UD1: Desarrollo del Software

Entornos de Desarrollo

Objetivos

- Reconocer la relación de los programas con los componentes del sistema informático
- Identificar las fases de desarrollo de una aplicación informática.
- Diferenciar los conceptos de código fuente, código objeto y código ejecutable.
- Clasificar los lenguajes de programación

Contenidos

- El software del ordenador
- Ciclo de vida del software
- Fases del desarrollo de una aplicación
- Concepto de programa y aplicación informática
- Lenguajes de programación. Tipos de lenguajes.
- El proceso de traducción/compilación
- Roles que forman parte del proceso de desarrollo del SW.
- Pseudocódigo y Diagramas de Flujo (ordinogramas)

Objetivo general de la unidad

En esta unidad aprenderemos a reconocer los elementos y herramientas que intervienen en el desarrollo de un programa informático, analizando sus características y las fases en las que actúan hasta llegar a su puesta en funcionamiento.

Índice

1. Introducción

El ordenador se compone de dos partes: el hardware y el software. El hardware lo forman los componentes físicos que se pueden ver y tocar: el monitor, el teclado, el ratón, la placa base, la memoria RAM, el microprocesador, el disco duro, etc.

En cambio, el software forma la parte lógica del ordenador que no se puede tocar. Comprende el conjunto de programas y aplicaciones que actúan sobre el hardware del ordenador y facilitan al usuario la realización de diferentes tareas.

2. El software del ordenador

Podemos dar varias definiciones sobre la palabra software, por ejemplo:

- El diccionario de la Real Academia Española (RAE) define software como <<*el conjunto de programas, instrucciones y reglas informáticas para ejecutar ciertas tareas en una computadora*>>.
- El estándar 729 del IEEE (*Institute of Electrical and Electronics Engineers*), define software como <<*el conjunto de los programas de cómputo, procedimientos, reglas, documentación y datos asociados que forman parte de las operaciones de un sistema de computación*>>.

2. El software del ordenador

2. El software del ordenador

Como todo el mundo sabe, el SW tiene una serie de características muy particulares que lo definen. Estas características son las siguientes:

- El software es lógico, no físico. Es intangible.
- El software se desarrolla, no se fabrica.
- El software no se estropea ni se rompe.
- En ocasiones puede desarrollarse a medida. Existe software a medida y software enlatado.

Actividad propuesta: Línea del tiempo sobre la evolución del software.

2. El software del ordenador

Investiga

Busca información sobre:

a) La vida y obra de Alan Turing, puesto que a la vez que interesante, forma parte del origen del software moderno.

b) El *firmware*

2. El software del ordenador

Actividades propuestas

Indica si las siguientes afirmaciones son verdaderas o falsas y razona tus respuestas:

1. El firmware puede considerarse un software. ____
2. La criptografía se ha utilizado desde la Segunda Guerra Mundial ____

2.1 El SW a medida y el SW estándar

El software a medida es una o varias aplicaciones realizadas según los requerimientos e instrucciones de una empresa u organismo. Dichos programas de amoldan o adecuan a la actividad desarrollada y son diseñados a la medida del organismo, su forma de trabajar y sus necesidades.

El software estándar o enlatado es un software genérico (válido para cualquier cliente potencial), que resuelve múltiples necesidades. Normalmente, para hacerlo más adaptable, dicho software tiene herramientas de configuración que lo parametrizan y lo adaptan a las necesidades del cliente.

2.1 El SW a medida y el SW estándar

Algunas características del SW a medida son las siguientes:

1. Como todo software, necesita tiempo de desarrollo.
2. Se adapta a las necesidades específicas de la empresa. Eso implica que, en ocasiones, ese software no sea trasladable a otras empresas diferentes (incluso a otras empresas del mismo sector).
3. Generalmente, suele contener errores y se necesita una etapa de mantenimiento en la que se subsana y se mejora dicho software.
4. En general es más costoso que el software estándar debido a que el precio lo soporta un único cliente.

2.1 El SW a medida y el SW estándar

Algunas características del SW estándar son las siguientes:

1. Se compra ya hecho. El software ya fue diseñado en su momento y lo único que podría hacerse es adaptarlo a las necesidades de la empresa.
2. Suele tener muchos menos errores que el software a medida, puesto que fue probado por múltiples empresas.
3. Suele ser más barato que el software a medida
4. Generalmente tiene funciones que la empresa no usará y también carecerá de otras opciones. Por regla general, no se adapta completamente a las necesidades de una empresa.

