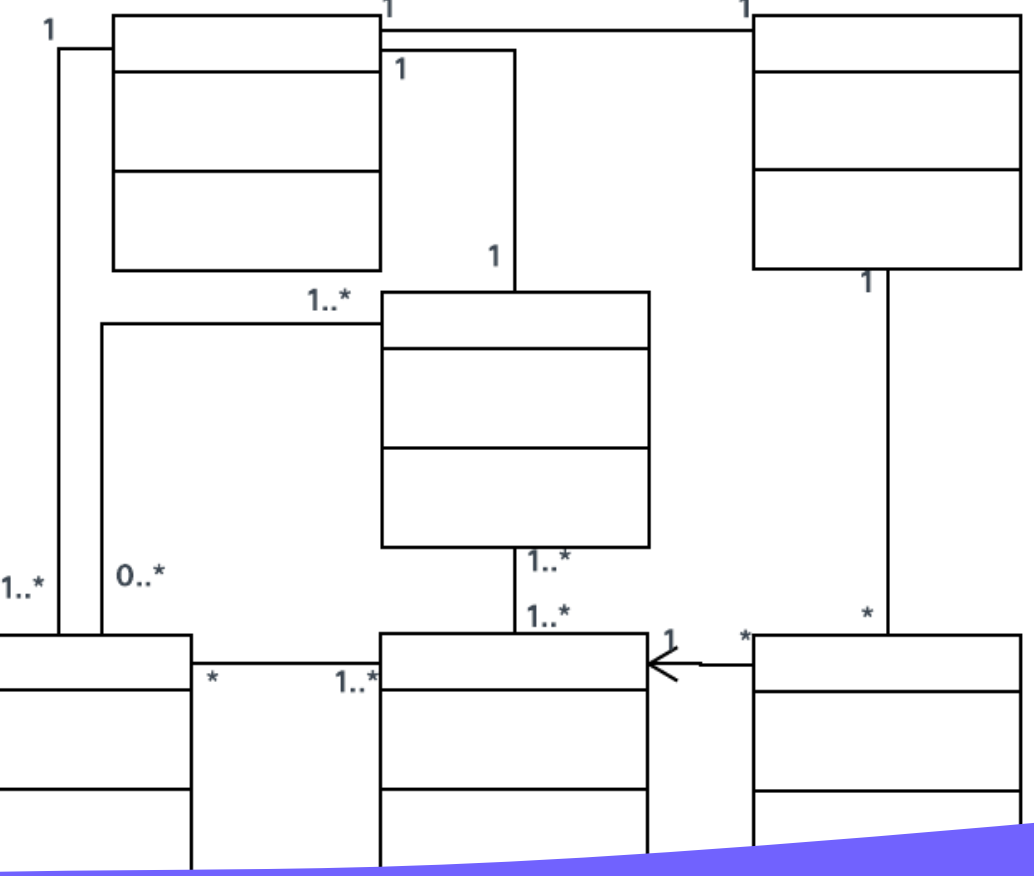
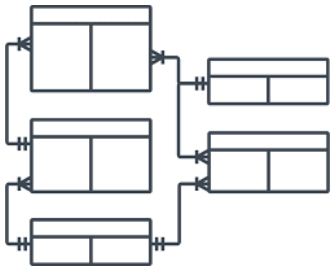




Diseño de Sistemas

UTN.BA
DPTO. INGENIERÍA EN SISTEMAS DE INFORMACIÓN
CÁTEDRA DISEÑO DE SISTEMAS

Agenda

- Cualidades de Diseño y/o Atributos de Calidad de Software
- Patrones de Diseño y sus Clasificaciones

Calidad de Software

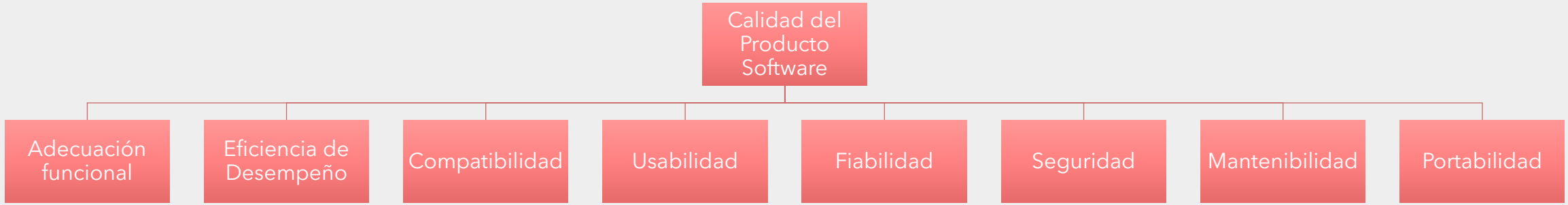
Para la norma ISO 25000:

"La calidad del producto software se puede interpretar como el grado en que dicho producto satisface los requisitos de sus usuarios aportando de esta manera un valor. Son precisamente estos requisitos (funcionalidad, rendimiento, seguridad, mantenibilidad, etc.) los que se encuentran representados en el modelo de calidad, el cual categoriza la calidad del producto en características y subcaracterísticas."

Calidad de Software – ISO 25000

El modelo de calidad del producto definido por la ISO 25010 se encuentra compuesto por las siguientes ocho características…

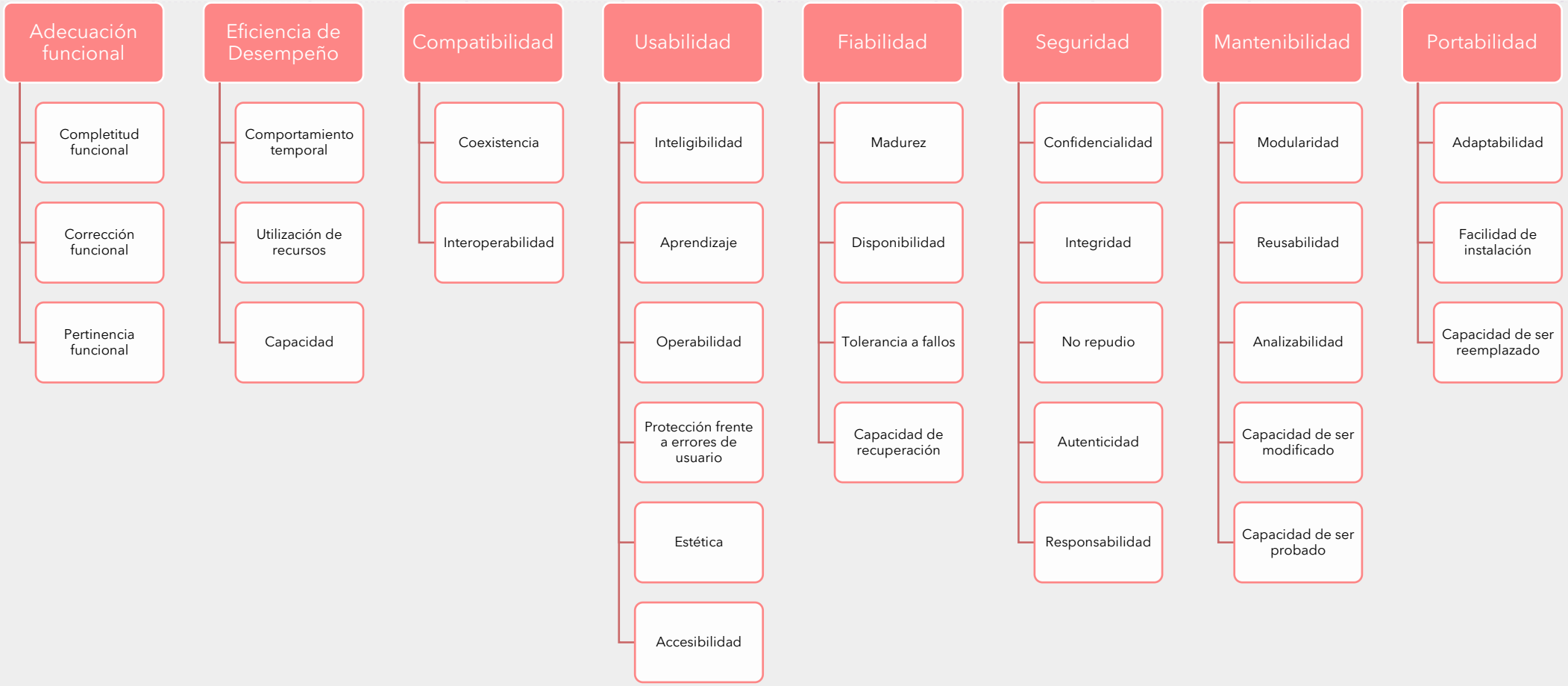
Calidad de Software – ISO 25000



Calidad de Software – ISO 25000

A su vez, estas grandes características se subdividen en varias subcaracterísticas…

Calidad de Software – ISO 25000



Adecuación funcional (Funcionalidad)

La capacidad del producto software para proveer las funciones que satisfacen las necesidades explícitas e implícitas cuando el software se utiliza bajo condiciones específicas.

