CENTRE ESPECÍFIC
D´EDUCACIÓ A DISTÀNCIA DE
LA COMUNITAT VALENCIANA

UD 01

DESARROLLO DE SOFTWARE

ENTORNOS DE DESARROLLO 23/24

SOLUCIONES ACTIVIDAD DE PROACTIVIDAD 2 - CUESTIONARIO SOBRE ISW

Autor: Sergio Badal
Modificado: Raúl Palao
Fecha: 26-9-2023

❶	***En las fases del ciclo de vida del software existen dos fases con nombre similar: implantación e implementación. ¿Podrías decir de qué fases estamos hablando, qué fases tienen antes y después y contextualizarlas con un ejemplo (imagina que diseñas una app para iphone)?***

El ciclo de vida del software comprende 7 fases: análisis, diseño, codificación/implementación, pruebas, documentación, implantación/explotación y mantenimiento.

La fase de implementación es otra forma de llamar a la fase de codificación y va justo después de la fase de diseño del software y antes de la fase de pruebas. En esta fase se genera el código y todos los elementos necesarios para que el aplicativo funcione acorde a las especificaciones definidas en la fase de análisis y a los modelos creados en la fase de diseño, como son la base de datos y cualquier sistema de información necesario.

En el caso de una app para iphone, en esta fase se crearía el código fuente en el lenguaje escogido en las fases previas según las características del proyecto o por imposición del cliente.

- **FUENTE**: https://elvex.ugr.es/idbis/db/docs/lifecycle.pdf (página 18)

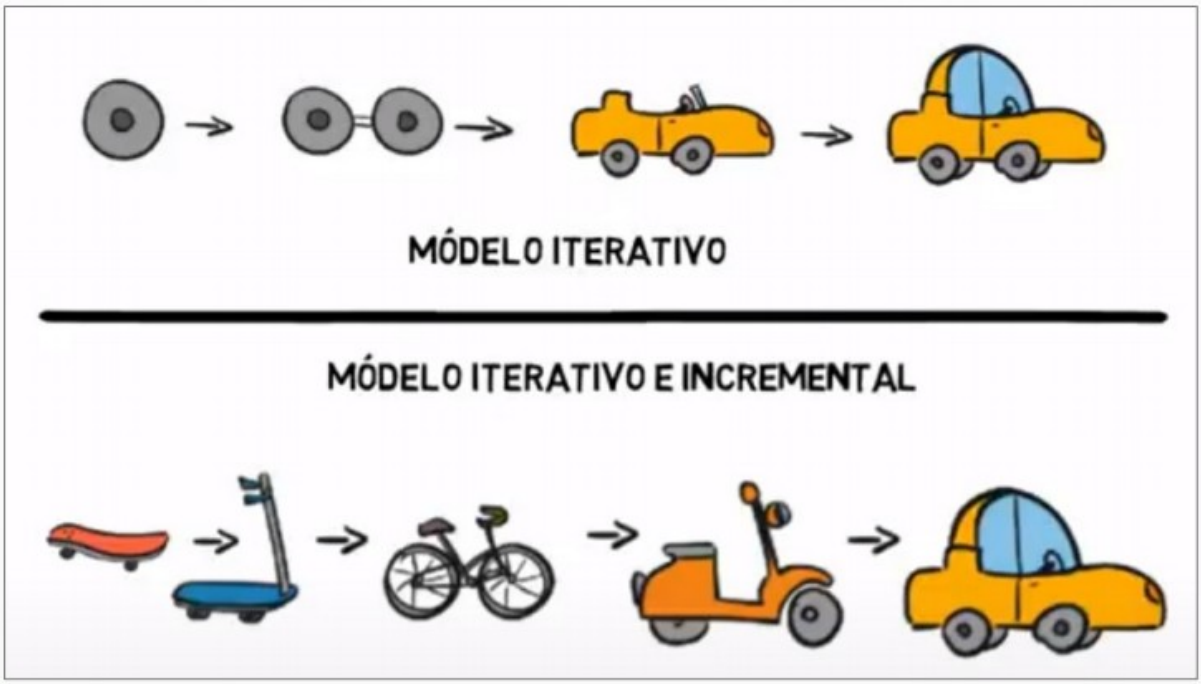
❷	***En el ciclo de vida incremental existe una fase llamada integración. ¿Podrías buscar en la Red y en los apuntes y decirnos, con tus propias palabras, en qué consiste?***

Existen diferentes variantes del ciclo de vida en cascada. Una de ellas es el ciclo de vida incremental. Tanto este como la variante iterativa se basan en repeticiones de las fases hasta conseguir el producto final.

