INGENIERIA DE SOFTWARE EN EL PROCESO DE DESARROLLO DE PRODUCTOS DE SOFTWARE

María Eugenia Valencia.
Profesor Asociado.
Departamento de Información y Sistemas.
Universidad del Valle.
Cali - Colombia.

RESUMEN

Este artículo hace una presentación de los orígenes y definiciones de la Ingeniería de Sofware (I. S.). Establece una caracterización de lo que es un producto de Software y destaca las diferentes etapas que aquella considera deben seguirse en su proceso de desarrollo.

Señala para cada una de las actividades que deben realizarse, los documentos a producirse y sus contenidos al igual que la conformación y características del equipo humano de trabajo.

1. INTRODUCCION.

El término "Ingeniería de Sofware" se introdujo al final de la década del 60 en una conferencia donde se discutió la llamada "Crisis del Software" (1968 Alemania Oriental y 1969 Italia) (7). La aparición de los computadores de la tercera generación, mucho más poderosos que los de la segunda, abrieron un abanico de posibilidades de implementación de aplicaciones que hasta el momento se consideraban imposibles de realizar. Dicha implementación requería desarrolar grandes sistemas de Software.

Los problemas encontrados en el desarrollo de tales sistemas no son en ninguna forma una versión ampliada de los problemas de escribir programas cortos de computador. Se puede ilustrar lo anterior comparando los problemas en el diseño y construcción de un rascacielos contra los de una vivienda unifamiliar de una sola planta. Aunque ambos pertenecen a la clase de edificaciones, un arquitecto nunca diseñaría y construiría el rascacielos pensando en la simple ampliación del diseño y construcción de la casa.

El nivel de complejidad de los programas pequeños (o diseño de casas) puede fácilmente entenderlos una sola persona y los detalles del diseño e implementación (o construcción), manejarlos en su cabeza.

Las especificaciones son informales y el efecto de los cambios es inmediatamente obvio.

Sin embargo, la complejidad de grandes sistemas de Software hace imposible que una sola persona maneje mentalmente y mantenga presente todos los detalles de cada aspecto del proyecto. Se necesita de técnicas formales para establecer requerimientos y especificaciones, técnicas de diseño y una documentación cuidadosa de cada etapa del proyecto.

2. DEFINICIONES.

2.1 Ingeniería de Software (I.S.).

I. Sommerville, define la Ingeriería de Software como aquella que se ocupa del desarrollo de grandes sistemas de Software, que son más grandes de lo que normal-

mente pueden ser desarrolados por un solo individuo, usa principios de ingeniería en el desarrollo de esos sistemas y se compone de aspectos técnicos y no técnicos (8).

Boehm, la define como la aplicación práctica del conocimiento científico en el diseño y construcción de programas para computadores y la documentación asociada requerida para desarrollarlos, usarlos y mantenerlos (4).

La IEEE, en su glosario de términos, la define como el enfoque sistemático para el desarrollo, operación, mantenimiento y eliminación de Software (6).

2.2 Productos de Software.

Son el conjunto de programas, reglas, procedimientos y documentación posible asociada con la computación, así como los datos pertenecientes a la operación de un sistema de cómputo.

2.3 Software de uso personal.

Son aquellos programas escritos por el programador para que le realice una tarea determinada, donde los aspectos de especificación de requerimientos y documentación durante su desarrollo se han tomado informalmente (9).

3. INFORMACION FUNDAMENTAL.

Cuando se habla de desarrollar productos de software debe establecerse una caracterización de éstos, para poderlos diferenciar del "Software de Uso Personal" y entender por qué el desarrollarlos trasciende el simple hecho de escribir un programa.

