Fragmento de lecturas de la OVI Etapas del desarrollo de software

**MPORTANCIA DEL DESARROLLO DE SOFTWARE**



Fecha: 23 abr. 2017   
Categorías:

Publicado por: amelo  
Leído por: 2674 visitantes

Para poder entender por qué es importante saber programar en el mundo moderno, uno debe comenzar por entender qué es una computadora. **Una computadora es una herramienta muy básica para hacer tareas repetitivas de forma más eficiente.**Una computadora no es capaz de analizar un problema y obtener una solución. Los seres humanos, por otra parte, somos realmente buenos para analizar y resolver problemas, pero nos aburren las tareas repetitivas.

Los seres humanos, usando sus habilidades analíticas y de resolución de problemas, inventaron los algoritmos (grupo limitado de instrucciones que trabajan con una serie de datos de entrada para producir un resultado) para aquellos problemas que pudieran ser de alguna manera computables. Una computadora puede seguir esas instrucciones y así producir una solución.

Programar es un proceso con tres (3) fases muy claramente definidas:

1. Fase de Identificación y Solución
2. Fase de Implementación
3. Fase de Mantenimiento

Todas estas fases son importantes para que los desarrolladores de software puedan resolver problemas. Muchos desarrolladores creen que programar es solo definir los algoritmos, pero eso es un error. Otros piensan que programar es traducir los algoritmos en un lenguaje de programación, que pueda ser compilado o interpretado y finalmente entendido por una computadora, pero eso solo es parte del proceso de programación. En mi humilde opinión creo que es la parte más sencilla.

En resumen, el desarrollo de software es el proceso de **resolver problemas con el uso de una computadora**.

En el mundo moderno, con la cantidad de problemas y su complejidad cada vez mayor, el desarrollo de aplicaciones, es una herramienta para obtener soluciones precisas, dándole las tareas repetitivas a la computadora y logrando así la "automatización", de los procesos que todavía requieren la intervención de las personas en tareas repetitivas.

El **lenguaje de programación** es importante porque define la relación, la semántica y la gramática, que permite a los programadores comunicarse efectivamente con las máquinas que programan.

Un lenguaje de programación cumple varios propósitos:

* Permite instruir a la computadora sobre qué hacer en una forma legible por humanos
* Facilita al programador estructurar las instrucciones en funciones, procedimientos, etc.
* Permite que el programa se divida en "trozos" que pueden ser desarrollados por grupos de desarrolladores
* Proporciona portabilidad: las instrucciones de bajo nivel de una computadora serán diferentes de las de otra computadora.

Los programadores informáticos entienden cómo funcionan los sistemas informáticos, incluyendo las limitaciones del sistema, y así pueden establecer expectativas realistas y trabajar en torno a esas limitaciones para maximizar plenamente el uso del equipo y sus accesorios.

Con el pasar del tiempo y el creciente uso de Internet como medio de comunicación y trabajo, los programas han evolucionado de funcionar únicamente en una computadora para empezar a trabajar en equipos mucho más potentes y al que pueden acceder múltiples usuarios usando conexiones de red. **Estas aplicaciones las llamamos Aplicaciones Web**.

El desarrollo de software y las redes ha permitido nuevas aplicaciones web interactivas que tienen acceso a los recursos del sistema y proporcionan el mismo nivel de control que las aplicaciones de escritorio.

Los principios de programación de computadoras implementados hoy probablemente influirán en cómo las tecnologías tales como reconocimiento de voz, inteligencia artificial y otras tecnologías sofisticadas cambiarán en el futuro y cómo se aplicarán a nuestras vidas diarias. Por ejemplo, la tendencia hacia la automatización de las búsquedas y compras en Internet para ser más localizada está en curso. Mientras que las plataformas de hardware desarrolladas jugarán un papel importante, la tecnología informática probablemente estará en el centro de todo y la programación de sistemas futuros será un aspecto importante.

No importa qué tipo de trabajo haga un desarrollador de software, porque siempre tendrá responsabilidades generales para realizar en su papel como programador. La primera es desarrollar nuevos métodos de programación de computadoras. Estos pueden incluir desde métodos básicos de programación de computadoras hasta métodos más profundos. Otra responsabilidad general que un programador de computadoras debe asumir en su rol diario de trabajo es el desarrollo de los programas mismos. Este es un concepto extremadamente importante asociado con el papel de un programador de computadoras ya que estas **son las personas que construyen los programas útiles que son utilizados por muchos**.

* . El ciclo de vida.

