AsianPLoP® Pattern Writing Bootcamp
National Taipei University of Technology in Taipei, Taiwan

Joseph (Joe) Yoder  joe@refactory.com
Rebecca Wirfs-Brock rebecca@wirfs-brock.com

Objectives

- At the end of this “bootcamp” you will:
  - Learn a little more about Patterns
  - Increase knowledge about pattern writing
  - Begin writing a new pattern

Copyright © 2011. The Hillside Group.
Introductions

Tell us your who you are and where you are from.

Have you been to a PLoP before?

Have you been to a bootcamp before?

What is your experience with writing Patterns?

Writing Patterns

“The Straight Scoop”

Good Judgment comes from Experience comes from bad Judgment….Patterns come from Experience

Capturing and communicating (sharing) our experiences is not always easy

While writing a pattern might not be difficult, writing good patterns can be very difficult

For most of us, any writing can be a difficult task, clear communication takes practice and iteration
Introduction to Patterns

The Reason for Patterns

• Experts at work often use their knowledge of how they solved similar problems to solve the current problem

• What if they could share their knowledge with others?

• Pattern descriptions share experts’ wisdom in ways others can use
An Example of a Pattern*

Window place

Christopher Alexander et al,
* A Pattern Language-
Towns, Building,
Construction
1977

The Window Place Pattern

Everybody loves window seats, bay windows, and big windows with low sills and comfortable chairs drawn up to them…A room without a place like this seldom allows you to feel comfortable
If the room contains no window which is a “place”,
a person will be drawn between:
1. Wanting to sit down and be comfortable
2. Being drawn toward the light
If comfortable places are away from the windows,
there is no way of overcoming this conflict…
Therefore: In every room where you spend any length of time during the day, make at least one window into a “window place”
Part 1: Pattern Basics

Patterns

“The pattern is, in short, at the same time a thing, which happens in the world, and the rule which tells us how to create that thing, and when we must create it. It is both a process and a thing; both a description of a thing which is alive, and a description of the process which will generate that thing”

—Christopher Alexander,
The Timeless Way of Building
**What is a Pattern?**

It is a solution to a problem in a context
It tells a problem solver what to do, how to solve the problem
It contributes to human comfort or quality of life
It is something that cannot be formalized or automated (if it can, do that instead of writing a pattern)

---

**A Pattern Should…**

- Provide facts (e.g. be a reference manual) about solving a problem
- Tells a problem solver what to do to solve the problem
- Describe a mature, proven solution (Rule of 3)
- Describe a solution built on the insight of the problem solver, that can be implemented many different ways
- Help you comprehend existing systems … and also build new ones
A Pattern is not

Just a simple solution:
• A simple rule
• A prescriptive recipe
• A procedure to always follow
• An algorithm
• A data structure

Patterns Will Not…

• … make you an instant expert
• … provide a “turn the crank” approach to software
• … eliminate the need for intelligence and taste
• … make you rich and famous
—Paul S. R. Chisholm, AT&T 10/94

• eliminate the need to think
• tell you how exactly how to adapt the solution to your context
Part 2: Pattern Forms

Pattern Styles

- There isn’t one “right” way to write a pattern
- Patterns can be written in various forms, depending on your audience and the pattern’s topic
- Major forms:
  - Christopher Alexander style
  - Software Pattern styles:
    - GOF (Design Patterns) style, POSA style, software pattern paper styles
  - Narrative styles (good for non-software patterns)
    - Story-based variation of Alexander’s style, Mary Lynn Manns and Linda Rising, Fearless Change
    - Pattern 3.0 format, Takashi Iba
Small Meeting Rooms

The larger meetings are, the less people get out of them. But institutions often put their money and attention into large meeting rooms and lecture halls.

It has been shown that the number of people in a group influences both the number who never talk, and the number who feel they have ideas which they have not been able to express.

There is no particularly natural threshold for group size; but it is clear that the number who never talk climbs very rapidly with group size. In a group of 12, one person never talks. In a group of 24, there are six people who never talk.

