Sample Peer Reviewing Process

Authors

Submit the paper (18 September)

Rebuttal (13–20 November)

Submit camera-ready version (15 January)

Present at the conference (26 April)

Conference

External researchers provide anonymous reviews (by late October)

Meta reviewer summarizes the reviews, adds own opinion (early November)

Program committee (PC) meeting (early December)

Criteria for a Good Paper

• **Contribution:** What new insights does it bring to the field?

• **Benefits:** What can you learn from this / do with this?

• **Novelty:** Prior publications?

• **Validity:** Are the claims properly backed up?

• **Applicability:** How good does the paper match the likely audience?

• **Format:** Readability and clarity

Structure of a Review

• Overall rating: 1: definite reject – 5: definite accept

• Short summary of the contributions and benefits
  • “This paper presents… (who) will benefit from (what)

• Concerns
  • Originality
  • Validity
  • Clarity

• Suggestions for improvement

• Reviewer’s expertise: 1: no knowledge – 4 expert
Reviewing Checklist

- **Recommending accept**
  - Convince yourself that it has no serious defects
  - Convince the editor that it is of an acceptable standard, by explaining why it is original, valid, and clear
  - List the changes that should be made before it appears in print
    - Where possible: indicating not just what to change but what to change it to
  - Take reasonable care in checking details, e.g., mathematics, formulas, and bibliography

- **Recommending reject**
  - Clearly explain the faults and, where possible, discuss how they could be rectified
  - Indicate which parts of the work are of value and which should be discarded
  - Check the paper to a reasonable level of detail

---

From *Writing for Computer Science* (Zobel, 2004)

---

**HCI Design Patterns**
Interdisciplinary Design

In-Class Exercise

You are a software developer working on a new software project. List all other disciplines/professions/stakeholders that you think you will need to involve as part of your team.

Problem: Interdisciplinary Design

What’s a Design Pattern?

- A design pattern describes a successful solution to a recurring contextualized design problem in a consistent format that is readable by non-experts and networked into a language.

Tratato I
Francesco di Giorgio
Renaissance Master Builder
1480
A New Literary Form

Poem Encyclopedia

Pattern Newspaper

Novel Letter

Urban architecture

253 patterns 1977

Patterns of Events and Space

"A building or town is given its character, essentially, by those events that keep on happening there most often."

1979

The Timeless Way of Building

Patterns idea and process

1979
Patterns of Events and Space

- QWAN
- Inhabitants create better environments
- Participatory design!

Pattern Languages

Patterns Balance Forces

- Patterns solve a problem of conflicting forces
- Example: WINDOW PLACE (psychological)
  - People naturally drawn towards light
  - But like to sit
- Forces can be social, economic, natural, or physical
Designing with Patterns

Design is unfolding

Piecemeal Growth

** Conceptual diagram indicating the design process of unfolding patterns.

** References

- CTHCI — Jan Borchers

** Examples

- Kent Beck (Apple), Ward Cunningham (Tektronix)
- Problem: E-R does not work for OOP
- End-user programming: Alexander
- Guiding designer
- 5 Smalltalk window design patterns (GUI!)
- Example: COLLECT LOW-LEVEL PROTOCOL
- Successful experiment with non-Smalltalk-programmers
- Started software design patterns

** References

- OOPSLA '87: The Smalltalk Experiment
- Kent Beck (Apple), Ward Cunningham (Tektronix)
- Problem: E-R does not work for OOP
- End-user programming: Alexander
- Guiding designer
- 5 Smalltalk window design patterns (GUI!)
- Example: COLLECT LOW-LEVEL PROTOCOL
- Successful experiment with non-Smalltalk-programmers
- Started software design patterns
The Gang Of Four Book

- Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides: Design Patterns (1995)
- 23 patterns for software engineering
  - Creational, structural, behavioral
- Famous: Singleton, AbstractFactory, Adapter, Façade
- Each pattern ~10 book pages of text

AbstractFactory Pattern: WidgetFactory Example

AbstractFactory Pattern: The General Solution

(Notation Cheat Sheet: See Gamma book, back cover)
GoF Book: Evaluation

- Highly successful among developers
  - Great for expert communication
  - Instead of reading code
- Not complete language
  - Workarounds instead of good design?
- Not readable by non-developers
  - 50% implementation details
  - Not empowering users
  - Language, intent, audience, values?
- The "Trial"
  - OOPSLA 1999

PLoP Conferences

- PLoP Conference Series
  - Special format: non-academic, shepherding, proceedings
  - Strangely omits HCI area for a long time
  - PLoP 1998: "Have we exhausted this [HCI] field?"
- The OOPSLA’96 keynote by Alexander

