SENG480A/CSC576A
Topics in Software Engineering
Software Development, Architecture & Evolution
Lectures, Nov 22, 26, 2001

Hausi A. Müller
University of Victoria

Fall 2001

Schedule

- Assignment 4
  - Due Dec 3
- Marks
  - A1, A2, midterm posted
  - A3 marks coming soon
- Final
  - Friday, December 21, 7-10 pm
  - Three hours, closed books, closed notes

Software Design Patterns

Hausi A. Müller
University of Victoria

Final Exam Questions

- What is the difference between and architectural style and a design pattern?
- What is the difference between an ABAS and a design pattern?
- What are the major components of an ABAS, a design pattern, and an AntiPattern?
- What is a pattern format?
- What is the difference between a structural and a behavioural pattern?
- Write a rationale to convince your software team to use design patters. Discuss advantages and disadvantages.

Final Exam Questions …

- Write a rationale to convince your software team to use design patters. Discuss advantages and disadvantages.
- Write a brief process description on how to use ABASs during software design.
- How can you keep ABAS or design pattern documentation up-to-date and current over long periods of time? Why is that useful?

Motivation

- Vehicle for reasoning about design or architecture at a higher level of abstraction (design confidence)
- Mining or discovering design patterns in legacy systems
- Software architecture
  - dissemination of good design, design reuse
- Engineering Handbooks
  - contain a wealth of experience

Software Patterns

- Conceptual patterns
- Architectural patterns
- Design patterns
- Generative patterns
- Programming patterns or idioms
- Analysis patterns
- AntiPatterns
- Organizational patterns

Pattern Definitions

- A pattern is a named nugget of insight that conveys the essence of a proven solution to a recurring problem within a certain context amidst competing concerns [Riehle]
- A pattern is the abstraction from a concrete form which keeps recurring in specific non-arbitrary contexts

Pattern Definitions …

- Description of communicating objects and classes that are customized to solve a general design in a particular context [GoF].
- Design patterns capture the static and dynamic structures of solutions that occur repeatedly when producing applications in a particular context [Coplien]

Historical Perspective

- 1979
  - Alexander's Timeless Way of Building
- 1987
  - OOPSLA workshop by Beck & Ward
- 1994
  - First PLoP conference
- 1995
  - GoF (Gamma, Helm, Johnson, Vlissides)
  - Design Patterns; Elements of Reusable Object-Oriented Software

A Good Pattern

- It solves a problem
  - Patterns capture solutions, not just abstract principles or strategies
- It is a proven concept
  - Patterns capture solutions with a track record, not theories or speculation
- The solution isn't obvious
  - The best patterns generate a solution indirectly; normal for many design problems

A Good Pattern

- It describes a relationship
  - Patterns describe more than black boxes: system structures and mechanisms
- The pattern has a significant human component
  - The best patterns explicitly appeal to aesthetics and utility

Pattern Formats

- GoF format
- Alexandrian form (canonical form)
- Essential components of a pattern format
  - Name, problem, context, forces
  - Solution, examples, context,
  - Rationale, related patterns, known uses

Pattern Format …

- Name
  - meaningful phrase
- Problem
  - a statement of the problem which describes its intent: the goals and objectives it wants to reach within the given context and forces

Pattern Format …

- Context
  - preconditions under which the problem and its solutions seem to occur
  - the pattern's applicability
  - may change over time
- Forces
  - relevant forces and constraints and their interactions and conflicts
  - motivational scenario for the pattern

Pattern Format …

- Solution
  - Static and dynamic relationships describing how to realize the pattern
  - instructions on how to construct the work products
  - pictures, diagrams, prose which highlight the pattern's structure, participants, and collaborations

Pattern Format …

- Examples
  - ➢ one or more sample applications to illustrate
    - a specific context
    - how the pattern is applied
- Resulting context
  - ➢ the state or configuration after the pattern has been applied
  - ➢ consequences (good and bad) of applying the pattern

Pattern Format …

- Rationale
  - ➢ justification of the steps or rules in the pattern
  - ➢ how and why it resolves the forces to achieve the desired goals, principles, and philosophies
  - ➢ how are the forces orchestrated to achieve harmony
  - ➢ how does the pattern actually work

Pattern Format

- Related patterns
  - ➢ the static and dynamic relationships between this pattern and other patterns
- Known uses
  - ➢ to demonstrate that this is a proven solution to a recurring problem

Qualities of a Pattern

- Encapsulation and abstraction
  - ➢ encapsulates a well-defined problem and its solution in a particular domain
  - ➢ provides crisp, clear boundaries to crystallize the problem and solution spaces
  - ➢ serves as an abstraction which embodies domain knowledge and experience
  - ➢ may occur at different levels of abstraction

Qualities of a Pattern …

- Openness and variability
  - is open for extension and parameterization by other patterns
  - is able to solve larger problems in concert with other patterns
  - can be realized by a variety of implementations (variants)

Qualities of a Pattern …

- Generativity and composability
  - applying a pattern once provides a context for further applications
  - patterns are easier to apply in another context than C++ code
  - can evolve into Golden Hammer AntiPattern

Qualities of a Pattern …

- Equilibrium
  - realizes a balance among its forces and constraints
  - realizes an invariant, heuristics, or a policy which minimizes conflict within the solution space
  - an invariant characterizes the problem solving philosophy

GoF Catalog of 23 Patterns

- Creational patterns
  - Abstract the instantiation process
  - Make the system independent on how the objects are created, composed, and represented
    - Abstract Factory
    - Builder
    - Factory Method
    - Prototype
    - Singleton

GoF Catalog of 23 Patterns …

- Structural patterns
  - Composition of classes and objects to form larger structures
  - Compose classes to form new interfaces
  - Compose objects to provide new functionality
    - Adaptor
    - Bridge
    - Composite
    - Decorator
    - Façade
    - Flyweight
    - Proxy

GoF Catalog of 23 Patterns …

- Behavioral patterns
  - concerned with algorithms and the assignment of responsibilities among objects
    - Chain of Responsibility
    - Command
    - Interpreter
    - Iterator
    - Mediator
    - Memento
    - Observer
    - State
    - Strategy
    - Template Method
    - Visitor

Summary

- Vehicle for reasoning about design, architecture, component technology
- GoF book is great but there are many other software patterns