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

User ↔ MAOCE ↔ Developer

Communication

interdisciplinary methods

values respect
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
The Timeless Way of Building

Christopher Alexander

Patterns idea and process

1979
Patterns of Events and Space

“A building or town is given its character, essentially, by those events that keep on happening there most often.”
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
... if all is well, the outdoor areas are largely made up of positive spaces—positive outdoor spaces (106); in some fashion you have marked boundaries between gardens and streets, between terraces and gardens, between outdoor rooms and terraces, between play areas and gardens—green streets (51), pedestrian street (100), half-hidden garden (111), hierarchy of open space (114), path shape (121), activity pockets (124), private terrace on the street (140), outdoor room (163), opening to the street (165), gallery surround (166), garden growing wild (172). With this pattern, you can help these natural boundaries take on their proper character, by building walls, just low enough to sit on, and high enough to mark the boundaries.

If you have also marked the places where it makes sense to build seats—seat spots (241), front door bench (242)—you can kill two birds with one stone by using the walls as seats which help enclose the outdoor space wherever its positive character is weakest.

In many places walls and fences between outdoor spaces are too high; but no boundary at all does injustice to the subtlety of the divisions between the spaces.

Consider, for example, a garden on a quiet street. At least somewhere along the edge between the two there is a need for a seam, a place which unites the two, but does so without breaking down the fact that they are separate places. If there is a high wall or a hedge, then the people in the garden have no way of being connected to the street; the people in the street have no way of being connected to the garden. But if there is no barrier at all—then the division between the two is hard to maintain. Stray dogs can wander in and out at will; it is even uncomfortable to sit in the garden, because it is essentially like sitting in the street.
The problem can only be solved by a kind of barrier which functions as a barrier which separates, and as a seam which joins, at the same time.

A low wall or balustrade, just at the right height for sitting, is perfect. It creates a barrier which separates. But because it invites people to sit on it—invites them to sit first with their legs on one side, then with their legs on top, then to swivel round still further to the other side, or to sit astride it—it also functions as a seam, which makes a positive connection between the two places.

Examples: A low wall with the children’s sandbox on one side, circulation path on the other; low wall at the front of the garden, connecting the house to the public path; a sitting wall that is a retaining wall, with plants on one side, where people can sit close to the flowers and eat their lunch.

Ruskin describes a sitting wall he experienced:

Last summer I was lodging for a little while in a cottage in the country, and in front of my low window there were, first, some beds of daisies, then a row of gooseberry and currant bushes, and then a low wall about three feet above the ground, covered with stonecress. Outside, a corn-field, with its green ears glistening in the sun, and a field path through it, just past the garden gate. From my window I could see every peasant of the village who passed that way, with basket on arm for market, or spade on shoulder for field. When I was inclined for society, I could lean over my wall, and talk to anybody; when I was inclined for science, I could botanize all along the top of my wall—there were four species of stonecress alone growing on it; and when I was inclined for exercise, I could jump over my wall, backwards and forwards. That’s the sort of fence to have in a Christian country; not a thing which you can’t walk inside of without making yourself look like a wild beast, nor look at out of your window in the morning without expecting to see somebody impaled upon it in the night. (John Ruskin, The Two Paths, New York: Everyman’s Library, 1907, p. 203.)

Therefore:

Surround any natural outdoor area, and make minor boundaries between outdoor areas with low walls, about 16 inches high, and wide enough to sit on, at least 12 inches wide.
Designing with Patterns

Design is unfolding

Piecemeal Growth
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
(Notation Cheat Sheet: See Gamma book, back cover)

- **AbstractClassName**: [Type] AbstractOperation()
- **ConcreteClassName**: [Type] Operation() [Type] InstanceVariable
- **PartialClassName**: Aggregation >1
- **AbstractClass**: Inheritance
- **ConcreteClass**: Fabrication
- **ReferencedClass**: Reference
- **InvolvedCaller**: Pseudo code for implementation
- **UninvolvedCaller**: Lifetime
- **Object1**: Instantiation active Operation()
- **Object2**: anObject ObjectReference
- **anotherObject**: InstanceVariable
AbstractFactory Pattern: WidgetFactory Example

WidgetFactory
  - CreateScrollBar()
  - CreateWindow()

MotifWidgetFactory
  - CreateScrollBar()
  - CreateWindow()

PMWidgetFactory
  - CreateScrollBar()
  - CreateWindow()

ScrollBar
  - PMScrollBar
  - MotifScrollBar

Window
  - PMWindow
  - MotifWindow

Client
AbstractFactory Pattern: The General Solution

AbstractFactory

CreateProductA()
CreateProductB()

ConcreteFactory1

CreateProductA()
CreateProductB()

ConcreteFactory2

CreateProductA()
CreateProductB()

AbstractProductA

ProductA1
ProductA2

AbstractProductB

ProductB1
ProductB2
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

Transcript available at
http://www.patternlanguage.com/archive/ieee/ieeetext.htm
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/ieeetext.htm
Mismatched Adoption

- Mitch Kapor’s 1990 “Software Design Manifesto”
Patterns in HCI
(X)PLML,...
Interactive technology likes to be told when something happens or when it is supposed to do something. But people easily forget that extra step, especially when in the middle of a high-energy brainstorming session.

