RESUMEN UNIDAD 01 ENTORNOS

¿Qué es el software?

Tipos de código

- Código fuente

  Compilación

- Código objeto

  Enlace (bibliotecas)

- Código ejecutable

  Ejecución

Código Fuente

Definición: Conjunto de sentencias entendibles por el programador que componen el programa o una parte.

Suele estar almacenado en fichero tipo texto. Estará escrito en un lenguaje de programación determinado.

- Es el código escrito por los programadores con algún editor de texto.
- Utiliza sentencias y órdenes derivadas del inglés.
- Más cercano al razonamiento humano.
- Lenguaje de programación de alto nivel y contiene el conjunto de instrucciones.
- Ejemplos: Java, C, C++, Python, PHP, etc.

Código objeto

Definición: Conjunto de instrucciones y datos escritos en un lenguaje que entiende el ordenador directamente. (binario o código máquina)

Provienen de la traducción de cierto código fuente, fragmento del programa final y específico de la plataforma de ejecución.

- Es el código binario resultado de compilar el fuente
- La compilación es la traducción de un sola vez del programa y realiza utilizando un compilador
- La interpretación es la traducción y ejecución simultánea del programa línea a línea
- No es directamente inteligible por el ser humano pero tampoco por la computadora
- Es un código intermedio entre el fuente y el ejecutable, solo existe si el programa se compila y se traduce y ejecuta en un sólo paso

Código ejecutable

Definición: Reúne diferentes códigos u objetos generados por los programadores junto con librerías componiendo el programa final.

Es el código que ejecutan los usuarios del sistemas y es específico de una plataforma concreta (Windows, linux, mac, etc.)

- Es el código binario resultante de nelazar los archivos de código objeto con ciertas rutinas y bibliotecas necesarias
- El SO será el encargado de cargar el código ejecutable en la RAM y proceder a ejecutarlo
- También es conocido como código máquina. Directamente inteligible por la computadora

Tipos de lenguaje

El ordenador sólo entiende un lenguaje conocido como código binario o código máquina, consistente en ceros y unos.

Los más próximos a la arquitectura hardware se denominan lenguajes de bajo nivel y los más cercanos a los programadores y usuarios se denominan lenguajes de alto nivel.

Lenguaje de bajo nivel

Son totalmente dependientes de la máquina, es decir, el programa resultante de este tipo de lenguajes no se pueden migrar o utilizar en otras máquinas.

Dentro de este grupo se encuentran:

Lenguaje máquina

Ordena a la máquina las operaciones fundamentales para su funcionamiento. Consiste en la combinación de 0 y 1, para formar las ordenes entendibles por el hardware. Mucho más rápido que los de alto nivel pero bastante más difíciles de manejar y usar. El código fuente es enorme y es casi imposible encontrar un fallo.

Lenguaje ensamblador

Derivado del lenguaje máquina y formado por abreviaturas de letras y números llamadas mnemotécnicos.

Con la aparición de este lenguaje se crearon los programas traductoress para pasar los programas escritos en ensamblador a máquina. Tiene casi las mismas desventajas que el de máquina añadiendo la dificultad de tener que aprender un nuevo lenguaje difícil de probar y mantener.

Lenguaje de alto nivel

Se trata de lenguajes independientes de la arquitectura del ordenador. En un principio se pueden migrar de una máquina a otra sin ningún tipo de problema.

Permiten al programador olvidarse por completo del funcionamiento interno de la máquina. Solo necesitan un traductor que entienda tanto el código fuente como las características de la máquina.

Etapas de la ingeniería del software

Análisis de requerimientos: Se extraen los requisitos del software, es crítica la habilidad y experiencia en ingeniería para reconocer requisitos incompletos. Usualmente el usuario tiene una visión incompleta de lo que necesita y es necesario ayudarle para obtener la visión completa. La comunicación es intensa e importante.

Especificación: La tarea de describir detalladamente el software a ser escrito. Se describe el comportamiento esperado y su interacción con el usuario.

Diseño y arquitectura: Determina cómo funcionará de forma general sin detalles, consiste en el diseño de los componentes del sistema que dan respuesta a las funcionalidades descritas en la anterior etapa. Se realiza en base a diagramas que permiten describir las interacciones entre entidades y su secuenciado.

Programación: Se traduce como el diseño a código. Es la parte más obvia del trabajo de ingeniería y la primera que obtiene resultados tangibles. No necesariamente la etapa más larga ni compleja aunque un diseño incompleto/ambiguo puede exigir que tareas propias de anteriores etapa se realicen en esta.

