Patrón – Concepto

“Un patrón describe un problema el cual ocurre una y otra vez en nuestro ambiente, y además describen el núcleo de la solución a tal problema, en tal una manera que puedes usar esta solución millones de veces, sin hacer lo mismo dos veces.” [Alexander et al.]
Aunque Alexander se refería a patrones en ciudades y edificios, lo que dice también es válido para patrones de diseño orientados a objetos.

Patrones de Diseño – Concepto

Los patrones de diseño tratan los problemas del diseño de software que se repiten y que se presentan en situaciones particulares, con el fin de proponer soluciones a ellas. Son soluciones exitosas a problemas comunes.
Estas soluciones han ido evolucionando a través del tiempo. Existen muchas formas de implementar patrones de diseño. Los detalles de las implementaciones se llaman estrategias.
En resumen, son soluciones simples y elegantes a problemas específicos del diseño de software orientado a objetos.

Historia

En 1994 se publicó el libro "Design Patterns: Elements of Reusable Object Oriented Software" escrito por Erich Gamma, Richard Helm, Ralph Johnson y John Vlissides. Ellos recopilaron y documentaron 23 patrones de diseño aplicados usualmente por expertos diseñadores de software orientado a objetos. Ellos no fueron los únicos que inventaron los patrones, pero la idea de patrones de diseño comenzó a tomar fuerza luego de la publicación de dicho libro.

Cuándo utilizarlos

Como analistas y programadores vamos desarrollando a diario nuestras habilidades para resolver problemas usuales que se presentan en el desarrollo del software. Por cada problema que se nos presenta pensamos distintas formas de resolverlo, incluyendo soluciones exitosas que ya hemos usado anteriormente en problemas similares.
Antes de comenzar nuestro diseño, deberíamos realizar un estudio meticuloso del problema en el que nos encontramos y explorarlo en busca de patrones que hayan sido utilizados previamente con éxito. Estos patrones nos ayudarán a que nuestro proyecto evolucione mucho más rápidamente.

Cuándo no utilizarlos

Si hay una sensación que describa lo que puede sentir un desarrollador o diseñador después de conocer los patrones de diseño esa podría ser la euforia. Esta euforia viene producida por haber encontrado un mecanismo que convierte lo que era una labor artesana y tediosa, en un proceso sólido y basado en estándares,

mundialmente conocido y con probado éxito.
Después de recién iniciarse en los patrones, probablemente el paso siguiente que tomará será emprender el rediseño de algunos proyectos aún vigentes y que intente aplicar todas las maravillosas técnicas que ha aprendido para que así estos proyectos se aprovechen de todos los beneficios inherentes al uso de patrones de diseño.
Esto termina en intentar que se aplique estos patrones en toda situación donde sea posible, aún en aquellas donde no deba aplicarse.

Dónde utilizarlos

No es obligatorio utilizar los patrones siempre, sólo en el caso de tener el mismo problema o similar que soluciona el patrón, siempre teniendo en cuenta que en un caso particular puede no ser aplicable. Abusar o forzar el uso de los patrones es un error muy común.
Antes de abordar un proyecto con patrones se debe analizar minuciosamente qué patrones nos pueden ser útiles, cuáles son las relaciones entre los diferentes componentes de nuestro sistema, cómo podemos relacionar los patrones entre sí de modo que formen una estructura sólida, cuáles son los patrones que refleja nuestro dominio, etc. Esta tarea obviamente es mucho más compleja que el mero hecho de ponerse a codificar patrones "porque sí". Lo ideal es que los patrones se vean plasmados en el lenguaje de modelado UML, que veremos en la próxima sección.

Qué es GOF

Los ahora famosos Gang of Four (GoF, que en español es la pandilla de los cuatro) formada por Erich Gamma, Richard Helm, Ralph Johnson y John Vlissides. Son los autores del famoso libro "Design Patterns: Elements of Reusable Object Oriented Software".
Aquí una breve reseña de cada uno de ellos:
Erich Gamma, informático suizo, es actualmente el director del centro tecnológico OTI en Zúrich y lidera el desarrollo de la plataforma Eclipse. Fue el creador de JUnit, junto a Kent Beck.
Richard Helm trabaja actualmente con The Boston Consulting Group, donde se desempeña como consultor de estrategias de IT aplicadas al mundo de los negocios.
Su carrera abarca la investigación de distintas tecnologías, desarrollo de productos, integración de sistemas y consultoría de IT. Antes de incorporarse a BCG, Richard trabajó en IBM: comenzó su carrera como científico investigador en "IBM Thomas J. Watson Research Center" en Nueva York. Es una autoridad internacional en arquitectura y diseño de software.
Ralph E. Johnson es profesor asociado en la Universidad de Illinois, en donde tiene a su cargo el Departamento de Ciencias de la Computación. Co-autor del famoso libro de Gof y pionero de la comunidad de Smalltalk, lidera el UIUC patterns/Software Architecture Group.

John Vlissides (1961-2005) era ingeniero eléctrico y se desempeñaba como consultor de la Universidad de Standford. Autor de muchos libros, desde 1991 trabajó como investigador en el "IBM T.J. Watson Research Center".

Beneficios

Proporcionan elementos reusables en el diseño de sistemas software, lo que significa que es aplicable a diferentes problemas de diseño en distintas circunstancias.
Efectividad comprobada en la resolución de problemas similares en ocasiones anteriores.
Formalizan un vocabulario común entre diseñadores.
Estandarizan el diseño, lo que beneficia notablemente a los desarrolladores.
Facilitan el aprendizaje de las nuevas generaciones de diseñadores y desarrolladores utilizando conocimiento ya existente.

Tipos de patrones

- Creacionales
- Comportamiento
- Estructurales

Clasificación según su propósito

Creacionales

Definen la mejor manera en que un objeto es instanciado. El objetivo de estos patrones es de abstraer el proceso de instanciación y ocultar los detalles de cómo los objetos son creados o inicializados.

Comportamiento

Permiten definir la comunicación entre los objetos del sistema y el flujo de la información entre los mismos.

Estructurales

Permiten crear grupos de objetos para ayudarnos a realizar tareas complejas.

Clasificación según alcance

De clase

Se basados en la herencia de clases.

De objeto

Se basan en la utilización dinámica de objetos.