Current Topics in Media Computing and HCI

HCI Design Patterns Part 2

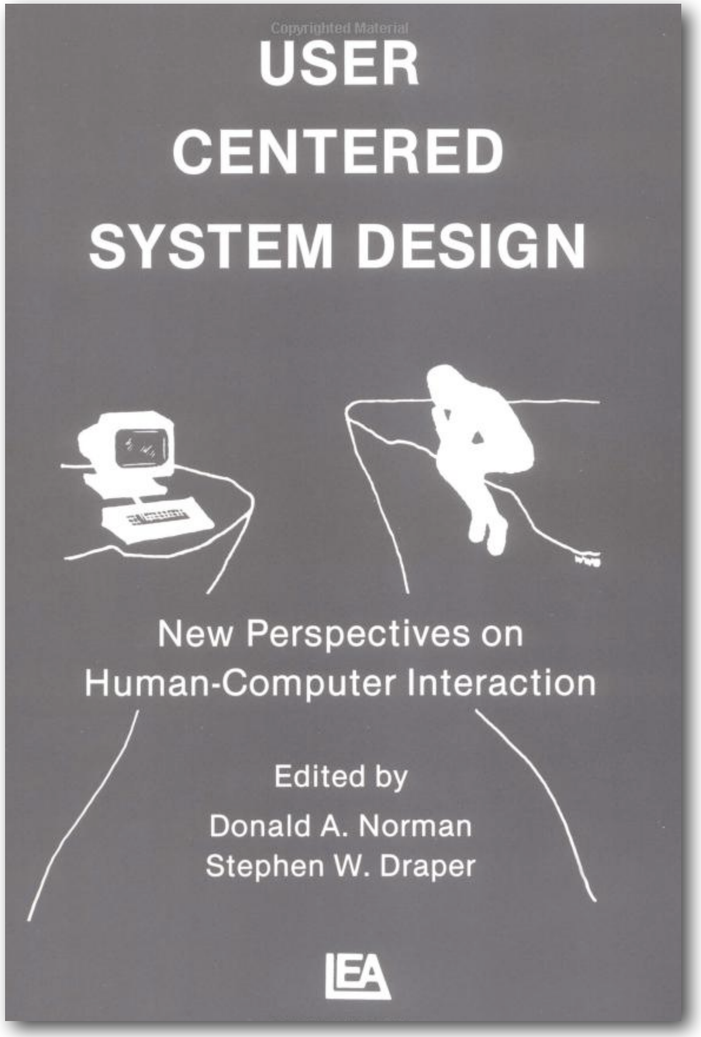
Prof. Dr. Jan Borchers
Media Computing Group
RWTH Aachen University

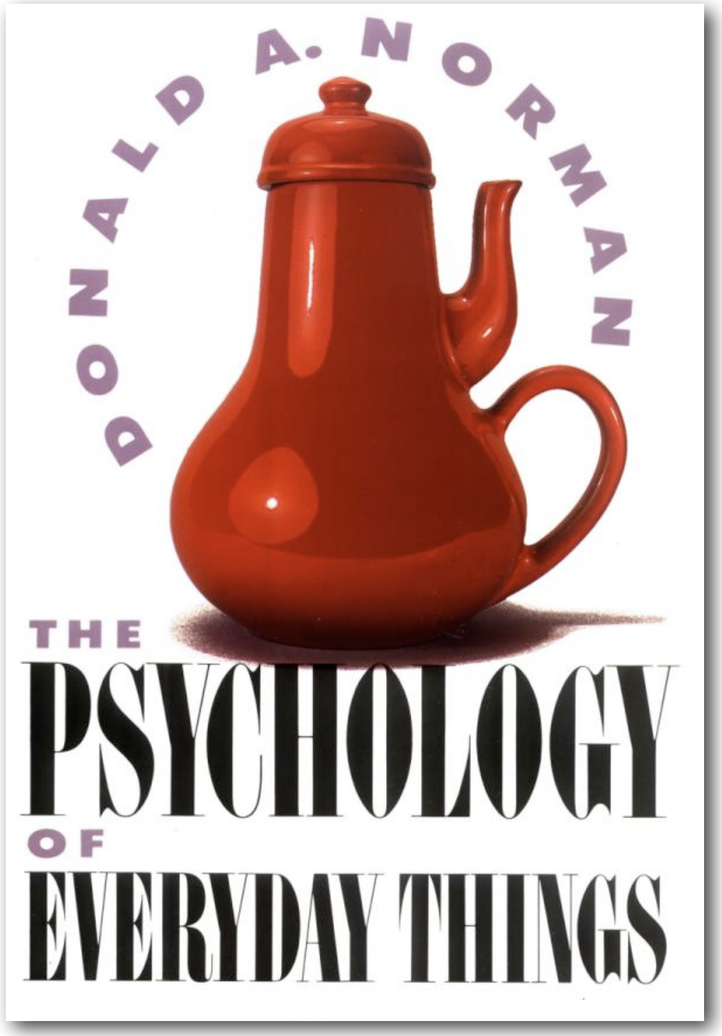
Summer Term 2020

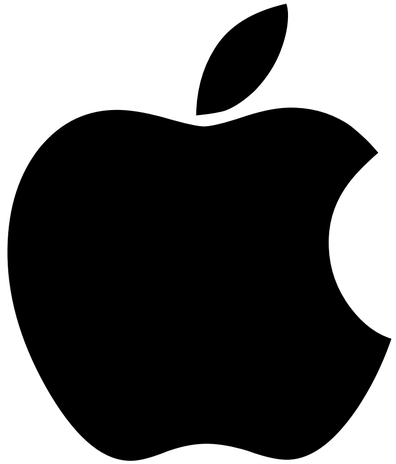
https://hci.rwth-aachen.de/cthci

RWTH AACHEN UNIVERSITY

Patterns in HCI







Patterns in HCI

(X)PLML,...

Name, Ranking

Sensitizer

Figure 17: Passing on a mouse for a group display.

Context

…you have picked your hardware to control the room and its services—ROOM CONTROLLER (15), and now need to decide how the technology is operated by the users.

♦♦♦

Problem

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 meeting 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 for human communication: If user A decides to pass control over the shared mouse cursor to remote user B in a shared application, he usually has to click a button to do so.

Examples

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 he 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.

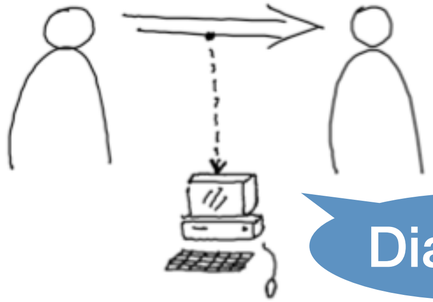
Examples include the design of the interaction for the iRoom's remote cursor control that allows "mouse fights" to occur, simply always using the last coordinate received; or its iClipboard feature that lets people cut and paste in a single shared clipboard for the room.

Winograd 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:

Solution

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. If a user issues a social protocol act, such as passing a wireless mouse to someone else, require an additional repetitive step from the user to tell the room what he just did for everyone else to clearly see.



Diagram

♦♦♦

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

Reference

[Borchers, 2008]

Evaluating Patterns

- By definition, patterns are not “new”
- Different evaluation from research contributions

Shepherding:

- Experienced pattern author provides feedback
- Usually part of the paper submission process
- Example: PLoP conferences (1994—today)

Writers' Workshops

- Originally invented for poets' meetings
- Adopted by Richard Gabriel for the software patterns community
- Designed to respect the author
  and create a relaxed, positive & friendly atmosphere


Richard Gabriel


EuroPLoP 2010


RWTH AACHEN UNIVERSITY

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 understand your work and topic very differently
- 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)

Writers' Workshops

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 my ChiliPLoP'99 HCI patterns workshop report for details.)

