Técnico en

DESARROLLO DE SOFTWARE

Diseño e Implementación del Software

Instituto Von Neumann

Diseño e Implementación del Software

< Unidad II >

Historia

El concepto de los patrones fue utilizado y especificado por Christopher Alexander en El lenguaje de patrones. El libro habla de un "lenguaje" para diseñar el entorno urbano. Las unidades de este lenguaje son los patrones. Pueden describir lo altas que tienen que ser las ventanas, cuántos niveles debe tener un edificio, cuan grandes deben ser las zonas verdes de un barrio, etcétera.

Esta idea fue tomada por cuatro autores: Erich Gamma, John Vlissides, Ralph Johnson y Richard Helm y en 1995, publicaron el libro Patrones de diseño, en el que aplicaron el concepto de los patrones de diseño a la programación. El libro presentaba 23 patrones que resolvían varios problemas del diseño orientado a objetos y se convirtió en un éxito de ventas con rapidez. Al tener un título tan largo en inglés, la gente empezó a llamarlo "el libro de la banda de los cuatro", lo que pronto se abrevió a "el libro GoF".

¿Qué es un patrón de diseño?

Son recetas ya validadas a problemas que ocurren con frecuencia cuando diseñamos software, es decir que son como planos prefabricados que se pueden personalizar para resolver un problema de diseño recurrente a nivel de código.

Para poder utilizar un patrón de diseño hay que tomar en cuenta que no se puede elegir y simplemente copiarlo en el programa como si se tratara de funciones o librerías ya preparadas ya que no es una porción específica de código, sino un concepto general para resolver un problema particular. Puedes seguir los detalles del patrón e implementar una solución que encaje con las realidades de tu propio programa.

Es muy común encontrar casos en los que patrones se confunden con algoritmos porque ambos describen soluciones típicas a problemas conocidos. Mientras que un algoritmo siempre define un grupo claro de acciones para lograr un objetivo, un patrón es una descripción de más alto nivel de una solución. El código del mismo patrón aplicado a dos programas distintos puede ser diferente.

Una analogía de un algoritmo sería una receta de cocina: ambos cuentan con pasos claros para alcanzar una meta. Por su parte, un patrón es más similar a un plano, ya que puedes observar cómo son su resultado y sus funciones, pero el orden exacto de la implementación depende de ti.

¿En qué consiste un patrón de diseño?

→ Dentro de las características que podemos encontrar y que conforman un patrón de diseño son:
→ El propósito del patrón es explicar de forma clara y breve qué problema se soluciona.
→ La motivación dando a conocer de forma detallada cómo resuelve dicho problema.
→ La estructura de las clases identificando cómo están conformadas y cómo se relacionan entre sí.

El ejemplo de código es una ayuda muy grande que podemos encontrar ya que nos da una idea clara de cómo funciona dicho patrón de diseño.

Algunos catálogos de patrones enumeran otros detalles útiles, como la aplicabilidad del patrón, los pasos de implementación y las relaciones con otros patrones.

Consideraciones

Toma en cuenta los siguientes aspectos antes de elegir algún patrón de diseño:

-Chapuzas para un lenguaje de programación débil

La necesidad por los patrones normalmente surge cuando la gente elige un lenguaje de programación o una tecnología que carece del nivel necesario de

abstracción. En este caso, los patrones se convierten en una chapuza que otorga al lenguaje unas súper habilidades muy necesitadas.

-Soluciones ineficientes

Los patrones de diseño son comúnmente muy conocidos y aplicarlos sin adaptarse al contexto de tu proyecto que estás trabajando puede llegar a perjudicar en lugar de beneficiar su utilización.

-Uso injustificado

"Si lo único que tienes es un martillo, todo te parecerá un clavo".

Esta práctica comúnmente se da lugar cuando se posee muy poco conocimiento de los patrones de diseño y se intenta aplicar un patrón en situaciones en las que la simplicidad de un código fuente puede ser una solución viable, llegando a complicar el proyecto que realizas.

Clasificación de los patrones

Existe gran cantidad de patrones de diseño los cuales varían en su complejidad, nivel de detalle y escala de aplicabilidad al sistema completo que se diseña.

