What is a pattern?

Aleksandra Tesanovic
Real-Time Systems Laboratory
Department of Computer and Information Science
Linköpings universitet
Sweden

September, 2001

Talk outline

• Pattern origins and history
• Definition
• Properties
• Types of patterns
• Describing design patterns
• Pattern language, system, catalog
• Reorganizing an object-oriented application using design patterns
Pattern origins and history

- writings of architect Christopher Alexander
  (coined this use of the term "pattern" ca. 1977-1979)
- Kent Beck and Ward Cunningham, Textronix, OOPSLA’87
  (used Alexander’s “pattern” ideas for Smalltalk GUI design)
- Gamma, Helm, Johnson, Vlissides ("Gang of Four" - GoF)
  Design Patterns: Elements of Reusable Object-Oriented Software, 1991-1994
- PLoP Conferences and books, 1994-present
- Buschmann, Meunier, Rohnert, Sommerland, Stal, Pattern - Oriented Software Architecture: A System of Patterns
  ("POSA book")

What is a pattern?

Definition

- ... a fully realized form, original, or model accepted or proposed for imitation...[dictionary]
- ... describes a problem which occurs over and over again in our environment, and then describes the core of the solution to that problem, in such a way that you can use this solution a million times over, without ever doing it the same way twice [Alexander]
- ... the abstraction from a concrete form which keeps recurring in specific non-arbitrary contexts [Riehle]
- ... both a thing and the instructions for making the thing [Coplien]
- ... a literary format for capturing the wisdom and experience of expert designers, and communicating it to novices
Patterns do...
- provide common vocabulary
- provide “shorthand” for effectively communicating complex principles
- help document software architecture
- capture essential parts of a design in compact form
- show more than one solution
- describe software abstractions

Patterns do not...
- provide an exact solution
- solve all design problems
- only apply for object-oriented design

Patterns can be
- non-generative (Gamma patterns)
  - observed in a system
  - descriptive and passive

- generative
  - generate systems or parts of systems
  - perspective and active
Ingredients

Pattern

Context
a design situation giving rise to a design problem

Problem

a set of forces occurring in that context

Solution

a form or rule that can be applied to resolve these forces

Example – window place

forces
- he wants to sit down and be comfortable
- he is drawn toward the light

solution
- in every room, make at least one window into a “window place”

Types of software patterns

- design patterns (software design) [Buschmann-POSA]
  - architectural (systems design)
  - design (micro-architectures) [Gamma-GoF]
  - idioms (low level)

- analysis patterns (recurring & reusable analysis models) [Flower]
- organization patterns (structure of organizations/projects)
- process patterns (software process design)
- domain-specific patterns
### Alexandrian form (canonical form)

- **Name**
  - meaningful name
- **Problem**
  - the statement of the problem
- **Context**
  - a situation giving rise to a problem
- **Forces**
  - a description of relevant forces and constraints
- **Solution**
  - proven solution to the problem
- **Examples**
  - sample applications of the pattern
- **Resulting context (force resolution)**
  - the state of the system after pattern has been applied

### Alexandrian form (canonical form)

- **Rationale**
  - explanation of steps or rules in the pattern
- **Related patterns**
  - static and dynamic relationship
- **Known use**
  - occurrence of the pattern and its application within existing system
<table>
<thead>
<tr>
<th>Pattern name and classification</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Intent</td>
<td>what does pattern do / when the solution works</td>
</tr>
<tr>
<td>Also known as</td>
<td>other known names of pattern (if any)</td>
</tr>
<tr>
<td>Motivation</td>
<td>the design problem / how class and object structures solve the problem</td>
</tr>
<tr>
<td>Applicability</td>
<td>situations where pattern can be applied</td>
</tr>
<tr>
<td>Structure</td>
<td>a graphical representation of classes in the pattern</td>
</tr>
<tr>
<td>Participants</td>
<td>the classes/objects participating and their responsibilities</td>
</tr>
<tr>
<td>Collaborations</td>
<td>of the participants to carry out responsibilities</td>
</tr>
</tbody>
</table>

| Consequences                    | trade-offs, concerns |
| Implementation                  | hints, techniques |
| Sample code                     | code fragment showing possible implementation |
| Known uses                      | patterns found in real systems |
| Related patterns                | closely related patterns |
What is a pattern?

