OBJECT ORIENTED ANALYSIS & DESIGN
DATA STRUCTURES & ALGORITHMS

Design Patterns

“Gang of Four” (GoF) Book

- Design Patterns: Elements of Reusable Object-Oriented Software, Addison-Wesley Publishing Company, 1994 Written by this "gang of four"
  1. Dr. Erich Gamma, then Software Engineer, Taligent, Inc.
  2. Dr. Richard Helm, then Senior Technology Consultant, DMR Group
  3. Dr. Ralph Johnson, then and now at University of Illinois, Computer Science Department
  4. Dr. John Vlissides, then a researcher at IBM

Object-Oriented Design Patterns

This book defined 23 patterns in three categories:

- Creational patterns deal with the process of object creation
- Structural patterns, deal primarily with the static composition and structure of classes and objects
- Behavioral patterns, which deal primarily with dynamic interaction among classes and objects

Key Points

- Design Patterns are recurring solutions to software design problems within a particular context
- There are 23 Design Patterns described by the GoF
- Design Patterns are categorized into Creational, Structural, and Behavioral
- There are other Design Patterns in existence

Definitions

- A design pattern is a documented best practice or core of a solution that has been applied successfully in multiple environments to solve a problem that recurs in a specific set of situations

- Design patterns are recurring solutions to software design problems you find again and again in real-world application development

- Design patterns represent solutions to problems that arise when developing software within a particular context (i.e., Pattern = problem/solution pair in context)

- Design patterns are standard solutions to common problems in software design

Benefits & Drawbacks

Benefits:

- Design patterns enable large-scale reuse of software architectures
- Patterns explicitly capture expert knowledge and design tradeoffs, and make this expertise more widely available
- Patterns help improve developer communication

Drawbacks:

- Patterns do not lead to direct code reuse
- Patterns are deceptively simple
- Teams may suffer from pattern overload
- Patterns are validated by experience and discussion rather than by automated testing

Kinds of Patterns

1. Analysis Patterns

   for modeling requirements

2. Architectural Patterns

   for major components of a software system

3. Design Patterns

   for smaller software components

4. Programming Patterns

   for specific languages

Design Pattern Catalog

CREATIONAL PATTERNS

1. Factory Method
2. Abstract Factory
3. Builder
4. Prototype
5. Singleton

STRUCTURAL PATTERNS

1. Adapter
2. Bridge
3. Composite
4. Decorator
5. Façade
6. Flyweight
7. Proxy

BEHAVIORAL PATTERNS

1. Chain of Responsibility
2. Command
3. Interpreter
4. Iterator
5. Mediator
6. Memento
7. Observer
8. State
9. Strategy
10. Template Method
11. Visitor

Thank You!