Adecuación funcional (Funcionalidad)

Completitud funcional

- Grado en el cual el conjunto de funcionalidades cubre todas las tareas y los objetivos del usuario especificados.

Corrección funcional

- Capacidad del producto o sistema para proveer resultados correctos con el nivel de precisión requerido.

Pertinencia funcional

- Capacidad del producto software para proporcionar un conjunto apropiado de funciones para tareas y objetivos de usuario especificados.

Eficiencia de desempeño

Esta característica representa el desempeño relativo a la cantidad de recursos utilizados bajo determinadas condiciones.

Eficiencia de desempeño

Comportamiento temporal

- Los tiempos de respuesta y procesamiento y los ratios de *throughput* de un sistema cuando lleva a cabo sus funciones bajo condiciones determinadas en relación con un banco de pruebas (*benchmark*) establecido.

Utilización de recursos

- Las cantidades y tipos de recursos utilizados cuando el software lleva a cabo su función bajo condiciones determinadas.

Capacidad

- Grado en que los límites máximos de un parámetro de un producto o sistema software cumplen con los requisitos.

Compatibilidad

Capacidad de dos o más sistemas o componentes para intercambiar información y/o llevar a cabo sus funciones requeridas cuando comparten el mismo entorno hardware o software.

Compatibilidad

Coexistencia

- Capacidad del producto para coexistir con otro software independiente, en un entorno común, compartiendo recursos comunes sin detrimento.

Interoperabilidad

- Capacidad de dos o más sistemas o componentes para intercambiar información y utilizar la información intercambiada.

Usabilidad

La capacidad del producto software de ser entendido, aprendido, usado y atractivo al usuario, cuando es usado bajo las condiciones especificadas.

Usabilidad

Adecuación
- Capacidad del producto que permite al usuario entender si el software es adecuado para sus necesidades.

Capacidad de aprendizaje
- Capacidad del producto que permite al usuario aprender su aplicación.

Capacidad para ser usado
- Capacidad del producto que permite al usuario operarlo y controlarlo con facilidad.

Protección contra errores de usuario
- Capacidad del sistema para proteger a los usuarios de hacer errores.

Estética de la interfaz
- Capacidad de la interfaz de usuario de agradar y satisfacer la interacción con el usuario.

Accesibilidad
- Capacidad del producto que permite que sea utilizado por usuarios con determinadas características y discapacidades.

Fiabilidad

Capacidad de un sistema o componente para desempeñar las funciones especificadas, cuando se usa bajo unas condiciones y periodo de tiempo determinados

Fiabilidad

Madurez

- Es la capacidad del producto software para evitar fallas como resultado de errores en el software.

Disponibilidad

- Capacidad del sistema o componente de estar operativo y accesible para su uso cuando se requiere

Tolerancia a fallos

- Es la capacidad del producto software para mantener un nivel especificado de funcionamiento en caso de errores del software o de incumplimiento de su interfaz especificada.

Recuperabilidad

- Es la capacidad del producto software para reestablecer un nivel especificado de funcionamiento y recuperar los datos afectados directamente en el caso de una falla.

Seguridad

Capacidad de protección de la información y los datos de manera que personas o sistemas no autorizados no puedan leerlos o modificarlos.

Seguridad

Confidencialidad

- Capacidad de protección contra el acceso de datos e información no autorizados, ya sea accidental o deliberadamente.

Integridad

- Capacidad del sistema o componente para prevenir accesos o modificaciones no autorizados a datos o programas de ordenador.

No repudio

- Capacidad de demostrar las acciones o eventos que han tenido lugar, de manera que dichas acciones o eventos no puedan ser repudiados posteriormente.

Responsabilidad

- Capacidad de rastrear de forma inequívoca las acciones de una entidad.

