UNIVERSIDAD DE MÁLAGA

Plan de calidad

Proyecto VIGIA

Autores:

Manuela Ruiz Montiel
Rafael Martínez González

Versión: 2.0

Fecha: 10/12/2008

Plan de calidad

¿Cómo vamos a medir la calidad del software?

Necesitamos una serie de atributos concretos que medir.

Nos basamos en el ISO 9126. En algunos atributos hemos puesto ejemplos de a lo que se pueden referir para que quede más claro:

- Funcionalidad - Un conjunto de atributos que se relacionan con la existencia de un conjunto de funciones que satisfagan los requisitos del software.
  - Idoneidad - ¿El software es idóneo para los requisitos que han de cumplirse?
  - Exactitud - ¿El software hace exactamente lo que queremos?
  - Cumplimiento de normas - ¿El software cumple las normas impuestas (idioma, estilo de programación, etc.)?
  - Seguridad
- Fiabilidad - Un conjunto de atributos relacionados con la capacidad del software de mantener su nivel de prestaciones bajo condiciones establecidas durante un período de tiempo establecido.
  - Recuperabilidad - Reacción a fallos externos
  - Tolerancia a fallos - Tratamiento de excepciones
- Usabilidad - Un conjuntos de atributos relacionados con el esfuerzo necesitado para el uso, y en la valoración individual de tal uso, por un conjunto de usuarios. (Aquí entra en juego el tema de la documentación).
  - Aprendizaje (¿El software sigue patrones típicos? ¿Tiene documentación?)
  - Comprensión - ¿La documentación y el código son fácilmente comprensibles y visuales?
- Eficiencia - Conjunto de atributos relacionados con la relación entre el nivel de desempeño del software y la cantidad de recursos necesitados bajo condiciones establecidas.

- Comportamiento en el tiempo - ¿El sistema cumple con los mínimos establecidos de tiempo de respuesta?
- Comportamiento de recursos - ¿Qué uso hace el sistema de los recursos disponibles?

- Mantenibilidad - Conjunto de atributos relacionados con la facilidad de extender, modificar o corregir errores en un sistema software.
  - Estabilidad
  - Facilidad de análisis
  - Facilidad de cambio
  - Facilidad de pruebas
- Portabilidad - Conjunto de atributos relacionados con la capacidad de un sistema software para ser transferido desde una plataforma a otra.
  - Capacidad de instalación
  - Capacidad de reemplazo
  - Adaptabilidad

¿Cómo vamos a asegurarnos de que lo que hacemos cumple unos mínimos de calidad?

A grandes rasgos, hemos de comprobar que se cumplen unos valores mínimos para los atributos y subatributos mencionados arriba.

Para llevar a cabo este análisis, es preciso incluir una tarea de auditoría en la planificación, en la que los testers y/o el jefe de proyecto, así como los programadores implicados, midan hasta qué punto se alcanzan los valores mínimos de cada atributo.

¿Y cuáles son estos mínimos?

Existen atributos para los que no tiene sentido hablar de mínimos, como por ejemplo la idoneidad o el cumplimiento de normas (tienen que cumplirse en su totalidad).

Otros dependerán del tipo de software que estamos desarrollando, como es la seguridad, la recuperabilidad y la tolerancia a fallos (dependiendo de las características del software, se pueden presentar diferentes situaciones a tratar y/o resolver).

Por último, el resto de atributos son comunes a todo el software:

- Exactitud: este atributo tiene sentido para el software que usemos que no sea desarrollado por nosotros (como las APIs), para el cual podemos poner la condición de que, aunque proporcione más funcionalidad de la que nos hace falta, como mínimo tiene que proveer lo necesario. Un ejemplo es el API del mando: aunque tiene mucha más funcionalidad de la que necesitamos, podemos obtener las coordenadas, así que el atributo estaría "aprobado".

- Aprendizaje: el sw tiene que seguir unos patrones conocidos o bien estar perfectamente documentados con su diagrama de clases correspondiente, amén de incluir documentación para todos sus elementos y comentarios donde sea preciso.

- Comprensión: el código no puede estar ofuscado con variables tipo "x", "xy", etc.; es decir, hay que seguir un estilo claro y limpio de programación. Además, el código y la documentación deberán estar escritos en un lenguaje técnico pero, al mismo tiempo, comprensible por cualquier persona. La documentación relativa a diagramas y tablas, principalmente, deberá proporcionar una visión clara de lo que intenta transmitir.

- Comportamiento en el tiempo: el software no puede tardar demasiado tiempo en ejecutarse (será tarea de los testers evaluar lo que es admisible y lo que no).

- Comportamiento de recursos: el software no puede agotar la memoria del sistema.

- Mantenibilidad: el software ha de ser fácil de analizar, probar y cambiar en el caso de que presente errores. Esto se consigue mediante un estilo de programación claro, un control de excepciones adecuado, la disponibilidad de la documentación y la existencia de un diagrama de clases correcto y completo.

- Portabilidad: el software ha de ser medianamente portable, para lo cual se definirán interfaces donde sea necesario y se desacoplarán los diferentes módulos para que puedan ser utilizados independientemente, en la medida de lo posible.