Diseño de sistemas año 2022

Luciano Villa

**Patrones**

**Introducción:**

**Los patrones de diseño representan soluciones** a problemas específicos de diseño de software orientado a objetos, **que han sido desarrolladas y han ido evolucionando a lo largo del tiempo**. No es algo que se hace de un día para el otro, sino que refleja todo el rediseño y la recodificación (tiene que ser mínima ese rediseño y recodificación) que los desarrolladores han ido haciendo a medida que luchaban por **conseguir mayor reutilización y flexibilidad en su producto software final**. Los patrones de diseño expresan las soluciones de manera breve, concisa, fácilmente aplicable.

Los patrones resuelven problemas concretos de diseño y hacen que los diseños orientados a objetos sean más flexibles, elegantes y reutilizables.

Los patrones ayudan a los diseñadores a reutilizar buenos diseños al basar los nuevos diseños en la experiencia previa.

Cada patrón definido en el libro, nomina, explica y evalúa un diseño importante y recurrente en los sistemas orientados a objetos.

Los patrones de diseño hacen que sea más fácil reutilizar buenos diseños y arquitecturas. Al expresar como patrones de diseño técnicas que ya han sido probadas, las estamos haciendo más accesibles para los desarrolladores de nuevos sistemas. Los patrones de diseño nos ayudan a elegir las alternativas de diseño que hacen que un sistema sea reutilizable, y a evitar aquellas que dificultan dicha reutilización. Pueden incluso mejorar la documentación y el mantenimiento de los sistemas existentes al proporcionar una especificación explicita de las interacciones entre clases y objetos.

Todas las arquitecturas orientadas a objetos que están bien estructuradas están repletas de patrones.

Hoy en día, pocos son los arquitectos de software que no han oído hablar de patrones de diseño, y su uso generalmente diferencia a un buen diseño de otro que, en el mejor de los casos, resuelve un problema concreto pero que se adapta muy mal, o no lo hace en absoluto, a nuevos requisitos o cambios en los ya existentes.

Los patrones de diseño se pueden dividir en tres tipos: de Creación, estructurales y de comportamiento.

Los patrones de comportamiento tienen que ver con algoritmos y con la asignación de responsabilidades a objetos. Los patrones de comportamiento describen el flujo de control complejo que es difícil de seguir en tiempo de ejecución, lo que nos permite olvidarnos del flujo de control para concentrarnos simplemente en el modo en que se interconectan los objetos.

Los patrones de comportamiento basados en clases usan la herencia para distribuir el comportamiento entre clases. Un método PLANTILLA es una definición abstracta de un algoritmo. Define el algoritmo paso a paso, y cada paso invoca o bien a una operación abstracta o a una operación primitiva. Una subclase es la encargada de completar el algoritmo definiendo las operaciones abstractas.

“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”

Por último, la solución de un patrón de diseño me describe los elementos que constituyen al diseño, sus relaciones, responsabilidades y colaboraciones. La solución NO DESCRIBE UN DISEÑO O UNA IMPLEMENTACION EN CONCRETO, SINO QUE UN PATRON ES MAS BIEN COMO UNA PLANTILLA QUE PUEDE APLICARSE EN MUCHAS SITUACIONES DIFERENTES. El patrón proporciona una descripción abstracta de un problema de diseño y cómo lo resuelve una disposición general de elementos (clases y objetos).

**Patrón Template Method**

Este patrón es un patrón de COMPORTAMIENTO. El objetivo de este patrón es que podamos definir en una operación el esqueleto de un algoritmo o de cierta lógica que luego iremos a ejecutar, y DEJAR algunos detalles DE IMPLEMENTACION, para que los hijos los puedan sobrescribir mediante HERENCIA, pero solo algunos detalles concretos, ya que, el cuerpo principal del algoritmo estará en el padre.

Otra definición: **Propósito:** *Define en una operación el esqueleto de un algoritmo, delegando en las subclases algunos de sus pasos. Permite que las subclases redefinan ciertos pasos de un algoritmo* ***sin cambiar su estructura****.* ¿el sin cambiar su estructura hace referencia a que una subclase no puede cambiar el orden de llamada de un método redefinido? Con PROPOSITO, nos referimos a qué hace este patrón de diseño, en qué se basa y cuál es el problema concreto de diseño que resuelve.

Un método plantilla define un algoritmo en términos de operaciones abstractas que las subclases deber redefinir para proporcionar un determinado comportamiento…

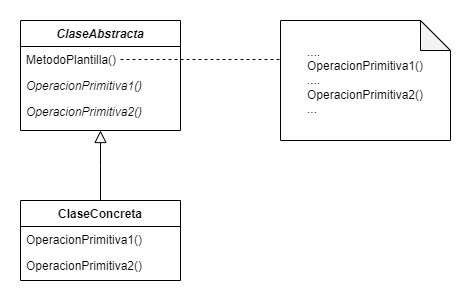
Al definir algunos de los pasos de un algoritmo usando operaciones abstractas, el método plantilla fija su ordenación, pero permite que las subclases modifiquen dichos pasos para adecuarse a sus necesidades.

APLICABILIDAD:

Este patrón debería usarse:

* Para implementar las partes de un algoritmo que no cambian y dejar que sean las subclases quienes implementen el comportamiento que puede variar.
* Cuando el comportamiento repetido de varias subclases debería factorizarse y ser localizado en una clase común para evitar el código duplicado.
* Para controlar las extensiones de las subclases. Podemos definir un método plantilla que llame a operaciones “de enganche” en determinados puntos, permitiendo así las extensiones sólo en esos puntos.

Estructura ESTATICA del patrón:



Explicación de estructura: En este patrón tenemos una clase abstracta que tiene definido un conjunto de operaciones abstractas que van a ser parte de la plantilla. Estas operaciones van a ser implementadas y redefinidas en una clase concreta que le DARÁ EL COMPORTAMIENTO ESPECÍFICO. Además, se encuentra una operación concreta, que es justamente la operación MetodoPlantilla () que es la que va a permitir realizar el conjunto de operaciones y la secuencia que corresponda, a partir de las implementaciones CONCRETAS.

PARTICIPANTES DE LA ESTRUCTURA?

CONSECUENCIAS

IMPLEMENTACION

EJEMPLO CODIGO