Tarea 2 (Patrones de Diseño)

Daniel Gutierrez - Carlos Guerrero - Ricardo López

14 de mayo de 2014

1. Definición

Los patrones de diseño son la base para la búsqueda de soluciones a problemas comunes en el desarrollo de software y otros ámbitos referentes al diseño de interacción o interfaces. Un patrón de diseño resulta ser una solución a un problema de diseño. 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 reutilizable, lo que significa que es aplicable a diferentes problemas de diseño en distintas circunstancias.

1.1. Objetivos

Los patrones de diseño pretenden:

- Reutilizar elementos en el diseño de Software.
- No reinventar la rueda...
- Facilitar el aprendizaje para las generaciones venideras.
- Estandarizar el modo de diseñar Software.
- Tener un idioma común entre desarrolladores.

1.2. Tipos de Patrones de Diseño

- Patrones creacionales.
- Patrones Estructurales.
- Patrones de Comportamiento.
- Patrones de Interacción.

2. MVC

El Modelo Vista Controlador (MVC) es un patrón de arquitectura de software que separa los datos y la lógica de negocio de una aplicación de la interfaz de usuario y el módulo encargado de gestionar los eventos y las comunicaciones. Para ello MVC propone la construcción de tres componentes distintos que son el modelo, la vista y el controlador, es decir, por un lado define componentes para la representación de la información, y por otro lado para la interacción del usuario. Este patrón de diseño se basa en las ideas de reutilización de código y la separación de conceptos, características que buscan facilitar la tarea de desarrollo de aplicaciones y su posterior mantenimiento.

2.1. Historia

MVC fue introducido por Trygve Reenskaug en Smalltalk-76 durante su visita a Xerox Parc en los años 70 y, seguidamente, en los años 80, Jim Althoff y otros implementaron una versión de MVC para la biblioteca de clases de Smalltalk-808 . Sólo más tarde, en 1988, MVC se expresó como un concepto general en un artículo9 sobre Smalltalk-80.

2.2. Descripción

El patrón de arquitectura "modelo vista controlador", es una filosofía de diseño de aplicaciones, compuesta por:

Modelo

- Contiene el núcleo de la funcionalidad (dominio) de la aplicación.
- Encapsula el estado de la aplicación.
- No sabe nada / independiente del Controlador y la Vista.

Vista

- Es la presentación del Modelo.
- Puede acceder al Modelo pero nunca cambiar su estado.
- Puede ser notificada cuando hay un cambio de estado en el Modelo.

Controlador

Reacciona a la petición del Cliente, ejecutando la acción adecuada y creando el modelo pertinente.
Para entender cómo funciona nuestro patrón Modelo vista controlador, se debe entender la división a través del conjunto de estos tres elementos y como estos componentes se comunican unos con los otros y con otras vistas y controladores externos a el modelo principal. Para ello, es importante saber que el controlador interpreta las entradas del usuario (tanto teclado como el ratón), enviado el mensaje de acción al modelo y a la vista para que se proceda con los cambios que se consideren adecuados.

Unión del modelo con la vista y el controlador

Como no todos los modelos pueden ser pasivos, necesitamos algo que comunique al controlador y a la vista, por lo que en este caso, si que necesitamos el modelo, ya que solo este puede llevar a cabo los cambios necesarios al estado actual en el que estos se encuentran.

Al contrario que el modelo, que puede ser asociado a múltiples asociaciones con otras vistas y controladores, cada vista solo puede ser asociada a un único controlador, por lo que han de tener una variable de tipo controler que notificara a la vista cual es su controlador o modelo asignado. De igual manera, el controlador tiene una variable llamada View que apunta a la vista. De esta manera, pueden enviarse mensajes directos el uno al otro y al mismo tiempo, a su modelo.

Al final, **la vista es quien lleva la responsabilidad de establecer la comunicación entre los elementos de nuestro patrón MVC**. Cuando la vista recibe un mensaje que concierne al modelo o al controlador, lo deja registrado como el modelo con el cual se comunicara y apunta con la variable controller al controlador asignado, enviándole al mismo su identificación para que el controlador establezca en su variable view el identificador de la vista y así puedan operar conjuntamente. El responsable de deshacer estas conexiones, seguirá siendo la vista, quitándose a si misma como dependiente del modelo y liberando al controlador.

3. Ejemplo

En mundo en constantes cambios respecto en tecnología, un ingeniero de software debe estar al corriente del estado del arte. Para esto es necesario actualizarse en contenidos e ir a la par con el avance, postulando y creando nuevos y/o mejorados sistemas de software orientado a solucionar un problema el cual demanda una tediosa intervención de una persona como por ejemplo en un día de elecciones, los votos, al igual como se hacen en otros países, podrían hacerse mediante ordenadores los cuales todos conectados mediante redes indicando en tiempo real los resultados correspondientes. En nuestro caso vamos a crear un sistema de software a baja escala que simule la propuesta anteriormente aludida utilizando la programación orientada a objetos en Java, así, a través de futuros cursos de programación y/o base de datos mejorar la idea y proponer una nuevas y mejoradas.

- Interfaz Gráfica (Vista)
- Procesos Internos de recepcion/respuesta a las operaciones entre el modelo y la vista [Listeners en Java por ejemplo] (Controlador)
- Métodos del software [PoblarBaseDeDatos(parametros) en java] (Modelo)

4. Anexo GitHub

GitHub/rlopezn: Grupo 1 Ingeniería de Software