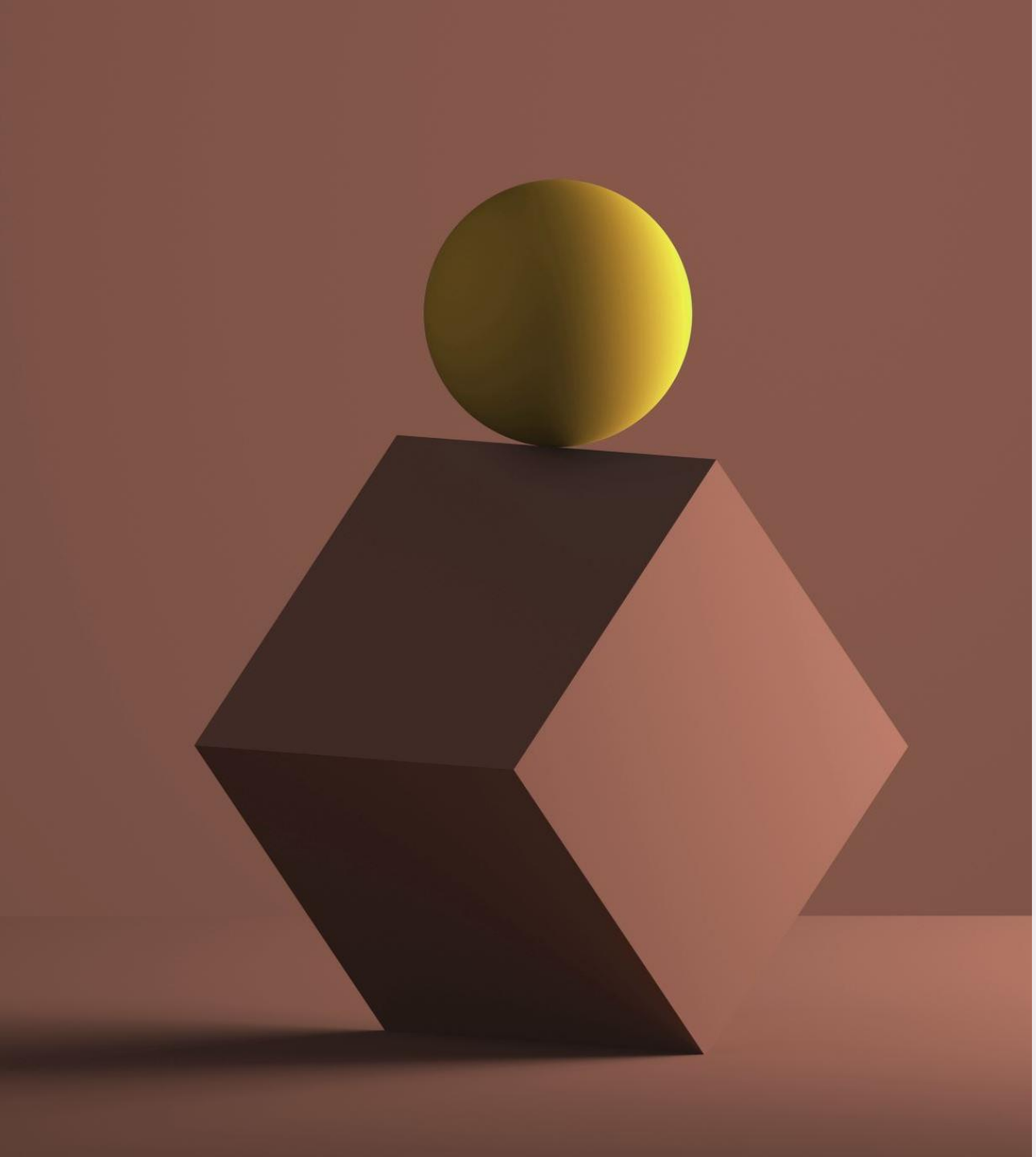


Calidad del Sofware

Ing Néstor López Luque

Universidad Nacional Autónoma de Honduras

Calidad del Software

- La Ingeniería de Software es una actividad intelectualy requiere la participación de personas
- El software se construye para cumplir un objetivo funcional satisfaciendo ciertas atributos de calidad
- El software es diferentede otros tipos de productos
- Es intangible, maleable y su construcción es una actividad creativa
- Los procesos de softwaretambién deben cumplir ciertos atributos de calidad

Atributos de calidad

Internas vs. Externas

- Externas:Visible a los usuarios y al cliente
- Internas: Visible al equipo de desarrollo
- Los atributos internos afectan a los atributos externos

