Universidad De San Carlos de Guatemala Facultad de Ingeniería Escuela de Ciencias y Sistemas Introducción a la Programación de Computadoras 1			**A**
Alumno:	Jefferson Geovanny Moreno Pérez		
Carne:	201603047	Fecha:	15/12/17

CICLO DE VIDA DEL SOFTWARE

Cuando surgió la necesidad de adaptar los sistemas informáticos a las exigencias del mercado, el programador realizaba un relevamiento de las solicitudes de quien necesitaba cierto programa o producto software, y con aquellos requerimientos bajo el brazo comenzaba la dura tarea de codificar. Esta tarea no estaba administrada, supervisada o gestionada de ningún modo, por lo que se iba corrigiendo a medida que surgían los errores, tantos los lógicos provenientes de la codificación, como los de requerimientos solicitados por el cliente o usuario final.

En la década de 1970 los programas fueron creciendo en complejidad, por lo que la antigua técnica de code & fix (codificar y corregir) terminó quedando obsoleta. Esta técnica se basaba en requerimientos ambiguos y sin especificaciones puntuales. Al no seguir normas para el proyecto, el cliente o usuario sólo impartían especificaciones muy generales del producto final. Se programaba, se corregía, y se volvía a programar sobre la misma marcha del proyecto. El ciclo de vida de este tipo de proyectos finalizaba cuando se satisfacían las especificaciones, no sólo las primeras por las cuales nació la necesidad del programa, sino también todas aquellas que fueron surgiendo sobre la marcha.

Esta técnica tiene las ventajas de no gastar recursos en análisis, planificación, gestión de recursos, documentación, etc., y bien sabemos que es muy cómoda y muchas veces recomendable cuando el proyecto es muy pequeño y es llevado adelante por uno o dos programadores. Por otro lado, cuando el sistema no es pequeño o es más complejo de lo creído (tengamos en cuenta que no hubo análisis) nos trae desventajas en lo que se refiere a costo de recursos, que siempre será mayor del previsto, aumentará el tiempo de desarrollo y la calidad del código será bastante dudosa.

Dado esto nace la necesidad de tener una metodología de software:

La metodología para el desarrollo de software es un modo sistemático de realizar, gestionar y administrar un proyecto para llevarlo a cabo con altas posibilidades de éxito. Esta sistematización nos indica cómo dividiremos un gran proyecto en módulos más pequeños llamados etapas, y las acciones que corresponden en cada una de ellas, nos ayuda a definir entradas y salidas para cada una de las etapas y, sobre todo, normaliza el modo en que administraremos el proyecto. Entonces, una metodología para el desarrollo de software son los procesos a seguir sistemáticamente para idear, implementar y mantener un producto software desde que surge la necesidad del producto hasta que cumplimos el objetivo por el cual fue creado.

Desde un punto de vista general puede considerarse que el ciclo de vida de un software tiene tres etapas claramente diferenciadas, las cuales se detallan a continuación:

- **Planificación**: Idearemos un planeamiento detallado que guíe la gestión del proyecto, temporal y económicamente.

- **Implementación**: Acordaremos el conjunto de actividades que componen la realización del producto.

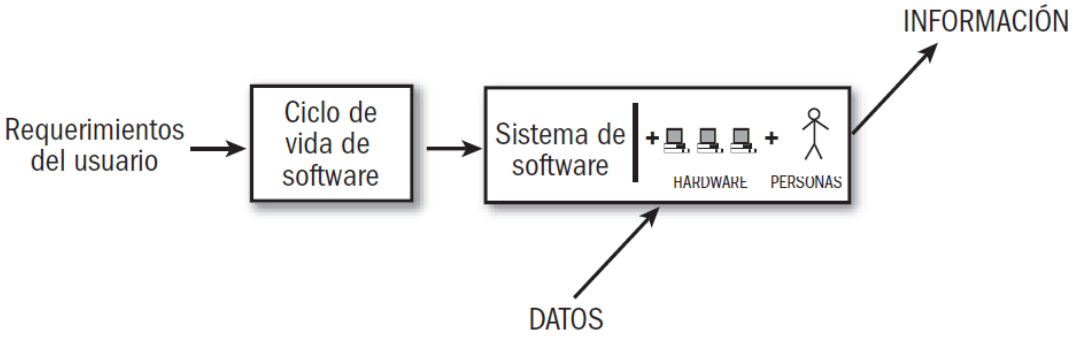
- **Puesta en producción:** Nuestro proyecto entra en la etapa de definición, allí donde se lo presentamos al cliente o usuario final, sabiendo que funciona correctamente y responde a los requerimientos solicitados en su momento. Esta etapa es muy importante no sólo por representar la aceptación o no del proyecto por parte del cliente o usuario final sino por las múltiples dificultades que suele presentar en la práctica, alargándose excesivamente y provocando costos no previstos.