Prueba: Consiste en comprobar que el software responda correctamente a las tareas indicadas en la especificación. Se realizan pruebas a diferentes niveles.

Documentación: Realización del manual del usuario y manual técnico con propósito de mantenimiento futuro y ampliación. Se inician en la primera fase y se terminar una vez terminadas las pruebas.

Mantenimiento: Se realiza mantenimiento correctivo y evolutivo.

Ciclo de vida del Desarrollo de Software

SDLC en sus siglas inglesas, es una secuencia estructurada y bien definida de las etapas en Ingenería de software para desarrollar el producto deseado.

El **paradigma de desarrollo de software** ayuda al desarrollador a escoger una estrategia para desarrollar. Tiene su propio set de herramientas, métodos y procedimientos.

Algunos paradigmas son:

- **Modelo de cascada**
- **Modelo en V**
- **Modelo en Espiral**

Modelo de cascada

Es el paradigma más simple en desarrollo. Sigue un modelo en que las fases funcionarán una detrás de otra de forma lineal. Solo cuando la primera termine podrá empezar la segunda, y así progresivamente.

Asume que todo se lleva a cabo y que tiene lugar tal y como se había planeado la fase anterior. No funcionará correctamente si se dejan asuntos de lado en la fase previa. La naturaleza secuencial no permite volver atrás a corregir.

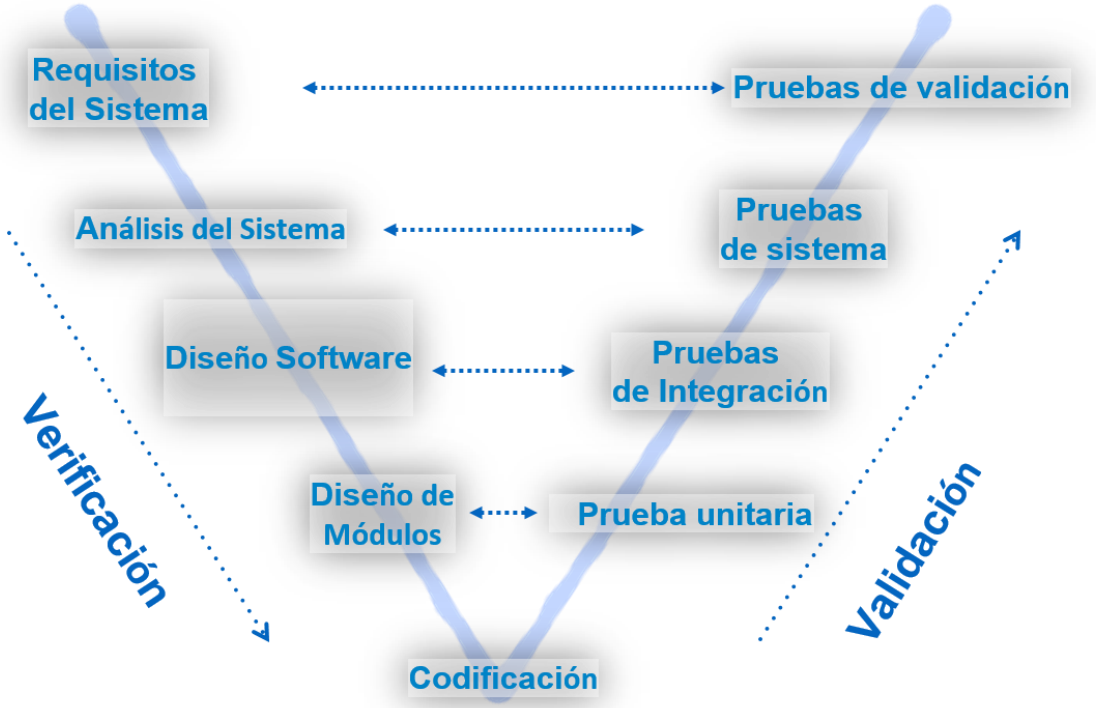
Es recomendable cuando el desarrollador ya ha diseñado y desarrollado softwares similares anteriormente.

Modelo en V

Su mayor inconveniente es que solo pasa a la siguiente fase cuando se completa la anterior por lo tanto no es posible avanzar si se encuentra un error en fases anteriores. Aporta opciones de evaluación en cada etapa de manera inversa.

