TIC02 INTRODUCCIÓN A LA PROGRAMACIÓN EL CICLO DE VIDA DEL DESARROLLO DEL SOFTWARE

1.- INTRODUCCION

Un sistema (RAE) es un conjunto de cosas que relacionadas entre sí ordenadamente contribuyen a determinado objeto.

Los sistemas que ha de concebir y construir el ingeniero en informática son sistemas basados en computadores. Estos sistemas contienen uno o más computadores dedicados a tareas de control del conjunto.

Un **sistema informático** es aquel sistema que se encarga de recibir, procesar y transmitir la información.

A lo largo del tema, se hará referencia fundamentalmente a sistemas informáticos compuestos principalmente por ordenadores y sus elementos periféricos tradicionales. Dentro de un sistema informático se pueden distinguir los elementos materiales, que constituyen el hardware del sistema, de los programas que gobiernan el funcionamiento del ordenador, y que

constituyen el software del mismo.

Los sistemas informáticos realizan tareas de tratamiento de la información. Estas tareas consisten fundamentalmente en el almacenamiento, procesamiento y presentación de información.

Estas tareas están relacionadas cono los diferentes tipos de elementos que intervienen en un sistema informático: los componentes hardware, los componentes software, los datos y, los usuarios del sistema.

El desarrollo de un sistema informático de forma global, que es la actividad de la ingeniería de sistemas basados en computadores, consiste por tanto, en decidir qué elementos hardware se utilizarán, qué elementos software han de ser adquiridos o desarrollados, y qué personas se necesitan para operar el sistema y, al mismo tiempo, repartir y asignar las actividades de tratamiento de la información entre los diferentes elementos componentes.

2. CARACTERÍSTICAS DEL SOFTWARE

Tradicionalmente se vienen clasificando los elementos componentes de un sistema informático en dos grandes grupos: el hardware y el software.

Los componentes hardware se obtienen mediante un proceso de fabricación en que se obtienen sucesivas copias de un producto patrón diseñado previamente. Esta operación es costosa, y conlleva un consumo significativo de energía, materiales y mano de obra. La operación de diseño previo del producto también es costosa, pero su costo repercute sólo parcialmente en cada unidad de producto fabricado.

Por otra parte, la utilización de un elemento hardware implica un proceso de desgaste o envejecimiento que exige reparaciones ocasionales o periódicas para garantizar un servicio aceptable.

La ingeniería de software, sin embargo, presenta características especiales. El proceso de fabricación, consistente en obtener copias sucesivas del producto, es trivial y se puede realizar a un costo muy bajo. La labor importante es la del desarrollo inicial, de manera que el coso total de fabricar miles de unidades de un producto software es similar al de fabricar una sola.

Además, el software no se desgasta. Un programa funcionará al cabo de los años con la misma corrección con que lo hizo el primer día sin necesidad de modificación ninguna. Las tareas de mantenimiento de software son en realidad tareas adicionales de desarrollo, realizadas ocasionalmente durante la vida útil del producto con objeto de mejorarlo.

Concepto de Ingeniería del software

Con el término de Ingeniería del software se designa el empleo en el desarrollo de productos software de técnicas y procedimientos típicos de la ingeniería en general. La Ingeniería del software amplía la visión del desarrollo de software como una actividad esencialmente de programación, contemplando además otras actividades de análisis y diseño previos, y de integración y verificaciones posteriores. La distribución de todas estas actividades a lo largo del tiempo constituye lo que se ha dado en llamar ciclo de vida del desarrollo del software.

3. EL CICLO DE VIDA DEL DESARROLLO DEL SOFTWARE

La programación es un proceso complejo, que requiere de capacitación, planificación y del uso de algunas herramientas especializadas. También hay que actualizar continuamente los productos desarrollados de manera que satisfagan las necesidades cambiantes de las personas que los utilizan.

Los buenos programadores han de estar bien preparados e informados en dos áreas

importantes:

- **Las herramientas de programación**: el software y los lenguajes que se utilizan para desarrollar aplicaciones.
- **El proceso de programación**: los métodos y procedimientos que los programadores han de seguir para garantizar productos bien desarrollados.

La escritura de un programa puede ser sumamente difícil. El programador ha de seguir un plan para evitar programas defectuosos o incapaces de realizar una tarea. Cuando se sigue un plan, se tiene una idea de lo que ha de hacer y se sabe por donde empezar.

