Introduction to Patterns

Outline

- Patterns origins and history
- Definitions
- Properties
- Types of patterns
- Design patterns
- Pattern language, system, catalog, etc.
- Usage Example: Reorganizing an object-oriented application using design patterns

Patterns 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)
- **Erich Gamma, Ph. D. thesis, 1988-1991**
- **James Coplien, *Advanced C++ Idioms book*, 1989-1991**
- **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")**
- **PDEAFs – Pattern Driven EAFs (today)**

Definitions

- **… 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**

Properties

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** occuring 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**
- **etc.**

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

GoF format

Pattern name and classification

Intent
what does pattern do / when the solution works

Also known as
other known names of pattern (if any)

Motivation
the design problem / how class and object structures solve the problem

Applicability
situations where pattern can be applied

Structure
a graphical representation of classes in the pattern

Participants
the classes/objects participating and their responsibilities

Collaborations
of the participants to carry out responsibilities

GoF format

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

Pattern templates

[PATTERN-NAME]
Author
[YOUR-NAME] ([YOU@YOUR.ADDR]).
Last updated on [TODAY'S-DATE]

Context
[PARAG-1]
[PARAG-2]
Problem
[ONE-ASPECT]
[ANOTHER-ASPEC
Examples
Forces
1.[FORCE-1]
2.[FORCE-2]
Design
[PARAG-1]
[PARAG-2]
An Implementation
[SOME-CODE]
Examples
Variants
[VARIANT]
[ANOTHER-VARIANT]
See Also
[ANOTHER-REF]

```
IF   you find yourself in CONTEXT
                for example EXAMPLES,
                with PROBLEM,
                entailing FORCES
           THEN for some REASONS,
                apply DESIGN FORM AND/OR RULE
                to construct SOLUTION
                leading to NEW CONTEXT and OTHER PATTERNS
```

http://g.oswego.edu/dl/pd-FAQ/pd-FAQ.html

http://hillside.net/patterns/Writing/Lea.html

More pattern templates:

- http://hillside.net/patterns/template.html
- http://www.patterndepot.com/pages (Templates)

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…**

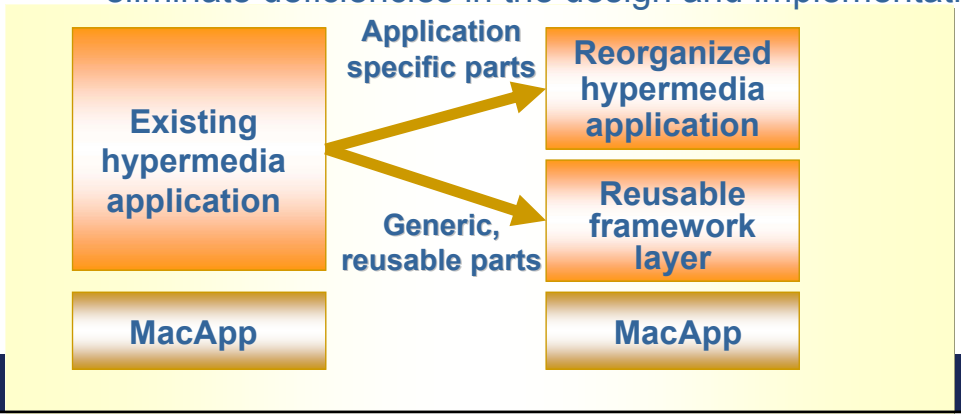
- **pattern system**
  - **…a cohesive set of related patterns, which work together to support the construction and evolution of hole architectures...**
  - **…e.g., pattern hierarchy**

Design pattern catalog - GoF

		Purpose		
		Creational	**Structural**	**Behavioral**
Scope	**Class**	• Factory Method	• Adapter	• Interperter
	Object	• Abstract Factory • Builder • Prototype • Singleton	• Adapter • Bridge • Composite • Decorator • Facade • Flyweight • Proxy	• Chain of Responsibility • Command • Iterator • Mediator • Momento • Observer • State • Strategy • Vistor

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 staring 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

} design patterns parts of the system

} critical application parts and their deficiencies

REOGANIZATIONAL 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