En cada etapa se crea la planificación de las pruebas para verificar y validar el producto según requisitos de la etapa.

Esto hace que tanto la verificación como la validación vayan en paralelo, también se conoce como modelo de validación y verificación.

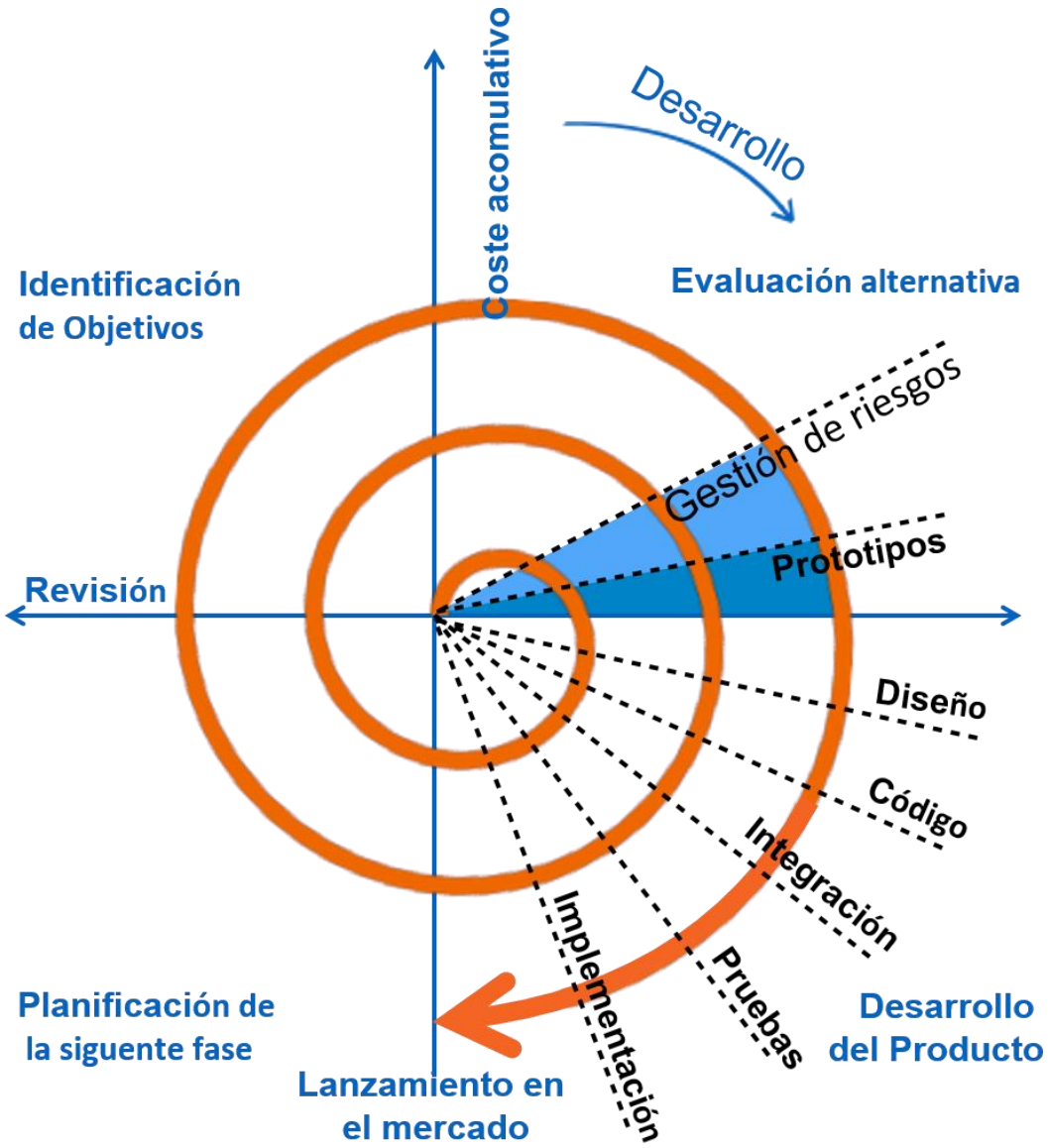


Modelo en espiral

Considera el riesgo, factor que otros modelos olvidan.

Empieza determinando los objetivos y limitaciones del software al inicio de cada repetición.

En la siguiente etapa se crean los modelos de prototipo del software. Esto incluye el análisis de riesgos. Un modelo estándar de SDLC se usa para construir el software y en la cuarta etapa es donde se prepara el plan de la siguiente repetición.

