MEDAC
Instituto Oficial de Formación Profesional

ENTORNOS DE DESARROLLO

Ingeniería del software

2

ÍNDICE

OBJETIVOS

Conocer los diferentes roles del desarrollo software.

Aprender las diferentes fases del desarrollo software.

Diferenciar las diferentes metodologías de desarrollo.

Conocer el concepto de ingeniería del software.

/ 1. Introducción y contextualización práctica

El desarrollo software ha sido tratado normalmente como únicamente la escritura de código. Por esta razón, muchos de los proyectos software no llegan a ver la luz, pues fracasan durante el desarrollo debido a la falta de un modelo que guíe todo el **ciclo de vida.**

El ciclo de vida de cualquier software comienza desde que se habla con el cliente hasta que el software deja de estar operativo. Por ello, es necesario establecer un proceso que ayude a realizar cada una de las etapas que requiere la implementación de cualquier aplicativo.

Desde el punto de vista arquitectónico, el software se diferencia de la arquitectura tradicional en el coste que tiene cambiar una línea de código, pues no sería lo mismo cambiar una línea, que una viga de carga de un edificio.

Con esto en mente, se tiende a menospreciar la arquitectura de software, sin embargo, en la actualidad es un campo que tiene suma importancia. La principal diferencia con la arquitectura tradicional es el coste que tiene la búsqueda de un error frente a su solución, siendo la búsqueda, normalmente más costosa.

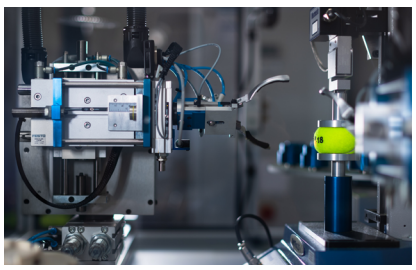

Fig. 1. Proceso de ingeniería del software

Por ejemplo, ¿qué haríamos para encontrar un error en 10000 líneas de código?

Escucha el siguiente audio en el que planteamos el caso práctico que iremos resolviendo a lo largo de esta unidad.

Audio intro. "Ingeniería del software"
https://bit.ly/2Cfohzc

/ 2. Fases del desarrollo software: Requisitos

El desarrollo de software requiere de la aplicación de técnicas de ingeniería conocidas como **ingeniería del software.** Es decir, el desarrollo del software debe seguir un proceso ingenieril, sin embargo, en la práctica, muy pocas veces se trata de esta forma.

Como todo proceso de ingeniería, la ingeniería del software requiere de unas fases por las que hay que pasar para poder tener un producto final acorde al proceso de ingeniería que estamos siguiendo. Tradicionalmente, se tiende a pensar que el software es únicamente código, sin embargo, en la práctica se demuestra que siguiendo un proceso de ingeniería el software es más que código.

Un ejemplo de ello, es qué realizar cuando ocurre un error en producción tras 10 años sin tocar el código. Si no seguimos un proceso de ingeniería sería muy complicado abordar y solucionar dicho error. Además del proceso de ingeniería, el software también sigue un proceso arquitectónico en el que podemos diferenciar no sólo el diseño del software, sino también la arquitectura en la que se va a ejecutar.

Por ejemplo, no es lo mismo crear un software que se va a ejecutar en un servidor y atenderá peticiones de los diferentes clientes que se conecten, que crear un software que se va a ejecutar en un dispositivo móvil.

El desarrollo software se divide en diferentes fases que las trataremos a continuación:

a. **Requisitos**

En esta fase se definen todas las funcionalidades que hay que desarrollar en la aplicación.

Normalmente, esta fase comienza con una entrevista con el cliente que es el que debe de indicar las funcionalidades que hay que implementar en la aplicación.

Es recomendable que tengamos más de una reunión con el cliente, pues normalmente puede ir cambiando de opinión sobre lo que él mismo quiere que implementemos.

Aunque es la primera fase, suele ser una de las más tediosas, ya que los clientes normalmente cambian algún requisito. Una vez que tengamos los requisitos, podremos generar un diagrama de casos de uso.

Video 1. "Diagrama de casos de uso "
https:/bit.ly/2zwvNAU

/ 3. Fases del desarrollo software: Del análisis al mantenimiento

a. **Análisis:** En esta fase se crea un diagrama sencillo en el que especificamos cada uno de los requisitos que nos ha indicado el cliente pudiendo también tener alguna relación, por ejemplo, a base de datos.

b. **Diseño:** En esta fase es recomendable definir todas las funcionalidades de manera general que va a tener la aplicación. El objetivo es poder identificar todos aquellos recursos del sistema, físicos, lógicos, etc. que serán necesarios para poder desarrollar la aplicación.

c. **Implementación:** En este punto nos centraremos en codificar todo lo diseñado previamente para tener una a primera aproximación al software.

d. **Pruebas:** Las pruebas de nuestra aplicación deben permitir identificar posibles errores que hayamos cometido a la hora de desarrollar.

e. **Documentación:** En la fase de documentación es necesario dejar por escrito todas las decisiones tomadas durante el desarrollo para facilitar el posterior mantenimiento de la aplicación.

f. **Explotación:** En esta parte del proceso, es necesario dejar preparado todo el software para poder ser lanzado a un entorno real para trabajar con usuarios reales y de esta forma entregarlo al cliente.

g. **Mantenimiento:** Una vez que el software se encuentra en ejecución en un entorno real es necesario mantenerlo, pues normalmente puede haber errores tanto lógicos como externos.

