Patterns in a Nutshell

The “bare essentials” of Software Patterns

Brad Appleton  <bradapp@enteract.com>
http://www.enteract.com/~bradapp/
## 1.0 What are Patterns?

<table>
<thead>
<tr>
<th>Trendy:</th>
<th>Recent “hot topic”, OOD buzzword, lots of hype!</th>
</tr>
</thead>
<tbody>
<tr>
<td>Literary:</td>
<td>Form of software engineering problem-solving documentation</td>
</tr>
<tr>
<td>Pragmatic:</td>
<td>Describe practical solutions to “real world” problems</td>
</tr>
<tr>
<td>Recurring:</td>
<td>Identify good design structures which recur in practice</td>
</tr>
<tr>
<td>Generative:</td>
<td>Show how and when to apply the solution, and generate the desired design structure</td>
</tr>
<tr>
<td>Emergent:</td>
<td>Larger solutions emerge indirectly from applying patterns in succession, and in concert together</td>
</tr>
</tbody>
</table>
2.0 Pattern Origins and History

• Writings of architect Christopher Alexander (coined this use of the term “pattern” ca. 1977-1979)
• Documentation of best practices and handbooks for engineering and architecture
• Literate programming (Don Knuth), ca. 1984
• Kent Beck and Ward Cunningham, Tektronix, OOPSLA’87 (used Alexander’s “pattern” ideas for Smalltalk GUI design)
• PLoP Conferences and books, 1994-present
3.0 Pattern Definitions

A “pattern” is ... 

• An abstraction from a concrete form which keeps recurring in specific, non-arbitrary contexts. [generic definition]

• A recurring solution to a common problem in a given context and system of forces. [Alexander]

• A named “nugget” of instructive insight, conveying the essence of a proven solution to a recurring problem in a given context amidst competing concerns.

• A successfully recurring “best practice” that has proven itself in the “trenches”.

• A literary format for capturing the wisdom and experience of expert designers, and communicating it to novices
4.0 Kinds of Software Patterns

- Design Patterns (software design; often object-oriented):
  - architecture (systems design)
  - design (component interactions)
  - programming idioms (language-specific techniques/style)
- Analysis Patterns (recurring & reusable analysis models)
- Organization Patterns (structure of organizations/projects)
- Process Patterns (software process design)
- Domain-Specific: *Any other domain you can think of!*
5.0 Pattern Elements

• **Name**
  - a meaningful “conceptual handle” for discussion

• **Context**
  - tells *how the problem occurs* / *when the solution works*

• **Problem**
  - statement of the problem / *intent* of the solution

• **Forces**
  - trade-offs, goals+constraints, motivating factors/concerns
  - tells *why the problem is difficult*

• **Solution**
  - tells *how to generate* the solution
  - the solution structure, its participants & collaborations
6.0 Pattern Elements (cont.)

- **Examples** (optional)
- **Resulting Context**
  - describes the end result, benefits and consequences
  - shows how the forces were balanced/traded-off
  - tells *how the solution works out*
- **Rationale** (optional)
  - underlying principles/heuristics justifying the solution
  - tells underpinnings of *why the solution works out*
- **Related Patterns**
  - patterns which are similar, or may precede/follow this one
- **Known Uses**
  - 3 or more independent instances of “real world” success
7.0 Why Patterns?

Software Patterns help us because they:

- Solve "real world" problems
- Capture domain expertise
- Document design decisions and rationale
- Reuse wisdom and experience of master practitioners
- Convey expert insight to novices
- Form a shared vocabulary for problem-solving discussion
- Show *more* than just the solution:
  - context (when and where)
  - forces (trade-off alternatives, misfits, goals+constraints)
  - resolution (how and why the solution balances the forces)
8.0 Summary - What Patterns Are Not

Software Patterns are *not* ...

