

Cualidades del Software

¿Qué es un buen software?

Cualidades del Software

- Correcto
- Confiable
- Robusto
- Eficiente
- Amigable
- Verificable
- Reusable
- Portable
- Interoperable
- Productivo
- A Tiempo
- Visible
- Coheso
- Desacoplado
- Comprensible
- Mantenible

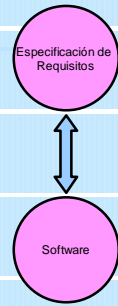


2

Correcto

Un software es correcto si se comporta de acuerdo con su especificación.

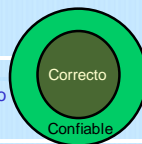
- La definición supone:
 - la existencia de una especificación de requisitos
 - la posibilidad de determinar sin ambigüedad la correspondencia entre la especificación y el diseño.
- La correctitud del software puede comprobarse probándolo o mediante análisis.



3

Confiable

El software se comporta de acuerdo con lo esperado por el usuario.



- A diferencia de la corrección, la confiabilidad es algo relativo.
- El mercado puede admitir algunos errores en el software siempre que en general se comporte en forma esperable.
- No sólo no damos garantías de corrección del software, sino que varios productos incluyen un "disclaimer".

è Esta es una señal de la inmadurez del área.

4

Robusto

Un software es robusto si se comporta en forma razonable aún en situaciones no anticipadas.



- Datos de entrada incorrectos o fallas de hardware son las situaciones más frecuentes.
- La cantidad de código que se dedica a hacer el software robusto depende de la experiencia de los usuarios o lo crítico de su misión.
- Si algo se especifica como requisito, cumplirlo es cuestión de corrección; si no está en los requisitos es cuestión de robustez.

5

Eficiencia

Un sistema de software es eficiente si usa sus recursos en forma económica.



- Muy lento è baja la productividad de los usuarios.
- Usa mucho disco è puede ser muy caro ejecutarlo.
- Usa mucha memoria è puede afectar la performance de otros sistemas.
- Los criterios de eficiencia varían con la tecnología y el tiempo.
- Métodos de evaluación de performance:
 - monitoreo,
 - análisis,
 - simulación.
- No es bueno evaluar la performance sólo después que el producto está listo.⁶

Amigable



Un software es amigable si sus usuarios lo encuentran fácil de utilizar.

- La interfaz con el usuario es parte esencial del ser amigable.
- Depende de los usuarios:
 - novicios: mejor largos mensajes explicativos.
 - expertos: aprecian los atajos.
- Los sistemas insertos son amigables si son fáciles de configurar.
- La consistencia de las interfaces es un factor determinante.
- También la performance y la confiabilidad.

7

Verificable

El software es verificable si sus propiedades pueden ser comprobadas.

- La corrección y la performance pueden verificarse fácilmente.
- La verificación puede hacerse mediante análisis o testing.
- Más verificable:
 - monitores en el código,
 - diseño modular,
 - disciplina en la codificación,
 - lenguaje de programación adecuado.

8

Reusable



Software ya construido se usa con pocos o ningún cambio.

- La reutilización es más apropiada para componentes que para sistemas completos.
- Las bibliotecas (¿librerías?) científicas FORTRAN son los ejemplos más conocidos. Java APIs son ejemplos más nuevos.
- El reuso es una cualidad difícil (¿imposible?) de conseguir a posteriori.
- La orientación a objetos tiene el potencial para mejorar la reutilización y la evolución.
- También los patrones de arquitectura y el desarrollo de familias de productos.

Portable



Un software es portable si puede ejecutarse en distintos ambientes (hardware, sistemas operativos, etc.).

- Una forma de lograr portabilidad es suponer la mínima configuración.
- Esto penaliza los sistemas que podrían ejecutar mejor haciendo uso del ambiente disponible.
- Otra opción es determinar sobre la marcha las disponibilidades del ambiente.

10

Interoperable



Un sistema es interoperable si puede coexistir y cooperar con otros sistemas.

- Las componentes reutilizables son (deben ser) inherentemente interoperables.
- La estandarización de las interfaces promueve la interoperabilidad.
- Los sistemas abiertos son casos típicos de sistemas interoperables.

11

Productivo



La productividad es la eficiencia del proceso de desarrollo del software.

- La productividad de un equipo de desarrollo es generalmente menor que la suma de las productividades individuales.
- Existen métricas para medir la productividad (LOC, puntos de función, etc.).
- La automatización y el soporte de software de desarrollo aumentan la productividad.

12

A Tiempo



El proceso de desarrollo debe obtener su producto en el tiempo planeado.

- Tener el producto a tiempo da, en general, una mejor oportunidad comercial, y a veces hace que el producto sea útil o inútil.
- Tener un producto a tiempo sin confiabilidad o eficiencia tampoco es útil.
- Requiere:
 - planificación
 - estimación del trabajo
 - hitos verificables

13

Visible



Un proceso de desarrollo de software es visible si todos sus pasos están claramente documentados, y se puede saber su estado de avance en cada momento.

- Diseño, testing, codificación e integración pueden suceder simultáneamente, pero deben coordinarse.
- La visibilidad ayuda a evaluar el impacto de las decisiones.
- También es esencial cuando existe rotación en el personal.

14

Cohesión

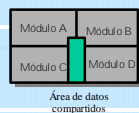
Medida de la relación entre las partes de una componente.

- Coincidental:
 - no relacionados.
- Lógica:
 - funciones similares.
- Temporal:
 - ejecución simultánea.
- Procedural:
 - secuencia de control.
- Comunicacional:
 - comparten el input.
- Secuencial:
 - output de uno es input del otro.
- Funcional:
 - todas las partes son necesarias para la función.
- Objeto:
 - todas las acciones actúan sobre los mismos datos del objeto.

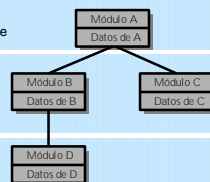
15

Acoplamiento

Medida de la interdependencia de distintas componentes.



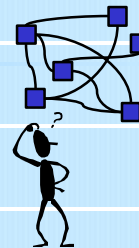
- Sistemas muy acoplados
 - comparten variables o información de control.
- Sistemas desacoplados
 - interfaces definidas con listas de parámetros.



16

Comprensible

Un sistema es comprensible si es fácil comprender cómo funciona.



- Características que afectan la comprensibilidad del sistema:
 - cohesión y acoplamiento
 - nombres
 - documentación
 - complejidad
- Si un sistema es comprensible, es también más mantenible y verificable.
- Desde el punto de vista del usuario, ser comprensible es ser amigable y robusto.

17

Mantenible



Un sistema es mantenible si es fácil modificarlo.

- Tipos de mantenimiento:
 - correctivo (aprox. 20%)
 - adaptativo (aprox. 20%)
 - perfectivo (más de 50%)
- Software Mantenible:
 - reparable: que permite corregir defectos,
 - evolucionable: facilita la introducción de nuevas funcionalidades.
- Condiciones:
 - número de componentes,
 - acoplamiento,
 - documentación
 - completa,
 - comprensible,
 - al día,
 - uso de componentes estándar.
- La evolucionabilidad decrece con cada versión del software.

18