2.2 El SW libre y el SW propietario

El software libre es aquél en el cual el autor cede una serie de libertades básicas al usuario, en el marco de una licencia, que establece las siguientes libertades:

1. Libertad de utilizar el programa con cualquier fin en cuantos ordenadores desee.
2. Libertad de estudiar cómo funciona el programa y de adaptar su código a necesidades específicas; para ello, como condición previa, es necesario poder acceder al código fuente.
3. Libertad de distribuir copias a otros usuarios (con o sin modificaciones).
4. Libertad de mejorar el programa (ampliarlo, añadir funciones) y de hacer públicas y distribuir al público las modificaciones; para ello, como condición previa, es necesario poder acceder al código fuente.

2.2 El SW libre y el SW propietario

A diferencia del software libre y de fuentes abiertas, el software propietario es aquél que, habitualmente, se distribuye en formato binario, sin posibilidad de acceso al código fuente según una licencia en la cual el propietario, por regla general, prohíbe alguna o todas las siguientes posibilidades: la redistribución, modificación, copia, uso en varias máquinas simultáneamente, transferencia de titularidad, difusión de fallos y errores que se pudiesen descubrir en el programa, entre otras.

3. Ciclo de vida del software

El estándar ISO/IEC 12207-1 define ciclo de vida del software como: *Un marco de referencia que contiene los procesos, las actividades y las tareas involucradas en el desarrollo, la explotación y el mantenimiento de un producto de software, abarcando la vida del sistema desde la definición de los requisitos hasta la finalización de su uso.*

El ciclo de vida de un producto software comprende el período que transcurre desde que el producto es concebido hasta que deja de estar disponible o es retirado. Normalmente se divide en etapas y en cada etapa se realizarán una serie de tareas. Usualmente se consideran las siguientes etapas: especificación y análisis de requisitos, diseño del sistema, implementación del software, aplicación y prueba, entrega y mantenimiento.

3.1 Fases del ciclo de vida

3.1 Fases del ciclo de vida

- **Fase inicial:** en esta fase se planifica el proyecto, se hacen estimaciones, se decide si el proyecto es rentable o no, etc. Es decir, se establecen las bases de cómo va a desarrollarse el resto de fases del proyecto. En un símil con la construcción de un edificio, consistiría en ver si se dispone de licencia de construcción de un edificio, consistiría en ver si se dispone de licencia de construcción, cuánto va a costar el edificio, cuántos trabajadores van a necesitarse para construirlo, etc.

- **Análisis:** construye un modelo de los requisitos. En esta etapa se debe entender y comprender de forma detallada el problema que se va a resolver. Es muy importante producir en esta etapa una documentación entendible, completa y fácil de verificar y modificar.

3.1 Fases del ciclo de vida

- **Diseño:** en esta etapa ya sabemos lo que hay que hacer, ahora hay que definir cómo se va a resolver el problema. Se deducen las estructuras de datos, la arquitectura de SW, la interfaz de usuario y los procedimientos. Por ejemplo, en esta etapa hay que seleccionar el lenguaje de programación, el SGBD, etc.

- **Codificación:** en esta etapa se traduce lo escrito en el diseño a una forma legible por la máquina. La salida de esta fase es código ejecutable.

- **Pruebas:** se realizarán pruebas para garantizar que la aplicación se ha programado de acuerdo con las especificaciones originales y los distintos programas de los que consta la aplicación están perfectamente integrados y preparados para la explotación.

- **Mantenimiento:** esta fase tiene lugar después de la entrega del SW al cliente. En ella hay que asegurar que el sistema pueda adaptarse a los cambios.

3.2 Modelos de ciclos de vida

Existen varios modelos de ciclo de vida, es importante tener en cuenta las características del proyecto software para elegir un modelo u otro. Los modelos más importantes son:

- En cascada
- Iterativo incremental
- En espiral
- Basado en prototipos

3.2 Modelos de ciclos de vida

ACTIVIDAD

Elige un modelo y elabora una presentación en la que indiques en qué consiste el modelo, es decir cómo funciona (sus características), ventajas e inconvenientes, y cuándo se recomienda.

