1. Desarrollo de software

Autor	Ⓧ Xerach Casanova
Clase	Entornos de desarrollo
Fecha	@Mar 29, 2021 9:05 PM

1. Software y programa. Tipos de software

Los ordenadores se componen de dos partes bien diferenciadas: software y hardware. El software es el conjunto de programas informáticos que actúan sobre el hardware para ejecutar lo que el usuario desea. Conforme a esta definición, podemos dividirlo en tres:

Software de sistema

Es el software base que ha de estar instalado en el ordenador para que las aplicaciones puedan ejecutarse y funcionar: sistema operativo: Windows, Linux, MacOs, etc...

Software de programación

Conjunto de herramientas para desarrollar programas informáticos, tales como editores de código, IDE, Frameworks...

Aplicaciones informáticas

Conjunto de programas que tienen una finalidad más o menos concretas, tales como procesadores de texto, programas de hojas de cálculo, reproductores de música, etc...

Un programa es un conjunto de instrucciones en algún lenguaje de programación que le indican a la máquina que operaciones realizar sobre determinados datos.

Relación hardware - software

La relación entre el hardware y el software es inseparable. El software necesita del hardware para poder ejecutarse y el hardware necesita del software para poder funcionar.

Desde el punto de vista del S.O.

Es el encargado de coordinar de manera oculta para el usuario y las apps, el hardware durante el funcionamiento del ordenador, haciendo de intermediario entre este y las aplicaciones. Así, se encargará de gestionar los recursos de hardware durante su ejecución: memoria RAM, tiempo de CPU, interrupciones...

Desde el punto de vista de las aplicaciones.

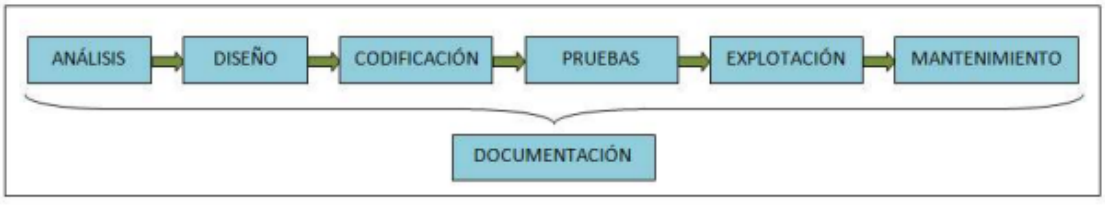
Una aplicación es un conjunto de programas escritos en algún lenguaje de programación que el hardware debe interpretar y ejecutar.

Todos los lenguajes de programación tienen en común que están compuestos de sentencias que el ser humano puede usar y entender fácilmente, mientras que el hardware funciona con código binario.

3. Fases a seguir en el desarrollo del software

Se entiende por desarrollo de software todo el proceso que ocurre desde que es concebida la idea hasta que un programa está implementado en el ordenador funcionando.

Es importante dedicar todos los recursos necesarios a las primeras etapas de su desarrollo para evitar errores que se propaguen durante toda la vida del proyecto.



3.1. Analisis

Es la primera fase y la de mayor importancia, ya que todo depende de lo bien que esta esté detallada.

En esta fase obtendremos dos salidas.

Documento especificación de requisitos de software

- **Requisitos funcionales.** Definen las funciones del software o sus componentes, pueden ser cálculos, detalles técnicos, manipulación de datos, etc...
- **Requisitos no funcionales.** Complementan a los requisitos funcionales y se refieren a todos los requisitos que no describen información a guardar, ni funciones a realizar, sino características de funcionamiento. Como por ejemplo el cumplimientos de normativas y estándares de calidad, tiempos de respuesta del programa, etc...

Documento de diseño de arquitectura

Contiene la descripción de la estructura relacional del sistema, la especificación de todas sus partes y la manera en que se combinan todas ellas. También se suele implementar en la primera fase de diseño.

El analista y el clicente deben asegurar buena comuniación entre ellos y...

- Planificar las reuniones
- Relación de objetivos del usuario cliente y del sistema
- Relación de objetivos prioritarios y temporización
- Mecanismos de actuación ante contingencias
- etc...