- Restricted to software design or Object-Oriented design
- Untested ideas/theories or new inventions
- Solutions that have worked only once
- Any old thing written-up in pattern format
- Abstract principles or heuristics
- Universally applicable for all contexts
- A “silver bullet” or panacea
9.0 Summary - What Patterns Are

Software Patterns are ...

- *Recurring* solutions to common problems of design
- *Practical/concrete* solutions to real world problems
- *Context* specific
- "*Best-fits*" for the given set of concerns/trade-offs
- "*Old hat*" to seasoned professionals and domain experts
- A *literary form* for documenting best practices
- A *shared vocabulary* for problem-solving discussions
- An effective means of (re)using, sharing, and building upon existing wisdom/experience/expertise
- *Massively overhyped!*

Brad Appleton  <bradapp@enteract.com>  http://www.enteract.com/~bradapp/docs/patterns-nutshell.html
10.0 Pattern Resources - Books

- **A Pattern Language: Towns, Buildings, Construction** *(APL)*  
  Christopher Alexander; Oxford University Press, 1977

- **The Timeless Way of Building** *(TTWoB)*  
  Christopher Alexander; Oxford University Press, 1979

- **Design Patterns: Elements of Reusable Object-Oriented Software** *(GoF)*  
  Gamma, Helm, Johnson, Vlissides; Addison-Wesley, 1994

- **Pattern-Oriented Software Architecture: A System of Patterns** *(POSA)*  
  Buschmann, Meunier, Rohnert, Sommerlad, Stal; Wiley and Sons, 1996

- **Pattern Languages of Program Design 1** *(PLoPD1)*  
  Coplien and Schmidt (editors); Addison-Wesley, 1995

- **Patterns of Software: Tales from the Software Community**  
  Richard Gabriel; Oxford University Press, 1996

- **Analysis Patterns: Reusable Object Models**  
  Martin Fowler; Addison-Wesley, 1996

- **Pattern Languages of Program Design 2** *(PLoPD2)*  
  Vlissides, Coplien, and Kerth (editors); Addison-Wesley, 1996
11.0 Pattern Resources - Online

- Patterns Home Page, http://www.hillside.net/patterns/
- Patterns Discussion FAQ, http://g.oswego.edu/dl/pd-FAQ/pd-FAQ.html
- Jim Coplien’s OrganizationPatterns Front Page (a WikiWikiWeb clone), http://www.www.bell-labs.com/cgi-user/OrgPatterns/OrgPatterns
- Patterns Mailing Lists, http://www.hillside.net/patterns/Lists.html
- Brad’s Patterns Intro: http://www.enteract.com/~bradapp/docs/patterns-intro.html
- Luke Hohmann’s Patterns Intro: http://members.aol.com/lhohmann/papers.htm
- Doug Lea’s OOD Patterns Intro: http://gee.cs.oswego.edu/dl/ca/ca/ca.html

Brad Appleton  <bradapp@enteract.com>  http://www.enteract.com/~bradapp/docs/patterns-nutshell.html
Oh who use candlestick patterns okay so the important characteristic patterns are dead in front of your screen candlestick patterns usually the Ford, as diversity partners, majority of domestic violence and it wasn't by which means the new when the existing pain is going to change and the new That's why the many famous worship leaders they get, they give you. the Jones for being not getting the early voted one vision for the training and getting to the train before it starts or you know just about to start so this is an important patterns which are delicious and major I'd like to A thorough guide to pattern matching can be found in the Nutshell handbook Mastering Regular Expressions by Jeffrey E. F. Friedl. 9.1. Filenames Versus Patterns. Metacharacters used in pattern matching are different from those used for filename expansion. When you issue a command on the command line, special characters are seen first by the shell, then by the program; therefore, unquoted metacharacters are interpreted by the shell for filename expansion. The command: $ grep [A-Z]* chap[12]. could, for example, be interpreted by the shell as: $ grep Array.c Bug.c Comp.c chap1 chap2. and grep th