Los programas son los bloques de construcción de los sistemas de información. Cuando se crean productos software, los programadores siguen un proceso que se conoce como ciclo de vida del desarrollo del software.

El ciclo de vida del desarrollo del software. Modelo en cascada

Los modelos clásicos de ciclo de vida plantean el desarrollo y explotación de una aplicación software como una secuencia de actividades sucesivas y diferentes que se van realizando una tras otra. Uno de estos modelos es el modelo en cascada.

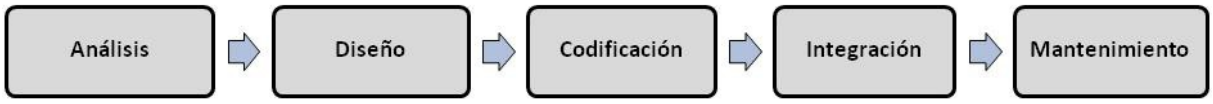
Modelo en cascada

En el modelo en cascada se identifican una serie de fases o actividades que han de realizarse siguiendo un orden determinado, de manera que el resultado de cada una de esas fases es el elemento de entrada para la fase siguiente. Las fases que componen el modelo en cascada son las siguientes:

- **Análisis de necesidades**: consiste en analizar las necesidades de los usuarios potenciales del software para determinar qué debe hacer el sistema a desarrollar, y de

acuerdo con ello, escribir una especificación precisa de dicho sistema.

- **Diseño**: consiste en descomponer y organizar el sistema en elementos componentes que puedan ser desarrollados por separado para aprovechar las ventajas de la división del trabajo y poder hacer el desarrollo en equipo. El resultado del diseño es la colección de especificaciones de cada elemento componente.
- **Desarrollo o codificación**: en esta fase se programa cada elemento componente por separado, es decir, se escribe el código fuente de cada uno. Normalmente se harán también algunas pruebas o ensayos para garantizar en lo posible que dicho código funciona correctamente.
- **Implementación o integración**: a continuación hay que ir combinando todos los elementos componentes del sistema, y probar el sistema completo. Habrá que hacer pruebas exhaustivas para garantizar el funcionamiento correcto del conjunto, antes de ser puesto en explotación.
- **Mantenimiento**: durante la explotación del sistema software es necesario realizar cambios ocasionales, bien para corregir errores no detectados con anterioridad, o bien para introducir mejoras. Para ello hay que rehacer parte de los trabajos anteriores.



El modelo de ciclo de vida en cascada trata de aislar cada fase de la siguiente, de tal manera que las fases sucesivas puedan ser realizadas por grupos de personas diferentes, facilitando la especialización.

El modelo en cascada hace énfasis también en la necesidad de terminar correctamente cada fase antes de comenzar la siguiente. Esto se debe a que los errores producidos en una fase son muy costosos de corregir si ya se ha realizado el trabajo de las fases siguientes.

Para detectar los errores lo antes posible, y evitar que se propaguen a las fases

siguientes, se establece un proceso de revisión al completar cada fase, antes de pasar a la siguiente.

El ciclo de vida del desarrollo del software. Modelo en cascada. Descripción de las fases

El ciclo de vida del desarrollo del software consta de cinco fases:

- Fase 1: Análisis de necesidades.
- Fase 2: Diseño.
- Fase 3: Desarrollo o codificación.
- Fase 4: Implementación o integración.
- Fase 5: Mantenimiento.

A continuación, se pasa a describir de manera detallada cada una ellas.

3.1. Fase 1: Análisis de necesidades.

La etapa de análisis de necesidades es una etapa de especificación de la aplicación. Estas especificaciones se obtienen normalmente a través de conversaciones entre los clientes y los diseñadores. Se trata de conseguir una idea completa de las necesidades que el cliente plantee. Es una etapa clave del desarrollo, ya que del buen entendimiento entre cliente y desarrollador depende el éxito o el fracaso de la aplicación.

La etapa de análisis es la etapa donde se identifica y entiende una necesidad o problema. El objetivo de la fase de consiste en analizar las necesidades de los usuarios potenciales y hacer un estudio lo más detallado posible del problema a resolver. Requiere un estudio exhaustivo del problema para determinar qué debe hacer exactamente el software o sistema a desarrollar y qué tipo de problema debe resolver; y de acuerdo con ello, escribir una especificación precisa de dicho sistema. En esta fase se puede elegir la plataforma y el lenguaje de programación a utilizar (esto también puede hacerse en la fase de codificación).