Make at least 70% of all meeting rooms really small -- for 12 people or less. Locate them in the most public parts of the building, evenly scattered among the workplaces.

See: Light on Two Sides of Every Room, Sitting Circle, Different Chairs, Pools of Light, The Shape of Indoor Space.

February 24, 2016 - 18
Parts of a Pattern (Alexander Form)

- Problem - when to use the pattern
- Solution - what to do to solve problem
- Context - where the pattern might apply
- Forces - considerations to balance
- Consequences - positive and negative effects of applying the pattern
- Examples – an application of the pattern
  - Teaches both problem and solution
  - Are proof of pattern-hood

Related patterns

Software Design Patterns

“Design Patterns are descriptions of communicating objects and classes that are customized to solve a general design problem in a particular context”

—Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides
Adapter

Intent
- Convert the interface of a class into another interface clients expect. Adapter lets classes work together that couldn't otherwise because of incompatible interfaces
- Use Interfaces in Java and C# to get the equivalence look of multiple inheritance

Parts of a Pattern in Design Patterns
(Also known as GOF for Gang of Four; Gamma et. al.)

Intent - brief description of problem and solution
Also Known As – aliases or other names
Motivation - prototypical example
Applicability - problem, forces, context
Structure/Participants/Collaborations - solution
Consequences - forces
Implementation/Sample Code - solution
Known Uses
Related Patterns
**Patterns-Oriented Software Architecture (POSA) Pattern Form**

<table>
<thead>
<tr>
<th>Name,Aliases</th>
<th>Implementation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Brief Description</td>
<td>Example Resolved</td>
</tr>
<tr>
<td>Example</td>
<td>Variants</td>
</tr>
<tr>
<td>Context</td>
<td>Known Uses</td>
</tr>
<tr>
<td>Problem</td>
<td>Consequences</td>
</tr>
<tr>
<td>Forces</td>
<td>Related Patterns</td>
</tr>
<tr>
<td>Solution</td>
<td>Credits</td>
</tr>
<tr>
<td>Structure</td>
<td></td>
</tr>
<tr>
<td>Dynamics</td>
<td></td>
</tr>
</tbody>
</table>

**POSA Example: Pipes and Filters**

**Brief Description:**

“The Pipes and Filters architectural pattern provides a structure for systems that process a stream of data. Each processing step is encapsulated in a filter component. Data is passed through pipes between adjacent filters. Recompiling filters allows you to build families of related systems.”
### POSA Example Pipes: and Filters Structure

<table>
<thead>
<tr>
<th>Filter</th>
<th>Pipe</th>
<th>Data Source</th>
<th>Data Sink</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Gets input data</td>
<td>• Transfers data</td>
<td>• Delivers input to processing pipeline</td>
<td>• Consumes output</td>
</tr>
<tr>
<td>• Performs a function on the data</td>
<td>• Buffers data</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• Supplies output data</td>
<td>• Synchronizes active neighbors</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### POSA Example: Pipes and Filters Dynamics

![Diagram showing data flow through pipes and filters](image)
Narrative Pattern Form Lead-in Story: Fearless Change

Ask for Help

Markita Andrews has generated more than $80,000 selling Girl Scout cookies since she was seven years old. She does not propose to be smarter or more extroverted than other people. Rather, she claims the difference is that she has discovered the secret of selling: Ask, Ask, Ask! The fear of rejection causes many people to fail before they begin because they don’t just ask for what they want.

Since the task of introducing a new idea into an organization is a big job, look for people and resources to help your efforts.

…

Narrative Pattern Form Ending Stories

Ask for Help

Don’t be discouraged if the help is slow in coming. Even a small start can help you promote your ideas, leading to more resources in the future.