3.2. Diseño

Una vez terminado el análisis y se definen los requisitos se divide el sistema en partes y establecer las relaciones entre ellas, decidiendo que hará exactamente cada parte, generando un módulo funbcional - estructural de los requerimientos del sistema global.

En este punto también se deciden las entidades y relaciones de la BBDD, lenguaje de programación que se va a utilizar, el gestor de BBDD, etc...

Se obtendrán dos salidas:

- **Documento de diseño de software**
- **Plan de pruebas** a utilizar en la fase de pruebas pero sin entrar en detalles.

3.3. Codificación

Se elige un determinado lenguaje y se codifica todo lo expuesto en el análisis y diseño, obteniendo el código fuente.

Las características del código deben ser:

- **Modularidad** (dividido en trozos más pequeños
- **Corrección** (que haga lo que se pide realmente)
- **Fácil de leer** (para facilitar su desarrollo y mantenimiento)
- **Eficiencia** (que haga buen uso de los recursos)
- **Portabilidad** (que se pueda implantar en cualquier equipo.

3.4. Compilación

Una vez realizado el código fuente, este debe ser traducido a lenguaje máquina, para que las computadoras sean capaces de entenderlo y ejecutarlo. Este proceso se puede realizar de dos formas.

- **Compilación.** La traducción se realiza sobre todo el código fuente en un solo paso a través de un compilador.
- **Interpretación.** La traducción del código se realiza línea a línea y se ejecuta simultáneamente a través del software responsable llamado intérprete.

3.5. Pruebas

La realización de pruebas es imprescindible para asegurar la validación y verificación del software construido.

Pruebas unitarias

Se prueban todas las partes del software una a una de manera independiente y se genera un documento de procedimiento de pruebas que viene desde la fase de diseño. Los resultados se comparan con los esperados que se habrán determinado de antemano.

Pruebas de integración

Consiste en la puesta en común de todos los programas desarrollados una vez pasadas las pruebas unitarias. Se genera un documento de procedimiento de pruebas de integración.

3.6. Explotación

La explotación es la fase en la que los usuarios finalez conocen la aplicación y comienzan a utilizarla.

A menudo se consideran esta fase y la de mantenimiento como la misma fase.

Esta fase consiste en la instalación y configuración del software en el/los equipo/s del cliente, se asignan los parámetros de configuración y se prueba que la aplicación está operativa.

Puede ocurrir que la configuración la realicen los usuarios finales con la guía de instalación.

3.7. Mantenimiento

La etapa de mantenimiento es la más larga de todo el ciclo de vida del software.

Se define como el proceso de control, mejora y optimización del software.

Este debe actualizarse y evolucionar con el tiempo, adaptándose a las mejoras del hardware en el mercado y afrontar situaciones nuevas que no existían cuando se construyó.

Se pacta con el cliente un servicio de mantenimiento de la aplicación, que tendrá un coste temporal y económico.

Según el tipo se dividen en:

- **Perfectivos** (mejoran la funcionalidad del software).
- **Evolutivos** (el cliente propone mejoras para el producto e implica nuevos requisitos).
- **Adaptativos** (modificaciones y actualizaciones para adaptarse a las nuevas tendencias del mercado de hardware, condiciones específicas de organismos reguladores, etc.).
- **Correctivos** (resolver errores detectados.

3.8. Documentación.

Todas las estapas de desarrollo de software deben quedar perfectamente documentadas.

Debido a esto no es considerada como una etapa más del desarrollo del proyecto.

La documentación permite pasar de una etapa a otra de una manera clara y definida, además permite reutilización de los programas en otras aplicaciones si la codificación ha sido modular.

La documentación se divide en:

Guías técnicas

Dirigidos a analistas y programadores.

Quedan reflejados: El diseño de la aplicación, codificación de programas y pruebas realizadas.

Objetivo: Facilita un correcto desarrollo y realizar correcciones y permitir mantenimiento en el futuro.

Guías de uso

Dirigido a los usuarios finales o clientes.

Quedan reflejados: descripción de funcionalidad de la aplicación, forma de comenzar a ejecutarla, ejemplos de uso, requerimientos de software y solución a problemas.