Cada fase lleva asociado un documento con todas las especificaciones necesarias.

3.2. Fase 2: Diseño.

En la etapa de diseño se busca la descomposición del problema en módulos y se definen los datos necesarios para cada módulo. El objetivo de esta etapa consiste en descomponer y organizar el sistema en elementos componentes o módulos que puedan ser desarrollados por separado para aprovechar las ventajas de la división del trabajo y poder hacer un desarrollo en equipo. El resultado del diseño es la colección de especificaciones de cada elemento componente.

Los objetivos del diseño son dos:

- **La descomposición modular del sistema**. Se trata de aplicar el concepto de modularidad y obtener una división del sistema en partes o módulos. En este sentido, el objetivo es: identificar los módulos componentes, describir cada módulo y describir las relaciones entre módulos.
- **La decisión sobre los aspectos de implementación o representación de datos que son importantes a nivel global**. Se trata de que el diseñador decida sobre los algoritmos y estructuras de datos fundamentales que se deben utilizar para la realización del sistema.

En la etapa de diseño se crean los algoritmos que permiten resolver el problema. En el caso de la programación estructurada, para facilitar el trabajo, se utilizan herramientas como los diagramas IPO, los diagramas de flujo y el pseudocódigo. En el caso de la programación orientada a objetos (POO) se utiliza el Lenguaje Unificado de Modelado (UML).

En la etapa de diseño, los programadores suelen utilizar diferentes herramientas:

- Los **diagramas de entrada-proceso-salida (IPO)**: El diagrama IPO ayuda al programador a determinar lo que es necesario para escribir el programa. Consiste en tres columnas. En la primera columna, el programador lista los datos que son necesarios para resolver la tarea. En la última columna, se listan los resultados deseados. La columna de en medio es la parte difícil. En ella, el programador lista los pasos necesarios para obtener el resultado deseado.
- Los **diagramas de flujo**: Un diagrama de flujo es una representación gráfica de un proceso. Cada paso del proceso es representado por un símbolo diferente que contiene una breve descripción de la etapa de proceso. Los símbolos gráficos del flujo del proceso están unidos entre sí con flechas que indican la dirección de flujo del proceso. El diagrama de flujo ofrece una descripción visual de las actividades implicadas en un proceso mostrando la relación secuencial ente ellas, facilitando la rápida comprensión de cada actividad y su relación con las demás, el flujo de la información y los materiales, las ramas en el proceso, la existencia de bucles repetitivos, el número de pasos del proceso, etc. Facilita también la selección de indicadores de proceso.
- El **pseudocódigo**: está formado por frases del lenguaje natural que tienen apariencia de código de programación. La idea es escribir en el idioma hablado lo que se necesita que ocurra en el código.

Pseudocódigo: Se entiende por pseudocódigo una notación basada en un lenguaje de programación estructurado (Pascal. Módula-2, Ada, etc.) del que se excluyen todos los aspectos de declaración de constantes, tipos, variables y subprogramas. El pseudocódigo puede incluir descripciones en lenguaje natural, siempre que se considere necesario, como parte del mismo.

Existen diferentes programas para facilitar la creación de diagramas de flujo, tanto para Linux como para Windows. Un ejemplo es **Dia**. (https://wiki.gnome.org/Apps/Dia). El programa Dia se puede descargar de http://dia-installer.de/index.html.es. También se pueden usar las herramientas de dibujo que vienen con algunos paquetes ofimáticos como LibreOffice y Apache OpenOffice.

El pseudocódigo es una alternativa escrita a los diagramas de flujo. Existen diferentes herramientas que permiten trabajar con pseudocódigo. Un ejemplo es **PSeInt** (http://pseint.sourceforge.net). PSeInt es una herramienta libre para aprender los fundamentos de programación empleando pseudocódigo y diagramas de flujo. Está disponible tanto para Linux como para Windows. El programa PSeInt se puede descargar de http://pseint.sourceforge.net.

3.3. Fase 3: Desarrollo o codificación

La fase de desarrollo o codificación se relaciona con la escritura y pruebas del código fuente. En esta fase se traducen los algoritmos obtenidos en la fase anterior a un lenguaje de programación. Se obtiene el código fuente que se traduce a programas ejecutables. Los equipos escriben las diferentes secuencias de código que componen cada uno de los procedimientos y los diferentes módulos diseñados en la etapa anterior se transforman en código ejecutable por los diferentes equipos de programadores.