Autenticidad

- Capacidad de demostrar la identidad de un sujeto o un recurso.

Mantenibilidad

Esta característica representa la capacidad del producto software para ser modificado efectiva y eficientemente, debido a necesidades evolutivas, correctivas o perfectivas.

Mantenibilidad

Modularidad

- Capacidad de un que permite que un cambio en un componente tenga un impacto mínimo en los demás.

Reusabilidad

- Capacidad de un activo que permite que sea utilizado en más de un sistema software o en la construcción de otros activos.

Analizabilidad

- Facilidad con la que se puede evaluar el impacto de un determinado cambio sobre el resto del software, diagnosticar las deficiencias o causas de fallos en el software, o identificar las partes a modificar.

Capacidad para ser modificado

- Capacidad del producto que permite que sea modificado de forma efectiva y eficiente sin introducir defectos o degradar el desempeño.

Capacidad para ser probado (“Testeabilidad”)

- Facilidad con la que se pueden establecer criterios de prueba para un sistema o componente y con la que se pueden llevar a cabo las pruebas para determinar si se cumplen dichos criterios.

Portabilidad

Capacidad del producto o componente de ser transferido de forma efectiva y eficiente de un entorno hardware, software, operacional o de utilización a otro.

Portabilidad

Adaptabilidad

- La capacidad del producto software para ser adaptado a diferentes entornos definidos sin aplicar acciones o medios diferentes de los previstos para el propósito del software considerado.

Capacidad para ser instalado

- Facilidad con la que el producto se puede instalar y/o desinstalar de forma exitosa en un determinado entorno.

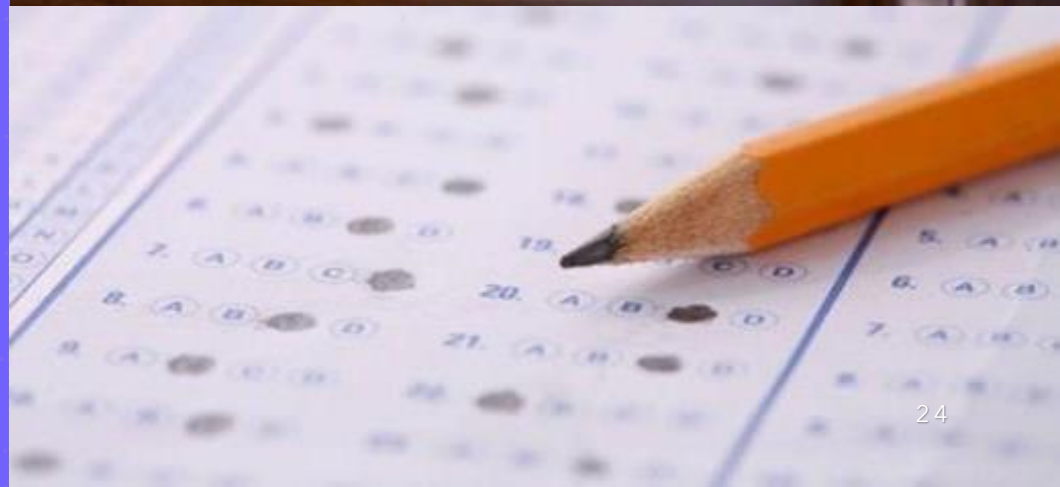
Capacidad para ser reemplazado

- Capacidad del producto para ser utilizado en lugar de otro producto software determinado con el mismo propósito y en el mismo entorno.

