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)?***

Entendemos la fase de implantación como el momento en el que el Software es completamente funcional sobre los requisitos exigidos, se lanza versión oficial del mismo, implantación en contrapartida, es el momento de codificación, el Software, aunque tenemos claro en que va a consistir y se ha realizado un análisis minucioso, está en una fase temprana de desarrollo, aun no existe como tal, solo conceptos, aunque ya se ha realizado también su diseño.

En el desarrollo pasaremos por 7 fases: Análisis, Diseño, *Codificación/Implementación*, Pruebas, Documentación, *Implantación/Explotación* y Mantenimiento.

Para tenerlo un poco mas claro, en una App para iPhone, en la fase de *Implementación*, crearíamos el código de programa, mientras que, en la fase de *Implantación*, la App estaría terminada y sería momento de ser lanzada en la Store.

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?***

En le tipo de Ciclo de Vida Incremental las fases no transcurren de forma unidireccional, las fases se retroalimentan, pueden pasar cada fase hasta tener el producto terminado y listo para ser lanzado, esta fase sería la llamada *Integración*, pero si fuese necesario para una mejora o para añadir nuevas funcionalidades, las fases se pueden volver a repetir y así mejorar el producto.

Imaginemos una aplicación de mensajería, inicialmente y con el método Iterativo, sería lanzada como una versión funcional pero que solo enviaría mensajes de un usuario a otro, el tiempo hará que los usuarios pidan nuevas funcionalidades, sin tener que hacer la aplicación desde cero, el modelo Incremental nos permitiría ir añadiendo las funcionalidades que hiciera falta poco a poco, envío multimedia, llamadas, geolocalización, etc. Este método hace a la aplicación ser mas viva, puesto que crece y se perfecciona constantemente.

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.***

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

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.***

Un ciclo de vida en cascada conlleva un análisis y diseño en las fases prematuras del proyecto, siempre con las directrices dadas por el cliente, un cambio de peso en alguno de estos requisitos una vez avanzado el proyecto podría tener unas consecuencias bastante importantes, sobre todo en lo que tiempo de desarrollo se refiere.

Antes de llegar a este inconveniente, y viendo de antemano que puede llegar a ocurrir, hay que intentar que el proyecto siga otros métodos mucho mas flexibles y menos sensibles a los cambios, si el desarrollo se alarga en el tiempo se hacen más probables estos cambios (cliente quiere nuevas funcionalidades, ve que la competencia se mueve y él quiere los mismos

movimientos, etc), hay que intentar predecir estos futuros problemas e inconvenientes antes de que sea demasiado tarde.

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).***

Considero que pese a que todas conllevan un riesgo el eliminarlas del proceso, si hay algunas con mas peso que otras, las pruebas son importantes, pero sin la codificación el producto no existiría directamente, por lo que, para mi punto de vista la *Codificación/Implementación* fase mas crítica del proceso, si el cliente da unas instrucciones claras de lo que quiere y necesita, siendo esto real o no, el proceso de Analisis podría llegar a ser prescindible, siempre con riesgos, la fase de Documentación por su parte, no tendría efectos directos sobre el producto, pero, es probable que una ampliación o revisión futura del producto se hiciera un tormento.

Bajo mi punto de vista eliminaría una de esas tres fases:

1- Análisis: Si el cliente da unas instrucciones claras.
2- Documentación: Si el producto no va a tener una larga vida o va ser poco revisado.
3- Pruebas: Esta sería la fase mas critica de quitar de las 3, un producto sin pruebas esta casi avocado al fracaso si algo no sale como esperamos

Usted decide de cuales quiere prescinda, todas tienen su importancia y se las he ordenado de mayor a menor, tenga claro que puede que algo salga mal por llevar un camino optimo, en sus manos está el correr los riesgos y lo problemas que pudieran surgir.