En principio, el ciclo de vida de un proyecto software incluye todas las acciones que se realizan sobre él desde que se especifican las características que debe tener, hasta que se mantiene en operación. A veces (aunque no será éste nuestro caso) se incluyen en el ciclo de vida las modificaciones que pueden realizarse al sistema para adaptarse a nuevas especificaciones. Podría pensarse que el ciclo de vida de un programa no tiene por qué seguir un desarrollo "lineal", entendiendo como tal una sucesión de etapas. En principio, las distintas actividades que se realizan son bastante independientes, y pueden llevarse (hasta cierto punto) en paralelo. Por ejemplo, para empezar a codificar hay que tener mínimamente claras las especificaciones que hay que cumplir. Pero (aunque no es una buena decisión, como veremos más adelante), podría pensarse en comenzar la producción de código mientras se completan las especificaciones, para poder irlo probando, por ejemplo. Más adelante se harían las modificaciones necesarias. Pero si el desarrollo de productos software ya es algo complejo en sí mismo (véase el capítulo sobre Medidas o Métricas de la Complejidad del Software), aún lo complicaremos más si intentamos "hacerlo todo a la vez", sin seguir una cuidadosa y detallada planificación. Y esto es precisamente lo que pretenden los modelos del ciclo de vida del software: simplificar en lo posible la gestión del proceso de desarrollo. La meta está en añadir la mínima complejidad que sea posible a la que de por sí ya implica el software. Desde el punto de vista del esquema HxIxO->IO, podríamos decir que los modelos del ciclo de vida son un instrumento conceptual (I) que permite a la persona encargada (H) de gestionar un desarrollo de software (el O será por tanto el propio proceso de desarrollo) tratar con un problema más sencillo (el IO resultante). Para ello, estos modelos dividen el proceso de desarrollo en unas cuantas etapas bien diferenciadas, y definen los posibles caminos por los que se deben relacionar. Además intentan que estos caminos lleven a un "progreso lineal", en el sentido de que antes de comenzar una etapa se haya concluido exitosamente (con las especificaciones cumplidas) la anterior. Sin embargo, esto no es siempre posible, y hay que recurrir a iteraciones (por ejemplo, entre el diseño y la codificación), que nos lleven mediante aproximaciones sucesivas a cumplir con los objetivos de la mejor forma posible. Desde el punto de vista jerárquico (véase el capítulo sobre las jerarquías) esta división en etapas puede verse como una jerarquía multicapa de toma de decisiones. Así, cada una de las etapas (capa de decisiones) termina cuando, tras haber hecho todas las elecciones necesarias, se han cumplido los objetivos marcados, sentando las bases para la siguiente etapa. Al dividirse el problema en estas capas, en cada momento del desarrollo nos enfrentamos con una complejidad menor (únicamente la debida a cada capa, ya que las anteriores habrán sido satisfactoriamente resueltas).

Análisis de requerimientos de los sistemas de información - LI.SIS.12

* La dirección de Tecnologías y Sistemas de la Información o quien haga sus veces debe incorporar un proceso formal de análisis y gestión de requerimientos de software en el ciclo de vida de los sistemas de información de manera que se garantice su trazabilidad y cumplimiento.

Código: LI.SIS.12

Ámbito: Ciclo de vida de los Sistemas de Información

Dominio: Sistemas de Información

Lineamiento:

La dirección de Tecnologías y Sistemas de la Información o quien haga sus veces debe aplicar un proceso formal de manejo de requerimientos, que incluya la identificación, la especificación y el análisis de las necesidades funcionales y no funcionales, la definición de los criterios de aceptación y la trazabilidad de los requerimientos a través del ciclo de vida de los sistemas de información.

Introduccion al Desarrollo del Software

El desarrollo del software puede ser un tema bastante complejo si así lo queremos, este día vamos a tratar de reducir esta complejidad a algo comprehensible en unas lineas.

Al momento de definir software podríamos verlo como una herramienta que nos sirve para agilizar nuestro trabajo, en los juegos que usamos en Facebook, las aplicaciones de nuestro smartphone, todo lo que usamos en la computadora fue creado por un equipo de desarrollo, pequeño, grande, distribuido o local, pero la pregunta que nos plantearemos es: Que hay detrás de este herramienta, como se construyo esta aplicación Es claro que hay un gran trabajo detrás de cada botón, detrás de cada información que mandamos a guardar.

Como todo proyecto el software tiene un ciclo para desarrollarse y consta de una serie de pasos que se van completando en diferentes tiempo; este ciclo de desarrollo de software depende directamente de la metodología que utilizamos para este desarrollo, y no es mas que una serie de pasos/tareas que tenemos que seguir como en cualquier otro proyecto, no hay nada escondido, nada mágico excepto la gran mente del equipo de desarrollo y las creaciones para tener una experiencia única al utilizar la aplicación o el paquete de software.