Objetivo: Dar a los usuarios finales toda la información necesaria para utilizar la aplicación.

Guías de instalación

Dirigido al personal informático responsable de la instalación

Quedan reflejados: aspectos de puesta en marcha, explotación y seguridad del sistema.

Objetivo: dar toda la información para garantizar la implantación de la aplicación.

4. Ciclo de vida del software

Es el conjunto de fases o etapas, procesos y actividades requeridas para ofertar, desarrollar, probar, integrar, explotar y mantener un software.

El concepto de desarrollo del software aparece por la necesidad de los inconvenientes del proceso individualizado y carente de planificación del mismo.

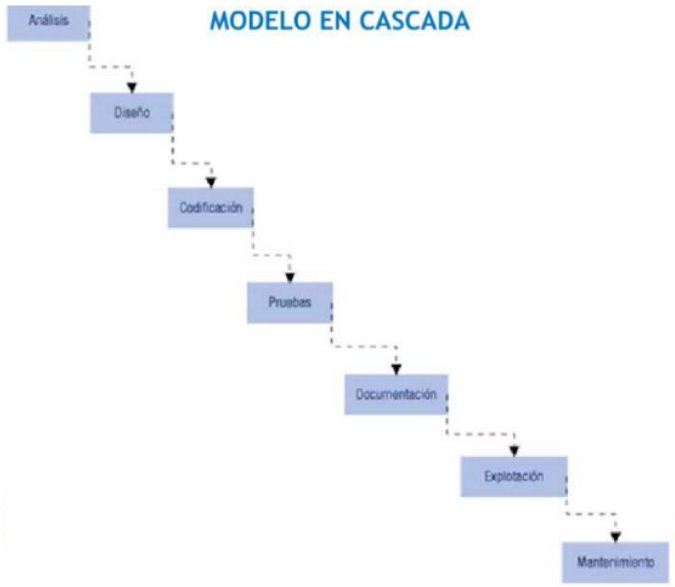
Ventajas:

- El análisis facilitará la codificación en un futuro a pesar de que en esté presente en esta fase.
- Asegura un desarrollo progresivo que permite detectar defectos con antelación.
- Control de plazos de entrega y costes excesivos.
- Aumenta la visibilidad y posibilidad de control para la gestión del proyeccto.
- Aporta una guía para el personal de desarrollo, marcando las tareas a realizar.
- Minimiza la necesidad de rehacer trabajo y problemas de puesta a punto.

Modelos de ciclos de vida

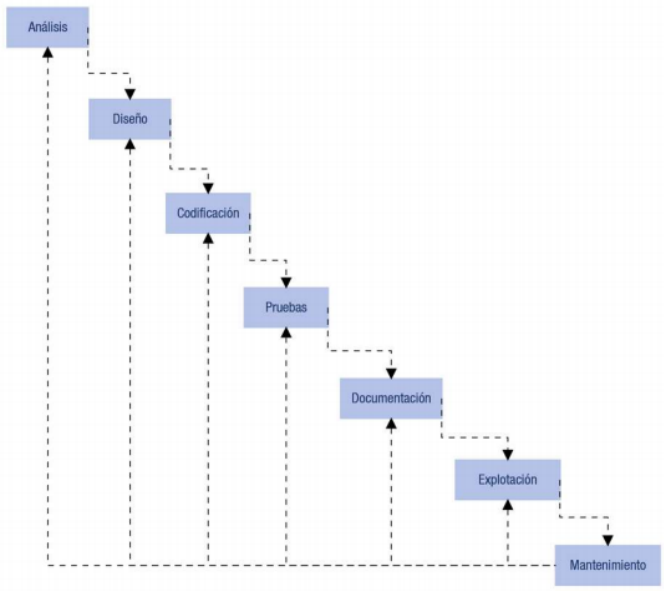
En cascada

Se pasa de una etapa a otra sin retorno, cualquier error en fases iniciales no es subsanable más adelante. Requiere conocimiento absoluto previo de los requerimientos del sistema y no permite modificaciones ni actualizaciones.



