PROCESO Y PROYECTO DE INGENIERIA DE SOFTWARE

*Carlos Barra Peñaloza**

El Ciclo de Vida del Software.

El Ciclo de Vida del Software (CVS) describe todo el proceso de software de un sistema dado, desde su concepción hasta su retiro (concepción, introducción, aceptación o crecimiento, madurez, saturación, obsolescencia y retiro). Esta descripción se realiza mediante un método basado en conceptos y técnicas que conforman el enfoque o paradigma de la forma de construir un sistema de software. Dicho método permite establecer el proceso de software, es decir, las actividades de desarrollo, cuya definición determinará el tipo de soporte requerido para llevarlas a efecto. En la figura 1, se muestran los elementos conceptuales que constituyen el desarrollo de software.

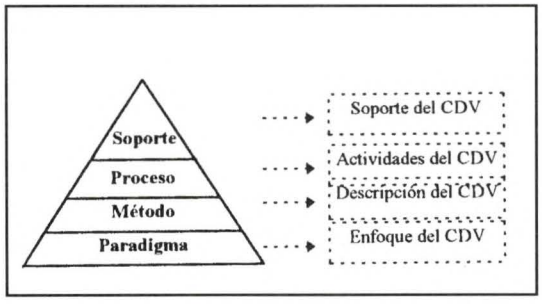

Figura 1: Conceptualización del desarrollo de software.

Básicamente, los métodos existentes se pueden agrupar en dos tipos de aproximaciones o paradigmas: orientados a la función y orientados al objeto.

Los métodos orientados a la función son aquellos que enfocan el tratamiento de las funciones y los datos en forma separada tales como las técnicas de diseño y análisis estructurado y el diseño guiado por requerimientos, donde las funciones son entidades activas que tienen un comportamiento y los datos son entidades pasivas que contienen información que es afectada por las funciones. Típicamente, el sistema se descompone en funciones, entre las cuales los datos son enviados, hasta que eventualmente dichas funciones se convierten en código fuente. Esta separación función/datos se origina de la arquitectura de hardware de von Neumann, donde se enfatiza la separación entre programa y datos.

Los métodos orientados al objeto estructuran el sistema a partir de entidades reales que existen en el dominio del problema, es decir, combinan el comportamiento y los datos del sistema como objetos integrados. Es por esto que la programación orientada al objeto requiere de una aproximación o enfoque diferente, donde el sistema se visualiza como un conjunto de objetos interactuando, cada uno con su propio estado privado, en vez de un conjunto de funciones. Estos objetos, siendo entidades independientes que encapsulan toda la información de estado, se comunican mediante el intercambio de mensajes en vez de compartir variables.

* Capitán de Corbeta Ing. Nv. Eln. Mg. en Ing. Informática.

El proceso de Software.

El reconocimiento de la "crisis del software" a fines de 1960 y la noción de que el desarrollo de software es una disciplina de ingeniería, llevaron a visualizar el proceso de desarrollo de software como el de otras ingenierías, de cuyas actividades se derivó un modelo para el proceso de desarrollo de software que, debido al escalonamiento entre fases, se le conoce como modelo de cascada (figura 2), y que tuvo una entusiasta aceptación en la administración de proyectos de software, ya que ofrecía un medio para hacer el proceso de desarrollo más visible. Este modelo describe el CVS mediante un enfoque sistemático y secuencial del desarrollo de software, abarcando las siguientes actividades:

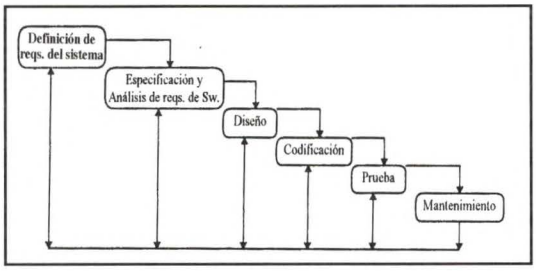

Figura 2: Modelo de cascada del Desarrollo de Software.