Los patrones más básicos y de más bajo nivel suelen llamarse modismos (idioms). Normalmente cada lenguaje de programación contiene sus propios modismos.

Existen también patrones más universales y de más alto nivel los cuales conocemos como patrones de arquitectura. Los desarrolladores pueden implementar estos patrones prácticamente en cualquier lenguaje. Al contrario que otros patrones, pueden utilizarse para diseñar la arquitectura de una aplicación completa.

Los patrones de diseño también pueden clasificarse por su propósito, algunos de ellos son:

Patrones creacionales

Estos son patrones de diseño que se ocupan de los mecanismos de creación de objetos, tratando de crear objetos de una manera adecuada a la situación. La forma básica de creación de objetos podría dar lugar a problemas de diseño o complejidad añadida al diseño. Los patrones de diseño creacional resuelven este problema al controlar de alguna manera la creación de este objeto.

Algunos patrones creacionales que podemos encontrar son:

- *Abstract Factory:* Crea una instancia de varias familias de clases.
- *Builder*:Separa la construcción del objeto de su representación.
- *Factory Method:*Crea una instancia de varias clases derivadas.
- *Object Pool:* Evite la adquisición costosa y la liberación de recursos reciclando objetos que ya no están en uso
- *Prototype:* Una instancia completamente inicializada para ser copiada o clonada.
- *Singleton:* Una clase de la que solo puede existir una única instancia.

Patrones estructurales

Los patrones de diseño estructural son patrones que facilitan el diseño al identificar una forma sencilla de realizar relaciones entre entidades.

- *Adaptader:* Hacer coincidir interfaces de diferentes clases.
- *Bridge:* Separa la interfaz de un objeto de su implementación.
- *Composite:* Una estructura de árbol de objetos simples y compuestos.
- *Decorator:* Agregar responsabilidades a los objetos dinámicamente
- *Facade:* Una sola clase que representa un subsistema completo
- *Flyweight:* Una instancia de granularidad fina utilizada para compartir eficientemente
- *Private class data:* Restringe el acceso del usuario/mutador
- *Proxy:* Un objeto que representa a otro objeto.

Patrones de comportamiento

Los patrones de diseño de comportamiento identifican patrones de comunicación comunes entre objetos y realizan estos patrones. Al hacerlo, estos patrones aumentan la flexibilidad para llevar a cabo esta comunicación.

- *Chain of responsibility:* Una forma de pasar una solicitud entre una cadena de objetos.
- *Command:* Encapsular una solicitud de comando como un objeto
- *Interprete*: Una forma de incluir elementos de lenguaje en un programa
- *Iterator:* Acceder secuencialmente a los elementos de una colección.
- *Mediator:* Define la comunicación simplificada entre clases.
- *Memento:* Capturar y restaurar el estado interno de un objeto
- *Null Object:* Diseñado para actuar como un valor predeterminado de un objeto
- *Observer:* Una forma de notificar el cambio a un número de clases.
- *State:* Alterar el comportamiento de un objeto cuando cambia su estado
- *Strategy:* Encapsula un algoritmo dentro de una clase.
- *Template method:* Diferir los pasos exactos de un algoritmo a una subclase
- *Visitor:* Define una nueva operación a una clase sin cambios.

<Bibliografía>

Referencias de contenido:

http://www.uml.org/what-is-uml.htm

Recursos adicionales

- http://www.juntadeandalucia.es/servicios/madeja/contenido/recurso/122
- https://docs.microsoft.com/es-es/azure/architecture/guide/architecture-styles/microservices
- https://msdn.microsoft.com/es-es/hh144976.aspx
- https://www.academia.edu/10102692/Arquitectura_de_n_capas

Libros:

- ✔ James A. Senn. (2001). Análisis y Diseño de Sistemas de Información. México: McGraw-Hill.
- ✔ Joseph Schmuller. (2000). Aprendiendo UML en 24 Horas. México: Pearson Education
- ✔ Shvets A. Sumergete en los patrones de diseño
- ✔ Robert C. Martin, Clean Code- A Handbook of Agile Software Craftsmanship.
- ✔ Alexander Shvets- (Source Making, 2020) ”Sumérgete en los patrones de diseño”.