Tema 2: Conceptos del desarrollo del software

¿Qué aprenderás?

- Identificar los elementos que intervienen en el desarrollo de una aplicación.
- Localizar as diferentes etapas del software para pasar un código fuente a obtener un ejecutable.
- Diferenciar las fases del desarrollo de un software.
- Reconocer el personal implicado en el desarrollo de software.

¿Sabías que...?

- Desde que el programador codifica el programa hasta que se ejecuta en un hardware determinado, el desarrollo del programa ha de pasar por diferentes fases y herramientas para ser tratado.
- El personal implicado en el desarrollo de software en cada empresa puede tener un nombre distinto (analista programador, junior analyst, applications analyst…).

2.1. Elementos que intervienen en el desarrollo de aplicaciones

2.1.1. Los elementos más importantes

2.1.1.1. Código fuente

Son las instrucciones que codifican los programadores. El código fuente debe cumplir con las normas del lenguaje de programación en el que estemos codificando.

2.1.1.2. Código objeto

Es el **resultado de compilar el código fuente**. Es decir, al traducir el código fuente se obtiene este código objeto, eso si, previamente se tiene que validar a nivel sintáctico y semántico.

Es un código intermedio pensado para enlazar todas las librerías con tal que lo entienda la máquina en la que se va a ejecutar. Este código intermedio se puede transformar en código máquina si es un lenguaje de programación compilado o en bytecode si es un lenguaje virtual como el ejemplo de Java.

2.1.1.3. Código ejecutable

El que podemos ejecutar. El fichero EXE. Es el que resulta de después de enlazar el código objeto con las librerías.

2.1.2. Herramientas implicadas para la obtención de código ejecutable

2.1.2.1. Compilador

El compilador es un programa que traduce el código fuente escrito en un determinado lenguaje de programación de alto nivel a lenguaje máquina.

El mismo compilador es el que se encarga de detectar los posibles errores del código fuente. Su objetivo es conseguir el programa ejecutable depurado.

2.1.2.2. Máquina virtual

La máquina virtual tiene el objetivo al compilador para que genere código para diferentes procesadores.

Seguro que en vuestro ordenador alguna vez os habrá salido una notificación de una taza de café indicando que necesitáis actualizar vuestra versión de Java. Eso es la máquina virtual de Java. Esta máquina virtual es capaz de interpretar el **bytecode** y de adaptarlo al equipo dónde se está ejecutando.

El **bytecode** es en realidad un código máquina de bajo nivel. Sólo que este se puede ejecutar en la máquina virtual de Java, la máquina virtual de Java es un programa nativo que se ejecuta en una plataforma específica y qué es capaz de interpretar y ejecutar las instrucciones del bytecode. Se sitúa en un nivel Superior al hardware.

La gran ventaja de la máquina virtual de Java para el programador es que no se ha de preocupar por el hardware de la máquina física donde se va a ejecutar el programa.

2.1.3. Etapas del proceso de obtención de código ejecutable

Desde que el programador codifica el programa hasta que se ejecuta en un hardware determinado, el desarrollo del programa ha de pasar por diferentes fases y herramientas para ser tratado.

2.1.3.1. El proceso de compilación

La compilación es la traducción de un programa escrito (código fuente) en un determinado lenguaje de programación con la finalidad de que se pueda ejecutar.

Partimos siempre del **código fuente**, nosotros escribimos el código una vez escribimos el código y nos parece válido tenemos que **pasar un análisis sintáctico y lexicográfico**.

Un ejemplo de análisis sintáctico, supongamos que en Java queremos hacer un hola mundo con un sistem.out.println. y nos dejamos un punto o escribimos mal esta instrucción. El análisis lexicográfico devolverá un error y no podremos continuar hasta escribirlo todo bien y no se produzca ningún error.

El análisis sintáctico-semántico que viene a continuación es el que se encarga de verificar que el orden de las instrucciones es correcto y tiene sentido (según el lenguaje de programación).

Por ejemplo instrucción *if-then-else* espera una secuencia en concreto. Primero la palabra reservada IF seguido de una condición, a continuación una instrucción para cuando se cumple la condición y otra instrucción opcional por si no se cumple la condición. Si colocamos la condición antes de *if* o en otro lugar que no le corresponde, a pesar de no tener ningún error lexicográfico, no pasaremos el análisis sintáctico-semántico ya que el compilador será incapaz de interpretar instrucción.

Una vez validados léxico y sintaxis **se pasa a generar el código intermedio**, a posteriori **se optimiza y se obtiene un código optimizado** y **con el generador de código finalmente se obtendrá el código optimizado.**

Una vez tenemos el código objeto entra al juego en el enlazador. Por ejemplo cuando codificáis en Java y queréis leer un dato por teclado necesitáis hacer servir la función *scan*. Esta función la tendréis que importar la de una librería, estas funciones ya están escritas y no es necesario que lo hagamos nosotros. **El enlazador se encargará de vincular estas librerías** que hace servir nuestro código fuente para que nuestro ejecutable pueda hacerlas servir. A partir de enlazar todas estas librerías con nuestro código objeto podremos generar el ejecutable.

