Design Patterns

Hausi A. Müller
University of Victoria

Software Architecture Course
Spring 2000

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

- *Design patterns [GoF]*
- *Pattern languages [Coplien]*
- *Software idioms [Coplien]*
- *Analysis patterns [Fowler]*
- *AntiPatterns [Brown]*
- *Frameworks*
- *STL (C++ Template Library)*
- *Algorithms and data structures*

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 ...

- *A pattern is a named nugget of instructive information that captures the essential structure and insight of a successful family of proven solutions to a recurring problem that arises within a certain context and system forces.*

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 on Design Patterns

- *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*

Patterns 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 Design Patters

- *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 Design Patters ...

- *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 Design Patters ...

- *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*