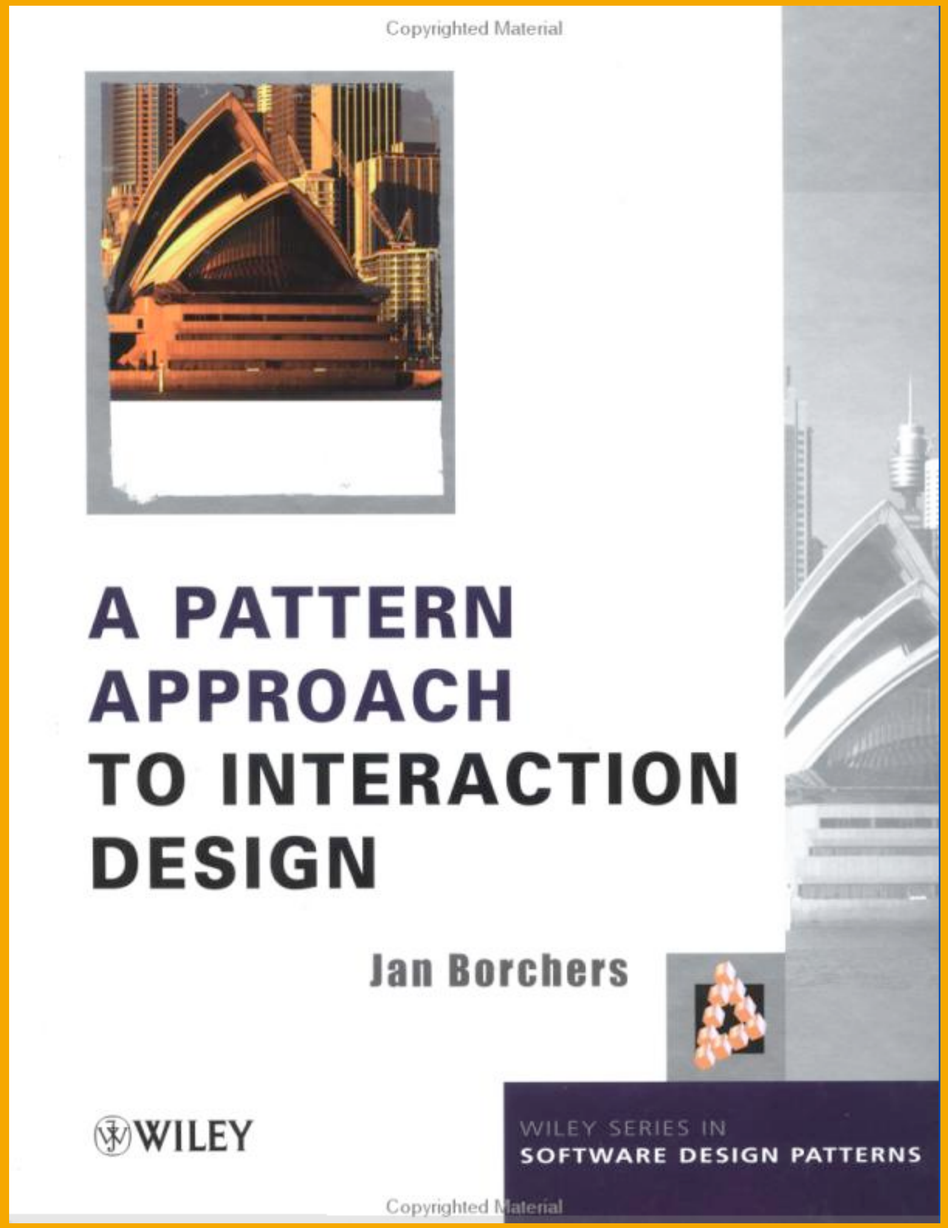
PLML 1.0

- 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

First book that brought design patterns to HCI



QWAN

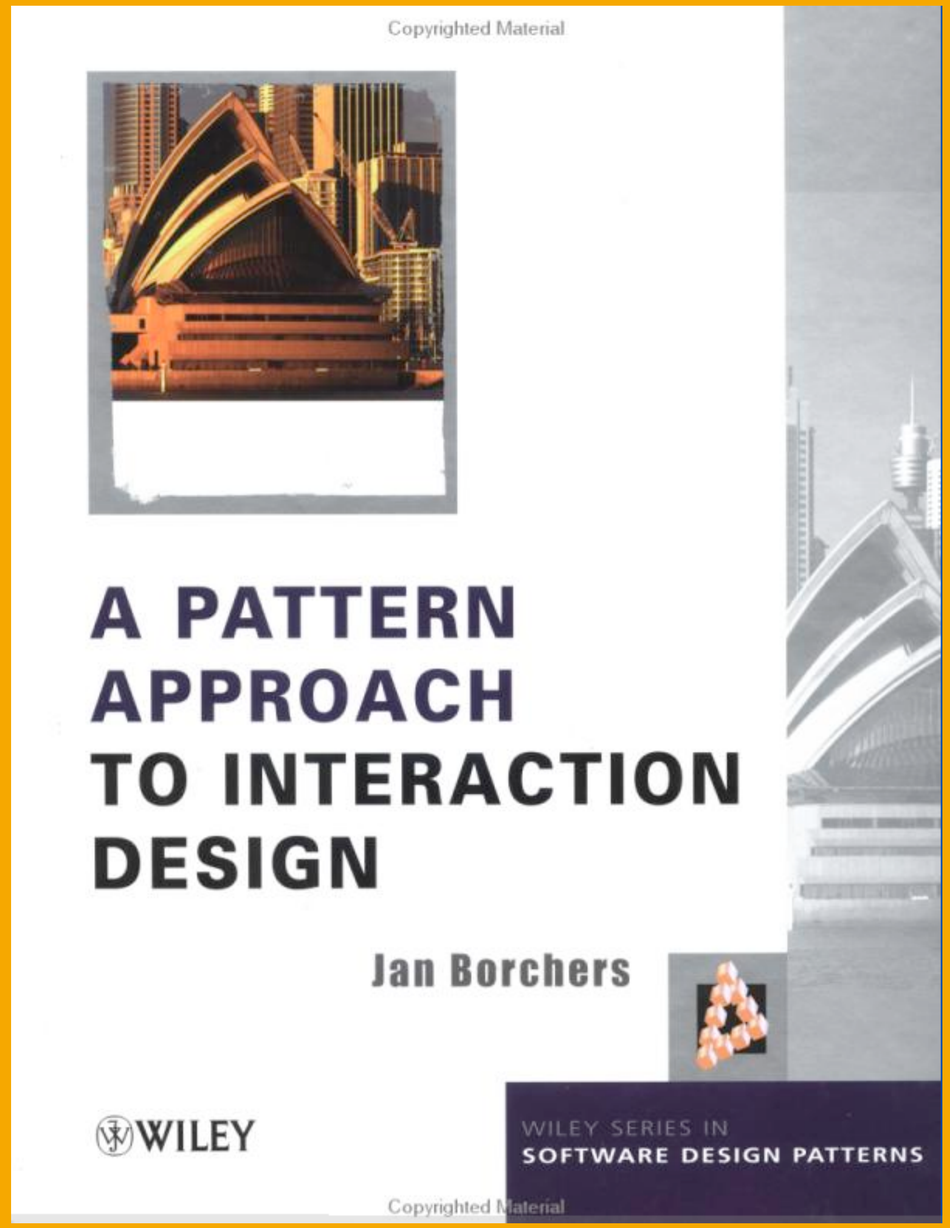
HCI is heir

lingua franca

Corporate Memory