A research video by MIT once showed a group of researchers having a discussion around the table, and the room was “listening in” on the conversation going on. Whenever a certain point was reached, such as deciding to add a new item to the agenda, or delegating a task to a member in the room, everybody had to shut up, and the moderator would speak the corresponding commands for the computer to keep up with what was going on. It was the worst group support interface imaginable.

Good group support software follows what’s going on in the room as good as it can, trying to detect from a variety of sensors, models, and other input what the current activity and actors are, and then takes initiative on a simple, reliable level to help the actors, without presuming to understand more than it can.

Computer scientists will argue that deriving this information from sensor values is not reliable, so the computer needs clear commands in order not to do something wrong. This is perfectly true in distributed settings with low bandwidth or human communication: If user A decides to pass control over the shared mouse cursor in a remote user B in a shared application, he usually has to click a button to do so.

In a collocated setting of an AE, an enormous advantage comes to the help of the system: social protocol. The people in the room can see and hear each other. If one person is controlling the mouse cursor using their laptop, and someone else wants to take over with their own laptop, they will just say so. The computer does not need to understand this verbal command, nor does it need to lock the cursor for everybody else but one user at a time: it can simply accept cursor movement from everybody in the room; if there’s a conflict of concurrent access, the users will quickly and easily notice and resolve it among themselves. This approach, on the other hand, saves the users having to send explicit messages each time they wish to pass control of that cursor to someone else, making the interaction much more fluid.

Examples include the design of the interaction for the iRoom’s remote cursor control that allows “mouse fights” to occur, simply always using the last coordinate received; or its iClipboard feature that lets people copy and paste in a single shared clipboard for the room.

Plogmann et al., in their chapter elsewhere in this book, reflect on this concept by suggesting room infrastructure in which “…users and social conventions in an environment take responsibility for actions, and the system infrastructure is responsible for providing a fluid means to execute those actions.”

Therefore:

Do not put unnecessary protocols into place that are aimed at avoiding overlapping access to technology, if that collision can be easily noticed and fixed by the users through social interaction. It is much better for one user to issue a social protocol that acts, such as passing a wireless mouse to someone else, than to deal with an additional repetitive step from the user to tell the room what he just did for everyone else to clearly see.

This is a basic pattern with no further references within this language.
Evaluating Patterns

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

- Writers’ Workshops
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, applaud, unrelated story
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

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
Writers’ Workshops

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
PERSONAL E-COMMERCE at a glance
Site Genre: Pattern Group A

L.L. Bean gives customers a sense of familiarity because the categories on the site are similar to what they find in L.L. Bean’s physical stores and catalogs. The bright colors, clean layout and navigation, and picture in the center work together to draw people in.

BACKGROUND
This pattern forms the core that makes online shopping possible. Start by using it in its most basic form, then expand and extend it as needed. Use it separately or in conjunction with other site genre patterns, such as NEWS MOSAICS (A2), VALUABLE COMPANY SITES (A7), and STIMULATING ARTS AND ENTERTAINMENT (A9).

PROBLEM
Customers appreciate the convenience of ordering online, but if a site is cumbersome, is veiled about its pricing and policies, or does not seem to provide a personal benefit, they leave.

SOLUTION
E-commerce sites should offer value, allowing customers to shop, collect multiple items, and then quickly check out in a secure manner.

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
Patterns in Game Design is that rare sort of book on game design: a useful one. Readers will find their understanding of games, and designers' tools, expanded by exposure to a wide variety of game design techniques, some of which may not have been previously aware.

—Greg Costikyan
Eminent game designer of Paranoia and member of the Adventure Gaming Hall of Fame

- Learn how to use game patterns to facilitate your designs, generate new ideas, and explore how other games work
- Customize the patterns for your own designs and use them to communicate ideas to the team
- Find 200 ready-to-use, customizable patterns on the companion CD-ROM, slides for lectures, and PDF condensed versions of each pattern

- Staffan Bjork, Jussi Holopainen, 2005
- 300 patterns
- Instantiates — Modulates — May Conflict
More Current Trends

- CHI 2009 XPLML

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

Diagram showing people using laptops and a shared display.
Using Patterns in the Application Domain

Power Shifts

User Interface + Software Design

Large-Scale Concepts
Small-Scale Concepts

Application Domain Pattern Language

Project Environment

Tasks
Interaction Objects

Human-Computer Interaction Pattern Language

Project Environment

Architecture Design
Implementation

Software Engineering Pattern Language
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
Stages and Pattern Use

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
Stages and Pattern Use

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
Stages and Pattern Use

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
Stages and Pattern Use

10. Iterative Design

- As in DIS I
- 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)
Stages and Pattern Use

11. Collect Feedback from Field Use

- After delivery: field tests, follow-up 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
Pattern Languages in HCI: A Critical Review

• In: Human-Computer Interaction Journal, 2006
• by Andy Dearden and Janet Finlay
### 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>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<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>
<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>
<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>
<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>
<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>
<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>
<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>
<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>
<td>⬜</td>
<td>⬜</td>
</tr>
</tbody>
</table>

• = direct statement  ? = implicit agreement
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
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.
Next Steps

• CTHCI: How to do HCI research

• DIS2, MCP, PDUl

• Hiwi jobs helping with HCI research and development (literature research, prototyping, coding, user studies,...)

• Bachelor/Master theses