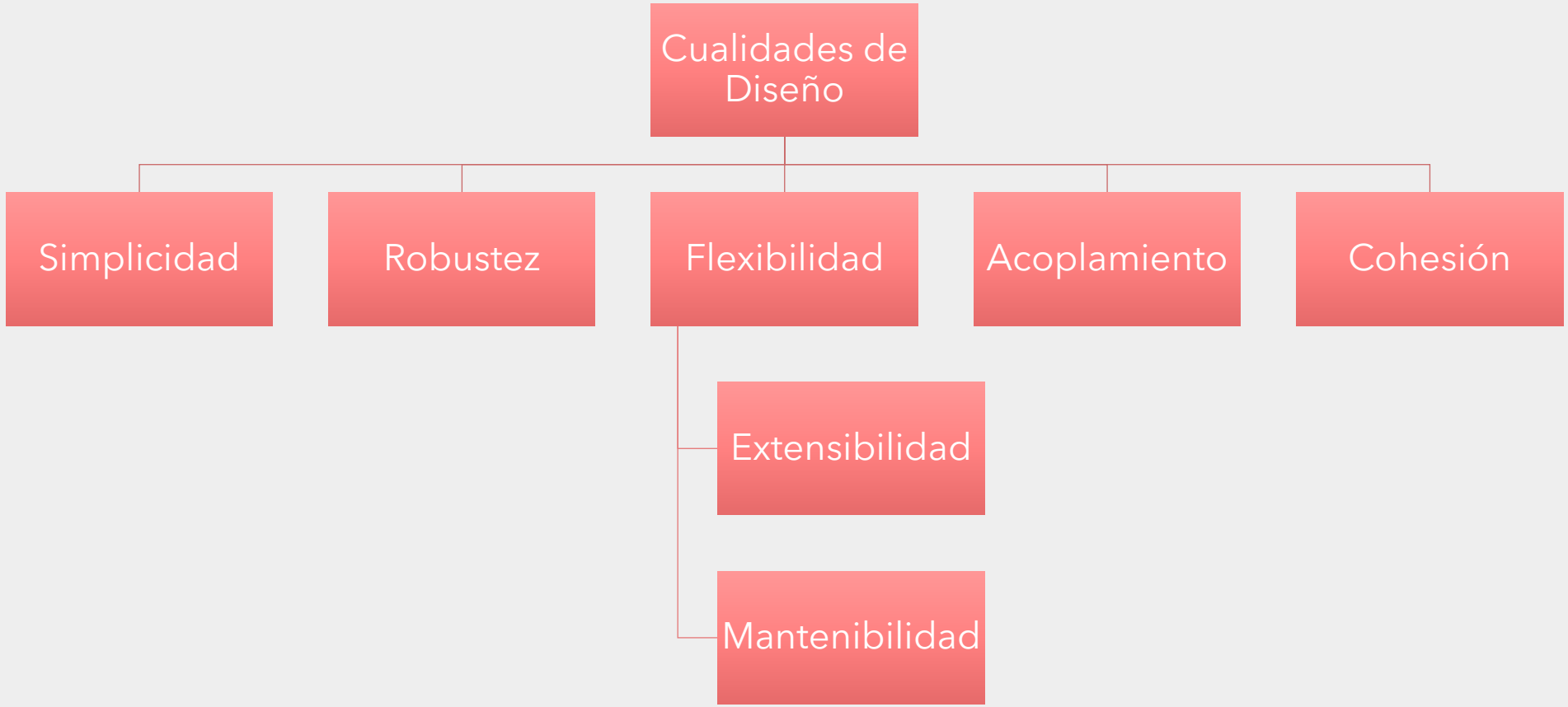
Cualidades de Diseño

- Las Cualidades de Diseño sirven como criterios para analizar distintas propuestas/alternativas y tomar decisiones más formadas respecto a ellas.

- No son los únicos criterios: experiencia y conocimiento de quien esté realizando el proceso de diseño, también serán elementos determinantes.



Cualidades de Diseño (Principales)



Acoplamiento

El acoplamiento se define como el grado de dependencia entre dos componentes, es decir, el grado de conocimiento que un componente tiene sobre el otro.

Cuanto mayor sea el acoplamiento entre dos componentes, los cambios o errores de uno de ellos impactarán en mayor medida sobre el otro componente.

Acoplamiento

Si minimizamos el acoplamiento podríamos:

- Mejorar la mantenibilidad
- Aumentar la reutilización
- Evitar que un defecto de un componente se propague a otros
- Evitar tener que inspeccionar y/o modificar múltiples componentes ante una modificación en uno solo de ellos

Cohesión

- *Un componente cohesivo tiende a tener todos sus elementos abocados a resolver el mismo problema.*
- *La cohesión se define como la cantidad de responsabilidades que están asignadas a un componente.*
- *Mientras más responsabilidades tenga un componente, menos cohesivo será.*

Simplicidad

KISS *– Keep it simple, stupid*

YAGNI *– You aren't gonna need it*

Simplicidad

KISS *– Keep it simple, stupid*

Nos propone evitar cualquier complejidad innecesaria.

Simplicidad

YAGNI *– You aren't gonna need it*

Nos propone no agregar funcionalidad nueva que no apunte a la problemática actual.