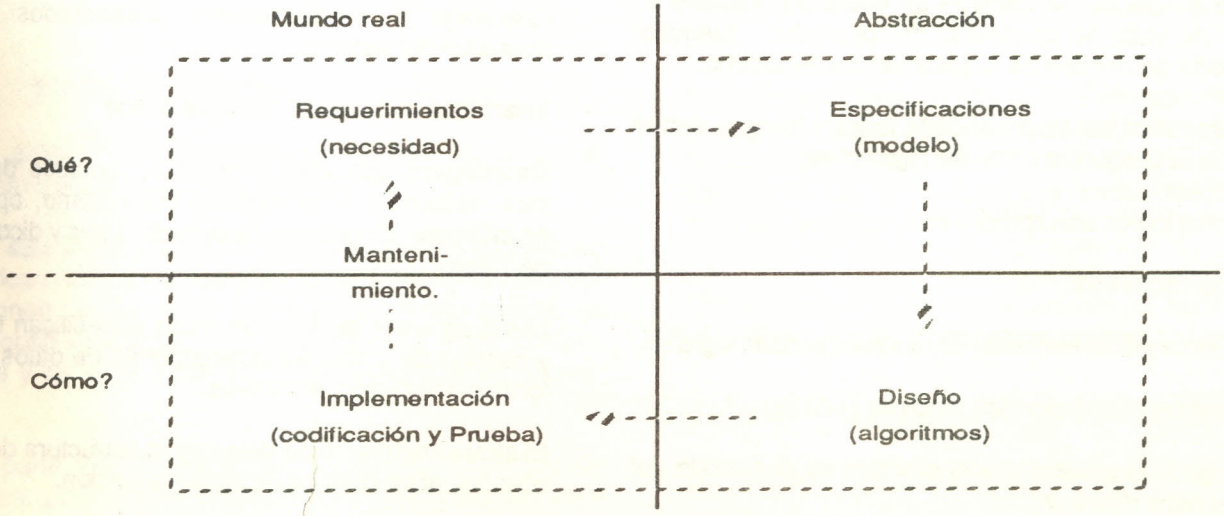
Los productos de Software se caracterizan porque son usados por gran cantidad de usuarios, son diferentes personas quienes desarrollan y dan mantenimiento a tales productos y, requieren de un enfoque más sistemático para su desarrollo que el usado para el Software de uso personal.

Para desarrollarlos:

- Deben determinarse explícitamente limitaciones y necesidades de los usuarios.
- El producto debe diseñarse considerando tanto a quienes lo implementan como a quienes lo usan y le dan mentenimiento.
- Debe prepararse documentación de apoyo como principios de operación, manual de usuario, instrucciones de implementación y guías de entrenamiento.
- Las labores de mantenimiento incluyen solicitudes de análisis de cambio, rediseño y modificación de código objeto, actualización de la documentación con los cambios realizados, pruebas completas de la versión modificada y distribución de la nueva versión en los lugares apropiados.

3.1 Ciclo de Vida de Desarrollo de Software.

El desarrollo de productos de Software supera la habilidad de un solo individuo para entenderlo y realizarlo. Para controlar mejor el desarrollo de proyectos de este tipo, la Ingeniría de Software ha identificado varias etapas por las que debe pasar el proyecto y se conocen como **CICLO DE VIDA DE DESARROLLO DE SOFTWARE.** Se pueden representar así (1):



3.1.1 Etapa de Análisis y Requerimientos.

Aquí se establecen en forma precisa las necesidades en el mundo real que dan origen al proyecto a desarrollar, es decir, se dice QUE es lo que hay que hacer sin mencionar cómo y cuáles son las restricciones que se tienen.

Consta de dos fases: definición del sistema y planeación del proceso de desarrollo.

Definición del Sistema.

Para hacerla se requiere:

- Enunciar el problema a resolver: descripción de la situación actual, restricciones del problema y metas a lograr. Debe emplearse terminología del cliente.
- Justificar la estrategia de solución computarizada para el problema.
- Identificar las funciones y restricciones a realizar, al igual que el equipo, personal y programas que se proporcionan (recursos).
- Determinar objetivos y prioridades para las características del sistema.
- Establecer criterios de aceptación del sistema.
- Conformar unglosario de términos.

Plan de Desarrollo del Proyecto.

Son los mecanismos que se usarán para lograr las metas y requerimientos. El alcanzar un objetivo se puede expresar en términos de los logros importantes a alcanzar a tiempo. Un logro es un hecho significativo en el ciclo de vida de desarrollo del producto. Ejemplo: terminación del análisis de requerimientos del diseño.

Para planear cómo alcanzar cada logro a tiempo, deben responderse preguntas como las siguientes:

- Cuántos logros son apropiados?.
- Cuándo ocurren?.
- Qué recursos se necesitan para alcanzar cada logro?.
- Exáctamente qué significa alcanzar cada logro?.

Para realizar la planeacióndel proceso de desarrollo del proyecto debe definirse:

- Un modelo del ciclo de vida con estimado de tiempos y costos de cada fase.
- Estructura del equipo de trabajo.
- Programa de desarrollo: logros y revisiones.
- Lenguajes de programación y herramientas de desarrollo.
- Requisitos de prueba y forma de demostración.
- Fuentes de información y glosario de términos.

Como memorias de esta etapa debe producirse el DOCUMENTO DE REQUERIMIENTOS, cuya guía de preparación se presenta en el artículo de referencia (2).

3.1.2 Etapa de Especificación de Requerimientos.

Representa la parte abstracta del QUE es lo que hay que hacer. Mientras los requerimientos describen las necesidades en el lenguaje del cliente, las especificaciones hacen una descripción de las necesidades en un lenguaje técnico, sin decir cómo.

