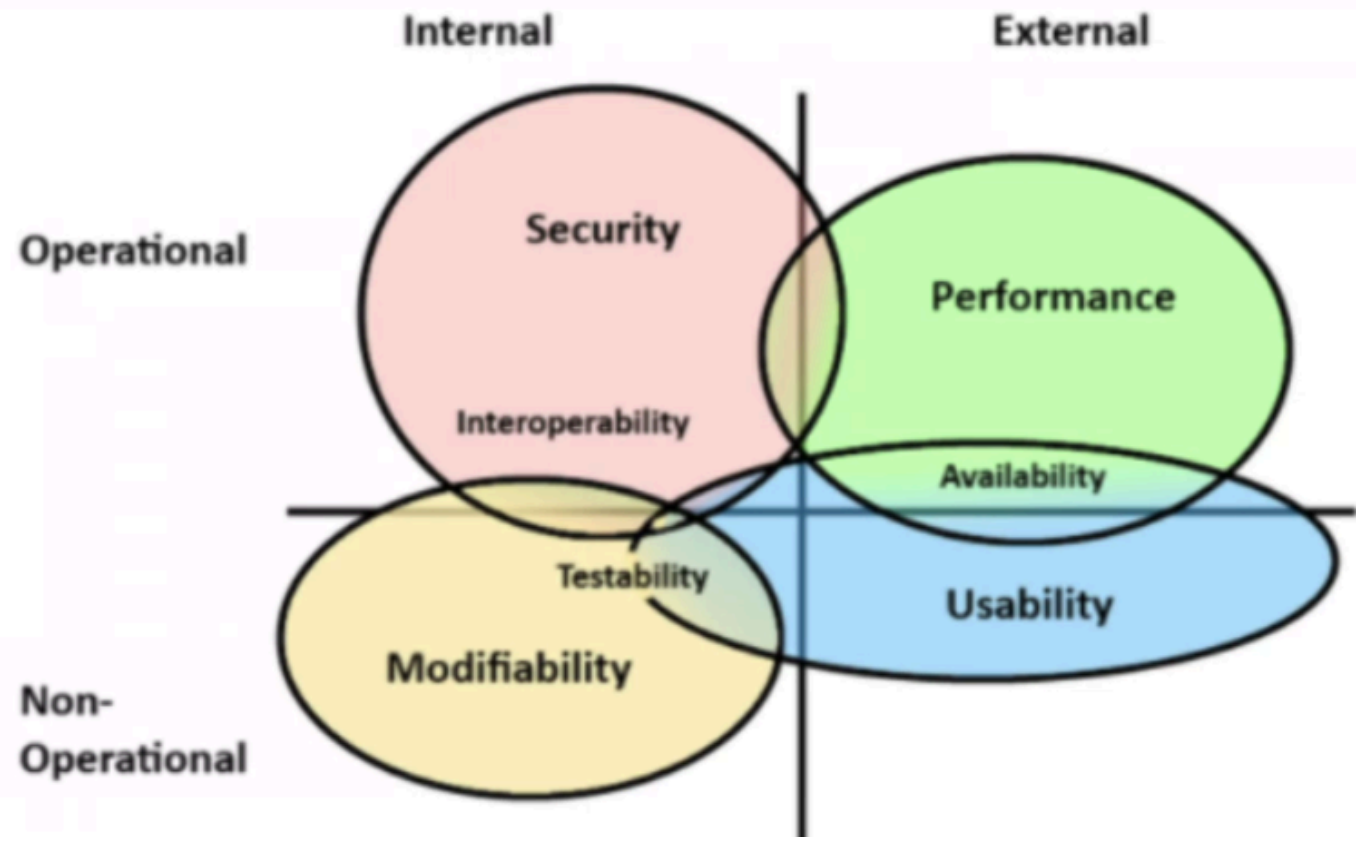
11. Atributos de calidad

Es una propiedad *testeable* de un sistema que es usada para*indicar cómo el sistema satisface las necesidades de sus stakeholders*

clasificación



clasificación: cómo se evaluan

Internos

- pueden ser evaluados *sin ejecutar el sistema*.
- Basado en análisis de código, la arquitectura o documentos del sistema

Externos

- solo se *evelúan ejecutando el sistema*
- punto de vista externo a la implementación: usuario

clasificación: qué comportamiento definen

Operacionales

- definen el comportamiento de uso diario, es decir, el *comportamiento durante su operación cotidiana*

No-Operacionales:

- define el comportamiento del sistema durante su mantenimiento y desarrollo.

Hay otra clasificaciones como los establecidos por normas como ISO/IEC FCD

Atributos de calidad y porqué tenerlos en cuenta

⚠ Porqué se rediseñan los sistemas?

no tiene relación directa con las definiciones funcionales, en realidad se rediseña porque se violan atributos de calidad (no funcionales):

- dificultades de mantenimiento
- falta de portabilidad

- problemas de escabilidad
- problemas de performance
- problemas de seguridad
- interfaces gráficas obsoletas

> *CÓMO EVITAR REDISEÑOS*: tenerlo en cuenta en la etapa de diseño y en la implementación de la entrega (a lo largo de todo el ciclo de vida del software).
>
> - No depende solo de la etapa de diseño (y/o arquitectura). pueden surgir en la implementación
> - No depende solo de la implementación o entrega (tambien en el diseño).