The OOPSLA’96 keynote by Alexander

- Annual ACM Conference on Object-Oriented Programming, Systems, Languages, and Applications
- Had been the location of patterns “birth” 9 years before
- Alexander was invited to comment on the efforts of the SW community in creating patterns, such as the GoF book and others
- His remarks were quite devastating, but also very helpful to understand his ideas…

The Origins of Pattern Theory
the Future of the Theory, and The Generation of a Living World

The Origins of Pattern Theory
the Future of the Theory,
and The Generation of a Living World

Transcript available at
http://www.patternlanguage.com/archive/ieee/ieee-text.htm

Mismatched Adoption

Architecture

Structural Engineering

HCI

Software Engineering

“User Experience”

“Technical Quality”

- Mitch Kapor's 1990 "Software Design Manifesto"

Patterns in HCI

(X)PLML,...
Evaluating Patterns

- Shepherding
  - Experienced pattern author provides feedback
  - Usually part of the paper submission process

- Writers’ Workshops
  - Immensely valuable experience for the author
  - Feedback as in a very thorough review of a paper, thesis, exam…
  - Plus, you get to listen to the review process
  - Often reveals that others have totally different views than yourself about your work and topic
  - Tip: Use this format also in other situations

Writers’ Workshops

- Originally invented for poets’ meetings
- Adopted by Richard Gabriel for the software patterns community
- Designed to respect the author and to create a relaxed, positive and friendly atmosphere
- Welcome, reading, positive first, constructive, sandwich, applause, unrelated story

Richard Gabriel
Writers’ Workshops

1. Everybody reads pattern before workshop
2. Welcome
3. Read part of work to remind of author
4. Author: Fly on the wall
5. Summary
6. Things to keep (form, content)
7. Suggestions for improvement (form, content)
8. Sandwich: Summarize positive points

9. Welcome author back
10. Author asks clarifying questions (no defending)
11. Applaud the author
12. Unrelated story =)

(See ChiliPLoP’99 HCI patterns workshop report for details.)

First book that brought design patterns to HCI

QWAN
HCI is heir
lingua franca
Corp. Memory
Values
hcipatterns.org
v. Duyne et al., 2006 (2nd ed.)
Successful book on HCI Design Patterns for web sites

Accompanying web site: designofsites.com (flash cards)

• Pawan Vora, 2009
• 100 patterns

Jenifer Tidwell, 2005
Developed from “Common Ground” Pattern Language (1997)
http://www.mit.edu/~jtidwell/common_ground.html
In part available at designinginterfaces.com
More Current Trends

- Staffan Bjork, Jussi Holopainen, 2005
- 300 patterns
- Instantiates – Modulates – May Conflict

Dearden & Finlay,
Human Computer Interaction 21(1), 2006

Interactions 1/2007

CHI 2009 XPLML


Chapter 10: Jan Borchers,
The Aachen Media Space: Design Patterns for Augmented Work Environments
Collocated Group Services

Using Patterns in the Application Domain

Nielsen’s Usability Engineering Lifecycle

- Described in detail in: Jakob Nielsen, Usability Engineering, Morgan Kaufmann 1993
- Nielsen is an often-cited usability expert, especially for the web
- His web site useit.com offers current, interesting articles on usability, including his regular Alertbox column

Nielsen’s Usability Engineering Lifecycle

- A software lifecycle model geared towards interactive systems
- Not all stages must be completed for a useful product, but they are recommended
- Not a strict step-after-step waterfall model; some “stages” are more like recommendations, overlapping others
Nielsen’s Usability Engineering Lifecycle

1. Know the User
2. Competitive Analysis
3. Setting Usability Goals
4. Parallel Design
5. Participatory Design
6. Coordinated Design
7. Design Guidelines & Heuristic Analysis
8. Prototyping
9. Empirical Testing
10. Iterative Design
11. Feedback from Field Use

Stages and Pattern Use

1. Know the User
   • Understand individual user characteristics of your target group and their tasks, then derive functional needs of your system
   • Create application domain pattern language during the task analysis
   • Not perfect patterns, but “work patterns”
   • Simplifies communication

2. Competitive Analysis
   • Study other products to find different solutions and compare usability
   • Generalize observations as HCI design patterns

3. Setting Usability Goals
   • Weigh and prioritize different usability aspects (e.g., simplicity vs. efficiency)
   • Use HCI design pattern forces to model design tradeoffs