Este modelo clásico de desarrollo fue puesto en práctica, pero pronto se evidenció que sólo era apropiado para algunas clases de sistemas de software. Aún cuando la administración encontraba que el modelo era útil para planificación y reporte, las realidades del desarrollo de software no concordaban con las actividades identificadas en el modelo. El proceso de software (las actividades involucradas en el desarrollo y mantenimiento de software) es un proceso complejo y variable que no puede ser fácilmente descrito mediante un modelo simple, por lo que los modelos más detallados del proceso de software aún son objeto de investigación. Aún así, es posible identificar diferentes modelos generales, algunos de los cuales son:

- *Aproximación en cascada (Waterfall).* Este modelo visualiza el proceso de software como el resultado de un número de etapas tales como especificación de requerimientos, diseño de software, implementación, pruebas, etc. Después de la conclusión y aceptación de cada etapa, el desarrollo procede a la etapa siguiente.
- *Programación exploratoria.* Esta aproximación involucra el desarrollo de un sistema que se pone en marcha tan pronto como sea posible, para luego ser modificado hasta que su desempeño sea adecuado. Usualmente esta aproximación se utiliza en el desarrollo de sistemas de inteligencia artificial (AI) en que los usuarios no pueden formular una especificación de requerimientos detallada y donde la adecuación más que la correctitud es el objetivo de los diseñadores.
- *Prototipado (Prototyping).* Esta aproximación es similar a la de programación exploratoria en que la primera fase del desarrollo involucra el desarrollo de un programa para experimentación por parte del usuario. Sin embargo, el objetivo del desarrollo es establecer los requerimientos del sistema. A esto le sigue una reimplementación del software para lograr un sistema con calidad de producción.
- *Transformación formal.* Esta aproximación involucra el desarrollo de una especificación formal del sistema de software y la transformación de dicha especificación, mediante transformaciones que preservan la correctitud, en un programa.
- *Ensamble de componentes reutilizables.* Esta técnica supone que los sistemas, en su mayoría, están constituidos por componentes existentes. El proceso de desarrollo del sistema es más un ensamble que una creación.

Actualmente, el desarrollo de un sistema es un proceso complejo, ya que existen diversos aspectos que deben ser considerados para producir un software confiable, efi-

caz, eficiente y con un rendimiento adecuado, lo que es muy difícil de lograr al enfrentar muchos requerimientos simultáneamente. Puede ser el caso de sistemas de software pequeños en que los requerimientos son directamente traducidos en programas, pero esto es imposible para aquellos sistemas involucrados en operaciones navales: sistemas de mando y control, de armas, etc., en que las características del software, antes mencionadas, son críticas. Por lo tanto, es necesario manejar dicha complejidad de un modo organizado, mediante diferentes modelos del sistema, a través de los cuales se introduce gradualmente esta complejidad en un orden específico y sucesivo. Por otra parte, una manera de administrar el desarrollo de software es dividiéndolo en fases, cada una de las cuales enfrenta diferentes problemas, mediante sus propios modelos conceptuales (del sistema), con notaciones para estos modelos y heurísticas que ayudan al desarrollador en la construcción de dichos modelos; así, el producto de cada fase es la base para la próxima.

Los modelos del CVS antes indicados, definen estas fases en base a diferentes enfoques pero, en general, todos concuerdan en la naturaleza de ellas, aunque es frecuente que existan desacuerdos respecto de los nombres y la ubicación exacta de las fronteras de cada fase.

En la figura 3, se muestran las fases del proceso de desarrollo de software incluyendo los distintos modelos del sistema.

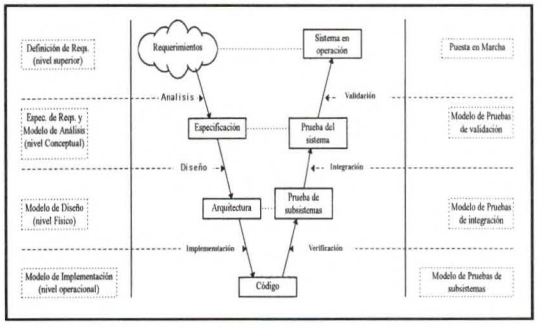

Figura 3: Fases del proceso de software.

