Intro: Design

MICHAEL BERNSTEIN
CS 376
Announcements

• Readings: the magic of Stanford’s EZProxy

• Project Brainstorm Round 2 due Friday
  • Find a team!
  • Mixer ten minutes before the end of class today
Round 1 feedback

• For this assignment, we’re sharing warnings rather than grading harshly. (That’s by design!) Now that you’re learning this, we’ll be more stringent with Round 2.

• The most common critiques:
  • Not being clear on what problem you’re solving, or why it matters
  • Not being clear about the method you’re using, or algorithm/system you’re proposing
  • “Bag of cool ideas” as opposed to one novel insight carried to its logical conclusion
  • Evaluation: how do you know if you’re right?
Course Overview

week 1  Intro to Interaction; Intro to Social Computing
week 2  Intro to Design; Interaction
week 3  Interaction; Social Computing
week 4  Social Computing
week 5  Design
week 6  AI+HCl; Media
week 7  Foundations
week 8  Access; Programming
week 9  Collaboration; Visualization
week 10 Education; Critiques of HCl
Design → Implement → Evaluate → Design
Design and creation are not static processes.

They can be studied, supported and improved.
How might we facilitate and empower this process?
Design
Brainstorming process
Early-stage design tools

Evaluate
Study strategies
Cognitive modeling

Implement
Programming tools
WYSIWYG design tools
Rapid prototyping tools
“Enlightened trial and error outperforms the planning of flawless intellect.”
- David Kelley
Threshold/Ceiling Tradeoff
[Myers, Hudson and Pausch, TOCHI 2000]

Difficulty of use

Sophistication of what can be created

C++

Web

Server-side

Client-side

Balsamiq, Sketch, proto.io
Major themes

- Design tools
- Design process
- End-user programming
Design tools
Design process
End-user programming
Goal: facilitate rapid iteration

- Prototypes enable exploration and iteration around concrete artifacts
- The more fluid the prototyping process is, the more you can learn before you sink time into engineering
Sketch the interaction to produce working systems

- SILK [Landay, CHI ’96]
Sketch the interaction to produce working systems

- Led to: Balsamiq
d.tools: Prototyping Physical Computing Experiences

- How might we prototype an iPhone in thirty minutes?
  - Plug-and-play sensors
  - Statechart authoring for logic
  - Runtime visualization of user states
Closed-loop parameter tuning

- Juxtapose
  [Hartmann et al., UIST 2009]
Closed-loop parameter tuning

- Led to: Inventing on Principle [Victor 2012]
Webzeitgeist
[Kumar et al., CHI ’13]

- Crawl the web and index large-scale design elements
- Main idea: what happens if we start data mining designs, rather than user behavior?
Design tools
Design process
End-user programming
Improve the process, improve the output.

- The design process we teach in human-computer interaction need not be fixed!
- Many techniques we use today were once prototyped in research labs.
Wizard-of-Oz Prototypes

• An iterative design methodology for user-friendly natural language office information applications [Kelley, TOIS ’84]

• “Central to the methodology is an experimental simulation which I call the OZ paradigm, in which experimental participants are given the impression that they are interacting with a program that understands English as well as another human would.”
Iterate on a design, or create parallel alternatives?

[Dow et al., TOCHI 2010]

- Feedback on five iterations or five parallel alternatives
- Quality measured via ad clickthrough
- Designs generated in parallel condition had ~1/3 more clicks
Participatory Design
[Schuler and Namioka ’93]

- Developed in Scandinavia, and later ported to the United States
design tradition
- Involve the eventual users deeply in the design process
  - Initial exploration
  - Problem definition
  - Develop and focus ideas
  - Evaluation
Quantifying Visual Preferences
[Reinecke and Gajos CHI 2014]

- LabInTheWild data via a quiz about which web sites you like
Design tools
Design process
End-user programming
Garbage in, garbage out

- The quality of the interactive systems we build depends on the tools we have at our disposal
- Toolkits and software engineering UIs…
  - Make programming easier to learn and debug, more powerful and more natural
- End-user programming…
  - Make programming more accessible to non-engineers
Programming toolkits

- Seek to understand programmers’ mental model and task goals
- Then, design better support!
- D3: Data-Driven Documents
  [Bostock, Ogievetsky and Heer, Visweek ’11]
Software engineering interfaces

- Augment the development environment rather than the programming language
- Programmers often ask ‘why?’ questions of their programs. Could we support this directly? [Ko and Myers, CHI ‘08]
End-user programming

- Lower the threshold to writing programs
- Allow users with little programming skill to author behaviors
  - e.g., Chickenfoot [Bolin et al., UIST 2008]

```javascript
isbn = find('number just after isbn')
with (fetch('libraries.mit.edu')) {
  pick('Keywords');
  enter(isbn)
  click('Search')
  link=find('link just after Location')
}
// back to Amazon
if (link.hasMatch) {
  insert(before('first rule after "Buying"'),
  link.html)
}
```
Programming by demonstration

- Induce a program behind the scenes
- EAGER [Cypher, CHI '91]
Getting the Right Design and Getting the Design Right

- **What?**
  - Showing users multiple versions of an interface produces more honest and more critical feedback
- **Why?**
  - It asks, how might we adapt the design thinking process to be more effective?
What’s difficult about design research?

- Design and programming tools:
  - Slight accelerations are easy; larger-scale improvements are not
- Design process:
  - Multidimensional and difficult to measure
What’s exciting about design research?

- Existing creation tools are getting better every day
- The design process is now an accepted practice in industry, but still malleable
- Your contributions are generative: they lead to new designs and programs that others will create tomorrow