A continuación dejamos las definiciones de las herramientas o elementos que intervienen en el proceso de compilación:

- **Código fuente**: líneas de texto con los pasos que se deben seguir para ejecutar el programa
- **Análisis lexicográfico**: a partir del código fuente genera una salida compuesta de tokens qué son los componentes léxicos
- **Análisis sintáctico-semántico:** se comprueba el texto de entrada en base a una gramática dada, la del lenguaje de programación
- **Generador de código intermedio**: es el que transforma el código lenguaje más próximo a la plataforma de ejecución.
- **Optimizador de código**: realiza una serie de transformaciones de mejora del código, se obtiene un código optimizado.
- **Generador de código**: el compilador convierte el programa sintácticamente correcto en una serie de instrucciones que deben ser interpretadas por una máquina. A partir del generador de código se obtiene el código objeto.
- **Enlazador**: programa que a partir del código generado y los recursos necesarios (bibliotecas) quita los recursos que no necesita y enlaza los que necesite al código objeto finalmente genera un código ejecutable.

2.1.4. Fases de desarrollo de una aplicación

Cuando desarrollamos un programa no nos ponemos directamente a codificar sin más, es necesario que nuestro proyecto de software siga una serie de fases de desarrollo con tal de tener éxito y evitar perder el tiempo en tareas que no son productivas.

A pesar de que existen muchas formas de abordar los proyectos de desarrollo de software. Las fases más comunes en el desarrollo del software son:

2.1.4.1. **Análisis**

En esta fase nos hemos de contestar a esta pregunta: **¿Qué necesita más hacer nuestro software?**

Para esto, hablamos con el cliente para que nos diga los requisitos que necesita su programa. Todo esto es lenguaje natural.

2.1.4.2. **Diseño**

Una vez tenemos los requisitos en lenguaje natural, lo siguiente es determinar **cómo funcionará el software de forma global** sin entrar en detalles. En esta fase se realizarán los diagramas de clases o de comportamiento.

2.1.4.3. **Codificación**

Es el momento de programar. Se pasa del diseño al código.

2.1.4.4. **Pruebas**

Las pruebas siempre se deben realizar durante y después la codificación. De hecho es una fase muy importante.

Las pruebas nos confirmarán que la codificación ha sido exitosa, nos permitirá verificar que el software no contiene errores y qué hace lo qué debe hacer.

2.1.4.5. **Documentación**

Como creadores de software hemos de asegurarnos nuestro código está lo mejor documentado posible. Pensad que en un momento dado podemos cambiar de proyecto, al llegar estaremos un tiempo tratando de entender el código. Para eso necesitaremos una buena documentación.

Normalmente la documentación es de carácter técnica y está pensada para que sea leída por otros desarrolladores por los clientes usuarios.

2.1.4.6. **Mantenimiento**

En esta fase se corrigen errores que se detectan cuando el programa ya está implementado y en explotación.
Las ampliaciones del software o modificaciones que se tengan que hacer tambien las consideraremos mantenimiento.

2.1.4.7. **Explotación**

Consiste en generar el código y ponerlo en marcha. Poner nuestro software en producción es un proceso sumamente delicado. Se puede dar el caso que pongamos en producción nuestro software por primera vez o qué hacemos con mejoras o parches que arreglan problemas detectados.

Se debe preparar el software para su distribución y tener en cuenta las posibles afectaciones que puedan surgir.

2.1.5. Personal que interviene en el desarrollo de software

En cada empresa o proyecto **es posible que estos perfiles tengan diferentes nombres**, pero normalmente los perfiles que podemos encontrar proyecto de desarrollo de software son los siguientes:

2.1.5.1. Analista de sistemas

Dirige proyectos y garantiza que se cumplen los requisitos del cliente. Es la persona que va a estar con el cliente. Se encarga de los requerimientos y de la parte más externa para garantizar las expectativas del cliente.

2.1.5.2. Diseñador de software

A partir del análisis es el que diseña la solución que hay que desarrollar. Normalmente es un experto qué es capaz de diseñar la solución sabiendo qué tecnologías debe usar cómo trabajo y dividirlo por fases y funcionalidades.

2.1.5.3. Analista programador

Tiene una visión general de todo el proyecto. Sabe lo que hay que diseñar pero además aterrizan los conceptos para que los desarrollen los programadores. En otras palabras, se encarga de traducir el diseño en funciones específicas que desarrollarán los programadores junto a los mismos analistas programadores.

En esta fase se especifican aspectos más técnicos como la interacción entre módulos y funciones valores de entrada, salida, resultados esperados...

2.1.5.4. Programador

A partir de las instrucciones del analista programador y de los documentos técnicos que se le proporcionan, se dedica a codificar y aportar soluciones.

Normalmente trabaja con el analista programador.