En cascada con realimentación

Uno de los más utilizado, igual que el modelo en cascada pero se introduce la realimentación entre etapas que sirven para introducir modificaciones o depurar errores. Ideal para proyectos rígidos con pocos cambios y poco evolutivos pero no para proyectos que prevén muchos cambios durante el desarrollo.

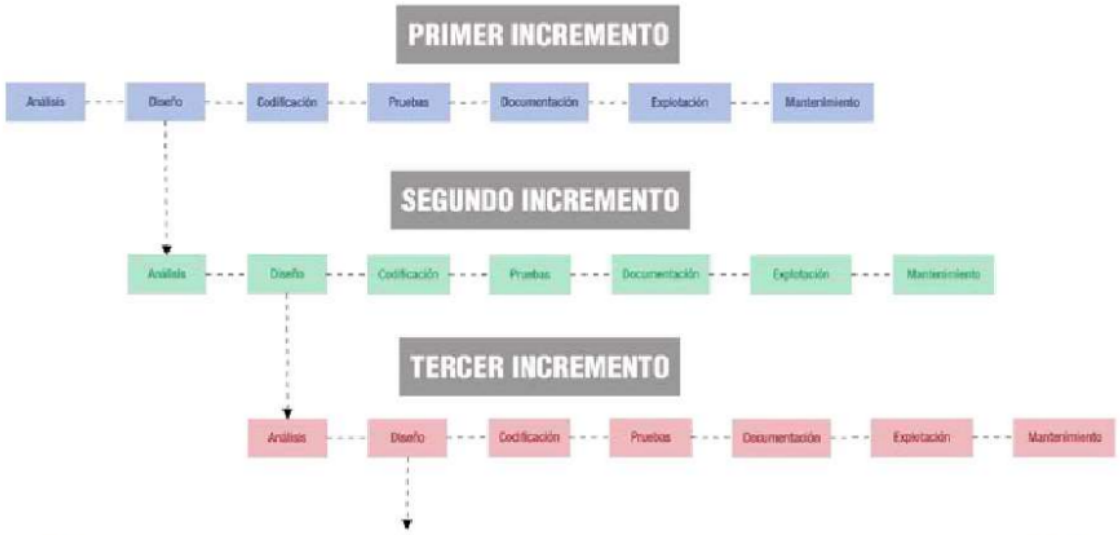


Evolutivos

Tienen en cuenta la naturaleza cambiante del software.

- **Modelo interactivo incremental.**

  Basado en el modelo en cascada con realimentación. Las fases de los ciclos se repiten en cascada y se refinan, de manera que se propagan sus mejoras a las fases siguientes. Estas generan versiones del producto cada vez más completas hasta llegar al producto final.



- **Modelo en espiral**

  Es una combinación del modelo anterior con el modelo en cascada. También se adapta a la evolución del software y reduce riesgos.

  El software se construye repetidamente en forma de versiones que son cada vez mejores. Se divide en 6 zonas llamadas regiones de tareas:

  - Comunicación con el cliente
  - Análisis de riesgos
  - Representación de la aplicación
  - Codificación y explotación
  - Evaluación del cliente.

Modelos Ágiles

Modelo que ha ganado presencia en el desarrollo de software, centrado en la satisfacción del cliente, gran flexibilidad a la aparición de nuevos requisitos incluso durante el desarrollo de la aplicación. Se trata de desarrollo incremental pero con incrementos cortos y abiertos a que las fases se solapen unas con otras. Un tipo de metodología ágil es Scrum.

5. Herramientas de apoyo al desarrollo del software.

Las herramientas CASE (Computer Aided Software Engineering) es el conjunto de aplicaciones que se utiliza en el desarrollo de software con el objetivo de automatizar sus distintas fases.

Estas herramientas permiten:

- Mejorar la planificación del proyecto.
- Darle agilidad al proceso.
- Reutilizar partes del software.
- Hacer aplicaciones que responda a estándares.
- Mejorar la tarea de mantenimiento.
- Visualizar las fases de forma gráfica.

Las herramientas CASE se clasifican en función del ciclo de vida en la que ofrecen ayuda.