…

Writing computer programs in pairs is part of a new agile software development approach. Programmers say that pairing makes it easier to admit they don’t know something. In the pair programming relationship, individuals lose the embarrassment that typifies the lone cowboy coder who would rather try to muddle through on his own. Asking for help has become a natural part of the software development process.
Pattern 3.0 Style: Focus on Human Action

“Note that it was a conscious choice not to show item names, such as “Problem” and “Solution”, in the format because of readability for students. Our format is midst between Alexander’s patterns and the design patterns.” - Takashi Iba and Toko Miyake, AsianPloP 2010

Contrasting Poetic Styles

- **SONNET** a poem of fourteen lines, usually in iambic pentameter that has one of two regular rhyme schemes

From fairest creatures we desire increase,  
That thereby beauty's rose might never die.  
But as the riper should by time decease,  
His tender heir might bear his memory:  
But thou, contracted to thine own bright eyes,  
Feed'st thy light's flame with self-substantial fuel,  
Making a famine where abundance lies,  
Thyself thy foe, to thy sweet self too cruel.  
Thou that art now the world's fresh ornament  
And only herald to the gaudy spring,  
Within thine own bud buriest thy content  
And, tender churl, mak'st waste in niggarding.  
Pity the world, or else this glutton be,  
To eat the world's due, by the grave and thee.  

-William Shakespeare
Contrasting Poetic Styles

• Haiku - three lines, with the first and last line having 5 moras, and the middle line having 7. A mora is a sound unit, much like a syllable, but is not identical to it. Since the moras do not translate well into English, in English it has been adapted and syllables are used as moras.

   An old silent pond...
   A frog jumps into the pond,
   splash! Silence again.

   - Basho Matsuo

Contrasting Poetic Styles

• Limerick a five-line joke of a poem — witty, usually involving place names and puns, and most often bawdy, sometimes unprintable.

   There was a small boy of Quebec
   Who was buried in snow to his neck
   When they said, “Are you friz?”
   He replied, “Yes, I is —
   But we don't call this cold in Quebec”

   - Rudyard Kipling
Pattern Styles and Forms

- Alexandrian style: Examples, simple writing, explicit connections to other patterns in the language
- Classic Software Patterns: Lots of elements, highly stylized (sonnet-like in their formality)
- Pattern 3.0 Iba-style: High level, essence, with drawings essential
- Fearless Change Patterns: Story telling (not exactly a limerick, but stories are essential)
- Joe-Rebecca Style: Mix of Alexandrian and Iba with good story telling

Qualify the Roadmap

“All you need is the plan, the roadmap, and the courage to press on to your destination”
— Earl Nightingale

- As systems qualities are a key factor in the success of any product, how can agile teams include these qualities as part of the roadmap and overall timeline?
- While developing and evolving the product feature roadmap, also plan for when system qualities should be addressed. Be sure to “Plan for Responsible Moments” to know when important system qualities should be implemented
Quality the Roadmap

Many agile teams include a product roadmap as part of their planning. The roadmap typically shows a rough plan for delivering features over time. This plan is useful for sharing a common understanding of the teams involved in the project and to help communicate stakeholders’ expectations and overall project plans and goals across the organization. The roadmap includes a timeline with expected milestones and targets for when key features are desired.

As system quality is a key factor in the success of any product, how can agile teams include three qualities as part of the roadmap and overall timeline?

Features that are delivered to end users are tangible and of obvious value to users, so they are easy to focus on and include in a roadmap. While the delivery of features may also depend on system qualities, it can be unseen how they are related and thus system qualities and architectural capabilities are often not included in the roadmap.

Asymmetrical design involves making tradeoffs between implementing functionality that is good enough to meet the important business requirements while adequately addressing system qualities. Sometimes when making design tradeoffs, there is a temptation to overdesign or put into too many details about technical qualities. On the other hand, trying to address important system qualities after basic functionality has been implemented can result in major roadblocks. While a more appropriate time to address these qualities would cause less disruption.

Therefore, while developing and evolving the product feature roadmap, also plan for when system qualities and the architectural features to support them should be addressed. Be sure to Plan for Responsible Abstraction to know when important system qualities should be implemented.