2.1.5.5. **Arquitecto de software**

Normalmente son expertos con muchas horas dedicadas a una tecnología y que han pasado por varios proyectos, en cada empresa o cada proyecto hay una persona que domina una tecnología y propone las mejores soluciones para cada caso. Siempre investigan sobre nuevas tecnologías y frameworks, y revisa que los proyectos se lleven a cabo de una forma óptima y aprovechando todos los recursos.

Recursos y enlaces

- Onlinegdb.com - Simulador de compilador online

- Video Youtube - Qué es y cómo funciona un compilador

Conceptos clave

- **Las fases del desarrollo:** Análisis, diseño, codificación, pruebas, documentación, mantenimiento y explotación.
- **Personal que interviene en el desarrollo de software:** Analista de sistemas, programador, diseñador de software, analista programador y arquitecto de software**.**
- **El proceso de compilación:** Código fuente, análisis lexicográfico, análisis sintáctico-semántico, generador de código intermedio, optimizador de código, generador de código, enlazador.

Test de autoevaluación

1. En las fases de desarrollo de una aplicación, ¿qué fase es la que consiste en generar el código y ponerlo en marcha?
   a) Explotación
   b) Mantenimiento
   c) Documentación
   d) Codificación

2. En el proceso de compilaciónm ¿qué herramienta es la que se encarga de transformar el código lenguaje más próximo a la plataforma de ejecución?
   a) Generador de código intermedio
   b) Generador de código
   c) Optimizador de código
   d) Enlazador

3. El bytecode es en realidad un código máquina de bajo nivel. Sólo que este se puede ejecutar en la máquina virtual de Java.
   a) Cierto
   b) Falso

Ponlo en práctica

Actividad 1

Completa el cuadro relativo al proceso de compilación

a. Se leen de manera secuencial todos los caracteres de nuestro código fuente, buscando palabras reservadas, operaciones, caracteres de puntuación y los agrupa en cadenas de caracteres que se llaman lexemas.

b. Una vez finalizado el análisis, se genera una representación intermedia a modo de pseudoensamblador con el objetivo de facilitar la tarea de traducir al código objeto.

c. Revisa el código pseudoensamblador generado en el paso anterior optimizándolo para que el código resultante sea más fácil y rápido de interpretar por la máquina.

d. Como se ha comentado anteriormente, se enlaza el código objeto con las librerías necesarias, produciendo en último término el código final ejecutable.

e. Genera el código objeto de nuestro programa en un código de lenguaje máquina relocalizable, con diversas posiciones de memoria sin establecer, ya que no sabemos en qué parte de la memoria volátil se va a ejecutar nuestro programa.

f. Agrupa los componentes léxicos estudiados en el análisis anterior en forma de frases gramaticales. Con el resultado del proceso del análisis sintáctico, se revisa la coherencia de las frases gramaticales.

Fase	**Descripción (Pon la letra de la definición)**
Análisis Lexicográfico	
Análisis Sintáctico-Semántico	
Generación de código intermedio	
Optimización de código	
Generación de código	
Enlazador de librerías	

SOLUCIONARIOS

Test de autoevaluación

1. En las fases de desarrollo de una aplicación, ¿qué fase es la que consiste en generar el código y ponerlo en marcha?

   a) **Explotación**
   b) Mantenimiento
   c) Documentación
   d) Codificación

2. En el proceso de compilaciónm ¿qué herramienta es la que se encarga de transformar el código lenguaje más próximo a la plataforma de ejecución?

   a) **Generador de código intermedio**
   b) Generador de código
   c) Optimizador de código
   d) Enlazador

3. El bytecode es en realidad un código máquina de bajo nivel. Sólo que este se puede ejecutar en la máquina virtual de Java.

   a) **Cierto**
   b) Falso

Ponlo en práctica

Actividad 1

Completa el cuadro relativo al proceso de compilación.

g. Se leen de manera secuencial todos los caracteres de nuestro código fuente, buscando palabras reservadas, operaciones, caracteres de puntuación y los agrupa en cadenas de caracteres que se llaman lexemas.

h. Una vez finalizado el análisis, se genera una representación intermedia a modo de pseudoensamblador con el objetivo de facilitar la tarea de traducir al código objeto.

i. Revisa el código pseudoensamblador generado en el paso anterior optimizándolo para que el código resultante sea más fácil y rápido de interpretar por la máquina.

j. Como se ha comentado anteriormente, se enlaza el código objeto con las librerías necesarias, produciendo en último término el código final ejecutable.

k. Genera el código objeto de nuestro programa en un código de lenguaje máquina relocalizable, con diversas posiciones de memoria sin establecer, ya que no sabemos en qué parte de la memoria volátil se va a ejecutar nuestro programa.

l. Agrupa los componentes léxicos estudiados en el análisis anterior en forma de frases gramaticales. Con el resultado del proceso del análisis sintáctico, se revisa la coherencia de las frases gramaticales.

Fase	Descripción (Pon la letra de la definición)
Análisis Lexicográfico	f
Análisis Sintáctico-Semántico	a
Generación de código intermedio	b
Optimización de código	c
Generación de código	e
Enlazador de librerías	d