

Patrones de diseño

- ➔ "Cada **patrón** describe un **problema** que ocurre una y otra vez en nuestro entorno, así como la **solución** a ese problema; de tal modo que se pueda aplicar esta solución un millón de veces, sin hacer lo **mismo** dos veces". [*Christopher Alexander*].
- ➔ Para que un **software** pueda **evolucionar** tiene que ser **reutilizable**; pues el software cambia.

Patrones de diseño

- ➔ Los futuros **cambios** en los **requisitos** exigen un diseño que considere **aspectos** que pueden cambiar.
- ➔ **”Diseñar software** orientado a objetos es difícil, y diseñar software orientado a objetos **reutilizable** es todavía más difícil” *Design Patterns, The Gang of Four [Erich Gamma]*.

Patrones de diseño

- ➔ Un **patrón** de **diseño** es una descripción de **clases** y **objetos** comunicándose entre sí, adaptada para resolver un **problema** de diseño **general** en un contexto **particular** [*Erich Gamma*].
- ➔ Un **patrón** de **diseño** consiste en un **diagrama** de **objetos** que forma una **solución** a un **problema conocido** y **frecuente**. Este diagrama está constituido por un conjunto de objetos descritos por **clases** y las **relaciones** que enlazan los objetos. [*Laurent DEBRAUWER*]

Patrones de diseño

- ➔ Los **patrones** de diseño responden a problemas de **diseño** de **aplicaciones** en el marco de la POO.
- ➔ Se trata de soluciones **conocidas** y **probadas** cuyo **diseño** proviene de la **experiencia** de los programadores.
- ➔ Los **patrones** de diseño están basado en las **buenas** prácticas de la programación orientada a objetos.

[Laurent DEBRAUWER].

Patrones de diseño

- ➔ Los patrones de diseño ayudan a **identificar abstracciones** menos obvias y los objetos que las expresan.
- ➔ Por ejemplo, los objetos que representan un **proceso** o **algoritmo** no tienen lugar en la **naturaleza**, y sin embargo son una parte crucial de los diseños **flexibles**.

[Laurent DEBRAUWER].

Patrones de diseño

- ➔ Un **patrón** es una **solución** a un **problema** en un contexto **particular**.
- ➔ Es **recurrente**; por lo tanto, **relevante** a otras situaciones.
- ➔ Permite entender cómo **adaptarlo** a la variante **particular** del problema donde se quiere aplicar.
- ➔ Los patrones de diseño están orientados al **cambio**.

[Juan Pavón Mestras].

Patrones de diseño

- ➔ Capturan la **experiencia** y la hacen accesible a los no expertos.
- ➔ El conjunto de sus **nombres** forma un **vocabulario** que ayuda a que los desarrolladores se **comuniquen** mejor.
- ➔ Permiten comprender un sistema más **rápidamente** cuando está **documentado** con los patrones que usa.

[Juan Pavón Mestras].

Patrones de diseño

- ➔ Son **descripciones** de **clases** y **objetos** relacionados que están **particularizados** para resolver un problema de **diseño general** en un determinado contexto.
- ➔ Un **patrón** de diseño nomina, abstrae e identifica los aspectos clave de una **estructura** de diseño **común**, haciéndolos útiles para crear un diseño O.O. **reutilizable**.
- ➔ Un **patrón** de **diseño** identifica las **clases** e instancias participantes, sus **roles** y **colaboraciones**, y la distribución de **responsabilidades**.

[Laurent DEBRAUWER].

Patrones de diseño

- ➔ Cada **patrón** de **diseño** se centra en un problema **concreto**, describe cuándo **aplicarlo** y si tiene sentido hacerlo según otras **restricciones** de diseño; así como las consecuencias y las **ventajas** e **inconvenientes** de su uso.
- ➔ Los **patrones** de diseño ayudan a definir **interfaces** identificando sus elementos clave y los **tipos** de datos que se envían a la **interfaz**.
- ➔ Un **patrón** de diseño también puede decir **qué no** debemos poner en la **interfaz**.

[Erich Gamma].

Patrones de diseño

- ➔ Los patrones de diseño también especifican **relaciones** entre **interfaces**; en concreto, muchas veces requieren que algunas clases tengan interfaces **parecidas**, o imponen **restricciones** a las **interfaces** de algunas clases.
- ➔ Al conjunto de todas las **signaturas** (**prototipo**) definidas por las operaciones (**métodos**) de un **objeto** se le denomina la **interfaz** del objeto.
- ➔ Dicha **interfaz** caracteriza al conjunto completo de **peticiones** que se pueden **enviar** al **objeto**.