ETAPAS DE DESARROLLO DE SOFTWARE

La ingeniería de software requiere llevar a cabo numerosas tareas agrupadas en etapas, al conjunto de estas etapas se le denomina ciclo de vida . Las etapas comunes a casi todos los modelos de ciclo de vida son las siguientes:

Etapa de Análisis Análisis:

Es el proceso de investigar un problema que se quiere resolver. Definir claramente el Problema que se desea resolver o el sistema que se desea crear. Identificar los componentes principales que integrarán el producto.

Etapa de Diseño Diseño:

Es el proceso de utilizar la información recolectada en la etapa de análisis al diseño del producto. La principal tarea de la etapa de diseño es desarrollar un modelo o las especificaciones para el producto o Componentes del Sistema.

Etapa de Desarrollo Desarrollo:

Consiste en utilizar los modelos creados durante la etapa de diseño para crear los componentes del sistema.

Etapa de Pruebas o Verificación Prueba:

Consiste en asegurar que los componentes individuales que integran al sistema o producto, cumplen con los requerimientos de la especificación creada durante la etapa de diseño.

Etapa de Implementación o Entrega Implantación:

Consiste en poner a disposición del cliente el producto.

Etapa de Mantenimiento Mantenimiento:

Consiste en corregir problemas del producto y re- liberar el producto como una nueva versión o revisión (producto mejorado).

Etapa final EOL (End-of-Life) :

El fin del ciclo del producto consiste en realizar todas las tareas necesarias para asegurar que los clientes y los empleados están consientes de que el producto ya no será vendido ni soportado.

Modelos y metodologías para el desarrollo de software

2.11.1 Ingeniería de software

Según Sommerville (2005), para muchas personas el software son solo programas de computadora, sin embargo nos comenta que son todos aquellos documentos asociados a la configuración de datos que se necesitan para hacer que estos programas operen de manera adecuada. Estos productos de software se desarrollan para algún cliente en particular o para un mercado en general. Para el diseño y desarrollo de proyectos de software se aplican metodologías, modelos y técnicas que permiten resolver los problemas. En los años 50 no existían metodologías de desarrollo, el desarrollo estaba a cargo de los propios programadores. De ahí la importancia de contar con analistas y diseñadores que permitieran un análisis adecuado de las necesidades que se deberían de implementar.

Aun así los resultados eran impredecibles, no se sabía la fecha exacta en que concluiría un proyecto de software, no había forma de controlar las actividades que se estaban desarrollando. Tampoco se contaba con documentación estandarizada. El nacimiento de técnicas estructuradas es lo que da origen al desarrollo de aplicaciones a través de métodos de ingeniería. La informática aporta herramientas y procedimientos que se apoyan en la ingeniería de software con el fin de mejorar la calidad de los productos de software, aumentar la productividad y trabajo de los ingenieros desarrolladores de software, facilitar el control del proceso de desarrollo de software y suministrar a los desarrolladores las bases para construir software de alta calidad en una forma eficiente, Gacitúa(2003).

El objetivo principal que busca la ingeniería de software es convertir el desarrollo de software en un proceso formal, con resultados predecibles, que permitan obtener un producto final de alta calidad y satisfaga las necesidades y expectativas del cliente. Según Gacitúa (2003), la Ingeniería de Software es un proceso intensivo de conocimiento, que abarca la captura de requerimientos, diseño, desarrollo, prueba, implantación y mantenimiento. Generalmente a partir de un complejo esquema de comunicación en el que interactúan usuarios y desarrolladores, el usuario brinda una concepción de la funcionalidad esperada y el desarrollador especifica esta funcionalidad a partir de esta primera concepción mediante aproximaciones sucesivas. Este ambiente de interacción motiva la búsqueda de estrategias robustas para garantizar que los requisitos del usuario serán descubiertos con precisión y que además serán expresados en una forma correcta y sin ambigüedad, que sea verificable, trazable y modificable.

El término ingeniería del software empezó a usarse a finales de la década de los sesenta, para expresar el área de conocimiento que se estaba desarrollando en torno a las problemáticas que ofrecía el software. En esa época, el crecimiento espectacular de la demanda de sistemas de computación cada vez más y más complejos, asociado a la inmadurez del propio sector informático (totalmente ligado al electrónico) y a la falta de métodos y recursos, provocó lo que se llamó la crisis del software. Durante esa época muchos proyectos importantes superaban con creces los presupuestos y fechas estimados. La crisis del software finalizó pues se comenzó a progresar en los procesos de diseño y metodologías.