Values

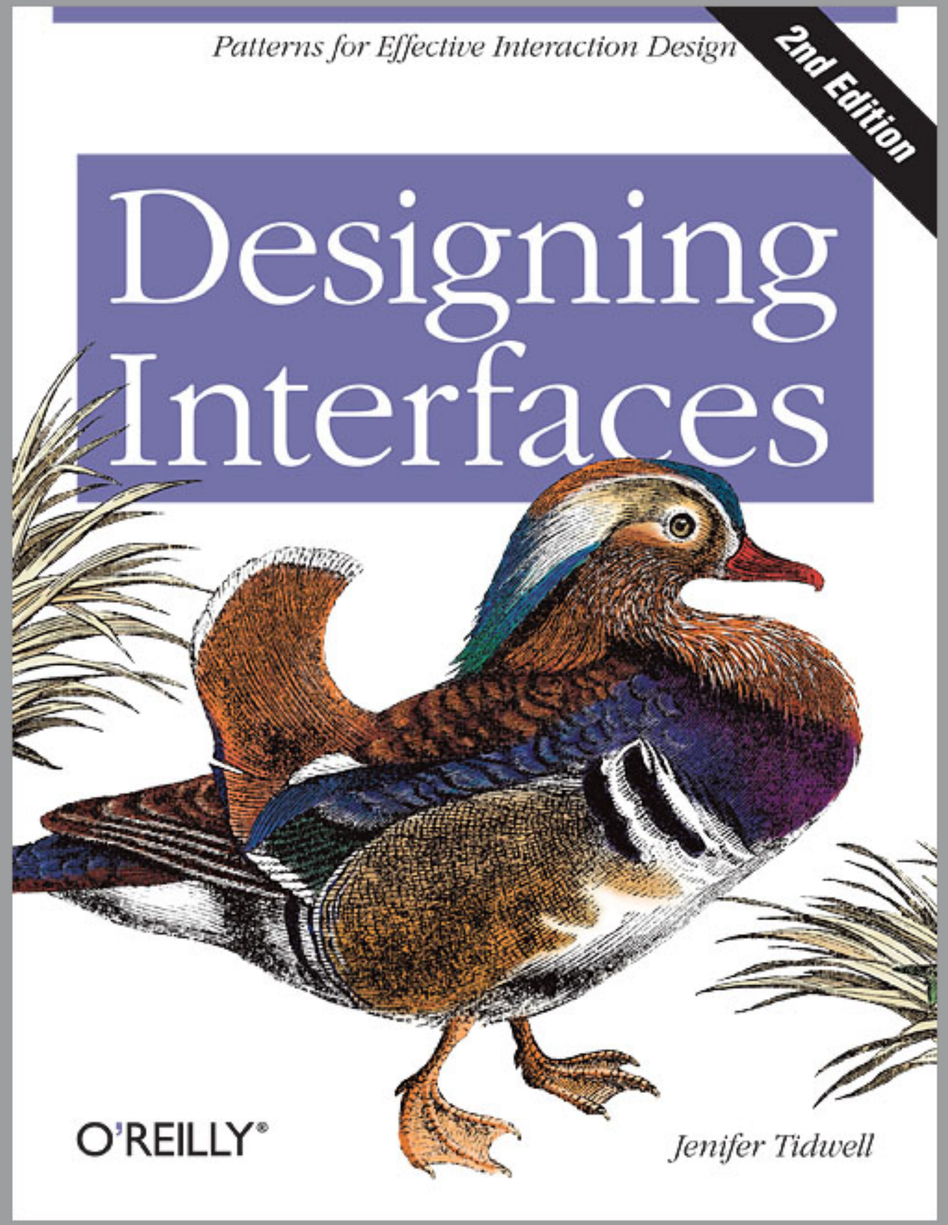
hcipatterns.org



Jenifer Tidwell, 2005

Developed from “Common Ground” Pattern Language (1997) [Common Ground]

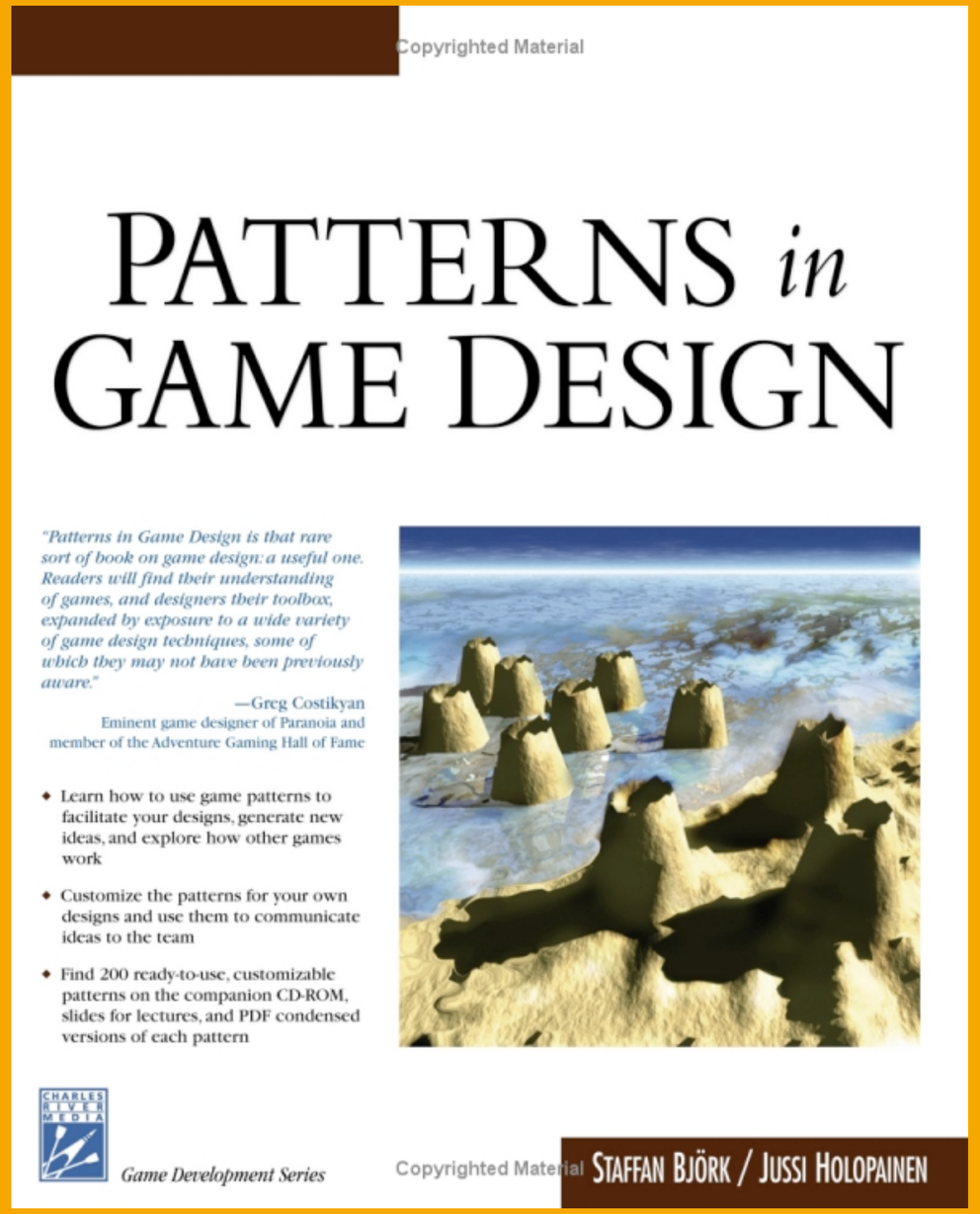
In part available at: designinginterfaces.com