- En cascada
- Iterativo incremental
- En espiral
- Basado en prototipos

4. Documentación

En cada una de las fases anteriores, se generan uno o más documentos. En ningún proyecto es viable comenzar la codificación sin haber realizado las fases anteriores porque eso equivaldría a un desastre absoluto. Además, la documentación debe ser útil y estar adaptada a los potenciales usuarios de dicha documentación (cuando se crea un coche, existen los manuales de usuario y los manuales técnicos para los mecánicos. "Para qué quiero yo saber dónde están situados los inyectores o las bujías si nunca la voy a cambiar. A mi lo que me interesa es saber es cómo se regula el volante, cómo funciona la radio, etc").

En cualquier aplicación, como mínimo, deberán generarse los siguientes documentos:

a) Manual de usuario: es, como ya se ha comentado anteriormente, el manual que utilizará el usuario para desenvolverse con el programa. Deberá ser autoexplicativo y de ayuda para el usuario. Este manual debe servirle para aprender cómo se maneja la aplicación y qué es lo que hay que hacer y lo que no.

4. Documentación

b) Manual técnico: es el manual dirigido a los técnicos (el manual para los mecánicos citado anteriormente). Con esta documentación, cualquier técnico que conozca el lenguaje con el que la aplicación ha sido creada debería poder conocerla tan bien como el personal que la creó.

c) Manual de instalación: en este manual, se explica paso a paso los requisitos y cómo se instala y pone en funcionamiento la aplicación.

Desde la experiencia, es preciso recalcar una vez más la importancia de la documentación, puesto que, sin documentación, una aplicación o programa es como un coche sin piezas de repuesto, cuando tenga un problema o haya que repararlo, no podrá hacerse nada.

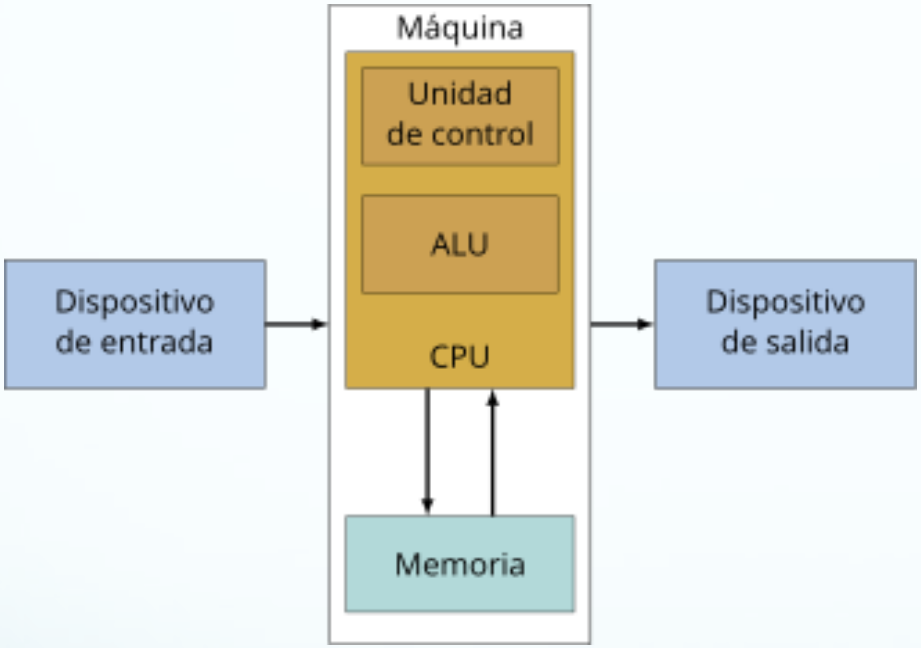
5. Programa informático y aplicación informática

Un programa informático es un conjunto de instrucciones escritas en un lenguaje de programación que aplicadas sobre un conjunto de datos resuelven un problema o parte del mismo. Para que el programa pueda ser ejecutado es necesario traducirlo a un lenguaje entendible por el ordenador, el lenguaje máquina; esta tarea es llevada a cabo por otro programa llamado compilador. Una vez tenemos el programa en código entendible por la máquina hay que cargarlo en memoria principal para que el procesador ejecute una a una todas las instrucciones.