Esta etapa se basa en la definición del sistema. Su objetivo es especificar total, consistentemente y sin ambigüedades los requerimientos técnicos del producto. Se concreta en el documento de especificaciones, el cual debe contener (5):

- **Resumen y panorama del producto.**

  Es la elaboración de las características del producto con un resumen del ambiente de procesamiento, operación y mantenimiento presentados en la definición del sistema.

- **Interfaces externas y flujo de datos.**

  Despliegues que verá el usuario, formatos de informes, resumen de comandos del usuario, opciones de informes, diagramas de flujo de datos y diccionario de datos.

  Los diagramas de flujo de datos especifican fuentes y destino de datos, almacenamiento de datos, transformaciones y flujo entre ellos.

  El almacenamiento de datos es la estructura de datos conceptual sin detalles de implementación.

- **Especificación de requerimientos funcionales.**

  Las relaciones entre entradas, acciones y salidas.

- **Especificación de requerimientos de operación.**

  Características de operación tales como tiempo de respuesta de varias actividades, tiempo de procesamiento de varios procesos, número de procesos ejecutados, restricciones de memoria primaria y secundaria, etc.

- **Manejo de excepciones.**

  Incluye acciones a tomar y mensajes de respuestas a enviar ante situaciones no deseadas. Se recomienda preparar una tabla con las condiciones de excepción y sus respuestas.

- **Subconjuntos iniciales y prioridades de implementación.**

  Ejemplo: desarrollo del proyecto por versiones. Versión prototipo, versión modesta y versión mejorada.

- **Modificaciones y mejoras previsibles.**

  Modificaciones al producto por cambio en presupuesto o en el objetivo, por adquisición de nuevo equipo, etc.

- **Criterios de aceptación.**

  Especifican las pruebas funcionales y de rendimiento que se deben realizar, los estándares aplicables al código fuente y documentación interna, y documentos de diseño, plan de pruebas, manual de usuario y procedimientos de instalación y mantenimiento.

3.1.3 Etapa de Diseño.

Es la descripción conceptual de COMO va a ser el producto que cumpla con las especificaciones.

Es en esta etapa donde se establece una estructura, se identifican tipos de datos y su almacenamiento, se descomponen funciones de alto nivel en subfunciones, se establecen relaciones e interconexiones entre componentes y se especifican en forma detallada los algoritmos.

Es la parte más crítica en el proceso de desarrollo de software, porque si el producto está bien diseñado será de fácil implementación y mantenimiento, muy entendible y confiable.

Si por el contrario es mal diseñado, será un producto con altos costos de mantenimiento, que presenta muchas dificultades para realizarle las pruebas y será poco confiable.

Existen varias técnicas de amplio uso en los últimos años para el diseño de productos de software. Ellas incluyen el refinamiento paso a paso, los niveles de abstracción, el diseño estructurado, el desarrollo integrado jerárquico hacia abajo (TOP-DOWN), el desarrollo jerárquico hacia arriba (BOTTON-UP) y la programación estructurada de Jackson. Todas ellas son metodologías de diseño que aportan un marco de trabajo a la actividad creativa del diseño de productos de software.

Como en las fases anteriores, en esta fase debe producirse el DOCUMENTO DE DISEÑO que contendrá:

- Especificaciones de diseño externo, conformadas por:

  Despliegue para el usuario y formatos de informe.
  Resumen de comandos para el usuario.
  Diagramas de flujo de datos detallados.
  Archivos de datos lógicos.
  Diccionario de datos lógicos.
  Formatos lógicos de archivos de datos y bases de datos.

- Especificaciones de diseño arquitectónico, conformadas por:

  Diagramas de estructura.
  Especificaciones de parámetros.
  Estructuras de datos lógicas.
  Descripciones funcionales.

- Especificaciones de diseño detallado, conformadas por:

  Especificaciones de interfaces de subprogramas.
  Prólogo de documentación para cada rutina.
  Seudocódigo para cada rutina.
  Estructura de datos físicos y especificaciones de archivos de datos.

3.1.4 Etapa de Implementación.

Consta de dos fases, la de codificación y la de prueba.

La de codificación, comprende la escogencia de un

lenguaje de programación adecuado que facilite la producción de programas confiables y mantenibles y, la producción del programa guardando un buen estilo de programación (uso de metodologías TOP-DOWN y BOTTOM-UP, documentación interna, uso de nombres significativos para las variables, manejo de módulos y manipulación de errores, entre otros).

La de prueba, tiene como objetivo establecer existencia de errores, localizarlos y corregirlos. Vale recordar que la prueba de software nunca puede demostrar la ausencia de errores.

Existen diversos niveles de prueba y pueden resaltarse como principales los siguientes: prueba de módulo o unidad, prueba de integración, prueba funcional, prueba del sistema y prueba de instalación y prueba de vida. Una buena práctica para realizar procedimientos de prueba es seguir la metodología TOP-DOWN (detalle de procedimientos y tipos de prueba, se presentan en el artículo de la referencia (3)).