Staffan Bjork,
Jussi Holopainen,
2005

300 patterns

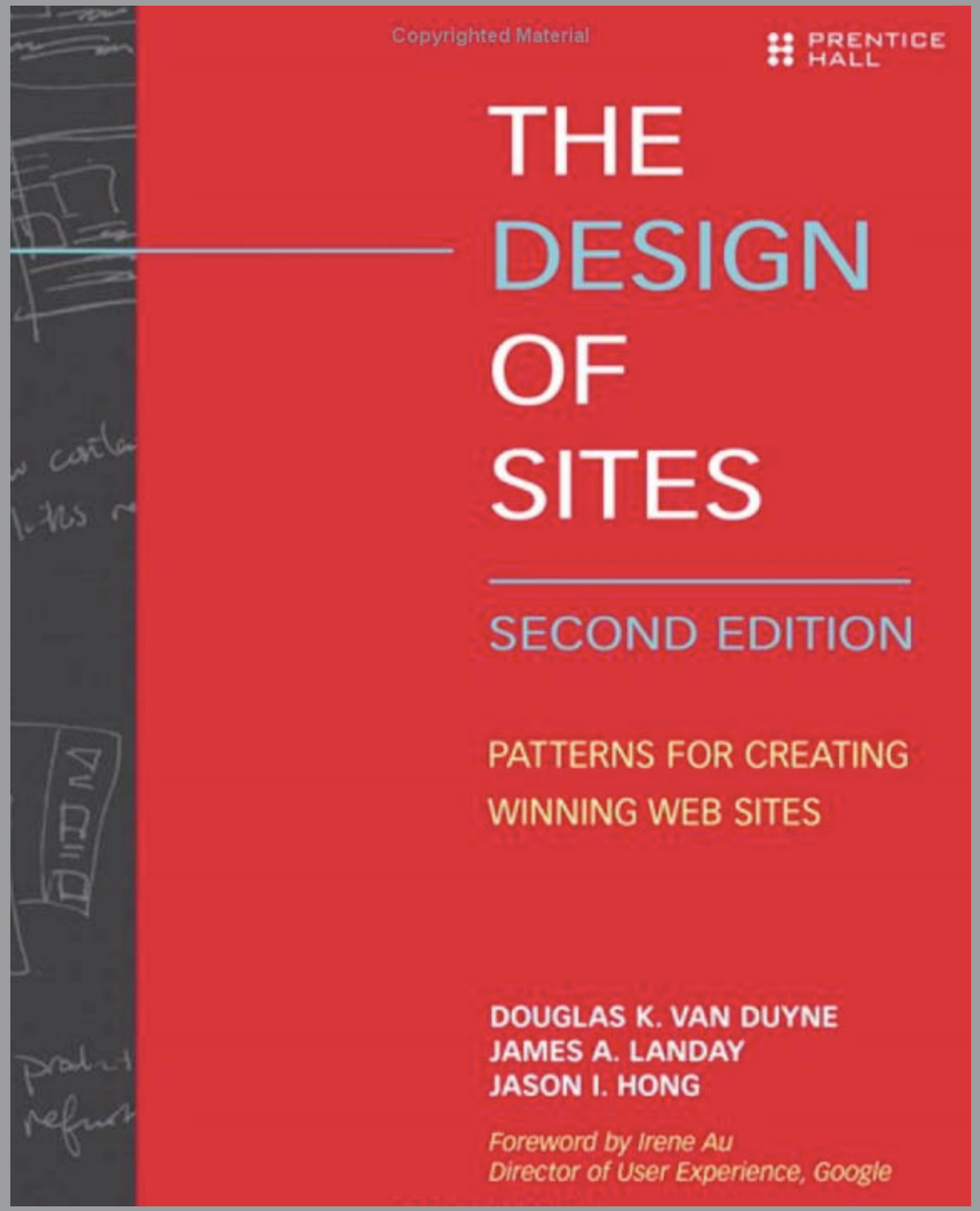
Instantiates – Modulates – May Conflict



v. Duyne et al., 2006 (2nd ed.)

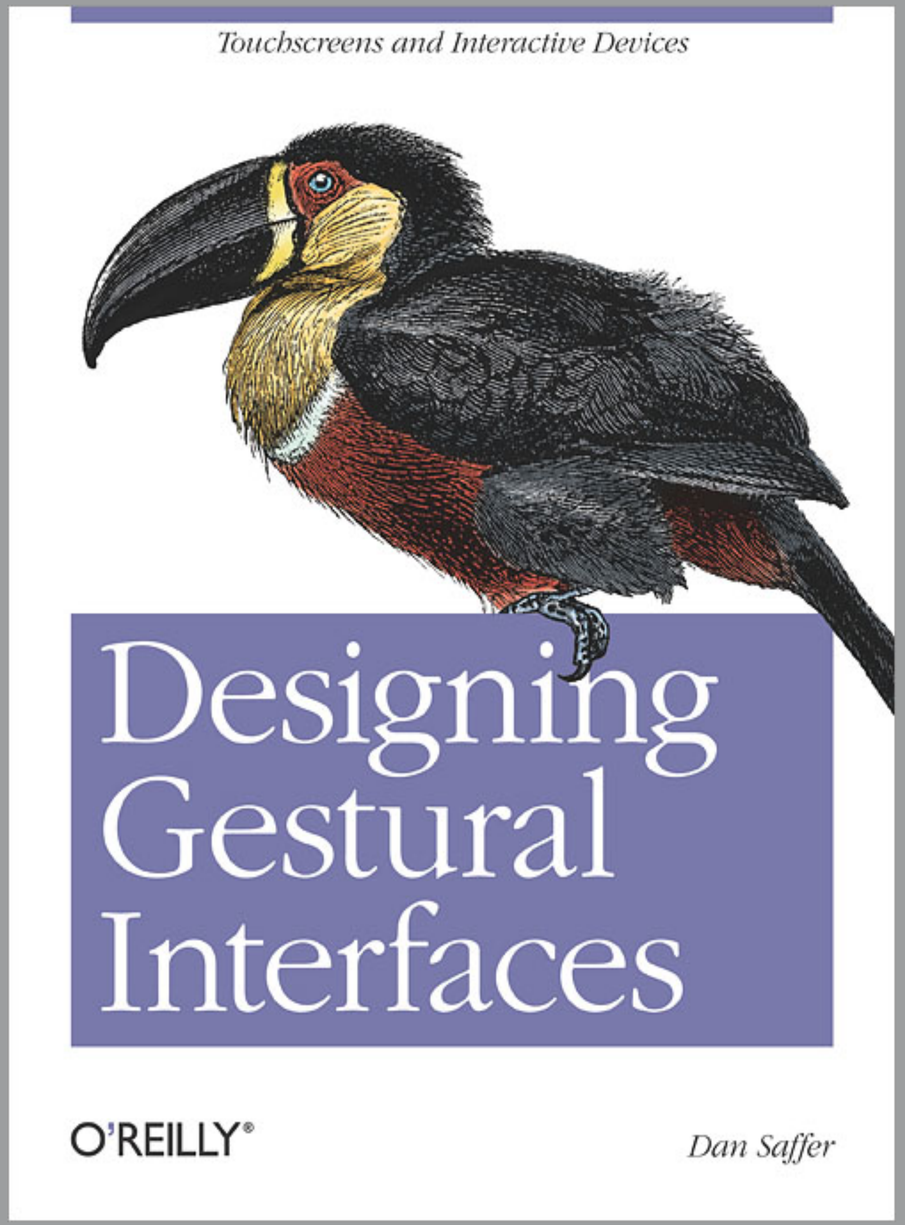
Successful book on HCI Design Patterns for web sites