La nube representa los requerimientos originales del sistema, cuya documentación normalmente no está bajo el control del desarrollador, ya que a menudo es provista por el usuario. Los rectángulos llenos representan los productos de salida de las diferentes fases de desarrollo y los punteados representan la documentación correspondiente y necesaria para la siguiente fase. Con el tiempo de izquierda a derecha: el flujo de bajada muestra las etapas desde los requerimientos hasta el código (producción), y el flujo de subida muestra el código en su integración y pruebas antes de la puesta en marcha (ensamble). Las líneas punteadas indican que el resultado de la fase de integración, en el flujo de la derecha, concuerdan con, y deben satisfacer, los requerimientos del flujo de la izquierda. Así, el proceso de producción del software es en tres fases, cada una de las cuales apunta al desarrollo de diferentes aspectos del sistema.

• *Definición de Requerimientos del Sistema.* Normalmente, el software es parte de un sistema que conforma su entorno o ambiente de operación, por lo que primero es necesario establecer todos los requerimientos del sistema, para luego asignar algún subconjunto de éstos al software. Esta actividad involucra establecer las fronteras del software a través de las cuales debe interactuar (interfaces) con el entorno operacional (requerimientos de nivel superior del software). El objetivo es lograr una clara comprensión de qué se trata el sistema y cuáles son los conceptos fundamentales.

• *Especificación y Análisis de Requerimientos.* Este proceso involucra una fuerte interacción con los usuarios para capturar a cabalidad los requerimientos específicos del software: objetivos, servicios, funcionalidades, interfaces, restricciones, rendimiento, etc. Estos requerimientos o requisitos se documentan en una Especificación de Reqs. de un modo comprensible tanto para el usuario como para el desarrollador, en base a la cual y mediante un análisis, se construye un modelo conceptual del sistema que especifica su comportamiento.

• *Diseño.* Este proceso traduce los requerimientos en un modelo físico que incluye: estructura de datos, arquitectura del software, funciones y/o procedimientos, interfaces. Esto significa incluir, aparte de las características del entorno operacional, las características de la plataforma de implementación, con lo cual se define cómo se logrará el comportamiento especificado a partir de los componentes de implementación.
• *Implementación.* El diseño se traduce en un modelo comprensible para la máquina, esto es, en un conjunto de programas o unidades de programa, mediante un lenguaje de programación obteniendo el *código.*
• *Pruebas.* Este proceso tiene por propósito asegurar la correctitud de la lógica interna del software y de los resultados que se requiere que la entrada definida produzca.
• *Mantenimiento.* Normalmente, esta es la fase de mayor duración del CVS. El sistema es instalado y puesto en marcha para su operación, durante la cual el mantenimiento involucra la corrección de errores que no se detectaron en etapas anteriores, mejoramiento de la implementación de unidades para lograr mejor rendimiento, aumentar las capacidades del sistema de acuerdo a nuevos requerimientos, etc.

Para llevar a la práctica los procesos de desarrollo del modelo del CVS, es necesario un *método* cuya semántica incluya las fases de dicho modelo. Debido a que el resultado más visible del trabajo de desarrollo de un sistema, es el sistema funcionando, normalmente existe una gran presión por iniciar la codificación lo más pronto posible, lo cual puede conducir a programas difíciles de mantener y que no satisfagan los requerimientos del usuario. Los métodos sistemáticos invierten mayor tiempo en las primeras fases del desarrollo de software y, consecuentemente, el código tarda más en aparecer, por lo que sus beneficios no siempre son apreciados: verificación de requerimientos, clarificación de conceptos, disminución del trabajo de rediseño, mejor descomposición y distribución del trabajo, mejor comunicación entre desarrolladores, menor esfuerzo en mantenimiento.