Atributos de calidad

buscamos evaluar cuantitativamente (tambien puede ser cualitativamente) múltipes atributos de calidad. Para ello:

1. *consensuamos una priorización de los atributos de calidad* con los interesados.
2. diseñar un sistema lo *suficientemente bueno para todos los interesados* (tanto para usuarios, como para developers, como deployers).

Algunos atributos de calidad son:

- disponibilidad
- reusabilidad
- performance
- robustez
- flexibilidad
- testeabilidad
- interoperatibilidad (compatibilidad)
- usabilidad (user error protection, facil de usar, rápido y cómodo para el usuario)
- mantenibilidad (modificabilidad)
- integridad
- confiabilidad:

    - tolerancia al fallo, disponibilidad, recuperabilidad, predecible, robusto y estable
    - se mide con tasa de fallos, tiempo medido entre fallos
- seguridad:
    - solo usuarios autorizados acceden al sistema, datos protegidos, resistente a ataques (confidencial, integridad)
    - no lo medimos con un número, pero podemos quedaros a cargo de tomar carta para prevenir la inseguridad: autenticación, autorización, cifrado, validación estricta de entrada)
- portabilidad

conflictos entre atributos

Algunos a.c. que pueden entrar en conflicto entre sí:

- *seguridad vs confiabilidad*: desde el punto de vista de la seguridad, ante un ataque aveces es mejor detener el sistema, respecto a la confiabilidad esto está mal. De igual forma, si una validación estricta deniega el acceso a un consilta legítima, se estará negando la availability
- *portabilidad vs performance*: respecto a performance, si sabes en qué ambiente se corre se pueden hacer varias optimizaciones, esto no es portable
- *seguridad vs usabilidad*: el exigir validaciones rigurosas hace que el acceso no sea tan fácil (lo que busca la usabilidad)
- *performance vs modificabilidad*: un sistema un poco más acomplado puede contemplar cierto hacks para ser más performante.

atributos en particular

disponibilidad-availability

Es la capacidad de un sistema o aplicación de estar disponible y operativo para los usuarios durante un período de tiempo

porqué es esencial:

porque se quiere cumplir con las expectativas de los usuarios y necesidades del negocio

entonces:
diseñamos y desarollamos aplicaciones teniendo en cuenta estrategias para mitigar fallos y garantizar continuidad del servicio

nos preguntamos:

> qué haría fallar el sistema?
> qué tan probable es que eso ocurra?
> cuánto tiempo lleva reparalo?

lo medimos:
en porcentajes que representen la proporción de tiempo en el que el ssitema está operativo en relación al tiempo total estimado.

factores que influencian:

- diseño robusto
- gestión de recursos
- mantenimiento y actualizaciones
- supervisión y detección de fallos
- recuperación ante desastres.

Ejemplos para *mejorar* la disponibilidad:

- balanceo de carga (distribuición)
- replica de datos
- respaldo y recuperación
- escalabilidad

Performance

son: los tiempos de respuesta de la aplicación en relación a las funcionalidad o actividades soportadas por la misma.
*métricas:

- latencia: tiempo dedicado solo a responder el evento.
- capacidad: número de eventos que se puede procesar/soportar en un tiempo determinado

Interoperatibilidad

capacidad para integrarse con con otros sistemas.
Mide la capacidad de intercambio de información con otros entornos de operación (del mismo sistema o de otros)
para mejorar la interoperatibilidad:

- usamos interfaces externas bien diseñadas
- nomas de intercambio (protocolos) y estándares
- Apis bien documentadas

Usabilidad

se comprueba a través de:

- *comprensibilidad*: refleja que tan fácil es para el susuario comprender cómo funciona el sistema (conocimientos previos que requiere el user para trabajar con el software)
- *fácil uso/eficiente*: permita realizar las operaciones de manera rápida y efectiva (no burocrático)
- *fácil de recordar, intuitiva*
- *interactiva, cómoda, atractiva*

Seguridad

mide la vulnerabilidad de las apps a ataques y las posibles defensas ante pérdidas o robp de info valiosa.

- capacidad de detección de ataques DoS y respuesta ante estos.
- *restricciones de acceso* de usuario de acuerdo a las políticas de autenticación y autorización.
- prevención de inyección de consultas SQL (se exige parseo de parámetros)
- encriptación de claves, y datos sencibles
- conexiones seguras (reliabilty)

escalabilidad

posibilidad de crecimiento sin perjudircar su funcionamiento operativo (el mismo trabajo pero con una escala más intensa, más grande o masiva).

cómo *mejorar la escalabilidad*:

- verticalmente: se agregan más recursos físicos a la infraestructura preexistente del sistema. por ejemplo: memoria, almacenamiento en disco, procesador, ancho de banda, etc.
- horizontalmente: incrementa el número de computadoras para dividir la carga de trabajo.

los *indicadores para medir la escalabilidad*:

- ¿el sistema permite escalamiento vertical o distribución en múltiples computadoras?
- cuánto demora en aumentar el escalamiento
- cuáles son las limitacones de crecimiento? (npumero más de servidores, memoria, discos,etc)
- evaluamos si el sistema puede responder ante un aumento en el volumen de transacciones o en la carga total de trabajo.