Ordered by design process



Dan Saffer (2008)

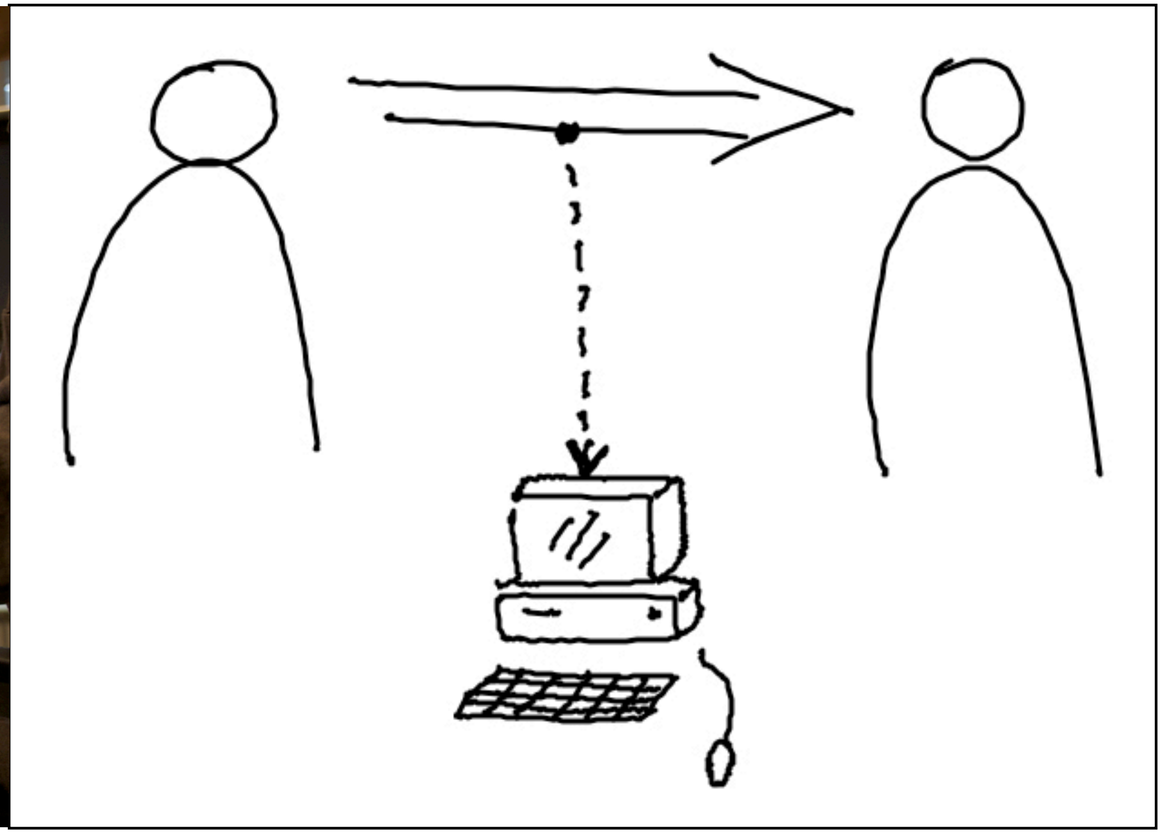
9 patterns for touchscreens and hand tracking devices



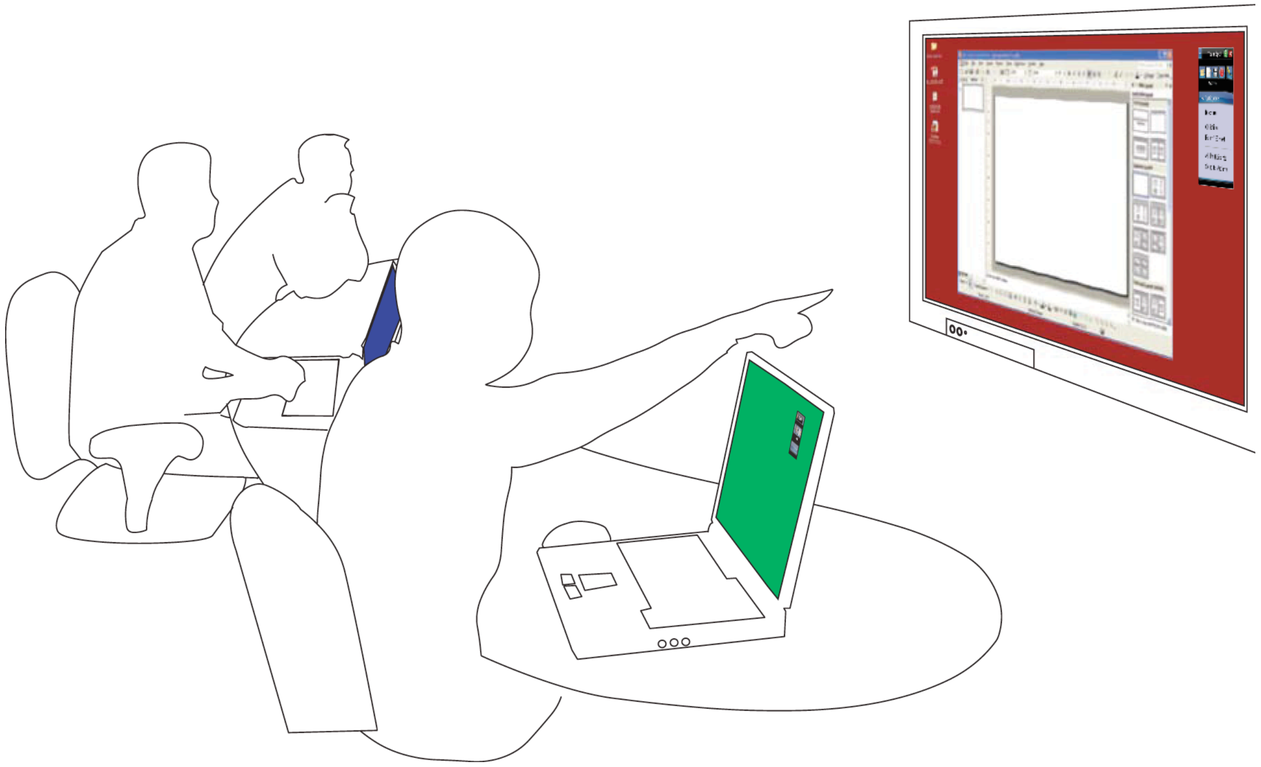
S. Lahlou (ed.), From Meeting Rooms to Digital Collaborative Spaces [CSCW, 2009]