Véase el tipico programa Hola mundo desarrollado en lenguaje C:

```
/*Codigo: Programa Holamundo*/

/* Programa holamundo.c*/

#include <stdio.h>  // Esta línea es necesaria para sacar información por pantalla

main( ) {

      printf("Hola mundo"); // Esta línea muestra las palabras Hola mundo por pantalla

}
```

5.1 Componentes de un sistema informático



Para ejecutar un programa se necesitan los recursos HW del ordenador: el **procesador** (también conocido como UCP *Unidad Central de Proceso* o CPU), la **memoria RAM** (o memoria principal), los **dispositivos de E/S**, la **UC** que interpreta y ejecuta las instrucciones máquina almacenadas en memoria principal, y la **ALU** (Unidad Aritmético-Lógica) recibe los datos sobre los que efectúa operaciones de cálculo y comparaciones, y toma decisiones lógicas (determina si una afirmación es cierta o falsa mediante las reglas del álgebra de Boole), y devuelve el resultado, todo ello bajo la supervisión de la UC.

5.2 Concepto de aplicación informática

Existen en el mercado muchas aplicaciones informáticas y cada una tiene una utilidad y finalidad concreta. Generalmente, las aplicaciones suelen estar formadas por varios programas con sus librerías correspondientes, aunque podrían constar solamente de un programa. Cuando son varios programas que pueden ejecutarse independientemente uno de otro, suele denominarse *suite* o *paquete integrado* como, por ejemplo, la *suite* ofimática de OpenOffice.

Una aplicación informática está en contacto con el usuario y no con el HW. Será el S.O el que haga de nexo de unión entre ambos (aplicación informática y HW). ¿Recuerdas la imagen de la transparencia 8?

5.2 Concepto de aplicación informática

Existe multitud de aplicaciones informáticas que automatizan o ayudan a la realización de ciertas tareas como, por ejemplo:

- Programas de contabilidad
- Bases de datos
- Programas de diseño gráfico
- Procesadores de texto
- Programa de facturación
- Aplicaciones multimedia
- Hojas de cálculo
- Presentaciones

6. Lenguajes de programación

Todos los programas se desarrollan en algún lenguaje de programación. Nadie programa directamente en con instrucciones máquina, debido a que son ininteligibles para el ser humano y realizar un programa de este tipo provocaría numerosos errores.

Los lenguajes de programación son, por lo tanto, lenguajes artificiales creados para que, al traducirse a código máquina, cada una de las instrucciones de dicho lenguaje dé lugar a una o varias instrucciones máquina-

Habitualmente, dichos lenguajes tienen una sintaxis y un conjunto de normas y palabras reservadas, de tal manera que la traducción sea lo más efectiva posible.

6. Lenguajes de programación

En la actualidad hay multitud de lenguajes de programación. Uno de los más conocidos es Java, pero no es el único.

6. Lenguajes de programación

Investiga

1. ¿Cuál es el Top ten de los lenguajes de programación en 2022?

2. ¿Qué es ReactJS y AngularJS y para qué sirven?

Toma nota: un ejemplo de framework es Boostrap, utilizado en Twitter y liberado para que cualquiera pueda realizar páginas y aplicaciones web con sus componentes.
Si quieres ver más ejemplos de sitios web diseñados con Bootstrap, puedes acceder a su zona de exposición.

6.1 Tipos de lenguajes de programación

a) Según su nivel de abstracción. Se clasifican en:

- **Lenguajes de bajo nivel:** se acercan al funcionamiento de un ordenador. El lenguaje de más bajo nivel por excelencia es el lenguaje máquina que es entendible directamente por la máquina. Las instrucciones están formadas por cadenas de ceros y unos.

A éste le sigue el lenguaje ensamblador. Este lenguaje es difícil de aprender y es específico para cada procesador. Un programa escrito en este lenguaje necesita ser traducido a lenguaje máquina para poder ejecutarse.

6.1 Tipos de lenguajes de programación

- **Lenguajes de nivel medio:** tienen ciertas características que los acercan a los lenguajes de bajo nivel, pero a la vez también tienen características de los lenguajes de alto nivel. Un lenguaje de programación de este tipo es el lenguaje C. Se suelen para aplicaciones como la creación de sistemas operativos.

- **Lenguajes de alto nivel:** son más fáciles de aprender porque están formados por palabras del lenguaje natural. Para poder ejecutarlos en el ordenador se necesita un programa intérprete o compilador que traduzca las instrucciones escritas en este lenguaje en instrucciones en lenguaje máquina que el ordenador pueda entender. Los lenguajes de programación de alto nivel son independientes de la máquina, es decir, no dependen del hardware del ordenador y no requieren ningún conocimiento de código máquina por parte del usuario que lo utiliza. Ejemplos: Java, C++, PHP etc.