Part 3: Pattern Parts

The product roadmap is a high-level view showing topics about how the product is likely to grow across several major releases. Typically, a product roadmap consists of high-level themes which are implemented by many small stories. Other processing and fulfillment, for example, may be a high-level feature that is expected to be delivered, and then it is represented on the roadmap. To implement other processes you may need to develop new or revised internal, external web services, implement security processes, and access a transactional database. These might be represented by technical classes on your roadmap. However, if performance or security are also important qualities, you may also want to add specific items for these concerns to your roadmap.

Quality-related roadmap items should either be placed just before or along with any functionality that depends on them. Now, the story moves contrary to that well-known agile mantra. “Make it work, make it right, make it fast.” However, if you have a side architecture feature, you might want to work out these features 1st before implementing functionality that depends on it. Is it similar to using a plug-in solution and then releasing that solution?

During planning, corresponding system quality-related items should be added to your product backlog or Trello list. Adding quality-related work to your backlog helps to more clearly identify where certain performance targets or security measures are expected and helps you understand priorities of quality-related items when you Qualify the Backlog.

Product roadmaps include a timeline for when major features are desired. Sometimes they can also include architectural features to achieve desired system qualities. Alternatively, teams may create a separate technology roadmap that outlines the expected delivery of architecture components and technology. Regardless of whether you have a separate technology roadmap or identify architecture features on your product roadmap, it is important to make visible when important system qualities should be considered and worked on. These are additional ways to make these qualities more visible such as Qualify the Backlog, Qualify Classes, and improve Quality Fabricator.

If you don’t make the delivery of system qualities explicit, then they might not be recognized as being needed. Waiting too long to implement certain system qualities can cause significant setbacks for the architecture. Critical qualities are often difficult to add into an existing or new project. Teams that do not properly consider these qualities can ensure to limit technical risks and increase the chances of timely completing your project.

These system qualities directly contribute in meeting your definition of done.

To randomly consider and specify necessary system qualities in product roadmaps, standards for software and system quality models such as DO-263A (34131) [DO-263A] can be considered. They classify typical system characteristics and provide an extensive framework for systemically considering quality concerns. Agile teams might focus on a few important system qualities, such as reliability and security, as important at the beginning of the project. When considering additional qualities, such as robustness and maintainability, you might be forced to make tradeoffs between revising your design to support qualities originally considered or adjusting your expectations. Although it is not necessary to specify and define requirements for all qualities from the very beginning, it is important to define essential qualities and identify where on the product roadmap they are expected to be delivered.
Name

Word or short phrase—the essence of the pattern, some say noun phrase
Naming is very important and communicates
Good names enhance communication—especially when you can guess the intent from the name
Patterns “build” something, the name should say what the pattern builds

ﱱ Iterator, Adaptor, Do Food, Inspiring Illustration …

Context

The setting—target user, patterns applied, size, scope, timing, memory constraints, anything that might invalidate the solution if changed

³ You’re an Evangelist or Dedicated Champion who has called a meeting to introduce a new idea. Members of the user community are free to attend or not. You have resources, your own personal contribution or those of a Local Sponsor or Corporate Angel.
Problem

Short, complete statement of the problem the pattern will solve

- Usually a meeting is just another ordinary, impersonal event
- How do we get people to want to attend our meeting?

Forces

Why the problem is hard …

The forces are often contradictory—create tension:
- You want to make your customers happy
- You have limited resources

- There’s always more important work to do
- Most people are curious about new ideas
Solution

Your proposed method of solving the problem. Resolve important forces determined by context; other forces may be ignored
Keep the target audience in mind
Best Patterns are Generative

- Have food at the meeting—donuts or bagels in the morning, with coffee, tea, and juice, and cookies and drinks in the afternoon, lunch at noon-time

Resulting Context

What happens if the solution is applied, what forces resolved, what problems may arise, what costs and benefits

Just “problem solved” is not enough

- Food will turn a mundane meeting, presentation or other gathering into a more special event. If offered in the beginning, it starts the meeting on a positive note
Other Optional Sections

**Known Uses:** A one-time occurrence is an event. A double occurrence is a coincidence. If it occurs more than twice, it’s a pattern...