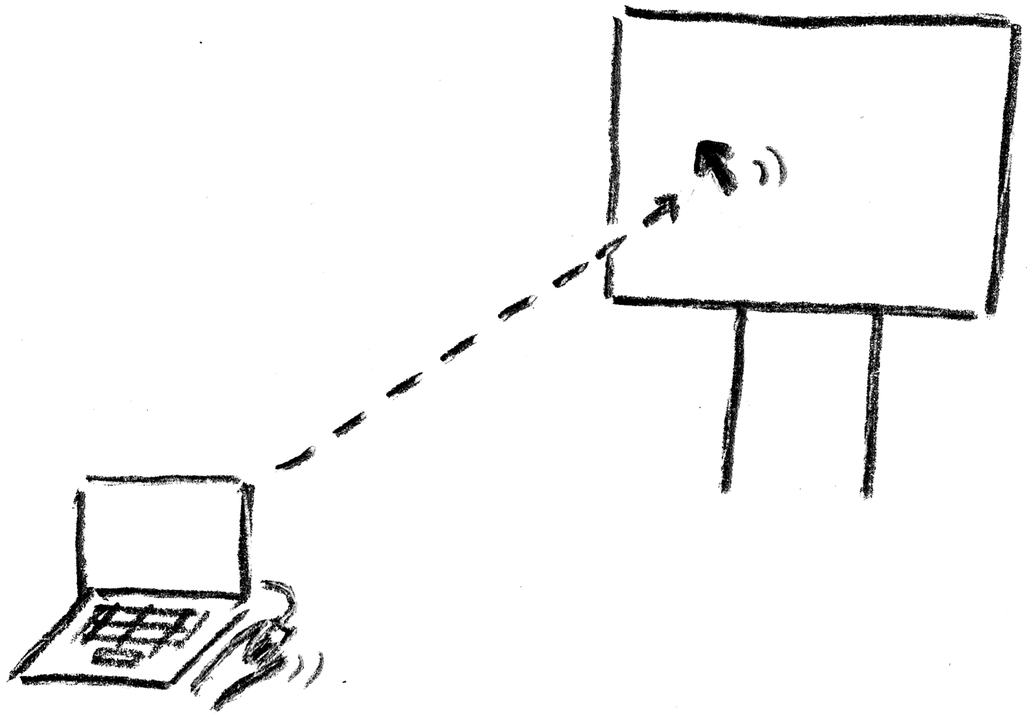
Chapter 10: Jan Borchers, The Aachen Media Space: Design Patterns for Augmented Work Environments