En esta etapa se producen, el programa fuente y el documento de prueba que contiene, el tipo de datos usados en los diferentes niveles de prueba y los resultados de las mismas.

A lo largo de todo el proceso de desarrollo se ha acumulado información que facilita la producción de un MANUAL DE USUARIO. Este es un documento fundamental y debe contener:

- Razón del producto y panorámica, terminología, características básicas resumen de formatos de despliegue e informes y descripción del manual;
- Iniciación (con una corrida de muestra);
- Modos de operación (comandos / despliegues / opciones);
- Características avanzadas y sintaxis de comandos y opciones del sistema.

PERSONAL DEL PROYECTO O EQUIPO HUMANO DE TRABAJO.

Como se ha dicho antes, un producto de software es desarrollado por un equipo humano de trabajo.

Según el enfoque de Terry Baker de IBM el equipo de trabajo debe estar constituido por:

- Programador jefe de equipo.
- Programador principal (o de respaldo).
- Bibliotecario.
- Programadores Junior: programadores / analistas.

El equipo es similar a un equipo de cirujanos donde cada miembro tiene adiestramiento crítico y responsabilidades separadas. Aunque es una organización democrática, ellos deben ser una cadena con jerarquía para asegurar una dirección consistente de esfuerzos y objetivos comunes.

El programador jefe tiene el papel principal y el programador de respaldo tiene el segundo lugar en el control. El tercer miembro es el bibliotecario del proyecto que juega un papel de soporte. Todos ellos tienen responsabilidades específicas.

El programador jefe es el líder o director técnico del equipo. Todos los demás miembros deben informarle de sus actividades. El es responsable por decisiones de diseño crítico y por algunas actividades de programación críticas, coordina actividades del equipo, supervisa la calidad del producto final y el desarrollo de cada etapa del proceso.

Debe ser un profesional con experiencia en procesamiento de datos, con suficientes conocimientos técnicos y con probada habilidad de dirección.

El programador de respaldo es tan experimentado como el Programador Jefe, trabaja estrechamente con él y puede reemplazarlo en cualquier momento que se requiera.

El Bibliotecario es el responsable de toda la documentación que se produzca durante el desarrollo del proyecto. El mantiene los módulos de programa fuente, módulos de carga, datos de prueba y demás datos del sistema. Es responsable de todo el control usado para mantener la integridad y separación de los diferentes niveles de los sistemas de biblioteca y proyectos relacionados con las tareas a ejecutar.

Programadores Junior tienen bajo su responsabilidad, junto con los tres miembros antes mencionados el desarrollo detallado de los diversos módulos del programa. A menudo, el trío conformado por el Programador Jefe, el Programador de respaldo y el Bibliotecario del proyecto se encargan de todas las tareas del ciclo de desarrollo, es decir, desde la etapa de requerimientos hasta la de implementación.

CONCLUSIONES

Se puede concluir que existe una clara diferencia entre

lo que es desarrollar software de uso personal, y productos de software.

- La ingeniería de software provee los elementos del enfoque sistemático y del conocimiento científico, para el desarrollo de productos de software.

- Es imperante la conformación de un equipo de trabajo liderado por su programador jefe, donde cada miembro tendrá responsabilidades bien definidas y enmarcadas dentro de un plan de desarrollo trazado priviamente.

- Desarrollar productos de software implica una tarea muchísimo más seria que la simple escritura de un programa. Lleva consigo procesos de análisis, de abstracción, de diseño, implementación y mantenimiento que exigen el uso de metodologías apropiadas y contar con un verdadero trabajo en equipo.

REFERENCIAS BIBLIOGRAFICAS

(1) ABADIA, José Antonio. *Ingeniería de Software.* Univalle 1985.

(2) ABADIA, María Eugenia Valencia de. *Guía para la preparación de un documento de Requerimientos. Revista Heurística,* V1, diciembre 1987.

(3) ABADIA, María Eugenia Valencia de. *Procedimientos para prueba de Software.* Publicaciones ICESI No. 18, Enero- Marzo 1986.

(4) BOEHM, B. W. *Software Engineering Economics.* Prentice Hall 1981.

(5) FAIRLEY, Richard. *Ingeniería de Software.* McGraw-Hill, 1987.

(6) IEEE *Standard Glossary of Software Engineering Terminology,* IEEE Standard 729-1983.

(7) KING, David. *Current Practices in Software Development.* Yourdon Press 1984.

(8) SOMMERVILLE I. *Software Engineering.* Addison Wesley Publishing Co. 1982.

(9) ZELKOWITZ - SHAW - GANNON. *Principles of Software Engineering and Design.* Prentice Hall. 1979.