En este punto, ya estamos en condiciones de hablar del lenguaje UML (Lenguaje de Modelado Unificado), que permite crear una representación gráfica de los diferentes componentes de cada una de las fases por las que pasa el desarrollo software. En UML existen diferentes tipos de diagramas como son los diagramas de casos de uso, de interacción de secuencia, de clases etc.

En el siguiente vídeo podrás ver un ejemplo de diagrama de casos de uso, utilizando una herramienta CASE (Ingeniería de Software Asistida por Computadora, en inglés, Computer Aided Software Engineering).

Audio 1. "Fases del desarrollo software"

https://bit.ly/30LFHNp

/ 4. Caso práctico 1: "Obviar fases del desarrollo"

Planteamiento. A la hora de crear un nuevo producto software, estimamos que tendrá una complejidad alta. En este contexto, nuestro responsable nos pide que nos pongamos cuanto antes con el diseño del nuevo producto, ya que el tiempo disponible para el proyecto es limitado.

Nudo. ¿Crees que se trata de una buena práctica?

Desenlace. Según hemos visto en el apartado de ingeniería del software, es necesario que antes de realizar ningún diseño, en primer lugar, realicemos una reunión con el cliente para poder realizar una toma de requisitos.

Después de esta reunión, es recomendable que realicemos un documento que enviemos al cliente con los requisitos que hemos entendido y programemos otra reunión para discutirlos. Esto es así, dado que la toma de requisitos es tan importante que, un fallo en ellos puede suponer una gran pérdida económica y de recursos para una empresa.

Una vez que los requisitos sean aprobados por nuestro cliente, deberemos pasar a una etapa de análisis en la que podemos estudiar las dependencias y mejor forma de realizar el proyecto. En este momento dispondremos de más información para poder abordar la etapa de diseño de una manera más sencilla y segura, de forma que podremos realizar un diseño que tenga altas posibilidades de ser aprobado para ser implementado.

Según se nos plantea en el enunciado, pasar directamente al diseño no es una buena opción, pues corremos el riesgo de tener malos requisitos. Además, en ningún caso se nos ha dado algo que modelar...

Fig. 2. En el desarrollo software hay que lograr que todas las piezas encajen

/ 5. Metodologías de desarrollo software

En el desarrollo software existen diferentes enfoques a la hora de enfrentarnos al problema que hay que solucionar, de ahí que podamos diferenciar diferentes metodologías de desarrollo.

Nótese que, una mala elección en la metodología de desarrollo puede provocar grandes retrasos a la hora de publicar nuestro proyecto. Además, es importante tener en cuenta las necesidades de los clientes en cada punto del ciclo en el que nos encontremos.

a. **Metodología de desarrollo en cascada**

Este modelo de desarrollo en cascada se plantea un desarrollo lineal que es fácil de seguir por todos los programadores.

La característica principal de este modelo es que no se pasa a la siguiente fase hasta que se completa la fase anterior. El principal problema que tiene este enfoque es el tiempo que se requiere para finalizar todo el proyecto. Esto es debido al propio funcionamiento del modelo, pues requiere que todas las fases previas estén finalizadas para pasar a la siguiente.

Existe una modificación de este modelo llamado modelo Iterativo en el que se va a iterar sucesivamente por cada una de las fases incluso al llegar al final.

b. **Metodología de desarrollo en espiral**

Este modelo de desarrollo software se centra en poder crear una pequeña funcionalidad que continuamente se va a ir agrandando a medida que vamos desarrollando el proyecto, terminando por tanto siguiendo un ciclo en una espiral.

Esta forma de desarrollo es más ágil que el modelo iterativo o cascada pues permite tener un producto funcional en menor tiempo.

c. **Metodologías ágiles**

Actualmente las metodologías ágiles se convierten en un referente que llevan al extremo el modelo en espiral.

Alguna de las metodologías más utilizadas son Extreme Programming y Scrum.

Video 2. "Modelos de ciclos de vida"
https://bit.ly/2HvMbWw

/ 6. Roles en el desarrollo del software

Durante el ciclo de vida del software es normal que existan diferentes individuos involucrados en el proyecto, que pueden ir saliendo y entrando acorde con las necesidades de éste. Por esta razón, es importante poder definir roles que van a intervenir en cada una de las diferentes fases que conforman el desarrollo de un proyecto software.

Algunos de los roles más comunes que nos encontramos en los proyectos son:

- -**Analista de sistema.** Es el encargado de realizar el estudio del sistema para llevar a cabo la resolución del problema y garantizar que se cumplen las expectativas del cliente. Interviene principalmente en la fase de análisis.

