implica los procesos de diseño y programación de software, pero, si se utiliza un enfoque evolutivo de desarrollo, también puede implicar un refinamiento de la especificación del software.

Un diseño de software es una descripción de la estructura del software que se va a implementar, los datos que son parte del sistema, las interfaces entre los componentes del sistema y, algunas veces, los algoritmos utilizados. Los diseñadores no llegan inmediatamente a un diseño detallado, sino que lo desarrollan de manera iterativa a través de diversas versiones. El proceso de diseño conlleva agregar formalidad y detalle durante el desarrollo del diseño, y regresar a los diseños anteriores para corregirlos.

El proceso de diseño puede implicar el desarrollo de varios modelos del sistema con diferentes niveles de abstracción. Mientras se descompone un diseño, se descubren errores y omisiones de las etapas previas. Esta retroalimentación permite mejorar los modelos de diseño previos.[...]

Las actividades específicas del proceso de diseño son:

1. *Diseño arquitectónico. Los subsistemas que forman el sistema y sus relaciones se identifican y documentan.*
2. *Especificaciones abstractas. Para cada subsistema se produce una especificación abstracta de sus servicios y las restricciones bajo las cuales debe funcionar.*
3. *Diseño de la interfaz. Para cada subsistema se diseña y documenta su interfaz con otros subsistemas.*
4. *Diseño de componentes. Se asignan servicios a los componentes y se diseñan sus interfaces.*
5. *Diseño de la estructura de datos. Se diseña en detalle y especifica la estructura de datos utilizada en la implementación del sistema.*
6. *Diseño de algoritmos. Se diseñan en detall y especifican los algoritmos utilizados para proporcionar los servicios. [...]*

Validación del software

La validación del software o, de forma más general, la verificación y validación (V & V) se utiliza para mostrare que el sistema se ajusta a su especificación y que cumple las expectativas del usuario que lo comprará. Implica procesos de comprobación, como las inspecciones y revisiones, en cada etapa del proceso del software desde la definición de requerimientos hasta el desarrollo del programa. Sin embargo, la mayoría de los costos de validación aparecen después de la implementación, cuando se prueba el funcionamiento del sistema.

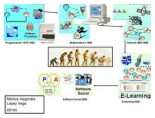
A excepción de los programas pequeños, los sistemas no se deben probar como una simple unidad monolítica.

Las etapas del proceso de prueba son:

1. *Prueba de componentes (o unidades). Se prueban los componentes individuales para asegurarse de que funcionan correctamente. Cada uno se prueba de forma independiente, sin los otros componentes del sistema. Los componentes pueden ser entidades simples como funciones o clases de objetos, o pueden ser agrupaciones coherentes de estas entidades.*
2. *Prueba del sistema. Los componentes se integran para formar el sistema. Este proceso comprende encontrar errores que son el resultado de interacciones no previstas en sus requerimientos funcionales y no funcionales y probar las propiedades emergentes del sistema. Para sistemas grandes, esto puede ser un proceso gradual en el cual los componentes se integran para formar subsistemas que son probados individualmente antes de que ellos mismos se integren para formar el sistema final.*
3. *Prueba de aceptación. Es la etapa final en el proceso de pruebas antes de que se acepte que el sistema se ponga en funcionamiento. Éste se prueba con los datos proporcionados por el cliente más que con datos de prueba simulados. Debido a la diferencia existen entre los datos reales y los de prueba, la prueba de aceptación puede revelar errores y omisiones en la definición de requerimientos del sistema. También puede revelar problemas en los requerimientos donde los recursos del sistema n cumplen las necesidades del usuario o donde el desempeño del sistema es inaceptable.*

Normalmente, el desarrollo de componentes y las pruebas se entrelazan. Los programadores definen sus propios datos de prueba y de forma incremental prueban el código que se va desarrollando. Éste es un enfoque económicamente razonable puesto que el programador es el que mejor conoce los componentes y es, por lo tanto, la mejor persona para generar los casos de prueba. [...]

Evolución del software



La flexibilidad de los sistemas software es una de las principales razones por la que más y más software se incorpora a los sistemas grandes y complejos. Una vez que se decide adquirir hardware, es muy costoso hacer cambios en su diseño. Sin embargo, se pueden hacer cambios al software en cualquier momento durante o después del desarrollo del sistema. Aun cambios importantes son todavía mucho más económicos que los correspondientes de los sistemas hardware.

Históricamente, siempre ha existido una separación entre el proceso de desarrollo y el proceso de evolución del software (mantenimiento del software). La gente considera el desarrollo de software como una actividad creativa en la cual un sistema software se desarrolla desde un concepto inicial hasta que se pone en funcionamiento. Sin embargo, a veces consideran el mantenimiento del software como algo aburrido y sin interés. Aunque los costos de `mantenimiento´ son a menudo varias veces los costos iniciales de desarrollo, el proceso de mantenimiento se considera a veces menos problemático que el desarrollo del software original." [7]

[7] Ian Sommerville. (2006). 7ma Edición. Ingeniería del Software. PEARSON EDUCACION, SA. Madrid – España. Pág. 66 - 76.

Ingeniería del Software Asistida por computadora

"Ingeniería del Software Asistida por computadora (CASE) es el nombre que se le da al software que se utiliza para ayudar a las actividades del proceso del software como la ingeniería de requerimientos, el diseño, el desarrollo de programas y las pruebas. Por lo tanto, las herramientas CASE incluyen editores de diseño, diccionarios de datos, compiladores, depuradores, herramientas de construcción de sistemas, etc.

Las tecnologías CASE proporciona ayuda al proceso del software automatizando algunas de sus actividades, así como proporcionando información acerca del software en desarrollo. Algunos ejemplos de las actividades que se pueden automatizar utilizando CASE son:

1. *El desarrollo de modelos gráficos del sistema como parte de la especificación de requerimientos o del diseño de software.*
2. *La comprensión del diseño utilizando un diccionario de datos que tiene información sobre las entidades y relaciones del diseño.*
3. *La generación de interfaces de usuarios a partir de la descripción gráfica de la interfaz que es elaborada de forma interactiva por el usuario.*
4. *La depuración de programas por medio de la provisión de la información proporcionada por los programas en ejecución.*
5. *La conversión automática de programas de una versión anterior de un lenguaje de programación, como COBOL, a una versión más reciente."* [8]

Para pensar

- Para encarar un desarrollo de software, ¿es posible combinar los diferentes modelos de procesos estudiados?
- Necesariamente si descubrimos que no se cumplen con los requerimientos del usuario, en una pequeña parte, ¿debes seguir con el desarrollo del software?
- Además de los usuarios del sistema, ¿existen otros "entes" que pueden afectar la evolución del software?

[8] Ian Sommerville. (2006). 7ma Edición. Ingeniería del Software. PEARSON EDUCACION, SA. Madrid – España. Pág. 79.