Universidad Tecnológica de Tijuana

Selection of Design Patterns

Student:
Canales Bernal Manuel Alejandro

Subject:
Desarrollo movil Integral

Grade & Group:
10A

Teacher:
Ray Brunett Parra Galaviz

Date:
January 9, 2025

Introduction

Design patterns are reusable solutions to common problems encountered during software development. They provide a template for solving recurring design challenges, improving code structure, and promoting maintainability. Selecting the right design pattern depends on the specific requirements, context, and constraints of the project. This process is crucial for creating scalable, efficient, and easily understandable software systems.

Understand the Problem

A thorough understanding of the problem domain is essential to select an appropriate design pattern. For instance, the Singleton pattern is suitable for scenarios requiring a single instance of a class, while the Observer pattern is ideal for event-driven systems.

Pattern Categories

Design patterns are categorized into Creational, Structural, and Behavioral patterns. Each category serves a unique purpose:

- **Creational Patterns:** Manage object creation (e.g., Factory, Singleton).
- **Structural Patterns:** Focus on class composition (e.g., Adapter, Composite).
- **Behavioral Patterns:** Define communication between objects (e.g., Strategy, Observer).

Impact on Scalability and Maintainability

The choice of a design pattern can significantly affect the scalability and maintainability of software. For example, using the Strategy pattern allows developers to dynamically change an object's behavior, facilitating extensibility without altering existing code.

Complexity vs. Simplicity

Balancing simplicity and functionality is key. Over-engineering a solution with unnecessary patterns can complicate the codebase.

Team Familiarity and Codebase Consistency:

The development team's familiarity with specific patterns and consistency with the existing codebase should guide the selection process.

Conclusion

The selection of design patterns is a vital aspect of software design, providing developers with proven solutions to complex problems. By understanding the problem domain, considering pattern categories, and balancing simplicity with functionality, developers can create software that is both efficient and maintainable. Similarly, key concepts in strategy versioning, such as semantic versioning, branching strategies, and version control tools, form the foundation for effective software lifecycle management, enabling teams to build scalable and reliable systems.

Bibliography

Gamma, E., Helm, R., Johnson, R., & Vlissides, J. (1994). *Design patterns: Elements of reusable object-oriented software.* Addison-Wesley. Retrieved January 9, 2025, from https://www.oreilly.com/library/view/design-patterns-elements/9780201633610/

Refactoring Guru. (n.d.). *Design Patterns.* Retrieved January 9, 2025, from https://refactoring.guru/design-patterns

GeeksforGeeks. (n.d.). *Introduction to Design Patterns.* Retrieved January 9, 2025, from https://www.geeksforgeeks.org/introduction-to-design-patterns/

IBM Developer. (n.d.). *Design patterns: What they are and why they matter.* Retrieved January 9, 2025, from https://developer.ibm.com/articles/design-patterns-overview/

Atlassian. (n.d.). *Software design patterns: The key to developing robust software.* Retrieved January 9, 2025, from https://www.atlassian.com/agile/software-development/design-patterns