Robustez

Nos dice que ante un uso inadecuado por parte del usuario, sistemas externos o ante fallas internas:

- El sistema no debe generar información o comportamiento inconsistente/errático
- El sistema debe reportar los errores y volver a un estado consistente
- El sistema debe facilitar tanto como sea posible la detección de la causa del problema

Flexibilidad

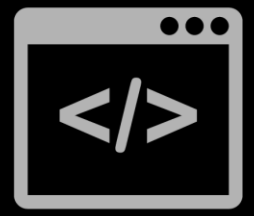
Es la capacidad de reflejar cambios en el dominio de manera simple y sencilla.

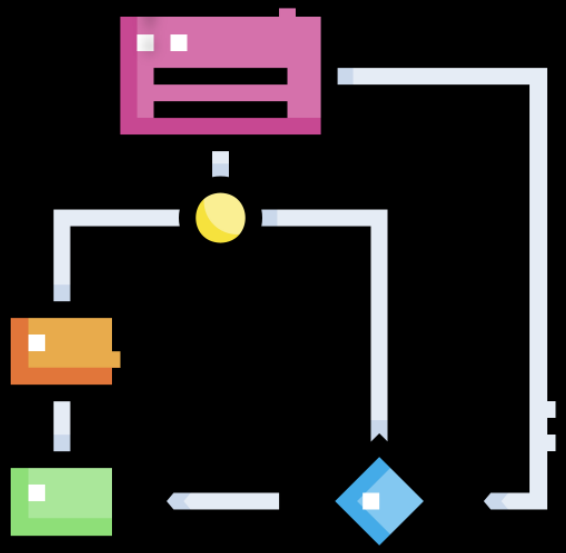
- *Extensibilidad*: capacidad de agregar nuevas características con “poco” impacto.
- *Mantenibilidad*: capacidad de modificar las características existentes con “el menor” esfuerzo posible.

Más sobre Cualidades de Diseño...

Link

Patrones de Diseño





Patrones de Diseño

"Son descripciones de clases y objetos relacionados que están adaptados para resolver un problema de diseño general en un contexto determinado"

Erich Gamma, Richard Helm, John Vlissides y Ralph Johnson

"Consiste en un diagrama de objetos que forma una solución a un problema conocido y frecuente"

Laurent Debrauwer - Patrones de diseño en Java

Patrones de Diseño

"Soluciones conocidas a problemas conocidos y reiterados en el mundo del Desarrollo de Software"

Patrones de Diseño

El objetivo es reutilizar la experiencia de quienes ya se han encontrado con problemas similares y han encontrado una buena solución.

Patrones de Diseño - Estructura

- Propósito
- Motivación
- Participantes
- Colaboraciones
- Consecuencias
- Implementación Código de ejemplo
- Usos conocidos
- Patrones relacionados

Patrones de Diseño – Partes esenciales

- ***Nombre***: Comunica el objetivo del patrón en una o dos palabras. Aumenta el vocabulario sobre diseño.
- ***Problema***: Describe el problema que el patrón soluciona y su contexto. Indica cuándo se aplica el patrón.
- ***Solución***: Indica cómo resolver el problema en términos de elementos, relaciones, responsabilidades y colaboraciones. La solución debe ser lo suficientemente abstracta para poder ser aplicada en diferentes situaciones.
- ***Consecuencias***: Indica los efectos de aplicar la solución. Son críticas al momento de evaluar distintas alternativas de diseño.

Patrones de Diseño – Clasificación

Creacionales	Comportamiento	Estructurales
• Abstraen el proceso de creación/instanciación de los objetos. Se los suele utilizar cuando debemos crear objetos, complejos o no, tomando decisiones dinámicas en momento de ejecución.	• Resuelven cuestiones, complejas o no, de interacción entre objetos en momento de ejecución.	• Resuelven cuestiones, generalmente complejas, de generación y/o utilización de estructuras complejas o que no están acopladas al dominio.

Patrones de Diseño

Creacionales	Comportamiento	Estructurales
• Factory Method • Simple Factory • Singleton • Abstract Factory • Builder • Prototype	• State • Strategy • Observer • Command • Template method • Iterator • Memento • Visitor • Interpreter • Chain of Responsability • Mediator	• Adapter • Composite • Facade • Decorator • Proxy • Flyweight • Bridge

Gracias