- **Diseñador de software.** Es el encargado de realizar el diseño del sistema que se va a implementar trabajando principalmente en la fase de diseño.

- **Analista programador.** Es el individuo que posee mayor dominio de la programación centrándose más en los detalles del proyecto. Trabaja principalmente en las fases de diseño, implementación, pruebas y mantenimiento.

- **Programador.** Es el encargado de codificar el estudio realizado por los analistas y diseñadores.

- **Arquitecto de software.** Se trata de un perfil híbrido que permite unir el proceso de desarrollo, pues es el encargado de conocer e investigar los entornos de desarrollo, frameworks y tecnologías diferentes.

Fig. 3. Grupo de trabajo del proyecto de desarrollo

/ 7. Caso práctico 2: "Metodologías ágiles"

Planteamiento. Nuestro mayor competidor ha empezado un proyecto que tiene grandes posibilidades de ganar más clientes que todos los productos que tenemos en el mercado. Dado los rumores que nos llegan, sabemos que están utilizando una metodología de desarrollo en cascada.

Nudo. ¿Cómo podríamos sacar un producto mejor en menor tiempo?

Desenlace. Según hemos visto en el tema, existen diferentes modelos de desarrollo software, siendo el modelo en cascada realmente lento y poco adecuado en la actualidad.

Una posibilidad para sacar un producto en menor tiempo (no necesariamente mejor) sería utilizando metodologías ágiles como Scrum.

Otra alternativa, podría ser utilizar modelos en espiral en el que nos permiten ir añadiendo pequeñas funcionalidades en cada ciclo de la espiral. Una desventaja de este modelo es que los ciclos de la espiral pueden tener una temporalización diferente y larga.

Sin embargo, al utilizar Scrum podemos plantear ciclos, denominados Sprints de unos 15 días. De esta forma, podemos sacar diferentes versiones del producto en menor tiempo.

Según Scrum, en cada sprint deberíamos ser capaces de tener una nueva funcionalidad en el producto. De esta forma, comenzaríamos con una funcionalidad muy básica que se iría ampliando en cada sprint.

Aplicando este enfoque, llegaríamos al mercado antes que nuestro competidor, lo que suele implicar quedarnos con el nicho del negocio.

Fig. 4. La búsqueda de soluciones requiere de un enfoque global

/ 8. Resumen y resolución del caso práctico de la unidad

A lo largo de este tema hemos presentado las diferentes estrategias para el desarrollo del software, considerando diferentes **metodologías de desarrollo.**

Además, hemos estudiado los **diferentes roles** que intervienen en el desarrollo de cualquier aplicación.

Para poder comprender las diferentes metodologías de desarrollo, hemos presentado las diferentes etapas por las que debe pasar el software durante su "construcción". De esta forma, se ha realizado una unión entre las diferentes **fases de desarrollo** y los diferentes roles que participan en el desarrollo.

En paralelo, utilizando las fases de desarrollo, hemos podido conocer diferentes **alternativas metodológicas** para el trabajo.

También hemos expuesto el concepto de **ingeniería de software,** siendo este un concepto fundamental para poder atacar el desarrollo de cualquier aplicación como un proceso ingenieril.

Resolución del caso práctico de la unidad

Como hemos visto en el tema existen diferentes roles a la hora de desarrollar un programa informático, que está directamente relacionado con el proceso ingenieril que hemos visto a largo de la unidad.

Cuando se intenta ahorrar costes reduciendo el número de roles que intervienen en el diseño, análisis e implementación del sistema, corremos el riesgo de gastar realmente más dinero a posteriori.

Si nada más recibir las especificaciones, que en este caso son muy escasas, pasamos a realizar algo que el cliente realmente no es lo que busca, porque hayamos entendido mal un requisito o simplemente éste no existe, tenemos todas las papeletas para no hacerlo bien.

Por ejemplo, ¿qué sucede con la consola si el usuario (padres) no toman una decisión determinada cuando reciben la notificación al móvil? Como se puede ver, el cliente no ha especificado nada sobe esto...

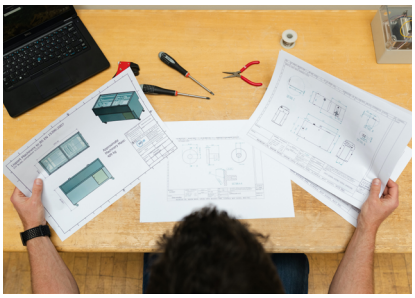

Fig. 5. La planificación en el desarrollo software es muy importante

/ 9. Bibliografía

Palomo, S. & Gil, E. (2020). Aproximación a la ingeniería del software. Madrid: Centro de Estudios Ramón Areces.

Sutherland, J., Rey, V. & Sutherland, J. (2018). Scrum: el revolucionario método para trabajar el doble en la mitad de tiempo. Barcelona: Ariel.

Beck, K. & Andres, C. (2005). Extreme programming explained: embrace change. Boston, MA: Addison-Wesley.