7. Obtención de código ejecutable

El código de un programa pasa por diferentes estados desde que se escribe hasta que se ejecuta en el ordenador:

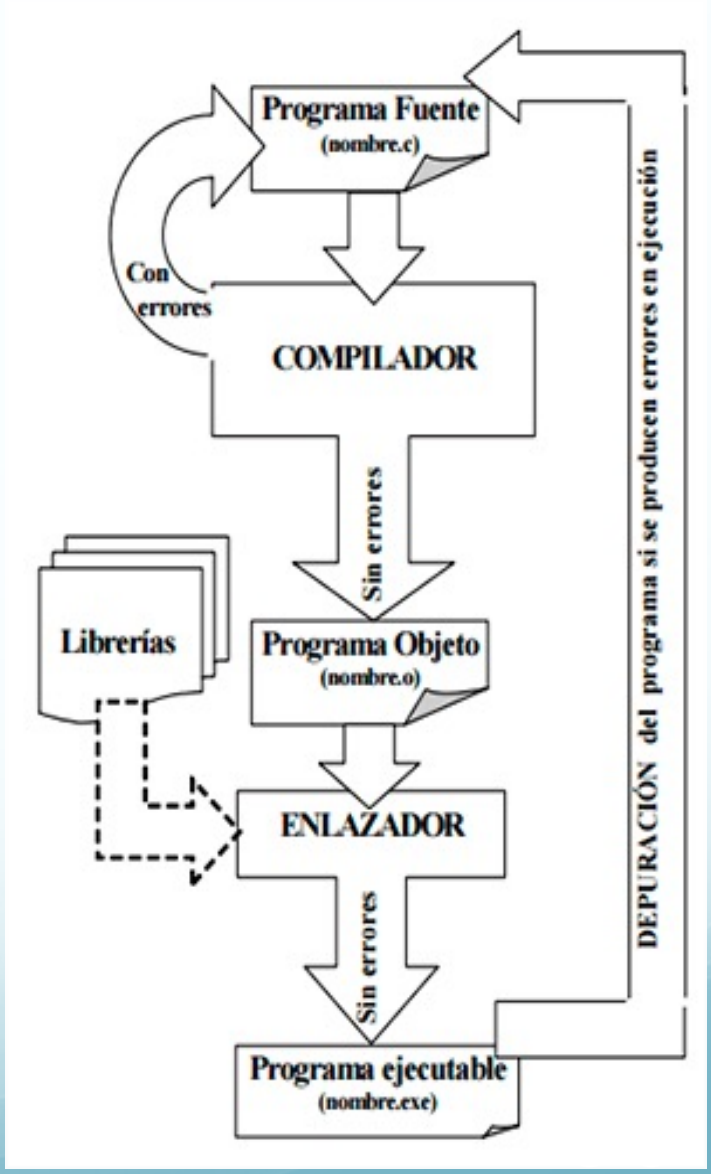
- **Código fuente:** es el código escrito por los programadores utilizando algún editor de texto o algún lenguaje de programación. Este código no es directamente ejecutable por el ordenador.
- **Código objeto:** es el código resultante de compilar el código fuente. No es directamente ejecutable por el ordenador ni entendido por el ser humano. Es un código de representación intermedia de bajo nivel.
- **Código ejecutable:** es el resultado de enlazar el código objeto con una serie de rutinas y librerías, obteniendo el código que es directamente ejecutable por la máquina.