La diferencia con el iterativo, en el incremental, con cada iteración se obtiene un prototipo funcional, es decir, que puede ejecutarse perfectamente y sirve de base para construir el producto final. En la fase de integración lo que hacemos es hacer que el prototipo crezca con las nuevas funcionalidades.

Podemos pensar en una web de venta de libros. Creada de manera iterativa, en una primera iteración le mostraríamos al cliente cómo sería el registro del usuario, en otra iteración cómo sería la compra de un libro… Creada de manera incremental, le mostaríamos la web funcionando solo con la información de la empresa, en el siguiente incremento con un catálogo de libros (sin compra), en el siguiente con compra pero solo con tarjeta… y así hasta completar el proyecto. En cada incremento, integramos lo nuevo con lo que ya aprobó el cliente en la iteración anterior.

Esta imagen y este video lo ilustran a la perfección:



- **FUENTE**: https://www.youtube.com/watch?v=_qUlL01th2s

❸	***El inicio de las metodologías ágiles se data el 12 de febrero de 2001, cuando 17 expertos firman lo que se llamó Manifiesto Ágil.*** • ***FUENTE: https://agilemanifesto.org/iso/es/manifesto.html*** ***Primero: Lista 12 principios que contiene (MANIFIESTO ORIGINAL)*** ***Segundo: Intenta comprimirlos en 5 (MANIFIESTO RESUMIDO)*** ***Sugerencia: Agrupa los puntos que hablan del equipo, de las entregas, del producto que se quiere obtener....*** ***Por ejemplo: Si seleccionamos todos los puntos que hablan de entregas. Tendremos unidos los puntos 1, 3 y 7 en un único punto.***

MANIFIESTO ORIGINAL:

1. Nuestra mayor prioridad es satisfacer al cliente mediante la entrega temprana y continua de software con valor.
2. Aceptamos que los requisitos cambien, incluso en etapas tardías del desarrollo. Los procesos Ágiles aprovechan el cambio para proporcionar ventaja competitiva al cliente.
3. Entregamos software funcional frecuentemente, entre dos semanas y dos meses, con preferencia al periodo de tiempo más corto posible.
4. Los responsables de negocio y los desarrolladores trabajamos juntos de forma cotidiana durante todo el proyecto.
5. Los proyectos se desarrollan en torno a individuos motivados. Hay que darles el entorno y el apoyo que necesitan, y confiarles la ejecución del trabajo.
6. El método más eficiente y efectivo de comunicar información al equipo de desarrollo y entre sus miembros es la conversación cara a cara.
7. El software funcionando es la medida principal de progreso.
8. Los procesos Ágiles promueven el desarrollo sostenible. Los promotores, desarrolladores y usuarios debemos ser capaces de mantener un ritmo constante de forma indefinida.
9. La atención continua a la excelencia técnica y al buen diseño mejora la Agilidad.
10. La simplicidad, o el arte de maximizar la cantidad de trabajo no realizado, es esencial.
11. Las mejores arquitecturas, requisitos y diseños emergen de equipos auto-organizados.
12. A intervalos regulares el equipo reflexiona sobre cómo ser más efectivo para a continuación ajustar y perfeccionar su comportamiento en consecuencia.

MANIFIESTO RESUMIDO:

1. Entregas funcionales, lo más frecuentes posibles, tempranas, continuas y con valor. (1,3,7)
2. Equipo motivado, auto-organizado y en continuo ajuste y perfeccionamiento. (5, 11, 12)
3. Los requisitos cambiantes aportan una ventaja competitiva al cliente (2)
4. Simplicidad, excelencia técnica y desarrollo a ritmo constante de forma indefinida (8, 9 ,10)
5. Cliente y desarrolladores trabajan juntos de forma cotidiana y se comunican cara a cara durante todo el proyecto. (4, 6)