Pattern language

[Coplien]

• ...is a structured collection of patterns that build on each other to transform needs and constraints into an architecture [Software Design Patterns: Common Questions and Answers]

• ...defines collection of patterns and rules to combine them into an architectural style...describe software frameworks or families of related systems [Patterns Home Page ->Patterns Definitions]
Pattern catalogs and systems

[Buschmann, POSA]

• pattern catalog
  ...a collection of related patterns, where patterns are subdivided into small number of broad categories...

• pattern system
  ...a cohesive set of related patterns, which work together to support the construction and evolution of hole architectures...

Design pattern catalog - GoF

<table>
<thead>
<tr>
<th>Purpose</th>
<th>Creational</th>
<th>Structural</th>
<th>Behavioral</th>
</tr>
</thead>
<tbody>
<tr>
<td>Creational</td>
<td>Factory Method</td>
<td>Adapter</td>
<td>Interperter</td>
</tr>
<tr>
<td>Structural</td>
<td>Chain of Responsibility</td>
<td>Command</td>
<td>Iterator</td>
</tr>
<tr>
<td>Behavioral</td>
<td>Mediator</td>
<td>Momento</td>
<td>Observer</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Scope</th>
<th>Object</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Class</td>
<td>Abstract Factory</td>
<td>Adapter</td>
<td></td>
</tr>
<tr>
<td>Class</td>
<td>Builder</td>
<td>Bridge</td>
<td>Command</td>
</tr>
<tr>
<td>Class</td>
<td>Prototype</td>
<td>Composite</td>
<td>Iterator</td>
</tr>
<tr>
<td>Class</td>
<td>Singleton</td>
<td>Decorator</td>
<td>Mediator</td>
</tr>
<tr>
<td>Scope</td>
<td>Object</td>
<td>Facade</td>
<td>Momento</td>
</tr>
<tr>
<td>Scope</td>
<td>Object</td>
<td>Flyweight</td>
<td>Observer</td>
</tr>
<tr>
<td>Scope</td>
<td>Object</td>
<td>Proxy</td>
<td>State</td>
</tr>
<tr>
<td>Scope</td>
<td>Object</td>
<td></td>
<td>Strategy</td>
</tr>
<tr>
<td>Scope</td>
<td>Object</td>
<td></td>
<td>Strategy</td>
</tr>
<tr>
<td>Scope</td>
<td>Object</td>
<td></td>
<td>Vistor</td>
</tr>
</tbody>
</table>
Reorganization using patterns

- Experiences using Design Patterns to Reorganize an Object-Oriented Application, Walter Zimmer
- hypermedia application
  - developed by the European Museum Network (EMN)
  - on top of MacApp
  - 50 classes
- goal of reorganization
  - eliminate deficiencies in the design and implementation

Steps in reorganization

PRELIMINARY STEPS
1 documentation
2 finding starting points
  - identification of classes / subsystems with design goals similar to ones of design patterns
  - experiences and future scenarios
  - metrics / design rules
  - analyzing the application for existing patterns

REORGANIZATIONAL STEPS
1 finding and exploring suitable design patterns
2 reconstructing and documentation
  - application classes corresponding to the design pattern
  - incorporate names of the application classes to the classes in the design pattern (e.g., LinkStrategy, HyperMediaDecorator)
Results and experiences

RESULTS
- drastic reduction of dependencies between subsystems
- short design documentation

EXPERIENCES
- common vocabulary - main advantage
- reorganization is time-intensive task
- good knowledge of design patterns needed
- combination of several design patterns required