- **U-CASE:** fase de planificación y análisis.
- **M-CASE:** fase de análisis y diseño.
- **L-CASE:** ayuda en la programación, detección de errores, depuración, pruebas y documentación.

6. Frameworks

Es una extructura de ayuda al programador a desarrollar proyectos sin partir desde cero.

Son plataformas software donde están difinidos programas, soporte, bibliotecas, lenguaje interpretado, etc. que ayudan a desarrollar los diferentes módulos o partes de un proyecto.

Ventajas de utilizar un framework:

- **Desarrollo rápido.**
- **Reutilización.**
- **Diseño uniforme.**
- **Portabilidad** (gracias a su ejecución sobre cualquier máquina virtual)

Por contra, uno de los inconvenientes es la dependencia del código al framework utilizado y los recursos del sistema que consume un framework en el equipo.

7. Lenguajes de programación

Un lenguaje de programación es aquel que nos permite comunicarnos con el hardware. Se trata de un conjunto de:

- **Alfabeto:** conjunto de símbolos permitidos.
- **Sintaxis:** normas de construcción permitidas.
- **Semántica:** significado de las construcciones para hacer acciones válidas.

7.1. Características de los lenguajes de programación.

Lenguaje máquina

- Código binario (unos y ceros).
- Es el único que entiende el hardware.
- Fue el primer lenguaje utilizado.
- Es único para cada procesador.
- Nadie programa en lenguaje máquina.

Lenguaje ensamblador

- Facilita la labor de programación en comparación al lenguaje máquina.

- Necesita traducción al lenguaje. máquina para ejecutrase.
    - Utiliza mnemotécnicos (instrucciones complejas que hacen referencia a la ubicación física de los archivos en el equipo).
- Difícil de utilizar.

Lenguajes de alto nivel basados en código

- Sustituyen al lenguaje ensamblador
- Utiliza sentencias y órdenes derivadas del inglés
- Necesita traducción al lenguaje máquina para ejecutarse
- Son utilizados hoy en día con tendencia a utilizarse cada vez menos.

Lenguajes visuales

- Sustituyen a los lenguajes de alto nivel.
- Se programa gráficamente utilizando el ratón.
- El código se genera automáticamente.
- Necesitan traducción al lenguaje máquina para ejecutarse.
- Son portables.

7.2. Clasificación de los lenguajes de programación.

La elección del lenguaje de programación para codificar un programa dependerá de las características del problema a resolver.

Según lo cerca que esté del lenguaje humano

- **Lenguajes de programación de alto nivel.** Tienen el inconveniente de no ser directamente ejecutados por el hardware así que debe traducirse.

Lenguajes de programación de bajo nivel

- **Lenguaje máquina.** registros con ceros y unos.
- **Lenguaje ensamblador.** Se trata de un primer nivel de abstracción con respecto al lenguaje máquina pero sigue siendo difícil de manejar y solo se puede utilizar en ordenadores de las mismas características.

Según su forma de ejecución

A la hora de traducir un programa en lenguaje máquina, esta se puede realizar de dos formas:

- **Lenguajes compilados.** Lenguajes que se traducen por el compilador. Se traduce todo el programa y se genera un fichero ejecutable con el código fuente escrito en lenguaje máquina. Para hacer cambios en el programa se debe volver a compilar. No se podrá ejecutar el programa hasta que la codificación no tenga ningún error.
- **Lenguajes interpretados.** Se traduce el programa cada vez que se ejecuta, los programas que realizan esta traducción se llaman intérpretes. Se puede ejecutar el programa aunque haya errores y se parará cuando llegue a ellos.
- **Lenguajes intermediarios.** Combinan los dos tipos de lenguaje, se hace una primera traducción a un lenguaje intermedio del cual se obtiene un fichero y en una segunda fase ese fichero se traduce en cada ejecución (interpretado).
    - **Código fuente.** Para obtener el código fuente de una aplicación informática se debe partir de las etapas de análisis y diseño, seguidamente se construye un algoritmo que simbolice los pasos a seguir y por último se elegirá el lenguaje de programación de alto nivel para condificar el algoritmo.

      Al terminar se obtendrá un documento con todos los módulos, funciones y bibliotecas necesarios.