4. Parallel Design
   • Have multiple teams develop divergent initial solutions to explore the design space better
   • Use high-level HCI design patterns as guidelines

5. Participatory Design
   • Involve users / application domain experts throughout the design process
   • Use the interdisciplinary vocabulary function of application and HCI design pattern languages
Stages and Pattern Use

6. Coordinated Design
   - Ensure consistent design of total UI, including help, documentation, earlier versions, and your other products
   - Low-level HCI design patterns support consistency

7. Apply Guidelines and Heuristic Analysis
   - Use style guides, guidelines, standards
   - Pattern languages can serve as “better guidelines” and corporate memory

8. Prototyping
   - Create limited prototypes (see DIS 1)
   - Software design patterns can help relating developer concepts and concerns to HCI team

9. Empirical Testing
   - Test all prototypes with or without users
   - Use application domain patterns for test scenarios
   - Relate usability problems to HCI design patterns

10. Iterative Design
    - As in DIS 1
    - HCI and software design patterns help because they are constructive
    - All languages will evolve, using “known” project examples
    - Capture the structural design rationale
    - (Patterns and anti-patterns for process rationale)

11. Collect Feedback from Field Use
    - After delivery: field tests, followup studies, helpline call analysis…
    - Application domain language as common language
    - HCI pattern language points designers to alternative solutions
    - Also strengthen / rethink patterns as result
PLML 1.0

• Early formalization: DAG, nodes = patterns

• PLML: Pattern Language Markup Language

• Goals:
  • Specify pattern language structure
  • Do not limit authors to specific pattern formats
  • Facilitate authoring and browsing tool support

• Formulated as XML DTD at CHI 2003 Workshop

PLML 1.0: Use

• Applied to several pattern languages, including Interactive Exhibits

• Recommended format for pattern submissions at CHI 2004 workshop

• Common data format for emerging tool support

Important Characteristics of a Pattern

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1. A pattern implies an artefact</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
</tr>
<tr>
<td>2. A pattern bridges many levels of abstraction</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
</tr>
<tr>
<td>3. A pattern includes its rationale</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
</tr>
<tr>
<td>4. A pattern is manifest in a solution</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
</tr>
<tr>
<td>5. A pattern captures system hot spots</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
</tr>
<tr>
<td>6. A pattern is part of a language</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
</tr>
<tr>
<td>7. A pattern is validated by use</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
</tr>
<tr>
<td>8. A pattern is grounded in a domain</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
</tr>
<tr>
<td>9. A pattern captures a big idea</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
</tr>
<tr>
<td>10. Patterns support a ‘lingua franca’</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
</tr>
<tr>
<td>11. Different patterns deal with problems at different ‘scales’</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
</tr>
<tr>
<td>12. Patterns reflect design values</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
</tr>
<tr>
<td>13. Patterns capture design practice</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
</tr>
</tbody>
</table>

• = direct statement  ? = implicit agreement

Pattern Languages in HCI: A Critical Review

• In: Human-Computer Interaction Journal, 2006

• by Andy Dearden and Janet Finlay
Patterns Compared to Other Guidance

- Style guides (Mac Human Interface Guideline, GNOME project, Microsoft)
- General guidelines applicable to a range of systems
- Standards (ISO 9241)
- Claims
- Heuristics

Languages vs. Catalogue

- Gamma describe their efforts as a catalogue
- Organization is not predictive
- Notions of generativity

Usage of Patterns

- Participatory Design
- Technical Lexicon
- Organizational Memory
- Lingua Franca
- Design Rationale

Values and Pattern Languages

- Properties examined: QWAN - holistic and experiential ‘quality’
- Values in the selection of patterns
- Values in the process of developing patterns
- Values in the process of using patterns
Future Issues

- Exploring Pattern Languages in Use
- Improving the production of pattern languages
  - Writers workshops (lots of cons)
  - Encouraging more collaboration

Next Steps

- CTHCI: How to do HCI research
- DIS2, MCP, PDUI
- Hiwi jobs helping with HCI research and development (literature research, prototyping, coding, user studies,...)
- Bachelor/Master theses

Summary

- HCI Design Patterns capture the essence of successful solutions to recurring problems in user interface design
- Architecture — software engineering — HCI
- Name, ranking: vocabulary
- Context, references: language network
- Problem (forces), solution: summary
- Sensitizing example, examples, diagram: grounding
- A literary form
- Writers’ workshops
- Middle ground between Golden Rules and Style Guides
- Now in standard HCI books (Shneiderman, Dix), many languages published
- Benefit today: lingua franca throughout design process