# Patrones de diseño en la Ingeniería de Software

Jonathan León, Francisca Sapiains, Cristian Mondaca

Santiago,14 de Mayo 2014

## 1. Introducción.

Los patrones de diseño son un conjunto de prácticas de óptimo diseño que se utilizan para abordar problemas recurrentes en la programación orientada a objetos.

Christopher Alexander, proponía el aprendizaje y uso de una serie de patrones para la construcción de edificios de mayor calidad, donde la mayor calidad se refería a la arquitectura antigua y la menor calidad a la arquitectura moderna, el romper con la arquitectura antigua había perdido esa conexión con lo que las personas consideraban que era calidad.

"Cada patrón describe un problema que ocurre infinidad de veces en nuestro entorno, así como la solución al mismo, de tal modo que podemos utilizar esta solución un millón de veces más adelante sin tener que volver a pensarla otra vez."

Los patrones son un intento de formalizar y plasmar de una forma práctica generaciones de conocimiento arquitectónico. Un patrón define una posible solución correcta para un problema de diseño dentro de un contexto dado, describiendo las cualidades invariantes de todas las soluciones.

En 1987, Ward Cunningham y Kent Beck, sobrepasados por el pobre entrenamiento que recibían los nuevos programadores en orientación a objetos, se preguntaban cómo se podían capturar las buenas ideas para luego de alguna manera traspasarlas a los nuevos programadores recién instruidos en herencia y polimorfismo. Leyendo a Alexander se dieron cuenta del paralelo que existía entre la buena arquitectura propuesta por Alexander y la buena arquitectura OO, de modo que usaron varias ideas de Alexander para desarrollar cinco patrones de interacción hombre-ordenador (HCI) y publicaron un artículo en OOPSLA-87 titulado "'Using Pattern Languages for OO Programs"'.

A principios de la decada de los 90 el concepto de patrones de diseño fue el resultado de un trabajo realizado por un grupo de 4 personas (Erich Gamma, Richard Helm, Ralph Johnson y John Vlissides, conocidos como "la pandilla de los cuatro") que se publicó en 1995 en un libro titulado "Patrones de diseño: Elementos de software orientado a objetos reutilizables" en el que se esbozaban 23 patrones de diseño.

## 2. Patrones de diseño en la ingeniería de software

El diseño de Software juega un papel importante en el desarrollo de software lo cual permite al ingeniero producir varios modelos del sistema que se va a construir, del mismo modo que forman una especie de plan de la solución de la aplicación. Estos modelos puede evaluarse en relacion con su calidad y mejorarse antes de generar código, de realizar pruebas y de que los usuarios finales se vean involucrados a gran escala. El diseño es el sitio en el que se establece la calidad del software. En el desarrollo de software existen problemas que son recurrentes entre distintos proyectos y cada vez que dichos problemas aparecen podemos utilizar una solución ya probada que sabremos funcionará, es por eso que aparecieron los patrones de diseño. Los patrones de diseño son una solución a dichos problemas de diseño. Cada patrón describe un problema que ocurre una infinidad de veces en nuestro entorno, así como la solución al mismo, de tal modo que podemos utilizar esta solución un millón de veces mas adelante sin tener que volver a pensarla otra vez. Para que una solución sea considerada

un patrón debe poseer ciertas características. Una de ellas es que debe haber comprobado su efectividad resolviendo problemas similares en ocasiones anteriores. Otra es que debe ser re utilizable, lo que significa que es aplicable a diferentes problemas de diseño en distintas circunstancias.

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, asi como los algoritmos que encapsulan.

Una arquitectura orientada a objetos bien estructurada esta 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.

La clave para la reutilización es anticiparse a los nuevos requisitos y cambios, de modo que los sistemas evolucionen de forma adecuada. Cada patrón permite que algunos aspectos de la estructura del sistema puedan cambiar independientemente de otros aspectos. Facilitan la reusabilidad, extensibilidad y mantenimiento. En resumen, un patrón es el denominador común, una estructura que tienen aplicaciones semejantes. Esto también ocurre en otros ordenes de la vida. Por ejemplo, en nuestra vida cotidiana aplicamos a menudo el esquema saludopresentación -mensaje-despedida en ocasiones diversas, que van desde un intento de conocer a una persona nueva hasta dar una conferencia.

## 2.1. Modelo Vista Controlador

El modelo vista controlador es un patrón de diseño realizado generalmente para el diseño de aplicaciones con sofisticados interfaces. Se trata de realizar un diseño que desacople la vista del modelo, con la finalidad de mejorar la reusabilidad. De esta forma las modificaciones en las vistas impactan en menor medida en la lógica de negocio o de datos. Si realizamos un diseño ofuscado, es decir, un software que mezcle los componentes de interfaz y de negocio, entonces la consecuencia será que, cuando necesitemos cambiar el interfaz, tendremos que modificar trabajosamente los componentes de negocio, lo cual implica mayor esfuerzo y más riesgo de error. El modelo vista controlador se compone de los siguientes elementos:

- Modelo: datos y reglas de negocio.
- Vista: muestra la información del modelo al usuario.
- Controlador: gestiona las entradas del usuario.

Un modelo puede tener diversas vistas, cada una con su correspondiente controlador. Un ejemplo clásico es el de la información de una base de datos, que se puede presentar de diversas formas: diagrama de tarta, de barras, tabular, etc. Veamos cada componente.

El modelo es responsable de:

- Acceder a la capa de almacenamiento de datos. Lo ideal es que el modelo sea independiente del sistema de almacenamiento.
- Define las reglas de negocio (la funcionalidad del sistema). Un ejemplo de regla puede ser: "Si la mercancía pedida no está en el almacén, consultar el tiempo de entrega estándar del proveedor".
- Lleva un registro de las vistas y controladores del sistema.

- Si estamos ante un modelo activo, notificará a las vistas los cambios que en los datos pueda producir un agente externo (por ejemplo, un fichero bath que actualiza los datos, un temporizador que desencadena una inserción, etc).

El controlador es el responsable de:

- Recibe los eventos de entrada (un clic, un cambio en un campo de texto, etc.).
- Contiene reglas de gestión de eventos, del tipo "SI Evento Z, entonces Acción W". Estas acciones pueden suponer peticiones al modelo o a las vistas. Una de estas peticiones a las vistas puede ser una llamada al método Actualizar(). Una petición al modelo puede ser.

Las vistas son responsables de:

- Recibir datos del modelo y los muestra al usuario.
- Tienen un registro de su controlador asociado (normalmente porque además lo instancia).
- Pueden dar el servicio de Actualización, para que sea invocado por el controlador o por el modelo (cuando es un modelo activo que informa de los cambios en los datos producidos por otros agentes).

Un ejemplo que podemos dar de este modelo es por ejemplo al crearnos una cuenta en algún foro. Los foros son el espacio donde las personas pueden dar su opinión en distintos temas. Al pinchar el link de Crear nuevo usuario"(o similar) el controlador toma esa instrucción y nos muestra la vista (una pagina) en donde ingresamos nuestros datos requeridos por un formulario. Al llenar los datos el usuario comúnmente se da click en un botón de ENVIAR (o similar) al hacer esto el controlador toma los datos y los manda a un método especifico del modelo el cual recibe los datos y realiza las consultas necesarias para agregar estos datos a la base de datos. El usuario no verá obviamente todo esto, el usuario solo verá que sus datos fueron ingresados y un mensaje de bienvenida al foro, pero por detrás se está realizando las acciones antes descritas.