Entornos de desarrollo – Actividades no evaluables 2

1. 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)?

· **Implantación** → Se refiere a la fase de explotación, cuando se entrega la solución (deploy). Antes va la fase de documentación y después la de mantenimiento. P.e.: subimos nuestra app a la AppleStore o se la entregamos al cliente.

· **Implementación** → Se refiere a la fase de codificación, cuando se programa atendiendo a los requisitos y al diseño. Antes va la fase de diseño y después va la de pruebas. P.e.: tras tener el diseño y los requisitos, codificamos en Swift la funcionalidad de la app.

2. 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?

Consiste en integrar, incremento a incremento, todas las funcionalidades necesarias en la entrega final; esto es, sumar todas las partes útiles de cada entrega para obtener la entrega final.

3. 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.

12 principios: 1- Satisfacción del cliente mediante entrega continua y temprana. 2- Requisitos cambiantes. 3- Entregas funcionales frecuentemente. 4- Trabajo en equipo en igualdad. 5- Motivación de los trabajadores y delegación de tareas. 6- Reuniones cara a cara. 7- Software funcional. 8- Desarrollo sostenible con ritmo constante. 9- Excelencia técnica y buen diseño. 10- Simplicidad, reducción de carga de trabajo. 11- Equipos auto-organizados. Sin “jefes”. 12- Mejora continua.

4. Enhorabuena, tu empresa ha firmado un contrato para realizar una aplicación de móvil por valor de 500.000 de euros, pero tu jefe 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.

Puesto que se trata de una metodología rígida, los requisitos no pueden cambiarse una vez terminada la fase de análisis (o requisitos) porque ello supondría volver a empezar todo el proceso, ya que se habría creado la memoria en la que se especifican los requisitos, se habría comenzado a diseñar la aplicación e incluso puede que se hubiera alcanzado la fase de implementación (teniendo en cuenta la brevedad del proyecto). En caso de darse unos nuevos requisitos o modificar los existentes, el plazo de entrega y los costes asociados cambiarían (entregar más tarde y costar más).

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).

Bajo mi criterio las fases del ciclo de vida del software deben mantenerse todas, pero teniendo en cuenta que hacen falta requisitos para desarrollar, diseñar y programar la solución a dichos requisitos, creo que mantendría sí o sí las fases de análisis, diseño y codificación.

Dicho lo cual y suponiendo que se eliminase una de las fases, esta sería la repercusión y esto sería lo firmado en cada caso, eliminando… :

- Análisis: Al eliminar esta fase no obtendríamos requisitos, con lo que no podría diseñarse la aplicación y, por tanto, no podría llevarse a cabo ninguna de las otras fases (al menos acorde a lo que busca el cliente, podría hacerle un buscaminas). Esta fase es indispensable, por lo cual, en caso de querer eliminarla, no participaría en el proyecto. Lo considero el "qué" del proyecto. Sin "qué", no hay proyecto.

- Diseño: Al eliminar esta fase no tendríamos una idea del funcionamiento global teniendo en cuenta los recursos. No se implementaría una solución y en la fase de programación no habría qué programar tampoco, únicamente requisitos. Lo considero el "cómo", y sin "cómo" tampoco hay proyecto.

- Codificación: Al eliminar esta fase tendríamos requisitos y solución, pero dicha solución no estaría implementada. Sin código no hay aplicación ni proyecto.

- Pruebas: explicado en el ejemplo.
- Documentación: Al eliminar esta fase no habría documentación y se tardaría un poco más en retomar el proyecto en caso de reemplazo o ampliación del equipo. Esta fase es prescindible y es, de hecho la primera en ser descartada, aún así usted se compromete a no pedir ningún cambio para evitar requerir un nuevo equipo o una ampliación del mismo, evitando así retrasos en la entrega final.

- Explotación: Al eliminar esta fase la aplicación se entregaría sin prepararla para su entrega / distribución y esto podría causar algún tipo de fallo inesperado. En dicho caso usted se compromete a asumir las consecuencias y/o costes derivados.

- Mantenimiento: Al eliminar esta fase no se dispondría de actualizaciones para ampliar / eliminar / modificar funcionalidades o para adaptar la aplicación a la legislación vigente u otras variaciones del contexto, ni correcciones de errores. Dicho esto, usted se compromete a nada más pedir ni reclamar tras la entrega del software.

Fuentes: Apuntes ED, https://agilemanifesto.org/iso/es/manifesto.html , https://es.wikipedia.org/wiki/Desarrollo_en_cascada ,