(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)? Implementación o codificación es aquella fase que realiza el programa atendiendo a todos sus componentes; esto incluye elementos como base de datos, servidores o comunicaciones. Implantación o explotación consiste en publicar la solución final en la plataforma de destino o entregar al cliente el producto final en el formato acordado.
App para iphone que mejore tu seguimiento en tu dieta para adelgazar, la fase de análisis inicial es conocer que debe de hacer la aplicación recopilando información sobre tus comidas y ejercicios, con los requisitos funcionales avisándote en las horas de ingesta donde indiques que has comido y cuanto ejercicio has hecho y con los requisitos no funcionales sobre características de sistema en este caso IOS, utilizando un determinado lenguaje de programación estableciendo plazos de entrega, tiempo de ejecución o más detalles técnicos. Posteriormente se diseñaría la app con diagramas indicando los flujos en su orden para el correcto funcionamiento de la app, se realizarían pruebas, se documentaría, se implantaría (explotaría) y se mantendría.

(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? Según los apuntes se entiende que es la fusión de las mejores diseños, implementaciones y pruebas integradas con los mejores resultados para la optimización del producto 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)
    1. Satisfacer al cliente con entrega temprana y de valor.
    2. Aceptar que los requisitos cambien en cualquier etapa del desarrollo.
    3. Entrega de software funcional de 2 semanas a 2 meses.
    4. Cooperación entre responsables de negocio y desarrolladores.

5. Equipos motivados para el proyecto a realizar. Entorno, ayuda y confianza al equipo desarrollador.
6. La comunicación de información es imprescindible entre el equipo.
7. Progreso significa software funcionando.
8. Agilidad en los procesos entre desarrolladores, promotores y usuarios.
9. Excelencia técnica y buen diseño mejoraran la agilidad.
10. La simplicidad es esencial.
11. Equipos auto-organizados con las mejores arquitecturas, requisitos y diseños.
12. Reflexión constante del equipo para ser mas efectivo y poder ajustar el proceso.

- Segundo: Intenta comprimirlos en 5 (MANIFIESTO RESUMIDO)

1. Entregas; puntos 1,3 y 7.
2. Requisitos; puntos 2, 11 y 12.
3. Agilidad; puntos 8 y 9.
4. Equipo; puntos 4, 5, 6 y 11.
5. Eficacia; punto 10

- 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.

(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.

Deberemos tener un buen conocimiento y análisis de nuevo sobre esos cambios de requisitos para poder mejorar la capacidad, tomar mejores decisiones y gestionar los requisitos cambiantes de manera efectiva.

Fuente: http://scielo.sld.cu/scielo.php?script=sci_arttext&pid=S2227-18992020000200131

(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).

En mi opinión creo que todas las fases son intocables, ya que, si no se sigue la estructura del ciclo, no habría un buen estudio de la aplicación y podría estar constantemente produciendo error tras error. Cada fase depende de la anterior y sin una continuidad en el proceso, no se alcanzaría el propósito inicial de lo que el producto final tiene que realizar. Pero me parece intocable la fase de análisis sin la que no habría una estructura clara a seguir con lo que debiéramos conseguir para que sirviera el software.