      El código fuente puede ser abierto (disponible para que se pueda estudiar, modificar o reutilizar) o cerrado, el cual no se tiene permisos para editarlo.
    - **Código objeto.** Se trata del código intermedio, no es ejecutable por el hardware directamente, sino que necesita un intérprete. Este código se obtiene a través del compilador solo cuando el código fuente está libre de errores.
    - **Código ejecutable.** Es el código máquina resultante de enlazar el código objeto con ciertas rutinas y bibliotecas. El S.O. se encarga de cargarlo en memoria y proceder a su ejecución. El ejecutable es el resultado de todos los archivos de código objeto enlazados a través de un software llamado linker.

Lenguajes declarativos e imperativos

- En la **programación imperativa** se describen paso a paso el conjunto de instrucciones que se van a ejecutar (Basic, C, Fortran, Pacal, PHP, Java...).
- En la **programación declarativa** se utilizan sentencias que describen el problema que se quiere solucionar pero no las soluciones para hacerlo (SQL, prolog, Haskell...).

Según la técnica de programación utilizada

- **Lenguajes de programación estructurada.**

  Permiten el uso de tres tipos de sentencia o estructuras:

  > Sentencias secuenciales
  >
  > Sentencias selectivas (condicionales)
  >
  > Sentencias repetitivas (iteraciones o bucles),

  Su éxito reside en la sencillez a la hora de construir y leer programas, mantenimiento sencillo y estructura clara. Evolucionó hacia la programación modular, que divide el programa en trozos de código llamados módulos.

  Su desventaja es que el programa se concentra en un único bloque demasiado grande para manejarlo y la reutilización del código no es eficaz.

- **Lenguajes de programación orientada a objetos.** Se tratan a los programas no como un conjunto ordenado de instrucciones, sino como un conjunto de **objetos** que colaboran entre ellos para realizar acciones.

  Se definen **clases** como una colección de objetos con características similares, dichas clases tienen una serie de **atributos** que caracterizan a esos objetos y **métodos** que corresponden a las acciones que pueden realizar.

  Mediante llamada a los métodos los objetos se comunican unos con otros cambiando el estado de los mismos.

  código reutilizable.

  Errores más fáciles de localizar.

- **Lenguajes de programación visuales.** Basados en las técnicas anteriores pero que permiten programar gráficamente: .Net, C#...

8. Máquinas virtuales

Una máquina virtual es un tipo especial de software que separa el funcionamiento del ordenador de los componentes hardware instalados. Esta capa desempeña un papel importante en el funcionamiento de lenguajes compilados e interpretados.

Con ellos podemos desarrrollar y ejecutar una aplicación sobre cualquier equipo, independientemente de sus características concretas.

- Consiguen que las aplicaciones sean portables.
- Reservan memoria para los objetos y liberan memoria no utilizada.
- Se comunican con el siste ma donde se instala la aplicación y controla los dipositivos hardware implicados.
- Cumplimiento de las normas de seguridad de las aplicaciones.

Características de la máquina virtual

- Cuando se compila el código fuente se obtiene el código intermedio.
- Para ejecutar ese código se requiere tener independencia respecto al hardware.
- La máquina virtual aísla la aplicación de los detalles físicos del equlpo.
- Funciona como una capa de software bajo nivel y actúa como puente entre el bytecode (código intermedio) y los dispositivos físicos.
- Verifica el bytecode antes de ejeecutarlo.
- Protege direcciones de memoria.

8.1 Entornos de ejecución

Un entorno de ejecución es un servicio de la máquina virtual que sirve para ejecuta programas. Puede pertenecer al S.O. pero también se puede instalar como software independiente.

Se denomina runtime al tiempo que tarda en ejecutarse un programa en la computadora.

Funcionamiento del entorno de ejecución.

Está formado por la máquina virtual y las API o bibliotecas de clase estándar necesarias para que la aplicación se aejecutada.

El entorno funciona como intermediario entre el lenguaje y el S.O.

8.2 Java runtime environment

Se denomina JRE al conjunto de utilidades que permitirá la ejecución de programas java en cualquier plataforma.

Está formado por la máquina virtual JVM y las bibliotecas de clase estándar (API).