Social Protocol ***



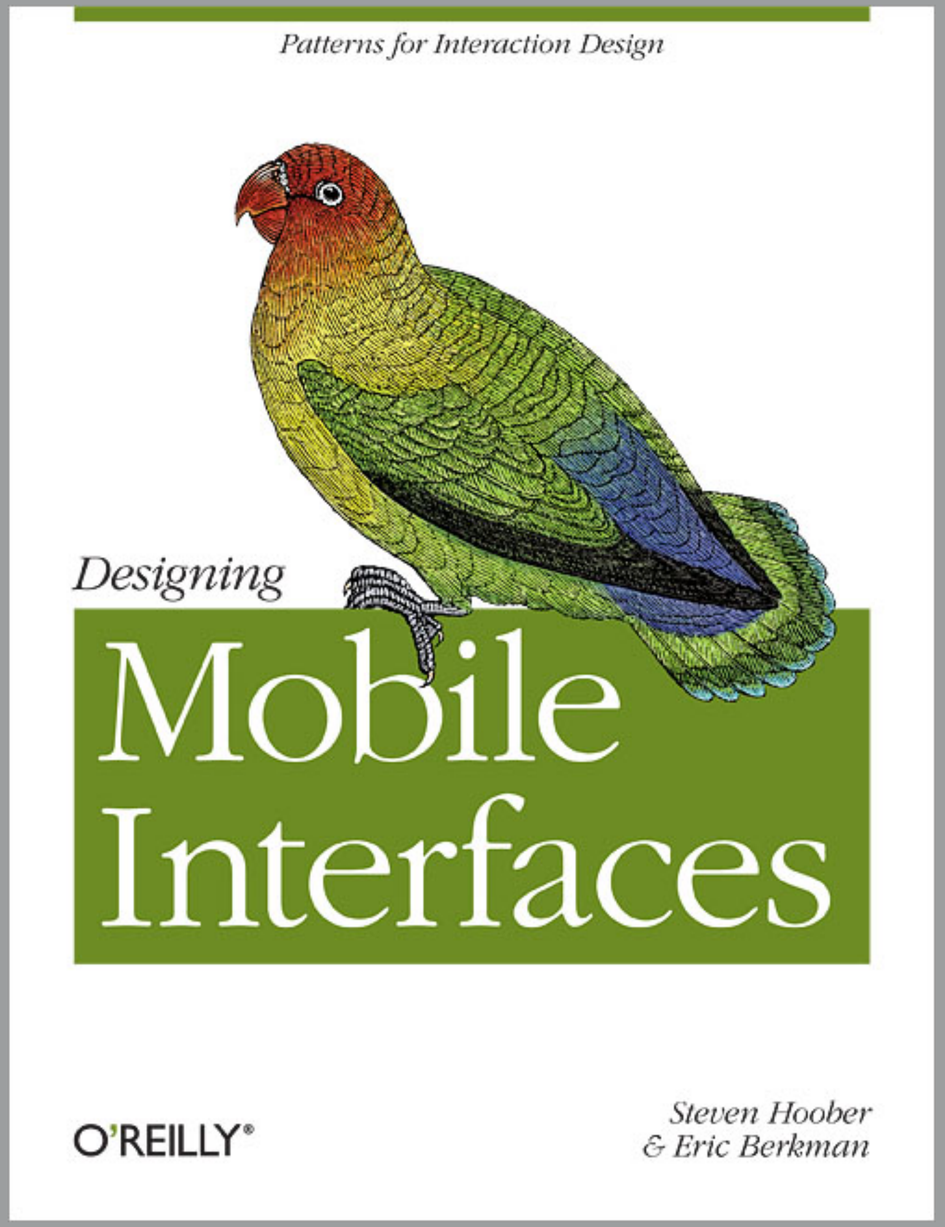
Collocated Group Services **





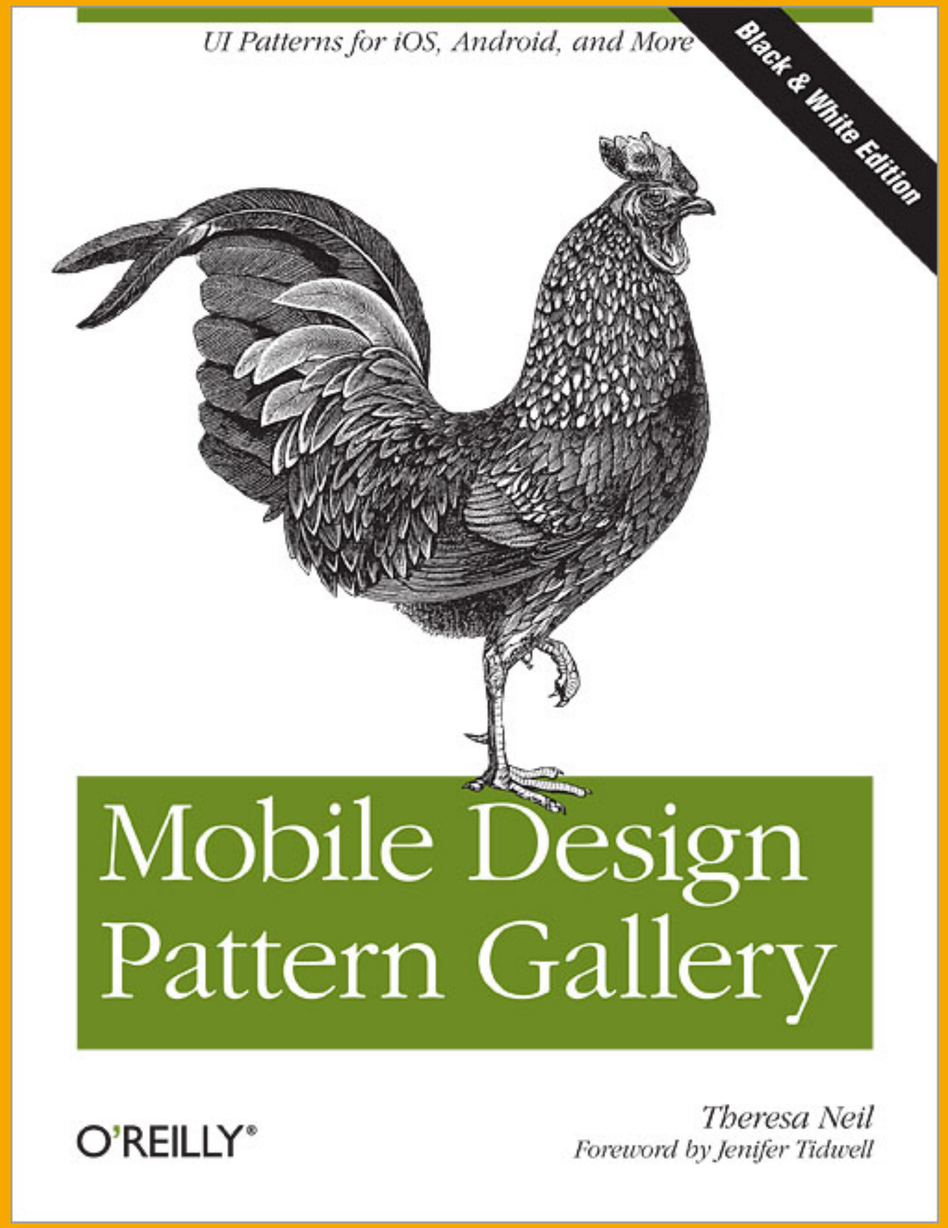
Steven Hoober and Eric Berkman (2011)

76 patterns



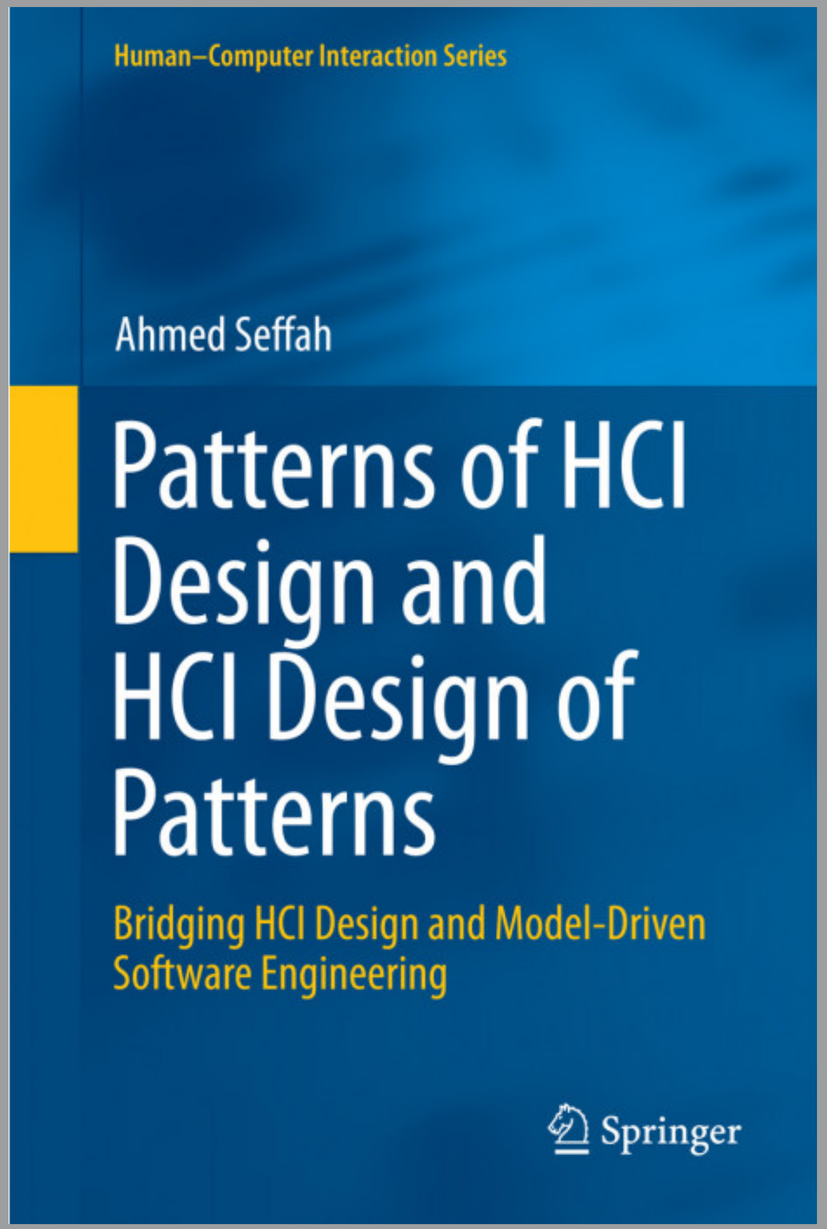
Theresa Neil (2012)

400 screenshots for 70 design patterns



Ahmed Seffah (2015)