7. Obtención de código ejecutable

COMPILACIÓN

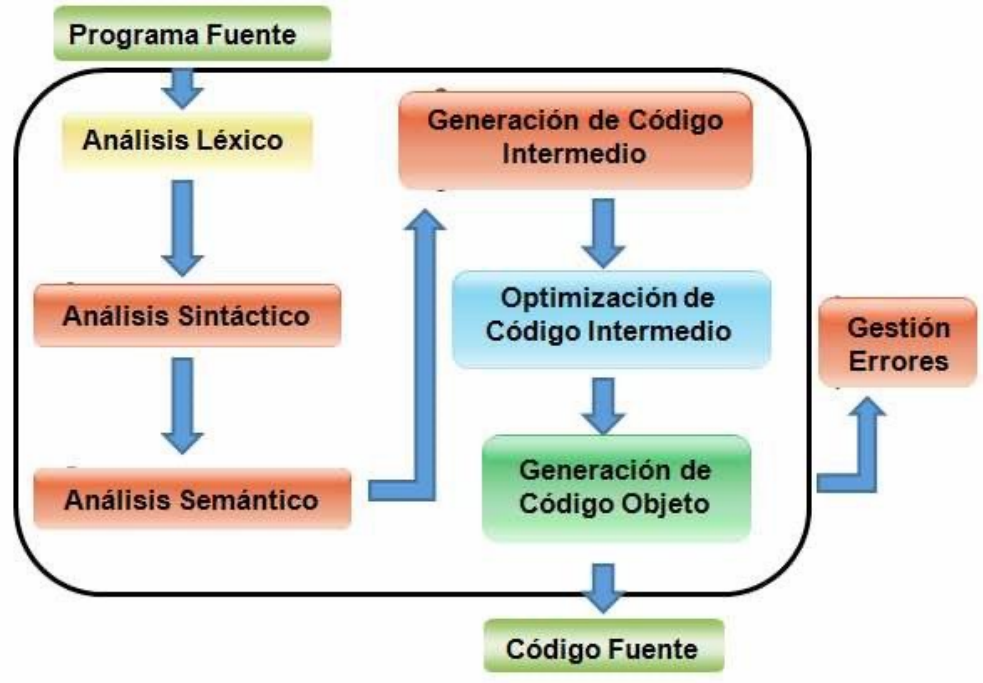
El proceso de compilación de un programa se lleva a cabo mediante dos programas, el compilador y el enlazador.

Si el compilador en el proceso de traducción devuelve algún error (falta algún signo de puntuación, sentencias mal escritas, tipos de datos no compatibles, variables no definidas...) no se generará programa objeto y será necesario modificar el programa fuente y pasarlo de nuevo por el compilador.



7. Obtención de código ejecutable

El compilador se compone internamente de varias etapas o fases que realizan distintas operaciones:



El programa enlazador inserta en el código objeto las funciones de librería necesarias para producir el programa ejecutable. Por ejemplo, en un programa escrito en C si el fichero fuente hace referencia a funciones de una biblioteca o a funciones que están definidas en otros ficheros fuente, entonces el enlazador combina estas funciones con el programa principal para crear un fichero ejecutable.

8. Roles o figuras que forman parte del proceso de desarrollo de SW

El equipo de desarrollo de SW se compone de una serie de personas, o más bien roles, los cuales tienen unas atribuciones y responsabilidades diferentes. A continuación, se describirán los principales roles en un proyecto de desarrollo de SW:

- **Arquitecto de software:** es la persona encargada de decidir cómo va a realizarse el proyecto y cómo va a cohesionarse. Tiene un conocimiento profundo de las tecnologías, los *frameworks*, las librerías, etc. Decide la forma y los recursos con los que va a llevarse a cabo un proyecto.

- **Jefe de proyecto:** dirige el proyecto. Muchas veces, un jefe de proyecto puede ser un analista con experiencia o una persona dedicada solamente a ese puesto. Tiene que saber gestionar un equipo y los tiempos, tener una relación fluida con el cliente, etc.

- **Analista de sistemas:** es un rol tradicional en el desarrollo de sw. Es una persona con experiencia que realiza un estudio exhaustivo del problema que ha de analizarse y ejecuta tanto el análisis como el diseño de todo el sistema.

8. Roles o figuras que forman parte del proceso de desarrollo de SW

- **Analista programador:** está a caballo entre el analista y el programador. Es un programador sénior, por así decirlo. Realiza funciones de análisis porque sus conocimientos lo permiten y también codifica. En proyectos pequeños, puede realizar ambas funciones (analista y programador)

- **Programador:** su función es conocer en profundidad el lenguaje de programación y codificar las tareas que le han sido encomendadas por el analista o analista-programador.

8. Roles o figuras que forman parte del proceso de desarrollo de SW

Investiga

Indica si las siguientes afirmaciones son verdaderas o falsas:

1. Los programadores intervienen en las fases iniciales de un proyecto.
2. Los analistas intervienen en las fases iniciales de un proyecto.

9. El paradigma de la POO

Accede a la plataforma y descarga “Para saber más. El paradigama de la POO”.

Después de leerlo realiza la siguiente actividad de investigación:

¿Qué es la programación orientada a componentes, o mejor dicho, los componentes en la POO?

10. Bibliografía

Ramos Martín, A; Ramos Martín, M. *Entornos de Desarrollo*. Garceta.

Moreno Pérez, JC. *Entornos de desarrollo*. Síntesis.