FUENTE: https://agilemanifesto.org/iso/es/manifesto.html

4	***Enhorabuena, tu empresa ha firmado un contrato para realizar una aplicación de móvil por valor de 500.000 de euros, pero tu jefe sigue se empeña en usar el ciclo de vida en cascada. Ahora, en noviembre, cerrará los requisitos con el cliente y en julio tenéis prevista la entrega. Busca en la Red qué ocurrirá si el cliente cambia los requisitos iniciales en enero, es decir, si os escribe un email u os llama para deciros que hay ciertas funcionalidades que quiere modificarlas.***

Aunque probablemente él no sepa de qué estamos hablando, el cliente ha firmado un desarrollo de software que sigue el método en cascada. Este modelo propone una serie de fases que van desde la fase de análisis a la fase de mantenimiento de manera lineal y unidireccional, es decir, que que no existe repetición, ni retroalimentación, ni vuelta atrás en ninguna de sus fases. Por tanto, no es posible cambiar los requisitos a mitad del ciclo de vida.

La primera opción es siempre hablar con el cliente para tener claro qué necesita, ya que muchas veces no tiene nada que ver “lo que pide con lo que necesita”. Si, tras esta conversación con el cliente, vemos que el proyecto realmente necesita un cambio de requisitos debemos cancelarlo y volver a redactar una nueva especificación de requisitos, asumiendo el cliente todos los costes que esto pueda suponer (económicos como temporales).

En la práctica, intentaríamos “hacer lo imposible” para reutilizar lo que ya teníamos y seguir adelante con el proyecto, reestructurando plazos y costes y, con casi total seguridad, asumiendo nuestra empresa parte de esos sobrecostes para no perder al cliente.

5	***Todas las fases del ciclo de vida del software son importantes pero queremos que valores y analices la importancia de eliminar cada una de ellas del ciclo de vida de tu proyecto.*** ***Imagina un escenario en el que tienes que desarrollar un proyecto software desde cero en un tiempo mínimo. El cliente asume que el producto no será "robusto" y tú te niegas a hacerlo pero tus jefes te presionan y accedes con la condición de que el cliente asuma ciertas consecuencias por escrito.*** ***Existe una etapa que no puedes eliminar bajo ningún concepto ¿Cuál es esa etapa? ¿Crees que hay más de una "intocable"? No hay una única respuesta si la argumentas correctamente.*** ***Repasa todas las etapas e indica en cada una de ellas las consecuencias de eliminarla de la planificación, es decir, qué le harías firmar al cliente si (para ahorrar tiempo) acordáis eliminar esa etapa.*** ***Por ejemplo:*** ***Pruebas. Si elimina esta etapa la aplicación no podrá probarse por una persona ajena a la fase de codificación y podrá contener errores, por tanto, usted asume el coste de solucionar estos errores (en caso de que surjan).***

1. **Análisis**: En esta fase nuestro analista experto en la captación de requisitos se pondría en contacto con usted para ayudarle a averiguar qué es lo que necesita. Si elimina esta fase este diálogo se limitará a una reunión rápida con nuestro programador en la que será usted quien tenga que decirle qué tipo de aplicación quiere, afectando a la calidad del producto que le entreguemos.

2. **Diseño**: Si eliminamos esta fase el programador tendrá que estudiar a fondo las Especificaciones de Requisitos y deducir qué lenguajes y estructuras de datos usar, pudiendo no optar siempre por la solución más óptima.

3. **Implementación**: Esta fase es la única que no se puede eliminar.

4. **Pruebas**: Si elimina esta etapa la aplicación no podrá probarse por una persona ajena a la fase de codificación y podrá contener errores, por tanto, usted asume el coste de solucionar estos errores (en caso de que surjan).

5. **Documentación**: Si elimina esta fase y el programador o los programadores asignados a su proyecto sufren algún tipo de indisposición o abandonan el proyecto, los nuevos programadores puede que necesiten un tiempo para poder entender el código. Usted asumiría ese tiempo.

6. **Implantación**: Si elimina esta fase deberá ser usted mismo quien publique el aplicativo en el servidor, la app en el app store, en el play store o en cualquier otro que sea el destino del producto, haciéndose usted cargo de todos los posibles problemas o costes que tenga esta fase.

7. **Mantenimiento**: Si elimina esta fase no estará cubierto ante posibles evolutivos (nuevas funcionalidades), correctivo (errores no detectados), o adaptativo (cambios en la legislación) y tendrá que pedirnos presupuesto en cada una de esas situaciones (si sucedieran) o cerrar un contrato de mantenimiento con posterioridad a la entrega.