A estas tres grandes etapas es conveniente añadir otras dos que, si bien pudieron enunciarse junto a las otras, es conveniente hacer una diferenciación ya que se tiende a menospreciarlas o a no darles la importancia que requieren.

- **Inicio**: éste es el nacimiento de la idea. Aquí definimos los objetivos del proyecto y los recursos necesarios para su ejecución. Hacia dónde queremos ir, y no cómo queremos ir. Las características implícitas o explícitas de cada proyecto hacen necesaria una etapa previa destinada a obtener el objetivo por el cual se escribirán miles o cientos de miles de líneas de código. Un alto porcentaje del éxito de nuestro proyecto se definirá en estas etapas que, al igual que la etapa de debugging, muchos líderes de proyecto subestiman.

- **Control en producción:** Control del producto, analizando cómo el proceso difiere o no de los requerimientos originales e iniciando las acciones correctivas si fuesen necesarias. Cuando decimos que hay que corregir el producto, hacemos referencia a pequeñas desviaciones de los requerimientos originales que puedan llegar a surgir en el ambiente productivo. Si nuestro programa no realiza la tarea para lo cual fue creada, esta etapa no es la adecuada para el rediseño. Incluimos también en esta etapa el liderazgo, documentación y capacitación, proporcionando directivas a los recursos humanos, para que hagan su trabajo en forma correcta y efectiva.

Dadas la etapa anteriormente mencionada podemos proceder a crear un modelo en el cual incluiremos las etapas con su objetivo claro ya que, en cada una de las etapas de un modelo de ciclo de vida, se pueden establecer una serie de objetivos, tareas y actividades que lo caracterizan.



- **Expresión de necesidades:** esta etapa tiene como objetivo el armado de un documento en el cual se reflejan los requerimientos y funcionalidades que ofrecerá al usuario el sistema a implementar (qué, y no cómo, se va a implementar).

- **Especificaciones:** formalizamos los requerimientos; el documento obtenido en la etapa anterior se tomará como punto de partida para esta etapa.

- **Análisis:** determinamos los elementos que intervienen en el sistema a desarrollar, su estructura, relaciones, evolución temporal, funcionalidades, tendremos una descripción clara de qué producto vamos a construir, qué funcionalidades aportará y qué comportamiento tendrá.

- **Diseño:** ya sabemos qué hacer, ahora tenemos que determinar cómo debemos hacerlo (¿cómo debe ser construido el sistema en cuestión? definimos en detalle entidades y relaciones de las bases de datos, seleccionamos el lenguaje que vamos a utilizar, el Sistema Gestor de Bases de Datos, etc.).

- **Implementación:** empezamos a codificar algoritmos y estructuras de datos, definidos en las etapas anteriores, en el correspondiente lenguaje de programación o para un determinado sistema gestor de bases de datos. En muchos proyectos se pasa directamente a esta etapa; son proyectos muy arriesgados que adoptan un modelo de ciclo de vida de code & fix (codificar y corregir) donde se eliminan las etapas de especificaciones, análisis y diseño con la consiguiente pérdida de control sobre la gestión del proyecto.

- **Debugging:** el objetivo de esta etapa es garantizar que nuestro programa no contiene errores de diseño o codificación. En esta etapa no deseamos saber si nuestro programa realiza lo que solicitó el usuario, esa tarea le corresponde a la etapa de implementación. En ésta deseamos encontrar la mayor cantidad de errores. Todas los programas contienen errores: encontrarlos es cuestión de tiempo. Lo ideal es encontrar la mayoría, si no todos, en esta etapa. También se pueden agregar testeos de performance.

- **Validación:** esta etapa tiene como objetivo la verificación de que el sistema desarrollado cumple con los requerimientos expresados inicialmente por el cliente y que han dado lugar al presente proyecto. En muchos proyectos las etapas de validación y debugging se realizan en paralelo por la estrecha relación que llevan. Sin embargo, tenemos que evitar la confusión: podemos realizarlos en paralelo, pero no como una única etapa.