Jim Coplien /Gerald Weinberg/Bunny Duhl

**Related Patterns:** Use, be used with, be similar to other patterns

– **While the prospect of free food is nice,** *Brown Bag* can be used when funding is not available. People can still eat together, even if they bring their own food

Pattern Mining

• You are interested in creating a pattern language in which you have little experience

The patterns extracted from people who have less experience can often be weak or wrong. People that are experienced in the domain are often busy and don’t understand patterns.

▼ Therefore

Conduct an interview with experienced people in the area of the pattern language you are creating. Use this information to gather the core information needed to write the patterns.
Pattern Mining

- **Keys Worth Sharing** – Gather Important Key Points
- **Problem Digging** – Get Details of the Patterns
- **Context Caching** – When to consider the patterns
- **Keys Worth Sharing** – Gather Important Key Points
- **Venture of Asking** – Ask even if you know the answer
- **Empathetic Response** – Understand the expert
- **Experience Overlapping** – Share similar experiences
- **Interesting Points** – Outline the main points you learned during the interview to use for the patterns

Ward’s Tips for Mining and Writing Patterns

- Pick a whole area, not just one idea
- Make a list of things you learned
- Cast each item on your list as a solution
- Now write each item as a Pattern
  - Try a four paragraph form where the second paragraph ends with the pivotal “therefore”
- Organize your patterns into sections
- Write an Introduction
Our Tips for Mining and Writing Patterns

• Pick a complete area
• Take a small 3x5 or equivalent card
• Write the problem and solution as simple sentences
• Outline the tradeoffs/forces
• Add any other ideas such as related patterns, resulting context, etc.
• Give it a candidate name that reflects the solution
• Use this to start writing more detail form of the pattern
• Choose the form and only add sections you need

Let’s Write a Pattern!

Name, Aliases
Context
Forces
Problem
Solution
Resulting Context (Consequences)
Rationale
Related Patterns
Known Uses
Sketch
Author
References
Examples
Shu – Ha - Ri

- Shu: First time pattern writers
- Ha: Pattern authors who want to focus on improving a part of their patterns
- Ri: Help us teach this workshop

守破離

Shu  Ha  Ri

Improving Your Writing

- Get people read to your work and give you comments
- Don’t wait until it is finished to do so
- Find trusted reviewers and friends to help, too
- Write clearly:
  - Shorten long sentences.
  - Eliminate extra words
  - Summarize main points
- Practice. Revise. Get help to improve
  - writing skills
  - understandable problem and solution
  - clear tradeoffs
Focus on Your Reader

• Know your intended audience
  – what can you assume that they already know?
  – what details do they need explained in detail?
  – how much detail do you need to include?
  – what form best keeps their interest?

• Keep your intended readers in mind

• Explain who your intended audience is in your paper and to your reviewers and shepherd

Collections vs. Languages

• Pattern collection: a group of patterns on a related topic.
  – no claim of completeness
  – patterns from a collection can be used in isolation

• Pattern language: a group of interconnected patterns that fit together and fully address a topic.
  – may include higher level patterns that guide you to more specific, lower-level patterns
  – discusses all the things you need to consider to build something
  – you are likely to use several patterns in a sequence from a language to solve your problem

• Both have value, languages more complete
Conclusions

• Good judgment comes from experience comes from bad judgment … Patterns come from experience
• Good writing is not an accident—it comes from dedication, focus, and practice … http://www.dreamsongspress.com/
• Writing patterns takes patience and iteration
• Just do it! Just thinking or theorizing will not help you write that pattern
• Ultimately Pattern Languages or catalogs are more powerful than individual patterns
• Get regular feedback from the patterns community http://www.hillside.net

Selected Bibliography

Design Patterns - Elements of Reusable Object-Oriented Software. Gamma, Helm, Johnson and Vlissides (The “Gang of Four”). Reading, MA: Addison-Wesley, 1995.

PLoP Conference Proceedings: http://hillside.net/plop/

For more books: http://hillside.net/patterns/books/index.htm