UNIVERSIDAD AUTÓNOMA DE BAJA CALIFORNIA

Facultad de Ciencias de la Ingeniería y Tecnología

Unidad Valle de las Palmas

Meta 1.1. Examinar los antecedentes, estructura y niveles de los patrones de software

Patrones de Software

José Humberto Moreno Mejia

AGOSTO 2024

Introducción

El objetivo de esta investigación fue explorar los patrones de software, entendiendo cómo funcionan y por qué son útiles en el desarrollo de programas. Los patrones de software son una especie de “receta” que los programadores usan para resolver problemas comunes en sus proyectos. Este estudio recopila información sobre los diferentes tipos de patrones, cómo están organizados, y los niveles en los que se usan, además de brindar una reflexión final sobre su importancia.

Historia de los patrones de software

El concepto de patrones de software viene de la arquitectura tradicional. Christopher Alexander, un arquitecto, fue quien en los años 70 comenzó a describir patrones como soluciones que se repetían en situaciones similares dentro del diseño de edificios. Esta idea fue adoptada más tarde por la comunidad de programadores, que la aplicó en la creación de software. A partir de la década de los 90, los patrones se convirtieron en una herramienta esencial para los desarrolladores de software, especialmente después de la publicación del libro “Design Patterns” de los famosos “Gang of Four” (GOF), quienes establecieron una base sólida para la aplicación de estos conceptos en el desarrollo de sistemas.

Definición de patrones de software

Un patrón de software es una solución probada que ayuda a resolver un problema común en un contexto específico. Básicamente, es una fórmula que los programadores pueden reutilizar para no tener que reinventar la rueda cada vez que se enfrentan a ciertos desafíos. Según Gamma y sus colegas (1995), los patrones de diseño permiten mejorar la eficiencia, evitando cometer los mismos errores y ahorrando tiempo al reutilizar soluciones que ya han funcionado bien en el pasado.

Patrones POSA: objetivos, estructura y clasificación

POSA (Pattern-Oriented Software Architecture) es una colección de patrones centrada en cómo estructurar bien las arquitecturas de software. El objetivo de estos patrones es mejorar aspectos clave como la flexibilidad, el rendimiento y la facilidad para hacerle cambios al sistema sin que todo se rompa. La estructura de los patrones POSA sigue un formato que incluye una descripción del problema, el contexto en el que surge y la solución propuesta.

Los patrones POSA se dividen en tres grupos principales:
1. Patrones arquitectónicos: Resuelven problemas generales de estructura en grandes sistemas de software.
2. Patrones de diseño: Se enfocan en cómo deben interactuar las clases y objetos dentro de una aplicación.

3. Patrones idiomáticos: Son soluciones más específicas que se aplican dentro de un lenguaje de programación particular.

Buschmann et al. (1996) explican que estos patrones ayudan a diseñar software que pueda evolucionar con el tiempo y adaptarse a nuevas necesidades sin grandes complicaciones.

Patrones GOF: objetivos, estructura y clasificación

Los patrones GOF, llamados así por los autores Gamma, Helm, Johnson y Vlissides, son tal vez los más conocidos en el mundo de la programación orientada a objetos. Su principal objetivo es estandarizar soluciones a problemas que aparecen una y otra vez en el diseño de software, facilitando la comunicación entre desarrolladores y haciendo que el código sea más fácil de mantener y entender.

Cada patrón GOF sigue una estructura sencilla: se presenta el problema, el contexto, la solución y las consecuencias de aplicarla. Estos patrones están divididos en tres categorías:
1. Patrones creacionales: Se centran en cómo crear objetos de manera que el código sea más flexible y menos complicado.
2. Patrones estructurales: Ayudan a organizar las clases y los objetos para que el sistema sea más fácil de modificar y entender.
3. Patrones de comportamiento: Se enfocan en cómo los objetos se comunican entre sí y cómo manejan sus responsabilidades.

Gamma et al. (1995) destacan que estos patrones no solo ahorran tiempo, sino que también permiten crear sistemas más eficientes.

Niveles de los patrones de software

Los patrones de software se aplican en diferentes niveles, dependiendo de la escala del problema:
1. Patrones arquitectónicos: Ayudan a resolver problemas de diseño a gran escala, como la estructura completa de un sistema.
2. Patrones de diseño: Son más específicos y ayudan a resolver problemas que surgen dentro de módulos o componentes de un sistema.
3. Patrones idiomáticos: Son soluciones más concretas que se aplican a lenguajes de programación específicos y resuelven problemas puntuales en el código.

Estos niveles permiten que los patrones de software se utilicen en varias etapas del desarrollo, desde la planificación del sistema hasta la implementación de detalles técnicos.

¿Por qué usar patrones de software?

Los patrones de software facilitan la vida de los programadores, ya que permiten reutilizar soluciones que han sido probadas y funcionan bien. Según Vlissides (1995), los patrones también crean un lenguaje común entre los desarrolladores, lo que hace que la comunicación dentro de los equipos sea mucho más fluida. Además, los patrones ayudan a

evitar errores repetitivos y hacen que el software sea más fácil de mantener y evolucionar a lo largo del tiempo.

Conclusión

Los patrones de software son una herramienta clave en el desarrollo de programas que deben ser eficientes y fáciles de mantener. Su uso permite a los desarrolladores ahorrar tiempo y esfuerzo, reutilizando soluciones que ya han demostrado ser efectivas. Sin embargo, es importante usarlos con criterio, ya que no todos los patrones son adecuados para todas las situaciones. Comprender cuándo y cómo aplicar estos patrones es lo que diferencia a un desarrollador experimentado de uno que solo sigue recetas sin entenderlas. Al final, los patrones son una guía, no una ley, y su buen uso depende de la capacidad de adaptarlos a las necesidades particulares de cada proyecto.

Referencias

- Alexander, C. (1977). A pattern language: towns, buildings, construction. Oxford University Press.
- Buschmann, F., Meunier, R., Rohnert, H., Sommerlad, P., & Stal, M. (1996). Pattern-oriented software architecture: A system of patterns. John Wiley & Sons.
- Gamma, E., Helm, R., Johnson, R., & Vlissides, J. (1995). Design patterns: Elements of reusable object-oriented software. Addison-Wesley.
- Vlissides, J. (1995). Pattern Hatching: Design Patterns Applied. Addison-Wesley.