Una de las propiedades más esenciales, si no la más esencial, de un sistema es que debe tener una estructura estable durante su vida útil; por lo tanto, al definir el proceso de desarrollo del sistema, es importante contar con conceptos y modelos que soporten sólidamente el desarrollo de dicha estructura. Estos fundamentos son los que conforman el *paradigma.* Un *método* describe cómo trabajar con estos conceptos y modelos en un desarrollo ideal. El método también debe permitir el logro de una estructura estable del sistema. Sin embargo, el método no es suficiente para llevar a la práctica el desarrollo del sistema, siendo necesaria una serie de actividades involucradas en todo su ciclo de vida. Estas actividades son el *proceso,* el cual expresa más que el método, ya que describe la administración o gestión completa del producto sobre todo el CVS. El desarrollo de esta gestión se apoya en la utilización de herramientas de soporte que guían o facilitan las actividades propias de administración y de desarrollo.

Proyecto de Software.

Para lograr un proyecto de software exitoso es necesario comprender el ámbito del trabajo a realizar, los riesgos en los que se puede incurrir, los recursos requeridos, las tareas a realizar, los hitos que hay que cumplir, el esfuerzo (costo) y la planificación. Por esto es que la gestión del proyecto de software comienza antes que se inicie el trabajo técnico, continúa a medida que el software evoluciona desde el concepto hasta la realidad y culmina con el retiro del software.

Antes de poder empezar a planificar un proyecto, se deben establecer el ámbito y los objetivos, considerar soluciones alternativas e identificar las restricciones técnicas y de gestión o administración. Sin esta información no es posible obtener estimaciones de costo razonables (y precisas), identificar en forma realista las tareas del proyecto o lograr una plan de trabajo adecuado que pro-

porcione una indicación significativa del estado del proyecto. Una condición necesaria, pero no suficiente, para la gestión y organización de cualquier proyecto de software, es una *buena administración de proyecto.*

Todos los proyectos tienen un aspecto técnico y uno de administración. El propósito de la administración es controlar, dirigir y monitorear el proyecto. El aspecto técnico cubre el qué se debe hacer y el cómo se debe trabajar para desarrollar el sistema o producto; es aquí donde se encuentran las actividades del proceso del software. Sin embargo, los dos aspectos mencionados deben ser correlacionados, para lo cual se definen *hitos* a cumplir. Un hito es un evento concreto, objetivamente definido o determinable, o bien un producto precisamente definido. Estos hitos, normalmente conllevan *revisiones y auditorías* del trabajo realizado hasta el momento. En la figura 4 se muestra un modelo típico de la gestión de un proyecto de software, donde se identifican las siguientes fases del aspecto administrativo del proyecto:

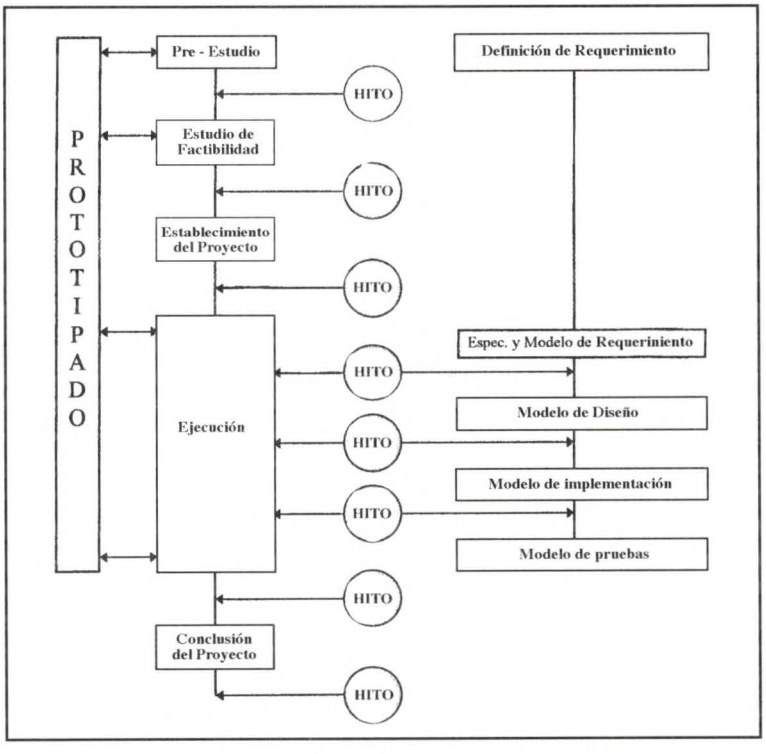

Figura 4: Modelo del proyecto de software.

