MDA 22/23 - Relación de problemas 1

Tema 1: Desarrollando software

1. ¿Cuáles crees que son los tres principales problemas en el desarrollo de software?

1. Estimaciones incorrectas de tiempo. Tenemos que ser realistas con el tiempo de desarrollo del producto y dar espacio a los incidentes que seguro van a surgir durante la implementación y desarrollo del software
2. Fallos en la comunicación y planteamiento. Es difícil determinar los requisitos que realmente quiere un cliente, por eso antes de empezar con la implementación hay que dedicar un tiempo considerable a reuniones con el cliente, especificaciones de requisitos, la aprobación de estos... Si no, acabaremos desarrollando un producto que no es lo que el cliente necesita y, por tanto, perderemos tiempo deshaciéndolo y planteándolo de nuevo.
3. Diseño incorrecto. Debemos desarrollar un producto que se adapte a las necesidades del cliente, pero también del desarrollador. Debe tener un diseño correcto, adecuado al uso que se va a dar, mantenible y extensible. Al igual que en el punto anterior, es preferible dedicar más tiempo al planteamiento del producto, que empezar a desarrollar y luego tener que rectificar problemas subyacientes.

2. ¿Por qué crees que es importante la calidad en el proceso de desarrollo de software?

El uso de buenas prácticas en todas las etapas del desarrollo genera un producto usable y mantenible, que se adecua a las expectativas del cliente y deja espacio a mejora. Alarga el tiempo de vida de nuestro producto, porque es sostenible mantenerlo y expandirlo.

Un producto que se desarrolla sin calidad, sin cuidado en su planificación e implementación, nos dejará con software mediocre. Quizá sirva al cliente en su necesidad, pero nos dará problemas a la larga en su mantenimiento y, si no ha sido bien planificado, no cumplirá con las expectativas del cliente.

3. ¿En qué se diferencian el modelo en cascada de los modelos iterativos e incrementales?

El modelo en cascada define una serie de etapas en el desarollo que se van cumpliendo secuencialmente. Es un enfoque un tanto simplista porque, en la práctica, es difícil dejar totalmente cerrada cualquiera de las etapas, es muy probable que en algún momento tengamos que volver atrás para revisar o modificar algo.

El modelo iterativo incremental repite todas las etapas de desarrollo en cada iteración: de esta forma, al final de una iteración siempre obtenemos un producto final con funcionalidad completa.

4. ¿Por qué crees que la documentación forma parte del software?

Porque es necesaria para el uso y mantenimiento del mismo. El objetivo de un producto software es ser usado y, si nadie sabe cómo utilizar nuestro producto, significa que hemos fallado en el desarrollo del mismo.

5. Comenta las dos características principales de los métodos ágiles que expone Martin Fowler, en su artículo "The New Methodology"

1. Adaptativo > Predictivo - Planificar un proyecto software de forma demasiado estricta sólo puede llevarnos a que a la mínima variación el plan se desmorone. Nuestro objetivo debería ser obtener el mejor producto posible para el cliente, no ceñirse a un plan, así que deberíamos aceptar el cambio y adaptarnos a las necesidades del proyecto en cada momento.
2. Orientado a las pesonas, no a los procesos - A la hora de desarrollar, no podemos pensar en los desarrolladores como máquinas de completar objetivos: esto da la sensación de que todo el equipo es reemplazable y el producto saldrá adelante siempre que haya cualquiera que complete la lista de objetivos. Rechazamos esta idea porque, simplemente, no es así. Un equipo bien formado, organizado y motivado llevará a buen puerto el proyecto porque lo desarrollará en grupo de forma colaborativa, como personas.

Tema 2: Principios y prácticas ágiles

6. La "Alianza Ágil" definió una serie de 12 principios que debería tener una metodología para alcanzar niveles aceptables de agilidad. Comentar cuales son los motivos o los problemas que intentan solucionar cada uno de estos principios.

1. ***Nuestra mayor prioridad es satisfacer al cliente mediante la entrega temprana y continua de software con valor.***

   Producir buen software, a tiempo y consistentemente. Si no tenemos buenas prácticas, probablemente no seamos capaces de cumplir con alguna o varias de esas tres expectativas.

2. ***Aceptamos que los requisitos cambien, incluso en etapas tardías del desarrollo. Los procesos Ágiles aprovechan el cambio para proporcionar ventaja competitiva al cliente.***

   El modelo iterativo nos da la ventaja de que un cambio a posteriori sólo afecta a un área, y no a todo el proyecto como un modelo en cascada.

3. ***Entregamos software funcional frecuentemente, entre dos semanas y dos meses, con preferencia al periodo de tiempo más corto posible.***

   No desarrollar en una caja negra, de esta forma podemos recibir feedback continuo e implementar los cambios y mejoras más pronto que tarde.

4. ***Los responsables de negocio y los desarrolladores trabajamos juntos de forma cotidiana durante todo el proyecto.***

   Contacto con el cliente, el software no se crea para los desarrolladores, sino para los usuarios.

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.***

   Un equipo de individuos motivados y capaces podrán gestionar y estimar el trabajo correctamente, sin necesidad de presiones por cargos medios.