- **Evolución:** en la mayoría de los proyectos se considera esta etapa como Mantenimiento y evolución, y se le asigna, no sólo el agregado de nuevas funcionalidades (evolución); sino la corrección de errores que surgen (mantenimiento). En la práctica esta denominación no es del todo errónea, ya que es posible que aun luego de una etapa de debugging y validación exhaustiva, se filtren errores.

Lo que buscamos guiándonos con una metodología es prolijidad, corrección y control en cada etapa del desarrollo de un programa. Lo que nos permitirá una forma sistemática para poder obtener un producto correcto y libre de errores.

COMENTARIO

Las etapas para la programación de software surgen a partir, de que cada dia los sistemas informáticos son mas complejos, ya que antes se usaba la técnica de codificar y corregir la cual consistía vagamente en programar corregir y seguir programando hasta que se satisfacían las necesidades del usuario final, cabe aclarar que no solo las ideas por las que nació la necesidad del programa, sino que también aquellas que fueron emergiendo durante la programación del mismo.

Entonces una metodología para programar a lo que nos ayuda es a administrar, supervisar y gestionar nuestro proyecto. Además de darnos una guía con los pasos a seguir para que nuestro producto en este caso un software cumpla las necesidades del Usuario final.

Un Modelo lo podemos desglosar en etapas, a cada etapa le podemos establecer ya sea objetivos, tareas, actividades o análisis. Las etapas de un ciclo de software para que sea exitoso son: Requerimientos, Especificaciones, Análisis, Diseño, Implementación, Depuración, Validación y Evolución o Mantenimiento.

Donde: Los Requerimientos serán todas aquellas ideas que se van a implementar en el software, aquí también irá todo aquello que se supondrá podrá realizar el programa al final. Las Especificaciones serán los requerimientos ya en un documento formal donde ambas partes estarán de acuerdo, este será el punto de partida para la siguiente etapa. El Análisis, aquí nos tomaremos el tiempo de pensar que relaciones existen y cuales son las funciones que tendremos que implementar en el software, esto también nos dará una idea clara de que es lo que vamos a hacer

y para que lo vamos a hacer. Diseño, en esta parte, ya que sabemos que hacer ahora tenemos que analizar como lo debemos de hacer, por ejemplo, si tiene que tener una base de datos o unas listas, en esta parte podemos utilizar diagramas UML para hacer un diagrama detallado de como realizaremos el sistema. En la Implementación nos tocara la tarea de codificar lo que se hizo en las etapas anteriores, cabe mencionar que muchos se saltan a esta parte y terminan en círculos o con preguntas de ¿con que continuo? o con funciones inservibles para la finalidad del programa. En la etapa de Depuración nuestro objetivo será encontrar el mayor numero de errores en nuestro programa, ya que nuestro deber es garantizar que el programa es funcional y sin errores. La mejor forma de encontrar errores es testeando el programa en situaciones reales y ver como reacciona a estas. En la Validación nosotros como programadores garantizaremos que el sistema cumple con los requerimientos del cliente. En el Mantenimiento deberemos actualizar el sistema con la corrección de algún error o el agregar funcionalidades nuevas que no se tenían cuando se realizo el sistema.

Dado esto tendremos un programa de software exitoso en su totalidad.

En la vida diaria como programadores las etapas para desarrollar un software nos ayudaran a separar el trabajo ya que empezar a programar a lo “loco” solo nos traerá crear líneas y líneas de código algunas inservibles y llegaremos a un punto donde no sabremos a donde vamos o donde seguir.

En los proyectos de la Universidad utilizaría las etapas para crear software y quedarían de la siguiente manera:

- ✓ **Requerimientos:** Que es lo que se pide juego, programa, sistema etc.
- ✓ **Especificaciones:** Enunciado de la practica o proyecto.
- ✓ **Análisis:** Cuales son las clases o las variantes que voy a implementar para la solución del programa.
- ✓ **Diseño:** Haría uso de diagramas UML para hacer un bosquejo detallado de las clases métodos y funciones que utilizare.
- ✓ **Implementación**: Se empezaría a codificar el programa para suplir el enunciado dado.
- ✓ **Depuración:** Se testearía el juego para ver los errores que podría tener el sistema, por ejemplo, un mal cálculo, errores, ciclos etc.
- ✓ **Validación**: Se seguiría probando el programa hasta constatar que tiene el menor número de errores.
- ✓ **Evolución o Mantenimiento:** Si el programa tiene algún error o necesita otra función en esta parte se suplirá esta necesidad.