• *Pre-estudio.* En esta fase se define la tarea mediante el desarrollo y evaluación de distintos tipos de requerimientos, necesidades e ideas con el propósito de juzgar, técnica y económicamente, si el proyecto es practicable.
• *Estudio de Factibilidad.* Se investigan diferentes alternativas técnicas y sus consecuencias, se planifica un programa principal de plazos y recursos y se evalúan los potenciales riesgos en el proyecto.
• *Establecimiento del Proyecto.* El proyecto se organiza, se planifica y se asegura su calidad. Se planifican en detalle los plazos y recursos.
• *Ejecución.* El proyecto es desarrollado de acuerdo a la planificación previa.
• *Conclusión.* El proyecto es completado y se presenta un sumario de propuestas para mejorar el proyecto y métodos de desarrollo utilizados.

En este modelo se ha incluido también el aspecto técnico, esto es, qué hacer en las fases específicas. Estas actividades corresponden al proceso de software y que principalmente se realizan en la fase de ejecución del proyecto, pero también es posible realizar el pre-estudio de la misma forma como una fase de ejecución simplificada.

Un ejemplo típico de la ejecución de otras fases mediante actividades incluidas en el aspecto técnico, es el prototipado en las dos primeras fases para lograr una rápida

comprensión del sistema a desarrollar: durante la fase de pre-estudio, el propósito principal es evaluar aspectos técnicos del sistema (por ejemplo, simulando ciertas partes críticas); durante el estudio de factibilidad, el propósito del prototipado frecuentemente se enfoca a la investigación de alternativas técnicas o para apoyar la elaboración de la especificación de requerimientos, donde el propósito es incrementar la precisión y calidad de los requerimientos y no para obviar fases posteriores. Es extremadamente importante tener presente el propósito del prototipo: ¿Este debe ser refinado hasta producir el sistema deseado? o ¿Sólo apunta a investigar ciertos aspectos?; ambos propósitos son buenos, pero frecuentemente un buen prototipo experimental se "transforma" en el producto real sin que este haya sido el propósito original: el prototipado debe apuntar al incremento de la calidad del producto y no a su detrimento.

Respecto de la figura 4, el primer paso corresponde a la definición del producto, cuya estabilidad es vital antes de que se inicie el desarrollo, ya que éste normalmente representa una gran demanda de recursos (esfuerzo). Una vez que los requerimientos del producto han sido establecidos, se define la arquitectura, es decir, los principales componentes de hardware y software del sistema, además, se hace una planificación del proyecto. Si no es posible decidir si algún componente debe ser realizado en hardware o software, el plan debe incluir una actividad para tomar dicha decisión. El plan también debe incluir la programación y los recursos para todas las actividades necesarias para producir la arquitectura considerando los riesgos. Posteriormente, se desarrollan los componentes de la arquitectura revisando regularmente los riesgos involucrados, lo que eventualmente puede conducir a actualizar la arquitectura.

En principio, este desarrollo se ajusta básicamente al modelo de "cascada", pero en realidad esto es sólo una aproximación, ya que normalmente es necesario iterar las fases debido a requerimientos, cambios o errores evidenciados durante el desarrollo. Por lo tanto, debe existir un mecanismo que permita asegurar la consistencia de las actualizaciones de los distintos modelos de análisis y diseño. Una manera de enfrentar este problema es el desarrollo incremental, el que se muestra esquemáticamente en la figura 5.

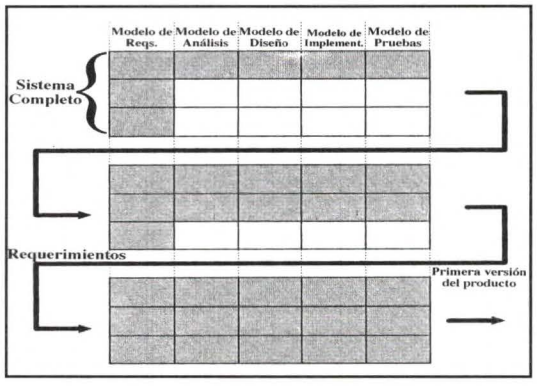

Figura 5: Desarrollo Incremental.