[Erich Gamma].

Patrones de diseño

- ➔ Buscan reducir las dependencias de implementación entre subsistemas, que lleva a aplicar el siguiente principio de reutilización: “**Programa** para una **interfaz**, no para una **implementación**”
- ➔ Es decir, no se deben declarar las variables como **instancias** de clases **concretas**.

[Erich Gamma].

Patrones de diseño

- ➔ Por lo tanto, se ajustarán simplemente a la **interfaz** definida por una clase **abstracta** o por una **interface** propiamente dicha.
- ➔ Es decir, toda instancia debería ser de tipo: clase **abstracta**, o de tipo **interface**.
- ➔ Sin embargo, en alguna parte del código estas instancias **siempre** se inicializan con clases **concretas**.

[Erich Gamma].

Patrones de diseño

- ➔ Los patrones de diseño hacen que sea más fácil **reutilizar** buenos **diseños** y **arquitecturas**.
- ➔ Los patrones de diseño expresan técnicas que ya han sido **probadas** y son más **accesibles** para desarrolladores de **nuevos** sistemas.
- ➔ Los patrones de diseño ayudan a elegir las alternativas de diseño que hacen que un sistema sea **reutilizable**, evitando las que **dificultan** dicha reutilización [*Erich Gamma*].

Patrones de diseño

- ➔ Los patrones de diseño mejoran la **documentación** y el **mantenimiento** de los sistemas existentes al proporcionar una **especificación** explícita de las **interacciones** entre clases y objetos y de cuál es su intención.
- ➔ Los patrones de diseño ayudan a un **diseñador** a lograr un **buen** diseño más **rápidamente**.

[Erich Gamma].

Patrones de diseño

Un patrón tiene **cuatro** elementos esenciales:

1. El **nombre** del patrón que permite describir, en una o dos palabras, un **problema** de diseño.
2. El **problema** que describe **cuándo** aplicar el patrón, explicando el problema y su **contexto** y opcionalmente incluye una serie de **condiciones** que deben darse para que tenga sentido **aplicar** el patrón. [*Erich Gamma*].

Patrones de diseño

3. La **solución** que describe los **elementos** que constituyen el **diseño**, sus **relaciones**, **responsabilidades** y **colaboraciones**; pero no describe un diseño o una implementación en **concreto**, si no que es como una **plantilla** que puede **aplicarse** en muchas situaciones **diferentes**. Por ello proporciona una descripción **abstracta** de un **problema** de diseño y cómo lo **resuelve** una disposición general de elementos (**clases** y **objetos**).

[Erich Gamma].

Patrones de diseño

4. Las **consecuencias** que son los **resultados**, así como las **ventajas** e **inconvenientes** de aplicar el patrón. Aunque cuando se **describen** decisiones de diseño muchas veces no se **reflejan** sus consecuencias, éstas son fundamentales para **evaluar** las **alternativas** de diseño y comprender los **costes** y **beneficios** de aplicar el patrón.

[Erich Gamma].

Patrones de diseño

Las **consecuencias** del uso de un patrón de diseño en el software suelen referirse al **equilibrio** entre **espacio** y **tiempo**; pero pueden tratar cuestiones de **lenguaje** e **implementación**; por lo tanto, las consecuencias de un patrón incluyen su impacto sobre la **flexibilidad**, **extensibilidad** y **potabilidad** de un sistema.

[Erich Gamma].

Patrones de diseño

- ➔ Incluir estas **consecuencias** de un modo explícito nos ayudará a **comprenderlas y evaluarlas**.
- ➔ Como tendremos que **implementar** nuestros diseños, un patrón también proporciona **código** de ejemplo.

[Erich Gamma].

Patrones de diseño

Los patrones de diseño se clasifican en tres:

1. Patrones de **construcción** tienen como objetivo organizar la **creación** de objetos.
2. Patrones de **estructuración** que facilitan la **organización** de la **jerarquía** de **clases** y de sus **relaciones**.
3. Patrones de **comportamiento** que proporcionan soluciones para organizar las **interacciones** y repartir el procesamiento entre los objetos. [*Laurent Debrauwer*].