6. ***El método más eficiente y efectivo de comunicar información al equipo de desarrollo y entre sus miembros es la conversación cara a cara.***

   Muchos intercambios de correos sin sentido se pueden comunicar en menos de 15 minutos cara a cara.

7. ***El software funcionando es la medida principal de progreso.***

   Un producto funcionando es la manera más efectiva de comunicar al cliente el trabajo que hemos desarrollado.

8. ***Los procesos Ágiles promueven el desarrollo sostenible. Los promotores, desarrolladores y usuarios debemos ser capaces de mantener un ritmo constante de forma indefinida.***

   El modelo iterativo y los procesos ágiles favorecen al desarrollo de un software mantenible. Si se hace correctamente, el desarrollo del producto puede alargarse en el tiempo indefinidamente sin causar problemas (aunque también hay que saber determinar cuándo un producto está completo).

9. ***La atención continua a la excelencia técnica y al buen diseño mejora la Agilidad.***

   Las buenas prácticas generan un producto fácilmente escalable y sostenible

10. ***La simplicidad, o el arte de maximizar la cantidad de trabajo no realizado, es esencial.***

    Optimizar siempre, menos es más.

11. ***Las mejores arquitecturas, requisitos y diseños emergen de equipos auto-organizados.***

Equipos capaces y autogestionados, sin jefes o cargos medios. El proyecto es de todos.

12. ***A intervalos regulares el equipo reflexiona sobre cómo ser más efectivo para a continuación ajustar y perfeccionar su comportamiento en consecuencia.***

    Si estamos desarrollando el software de esta manera es para poder recibir e implementar el feedback que recibimos.

7. ¿Qué tipos de contratos ágiles podemos encontrar? (ver aquí)

Podemos clasificarlos con tres baremos: según variabilidad, según alcance y según plazo

- Contrato cerrado: importe fijo, alcance fijo, plazo indefinido. Un proyecto con ciertas características definidas en un contrato.
    - Sin complejidad: importe fijo, alcance fijo, plazo fijo. Un proyecto pequeño, quizá una función específica sobre un framework... Tareas sin mayor dificultad técnica con una deadline impuesta
- Alcance no vinculante: importe fijo, alcance variable, plazo indefinido. Los límites del proyecto están difusos y dependerán de otros factores externos al contrato, donde no se definen.
    - Tanto como puedas: importe fijo, alcance variable, plazo fijo. Autodescriptivo.
- Coste objetivo: importe variable, alcance fijo, plazo indefinido. La compensación dependerá de la calidad del producto software y de los objetivos alcanzados.
- Progresivos: importe variable, alcance variable, plazo indefinido. Carrera de fondo, un producto con un desarrollo y mantenimiento muy extendido en el tiempo.

8. Los equipos ágiles son auto-organizados. ¿Qué ventajas tienen estos equipos frente a los equipos jerarquizados?

Que todos los miembros del equipo se involucran y sienten el proyecto como suyo. Eliminando el puesto de *Product Manager*, la figura de un jefe que dirige pero no participa, te aseguras de que todos los miembros del proyecto están cualificados para su desarrollo, obteniendo un equipo cualificado, realista con los objetivos, mejor organizado.

Seminario 1: Los escenarios

9. ¿Qué diferencias encuentras entre los escenarios y los casos de uso?

Un escenario describe la forma en la que un usuario interactúa con el sistema en un contexto para alcanzar un objetivo. Es una interacción muy específica, en la que tenemos en cuenta qué clase de persona está interactuando el sistema, cómo y bajo qué condiciones lo hace, sus expectativas, lo que piensa... El escenario es una descripción detallada de una situación específica.

Un caso de uso es mucho más genérico que un escenario, ya que se limita a describir qué salida obtendremos del sistema partiendo de cierta entrada.

10. Indica los elementos de los escenarios.

- Objetivos: ¿qué pretende conseguir el usuario?
- Procesos: ¿qué pasos sigue el usuario para lograr su objetivo?
- Entradas y salidas: durante esos pasos, ¿que información o material necesita el sistema? ¿y el usuario?
- Experiencia: ¿el usuario está familiarizado con cómo se llevan los procesos? ¿ha hecho esto o algo similar antes?
- Restricciones: ¿existe alguna restricción física, temporal, logística?
- Entorno físico: ¿qué necesita el usuario para llevar a cabo la actividad? ¿dónde?
- Herramientas a usar: ¿qué hardware y software?
- Relaciones: ¿existen terceros que se vean afectados por la tarea?

11. ¿Se pueden utilizar los escenarios como base para definir las pruebas de un sistema software? Justifica tu respuesta.

Sí y no. Las pruebas del sistema software se pueden definir hasta cierto punto basándonos en los escenarios, pero también han de incluir un testing más formal a nivel técnico, situaciones que posiblemente no se plantearían en un escenario.

12. De todos los posibles usos de los escenarios cuál crees que es el más importante.

El poder determinar de forma orgánica cómo se interactúa con el sistema. Los flujos de una tarea a otra, cómo lo maneja el usuario, su objetivo... Es una forma muy eficaz para determinar las funciones del sistema de manera natural.