Patrones de Diseño

Axel Rodríguez Espinoa
Nicolas Oyarzún Hernandez
Pedro Salas Vergara

11 de diciembre de 2013

PATRONES DE DISEÑO DE SOFTWARE

Introducción

El diseño es un modelo del sistema, realizado con una serie de principios y técnicas, que permite describir el sistema con el suficiente detalle como para ser implementado. Pero los principios y reglas no son suficientes, en el contexto de diseño podemos observar que los buenos ingenieros tienen esquemas y estructuras de solución que usan numerosas veces en función del contexto del problema. Este es el sentido cabal de la expresión "tener un mente bien amueblada", y no el significado de tener una notable inteligencia. Estos esquemas y estructuras son conceptos reusables y nos permiten no reinventar la rueda. Un buen ingeniero reutiliza un esquema de solución ante problemas similares.

Antecedentes

El concepto de "patrón de diseño"que tenemos en Ingeniería del Software se ha tomado prestado de la arquitectura. En 1977 se publica el libro .ª Pattern Language: Towns/Building/Construction", de Christopher Alexander, Sara Ishikawa, Murray Silverstein, Max Jacobson, Ingrid Fiksdahl-King y Shlomo Angel, Oxford University Press. Contiene numerosos patrones con una notación específica de Alexander.

Alexander comenta que "Cada patrón describe un problema que ocurre una y otra vez en nuestro entorno, para describir después el núcleo de la solución a ese problema, de tal manera que esa solución pueda ser usada más de un millón de veces sin hacerlo siquiera dos veces de la misma forma". El patrón es un esquema de solución que se aplica a un tipo de problema, esta aplicación del patrón no es mecánica, sino que requiere de adaptación y matices. Por ello, dice Alexander que los numerosos usos de un patrón no se repiten dos veces de la misma forma.

La idea de patrones de diseño estaba .en el aire", la prueba es que numerosos diseñadores se dirigieron a aplicar las ideas de Alexander a su contexto. El catálogo más famoso de patrones se encuentra en "Design Patterns: Elements of Reusable Object-Oriented Software", de Erich Gamma, Richard Helm, Ralph Johnson y John Vlissides, 1995, Addison-Wesley, también conocido como el libro GOF (Gang-Of-Four).

Siguiendo el libro de GOF los patrones se clasifican según el proposito para el que han sido definidos:

- **Creacionales**: solucionan problemas de creación de instancias. Nos ayudan a encapsular y abstraer dicha creación.

- **Estructurales**: solucionan problemas de composición (agregación) de clases y objetos.

- **De Comportamiento**: soluciones respecto a la interacción y responsabilidades entre clases y objetos, así como los algoritmos que encapsulan.

Concepto de patron de diseño

-Una arquitectura orientada a objetos bien estructurada está llena de patrones. La calidad de un sistema orientado a objetos se mide por la atención que los diseñadores han prestado a las colaboraciones entre sus objetos. Los patrones conducen a arquitecturas más pequeñas, más simples y más comprensibles.- (Grady Booch).
Los patrones de diseño son descripciones de clases cuyas instancias colaboran entre sí. Cada patrón es adecuado para ser adaptado a un cierto tipo de problema. Para describir un caso debemos especificar:

- Nombre
- Propósito o finalidad
- Sinónimos (otros nombres por los que puede ser conocido)
- Problema al que es aplicable
- Estructura (diagrama de clases)
- Participantes (responsabilidad de cada clase)
- Colaboraciones (diagrama de interacciones)
- Colaboraciones (diagrama de interacciones)
- Otros patrones con los que está relacionado

Ventajas de los patrones de diseño

Clasificación de patrones

EL patron de diseño que vamos a ejemplificar

- Modelo
- Vista
- Controlador

Figura 1: MVC aplicado en un software

La capa de modelo mapea los datos de la aplicación; en está es en la que almacenaremos datos para buscar/inserción/eliminación y/o modificación. La capa del controlador es muy importante para este patrón de diseño, ya que es la que maneja la lógica con la que se manipulan los datos (Es decir se relaciona íntimamente con la capa modelo) y a su vez interactúa con la capa vista, mostrando al cliente la representación de los datos según sus peticiones.

EJEMPLO DE MVC

Problema: Realizar la venta de un producto en una tienda.
Aplicación del patrón MVC al problema de la tienda:
En este problema podemos mapear las 3 capas de MVC a la tienda, teniendo lo siguiente:

- Modelo: Libro de ventas, productos de la tienda, etc.
- Vista: Estantes con productos, precio de los productos.
- Controlador: Vendedor realizando ventas, vendedor realizando boletas.

El cliente solicitará al vendedor que le venda un producto (Una frucola grande), a lo que el vendedor le dirá que esta tiene un precio de $350 y le pedirá el dinero especificado al cliente.
El cliente pagará su producto y el vendedor se lo entregará al cliente con su boleta correspondiente.
Cada petición que hace el cliente al vendedor es una interacción con un controlador, esté a su vez entrega información sobre productos y solicita información al cliente mediante vistas.
Se producirán cambios al modelo (ej: Cuando el vendedor le entrega un producto o la boleta al cliente) y estos son un efecto colateral a las peticiones realizadas al controlador.