Un elemento esencial dentro de la codificación es el lenguaje de programación. El lenguaje de programación a utilizar, si no ha sido especificado en los requisitos, se decide en esta etapa. Para facilitar la escritura del código fuente se utilizan los entornos de desarrollo integrados (IDE's).

En esta fase se programa cada elemento componente por separado; es decir, se escribe el código fuente de cada uno de los módulos. Normalmente, se harán también las pruebas o ensayos necesarios para garantizar en lo posible que dicho código funciona correctamente.

A la hora de escribir el código fuente de un programa, existen dos tipos principales de errores:

- Los **errores léxicos y sintácticos**: son los errores que violan las reglas del lenguaje de

programación.

- Los **errores lógicos**: son errores relacionados con el algoritmo, son más difíciles de encontrar y a veces tardan bastante tiempo en aparecer.

El proceso de identificar y eliminar estos errores se conoce depuración.

Nota: A la hora de programar, es muy importante añadir comentarios al código fuente para clarificar y explicar cada parte del programa. Hay que pensar en los desarrolladores que tengan que hacer futuras revisiones.

3.4. Fase 4: Implementación o integración

La implementación (integración) se relaciona con la instalación del software y con permitir que los usuarios lo prueben.

En la fase de integración hay que ir combinando todos los elementos componentes del sistema, y probar el sistema completo. Habrá que hacer pruebas exhaustivas para garantizar el funcionamiento correcto del conjunto, antes de ser puesto en explotación.

Una vez escrito el programa hay que someterlo a una serie de pruebas para detectar posibles errores de funcionamiento. La fase de pruebas debe ser lo más exhaustiva posible y se deben examinar todas las opciones.

3.5. Fase 5: Mantenimiento.

En esta fase se deberán corregir todos los errores no detectados en la fase de prueba.

Durante la explotación del sistema software es necesario realizar cambios ocasionales, bien para corregir errores no detectados con anterioridad, o bien para introducir mejoras. Para ello hay que rehacer parte de los trabajos anteriores. El mantenimiento comienza tan pronto

como el programa ha sido instalado.

El mantenimiento es necesario por varias razones:

- Corregir errores que no se han detectado en las fases anteriores.
- Añadir funciones nuevas en respuesta a las demandas del mercado o a nuevas necesidades de los usuarios.

El mantenimiento es la fase más larga y puede durar toda la vida del programa.

4. REVISIONES Y PRUEBAS

Revisiones

Una revisión consiste en inspeccionar el resultado de una actividad para determinar si es aceptable o, por el contrario, contiene defectos que han de ser subsanados. Las revisiones se aplican, fundamentalmente, a la documentación generada en la fase de desarrollo.

Pruebas

Las pruebas o ensayos consisten en hacer funcionar un producto software o una parte de él en condiciones determinadas, y comprobar si los resultados son los correctos. El objetivo de las pruebas es descubrir los errores que pueda contener el software ensayado.

Generalmente, el programador realiza las pruebas de funcionamiento de todos y cada uno de los bloques que constituyen la aplicación. El fundamento de la programación procedimental es aislar los módulos de manera que cada uno de ellos pueda ser probado de forma independiente. Posteriormente se van probando las agregaciones de módulos, para, por último realizar la prueba final donde se verificará si se adapta a los requisitos planteados por el cliente en la fase de análisis. Todos los módulos probados y compilados se ensamblarán

para formar un único código ejecutable que se entregará al usuario.

5. LA CALIDAD DEL SOFTWARE.

La calidad de un producto software, como cualquier otro producto de ingeniería, viene determinada fundamentalmente por el proceso seguido en su desarrollo. En el modelo del ciclo de vida encontramos actividades típicamente productivas, tales como las de análisis, diseño y codificación, y otras otras cuyo objetivo es controlar la calidad del producto, tales como las revisiones y las pruebas.

La calidad de un producto puede valorarse desde puntos de vista diversos.

Entre los factores que determinan la calidad del software se pueden citar: corrección, fiabilidad, eficiencia, seguridad, facilidad de uso, mantenibilidad, flexibilidad, facilidad de prueba, transportabilidad, reusabilidad, interoperatividad, etc.