

Introduction to "Design Patterns"

Kai SHI

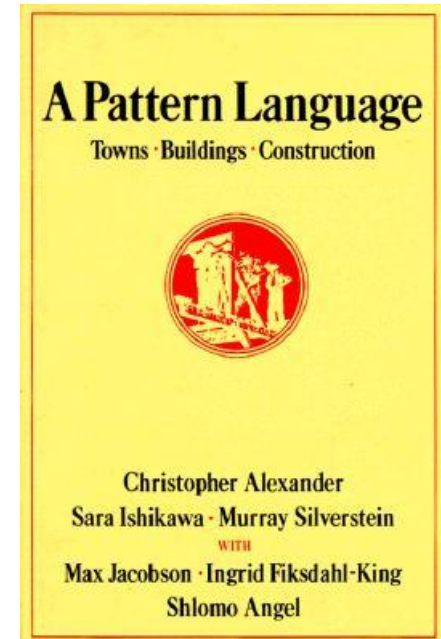
Outline

- Patterns History
 - What Is A Pattern?
 - Why Patterns?
 - Problems with Design Patterns
 - Types Of Software Patterns
 - GoF Design Patterns
 - Pattern List
-

Patterns History

Christopher Alexander, 1977

- A 1977 book.
- The book creates a new language, what the authors call a pattern language derived from timeless entities called patterns.
- Pattern is a solution to a problem in a context.



Eishin Campus, Iruma-shi, Saitama, Japan, 1985 (1/2)



<http://eishin.ac/school-life/facilities/>

Eishin Campus, Iruma-shi, Saitama, Japan (2/2)



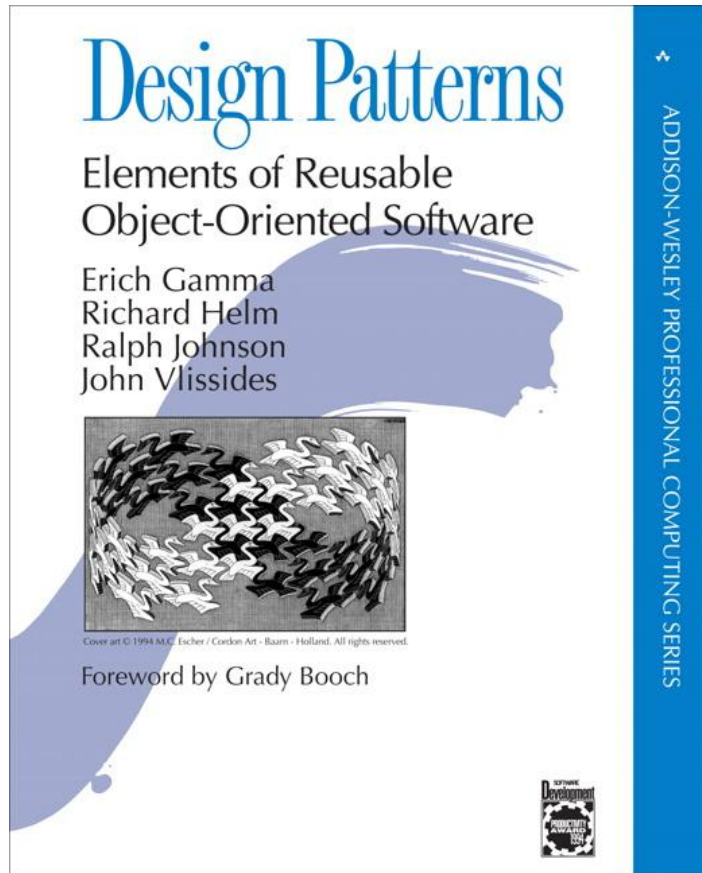
Laying out the campus, with flags, so that we could all, together, feel the buildings and the spaces of the neighborhood, adjust them, again and again, until they felt just right.



This is the drawing made after placing of all the flags, as shown in the process at head of this page. These flags were carefully transcribed, by position, from the exact position in the field. These became the basis of the plan, and gave the life to the positions of the buildings as they may be seen in photographs: a rather simple naturalness, which arises, actually, from careful, careful attention to the land, when placing the flags originally.



GoF (E. Gamma, R. Helm, R. Johnson, J. Vlissides), 1994



- **23** classic software design patterns
- in C++ and Smalltalk.



What Is A Pattern?

- Current use comes from the work of the architect Christopher Alexander.
 - Alexander studied ways to improve the process of designing buildings and urban areas.
 - “Each pattern is a three-part rule, which expresses a relation between a certain context, a problem and a solution.”
 - Hence, the common definition of a pattern: **“A solution to a problem in a context.”**
 - Patterns can be applied to many different areas of human endeavor, including software development.
-

Why Patterns?

- "Designing object-oriented software is hard and designing reusable object-oriented software is even harder." - Erich Gamma
- Experienced designers reuse solutions that have worked in the past.
- Well-structured object-oriented systems have recurring patterns of classes and objects.
- Knowledge of the patterns that have worked in the past allows a designer to be more productive and the resulting designs to be more flexible and reusable.