Producto vs. Proceso

- Los atributos de calidad pueden aplicarse sobre productos y procesos
- La calidad del proceso afecta la calidad del producto

Atributos de calidad

- Correctitud
- Confiabilidad
- Robustez
- Rendimiento
- Usabilidad
- Verificabilidad
- Reusabilidad
- Portabilidad
- Interoperabilidad
- Acoplamiento
- Cohesión
- Comprensibilidad
- Mantenibilidad
- Productividad
- A Tiempo
- Visibilidad

Correctitud

- 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 la realización
- La correctitud es absoluta: sí o no
- La corrección del software puede comprobarse mediante pruebas o análisis

Confiabilidad

Se comporta de acuerdo a lo esperado por el usuario

- A diferencia de la correctitud, la confiabilidad es algo relativo

  El mercado puede admitir algunos errores en el software siempre que en general se comporte en forma esperable

  No hay garantías de corrección del software: varios productos incluyen un "disclaimer"

Esta es una señal de la inmadurez del área

Robustez

- Se comporta en forma razonable aún en situaciones no anticipadas

  Datos de entrada incorrectos o fallas de hardware son las situaciones más frecuentes

  Si algo se especifica como requisito, cumplirlo es cuestión de correctitud

  Si no está en los requisitos es cuestión de robustez

  El esfuerzo dedicado a robustez depende de la experiencia de los usuarios o lo crítico de la misión del producto

Rendimiento

- Usa los recursos en forma económica
- Los criterios de rendimiento varían con la tecnología y el tiempo
- Muy lento: baja productividad de usuarios
- Mucha memoria o disco: puede afectar a otros sistemas
- El rendimiento parte con la arquitectura
- Métodos de evaluación de rendimiento
- Análisis, monitoreo, simulación

Usabilidad

- Los usuarios lo encuentran fácil de usar
- La interfaz con el usuario es esencial
- Usuarios novicios: más auto-explicativo, más ayuda, más simple (menos “funciones”)
- Usuarios expertos: más “funciones”, más atajos
- Usualmente es subjetiva y difícil de evaluar
- Factores críticos: consistencia, rendimiento y confiabilidad

Verificabilidad

Sus propiedades pueden ser comprobadas

- Interesan todas las propiedades: correctitud, rendimiento, seguridad, etc.
- La verificación puede hacerse mediante análisis o pruebas

Más verificable:

- monitores en el código
- diseño modular
- disciplina en la codificación
- lenguaje de programación adecuado

Reusabilidad

- Se reutiliza a bajo costo
- Productos existentes (o partes) se utilizan (con modificaciones menores) para construir otro

  Las bibliotecas científicas FORTRAN son los ejemplos más antiguos

  APIsy Frameworksson ejemplos más nuevos
- La reusabilidad es difícil de conseguir a posteriori

  La orientación a objetos tiene el potencial para mejorar la reutilización

  También los patrones de arquitectura y el desarrollo de familias de productos

Interoperabilidad

- Puede coexistir y cooperar con otros sistemas

  Promueven la interoperabilidad:

  La especificación y uso de interfaces

  El uso de mecanismos de conexión y comunicación estándares

  Hay distintos niveles en que los sistemas pueden interoperar

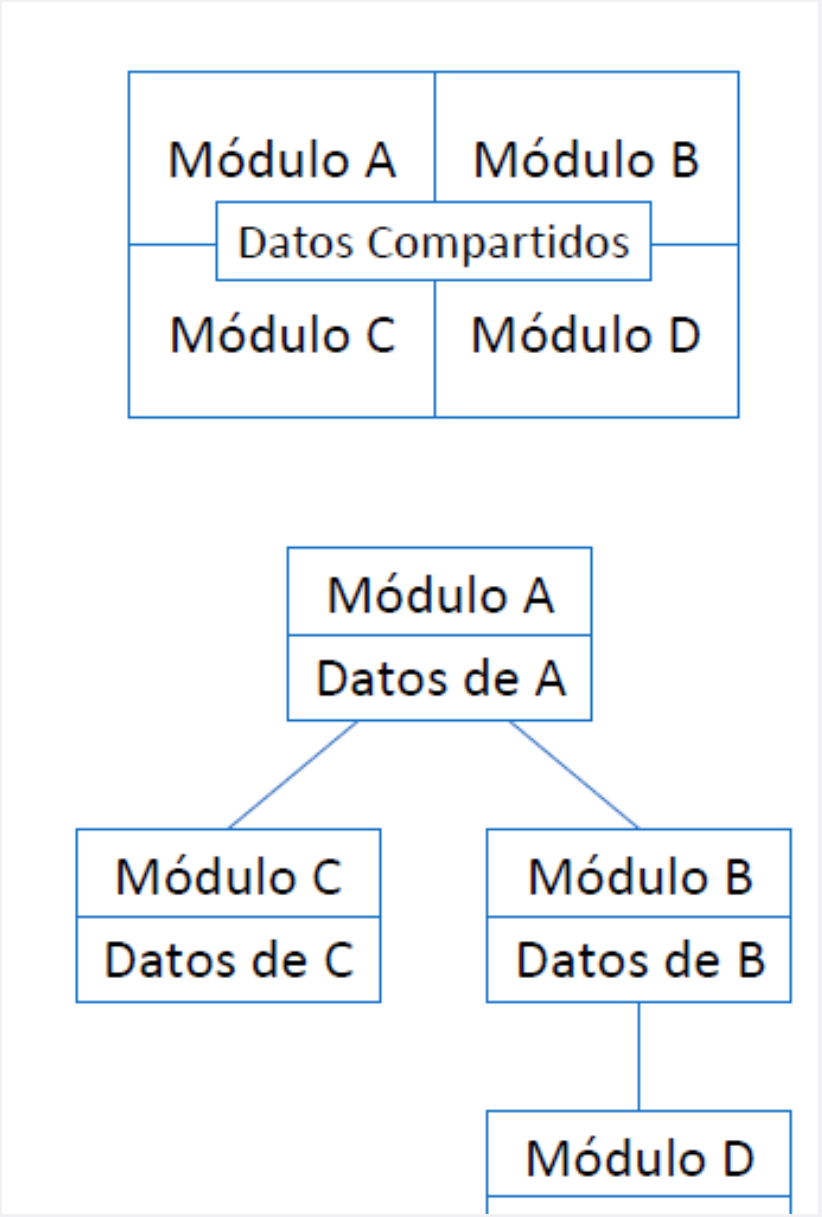
  e.g., archivos, base de datos, llamada a función, sistema de mensajería

Acoplamiento

- Es una medida de la interdependencia de distintos componentes
- Sistemas muy acoplados:

  Comparten variables o información de control
- Sistemas desacoplados:

  Interfaces definidas con listas de parámetros



Cohesión

- La cohesión es un tipo de medición ordinal y se describe generalmente como "cohesión alta" o "cohesión baja". Se prefieren los módulos con una alta cohesión debido a varios rasgos deseables del software con los que se relaciona como la robustez, la fiabilidad, la reutilización y el grado de comprensión. Por otro lado, la baja cohesión se asocia con rasgos indeseables, tales como ser difícil de mantener, probar, volver a utilizar o incluso entender.
- Se trata que las clases hagan bien una sola cosa y no se llenen de multiples responsabilidades diferentes

Mantenibilidad

- Es posible mantener el sistema

  Es reparable si permite la corrección de defectos

  Es evolucionable si permite cambios en su funcionalidad

- Tipos de mantenimiento:

  Correctivo ≈ 20%

  Adaptativo ≈ 20%

  Perfectivo > 50%

- Factores:

Tamaño, acoplamiento, documentación (completa, comprensible, al día), componentes estándar, antigüedad del software

Calidad en el proceso

- Los atributos de calidad del producto pueden aplicarse también a procesos de desarrollo
- Atributos de calidad específicos de procesos:
  - *Productividad*
  - A tiempo
  - Visibilidad

Productividad

- La productividad es la eficiencia del proceso de desarrollo del software
- Es tamaño dividido por esfuerzo

  e.g., KLOC/mes, puntos de función por semana, páginas de documentación por día

  La productividad de un equipo es distinta que la suma de las productividades individuales
- La automatización y la reusabilidad aumentan la productividad

A tiempo

- El proceso de desarrollo debe obtener su producto en el tiempo planeado
- Esto da una mejor oportunidad comercial y puede determinar la utilidad del producto
- Tener un producto a tiempo sin confiabilidad o rendimiento no es útil
- Requiere:

  Estimación del trabajo

  Planificación con hitos verificables

  Gestión de riesgos, de requisitos y de cambios

Visibilidad

- Un proyecto de software es visible si se puede conocer el estado de avance

  La planificación y la gestión son esenciales para la visibilidad

  La visibilidad permite evaluar el impacto de las decisiones

  El uso de hitos y la construcción por incrementos planificados favorece la visibilidad

Dificultades con la calidad

- En el desarrollo de software, los recursos son limitados y es necesario invertir para lograr cada atributo, y no todos pueden lograrse
- ¿Cómo priorizar?

  Algunas veces, algunos atributos están en contraposición con la éticaprofesional:

  e.g., si tenemos poco tiempo para el desarrollo y el sistema es muy crítico, ¿hacemos todas las pruebas planificadas?