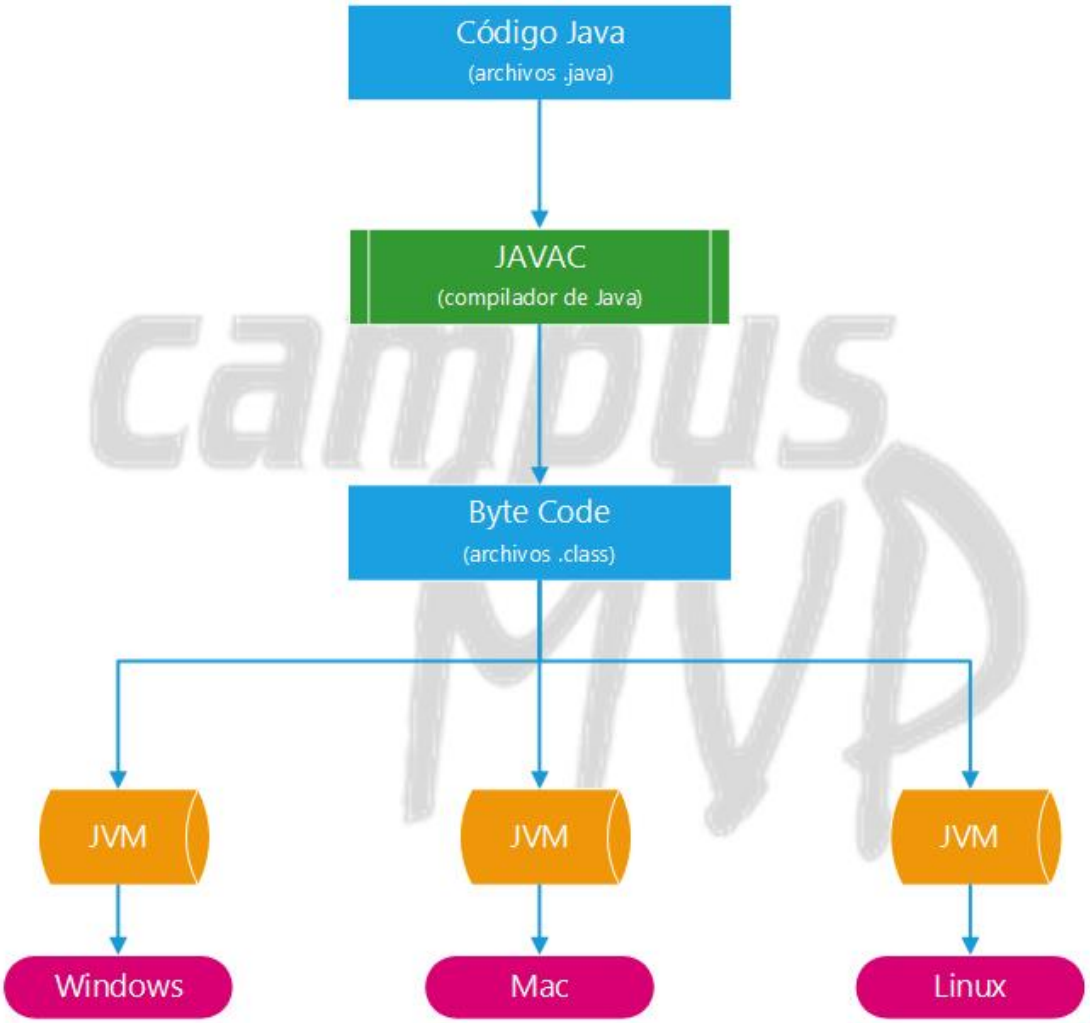


Máquina Virtual de Java (MVJ o JVM)

La máquina virtual de java o java virtual machine es un entorno de ejecución para aplicaciones Java cuya finalidad es adaptar los programas Java compilados a las características del SO donde se van a ejecutar.

Cuando compilas una aplicación escrita en Java, en realidad éste no se compila a lenguaje máquina sino a lenguaje intermedio denominado Byte Code.

Entre el Byte Code y el SO se coloca un componente especial llamadao Máquina virtual que es el que realmente ejecuta el código.


Código Java
(archivos .java)
JAVAC
(compilador de Java)
Byte Code
(archivos .class)
JVM
JVM
JVM
Windows
Mac
Linux