Benefits Of Design Patterns

- Capture expertise and make it accessible to non-experts in a standard form
- Facilitate communication among developers by providing a common language
- Make it easier to reuse successful designs and avoid alternatives that diminish reusability
- Facilitate design **modifications**
- Improve design documentation
- Improve design understandability

Software Patterns History

- 1987 - Cunningham and Beck used Alexander's ideas to develop a small pattern language for Smalltalk.
- 1990 - The Gang of Four (Gamma, Helm, Johnson and Vlissides) begin work compiling a catalog of design patterns
- 1991 - Bruce Anderson gives first Patterns Workshop at OOPSLA
- 1993 - Kent Beck and Grady Booch sponsor the first meeting of what is now known as the Hillside Group
- 1994 - First Pattern Languages of Programs (PLoP) conference
- 1994 - The Gang of Four (GoF) publish the Design Patterns book

Why study design patterns? (1/2)


- Up to now in your degree you have been taught the OO basics and some higher level principles, but you cannot design and/or do wrong.
 - You should study some existing patterns produced and tested by experts; When you're in the situation where you think: "Oh I've seen this problem before and I can solve it by ..."
-

Why study design patterns? (2/2)

- Reuse of design expertise.
- Support software that is **flexible to change** (extension).
- Improve communication between engineers: a common **vocabulary**.

Problems with Design Patterns

- Trade-off: design can become a little more complicated.
- Inexperienced users often try to use more design patterns than they need to.
 - You need to ask yourself why you're using the design pattern.



Patterns are tools.
You use tools.
Not tools use you.

Types Of Software Patterns

- Analysis
 - Design
 - Organizational
 - Process
 - Project Planning
 - Configuration Management
-

Types Of Software Patterns

- Riehle and Zullighoven in “Understanding and Using Patterns in Software Development” mention three types of software patterns.

- Conceptual Pattern

- Pattern whose form is described by means of terms and concepts from the application domain

- Design Pattern

- Pattern whose form is described by means of software design constructs, such as objects, classes, inheritance and aggregation

- Programming Pattern (Programming Idiom)

- Pattern whose form is described by means of programming language constructs

Design Pattern Levels Of Abstraction

- Complex design for an entire application or subsystem



More Abstract

- Solution to a general design problem in a particular context

More Concrete



- Simple reusable design class such as a linked list, hash table, etc.

GoF Design Patterns

- The GoF design patterns are in the middle of these levels of abstraction.
- “A design pattern names, abstracts, and identifies key aspects of a common design structure that makes it useful for creating a reusable object-oriented design.”
- The GoF design patterns are “descriptions of communicating objects and classes that are customized to solve a general design problem in a particular context.”

GoF Classification Of Design Patterns (1/2)

- Purpose - what a pattern does
 - Creational Patterns
 - Concern the process of object creation
 - Structural Patterns
 - Deal with the composition of classes and objects
 - Behavioral Patterns
 - Deal with the interaction of classes and objects
-

GoF Classification Of Design Patterns (2/2)

- Scope - what the pattern applies to
 - Class Patterns
 - Focus on the relationships between classes and their subclasses
 - Involve inheritance reuse
 - Object Patterns
 - Focus on the relationships between objects
 - Involve composition reuse
-

GoF Essential Elements Of Design Patterns

■ Pattern Name

- Having a concise, meaningful name for a pattern improves **communication among developers**

■ Problem

- What is the problem and context where we would use this pattern?
- What are the conditions that must be met before this pattern should be used?

■ Solution

- A description of the elements that make up the design pattern
- Emphasizes their relationships, responsibilities and collaborations
- Not a concrete design or implementation; rather an abstract description

■ Consequences

- The pros and cons of using the pattern
 - Includes impacts on reusability, portability, extensibility
-

GoF Pattern Template (1/3)

- Pattern Name and Classification
 - A good, concise name for the pattern and the pattern's type
 - Intent
 - Short statement about what the pattern does
 - Also Known As
 - Other names for the pattern
 - Motivation
 - A scenario that illustrates where the pattern would be useful
 - Applicability
 - Situations where the pattern can be used
-

GoF Pattern Template (2/3)

- Structure
 - A graphical representation of the pattern
- Participants
 - The classes and objects participating in the pattern
- Collaborations
 - How to do the participants interact to carry out their responsibilities?
- Consequences
 - What are the pros and cons of using the pattern?
- Implementation
 - Hints and techniques for implementing the pattern

GoF Pattern Template (3/3)

- Sample Code
 - Code fragments for a sample implementation
 - Known Uses
 - Examples of the pattern in real systems
 - Related Patterns
 - Other patterns that are closely related to the pattern
-

Pattern List – Three types of pattern (1/3)

- Creational Patterns 创建型
 - Abstract Factory (抽象工厂)
 - Builder (建造者)
 - Factory Method (工厂方法)
 - Singleton (单例)
 - Prototype (原型)

Pattern List – Three types of pattern (2/3)

■ Structural Patterns 结构型

- ❑ Adapter (适配器)
- ❑ Bridge (桥接)
- ❑ Composite (组合)
- ❑ Decorator (装饰器)
- ❑ Facade (门面)
- ❑ Flyweight (享元)
- ❑ Proxy (代理)

Pattern List – Three types of pattern (3/3):

- Behavioral Patterns 行为型
 - Chain of Responsibility (职责链)
 - Command (命令)
 - Interpreter (解释器)
 - Iterator (迭代器)
 - Mediator (中介者)
 - Memento (备忘录)
 - Observer (观察者)
 - State (状态)
 - Strategy (策略)
 - Template Method (模板方法)
 - Visitor (访问者)

We use Java in this course

- Java's “field” --> C++'s “member variable”
- Java's “method” --> C++'s “member function”

Outline of Design Pattern Course

- UML Notation
 - Principles of Object Oriented Design
 - 23 classic software design patterns
 - Each pattern
 - Solve a problem without using the pattern
 - Learn the pattern
 - Solve more problems using the pattern
 - Summary of 23 design patterns
-