Aún cuando el desarrollo incremental de componentes es sensible, debido al natural deseo del usuario de tener acceso a un sistema en operación lo más pronto posible, tiene la ventaja de que la integración se puede hacer de a poco, evitando así los problemas que se derivan de una integración total tipo "big bang". Por otra parte, el trabajo en diferentes fases puede ser perfectamente traslapado, lo que hace esencial la administración de diferentes versiones de modelos.

Los principales productos del proyecto son: el sistema mismo junto con su documentación de usuario y pruebas de validación; las versiones finales de la arquitectura del sistema junto con las especificaciones de componentes, que constituyen los productos técnicos del proyecto (sin esta documentación, el resto de las etapas del CVS serán innecesariamente costosas).

Consideraciones.

En resumen, ¿para qué sirve el modelo del CVS?

- Organizar, planificar, asignar personal, presupuestar (costos), programar y gestionar el trabajo de un proyecto de software, en el tiempo, espacio y ambiente computacional de la organización.
- Como un lineamiento directivo sobre qué documentos producir para entrega al usuario.
- Como una base para determinar qué herramientas y metodologías de ingeniería de software serán las más apropiadas para apoyar las actividades del CVS.
- Como marco conceptual de patrones de análisis y estimación de la asignación y consumo de recursos durante el CVS.
- Como una información comparativa, descriptiva o directiva de cómo los sistemas de software son lo que son.
- Como una base de conducción de estudios empíricos para determinar qué afecta a la productividad, costo y calidad total del software.

En la Armada existe una vasta experiencia en cuanto a la elaboración y ejecución de proyectos asociados con la adquisición o desarrollo de sistemas, pero en la actualidad, la mayoría de ellos son del tipo hardware-software, es decir, se pueden distinguir dos líneas de desarrollo paralelo, una de las cuales corresponde al software. Si no se comprende claramente la relación entre el proceso y el proyecto de software, es muy probable que una de las principales dificultades a las cuales se verá enfrentado el administrador del proyecto, es precisamente aquella representada por la gestión de software.

Aún cuando los procesos de desarrollo de software poseen características particulares que lo diferencian de otros desarrollos, desde el punto de vista de la ingeniería, éstos pueden ser definidos mediante una metodología que permita modelar el paradigma del CVS. De esta manera es posible establecer los procesos de software y, por consiguiente, las actividades involucradas pueden ser correspondientes con las fases de desarrollo de un proyecto. Es claro que las fases de desarrollo de un proyecto de software no son diferentes de las de cualquier proyecto, pero es pre-requisito comprender el proceso de software para visualizar cómo administrar el desarrollo o la adquisición de software.

Un desarrollo de software exitoso resulta de la administración de procesos, productos y personal involucrados en el proyecto. El mecanismo utilizado para estructurar el proceso y definir las principales actividades asociadas, es el modelo del CVS (paradigma). Este modelo sirve como mecanismo para comunicar cuáles son las tareas a realizar, cuándo y por quién, al personal administrativo, técnico y usuario asociado con el proyecto.

BIBLIOGRAFIA

- Pressman, Roger S.: "Ingeniería del Software, un enfoque práctico",Tercera edición.
- Sommerville, Ian: "Software Engineering", Fourth edition, Addison-Wesley, 1992.
- Walt Scacchi: "Models of Software Evolution: Life Cycle and Process", SEI Curriculum Module SEI-CM-10-1.0, Oct. 1987.
- Jacobson, Ivar: "Object-Oriented Software Engineering", Addison-Wesley, 1994.
- Humphery, Watts S.: "Managing the Software Process", Addison-Wesley, 1989.
- DoD: "Software Development and Documentation", MIL-STD-498, Dec. 1994.
- Barra, Carlos y Visconti, Marcello: "Modelo para Administración de Proyectos de Desarrollo de Software O-O", INFONOR '96, Antofagasta, Nov. 1996.
- Barra, Carlos: "Software e Ingeniería de Software", Revista de Marina N° 1/98.