Making use of HCI design patterns while designing interactive systems

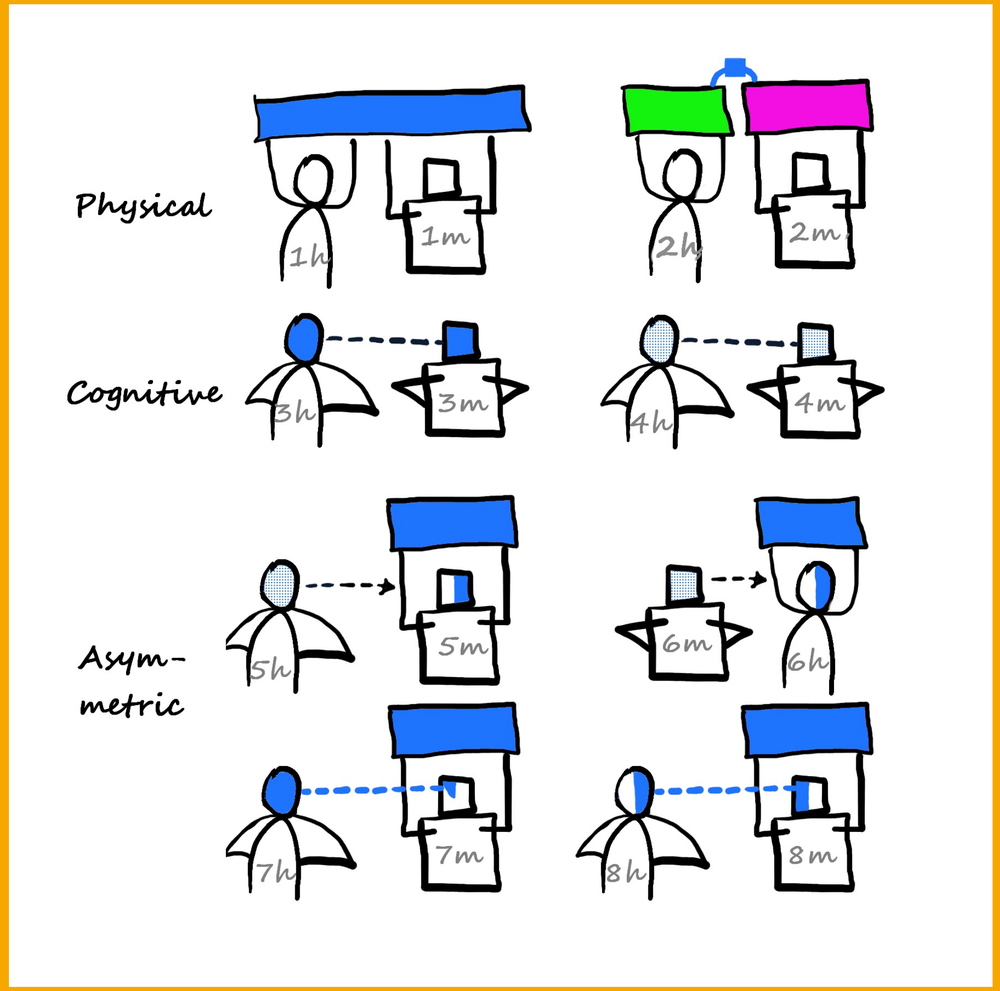


Yixiao Wang & Keith Evan Green (TEI, 2019)

A Pattern-Based Design Framework for Designing Collaborative Environments

Jurriaan van Diggelen & Matthew Johnson (HAI, 2019)

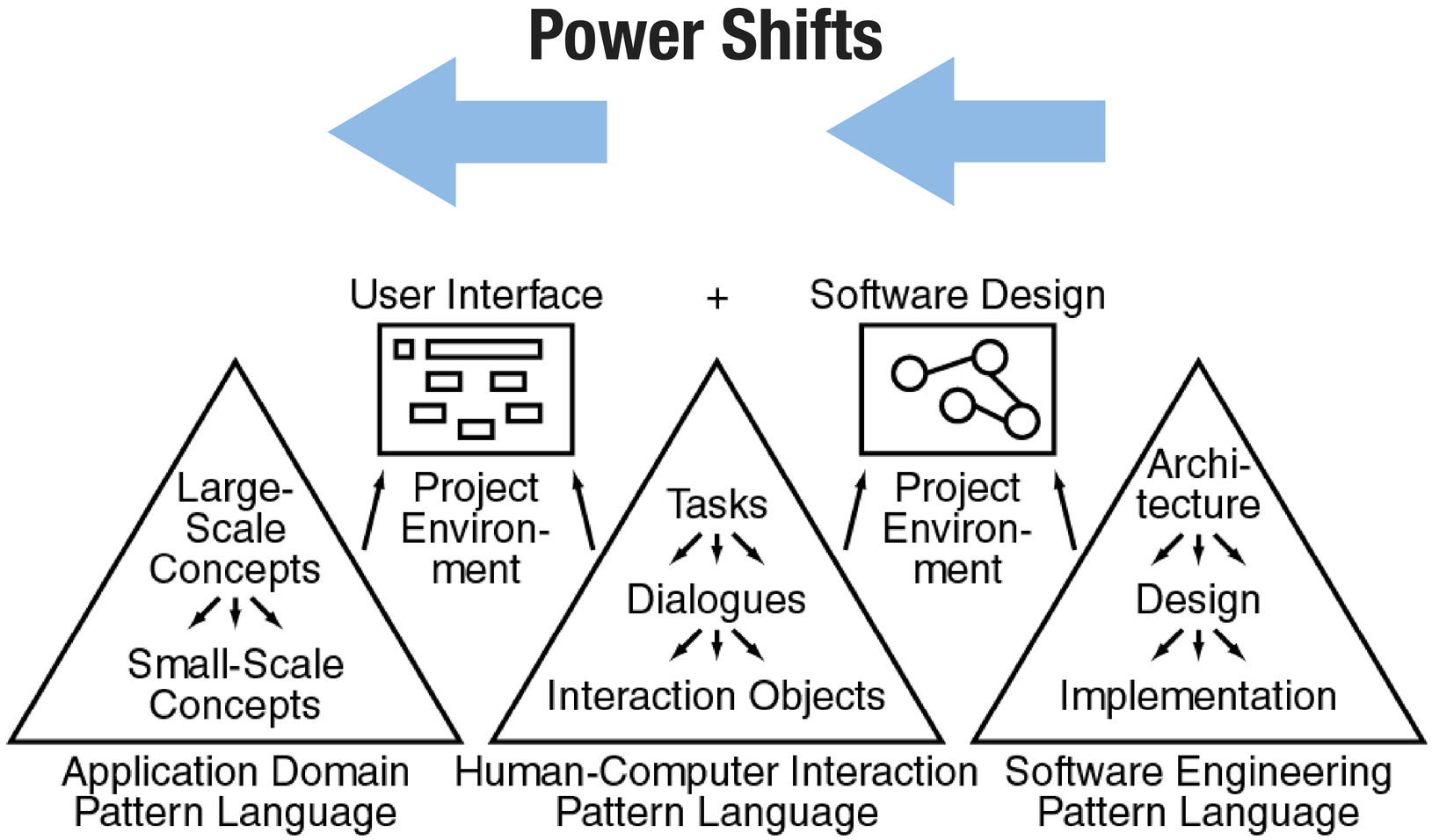
Team design patterns for human-agent teaming concepts



More Trends

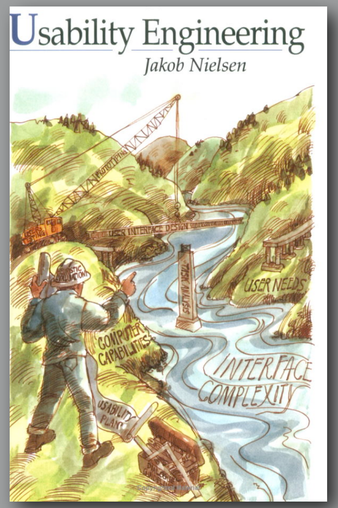
- Dearden & Finlay,
  Human Computer Interaction 21(1), 2006
- *Interactions* 1/2007
- CHI 2009 XPLML

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

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 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)

Stages and Pattern Use

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

Pattern Languages in HCI: A Critical Review

- In: Human-Computer Interaction Journal, 2006
- by Andy Dearden and Janet Finlay

Important Characteris-tics of a Pattern

Characteristic	Wim & Calder2002	Bayle et al. 1998	van Welie et al. 2000	Granlund et al. 2001	Borchers 2000	Finlay et al. 2002	Fincher & Utting 2002	Erickson 2000a	van Duyne et al. 2003	Tidwell 1998, 1999a
1. A pattern implies an artefact	●		?	?	●	?	?	?	●	?
2. A pattern bridges many levels of abstraction	●						?		?	
3. A pattern includes its rationale	●	?	●	?	●	?	?	?	●	?
4. A pattern is manifest in a solution	●									
5. A pattern captures system hot spots	●									
6. A pattern is part of a language	●		?	●	●	●	●	●	●	?
7. A pattern is validated by use	●	?	●	?		●		?		●
8. A pattern is grounded in a domain	●	?	?	●	●	?		●	●	
9. A pattern captures a big idea	●						●			
10. Patterns support a 'lingua franca'		●	?	?	●	●	●	●	?	●
11. Different patterns deal with problems at different 'scales'		●	●	●	●	?	?	●	●	●
12. Patterns reflect design values		●	?		●	●	●	●	?	●
13. Patterns capture design practice		●	●	?	?	?	●		●	?

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