ICS 667
Advanced HCI Design Methods

07. Prototyping and Agile Methods

Dan Suthers
Spring 2005

Outline

• About Prototyping
  - What prototyping is
  - Advantages and disadvantages
  - Types of prototyping
  - Prototyping’s Role: how prototyping is used

• Agile development (where prototyping is the primary design method)
  - A good time to think about the relationship between HCI and Software Engineering

• Your Projects
About Prototyping

Goals of Prototyping

- Exploring Requirements
  - Participatory design
  - Market analysis
- Choosing among alternatives
  - Risky or critical features
  - Go/no-go decisions
- Empirical usability testing
  - More about this in a few weeks
- Evolutionary development
  - Build in incremental iterative fashion
Arguments for Prototyping

- Structured design has limitations
  - Abstract notations may be hard to understand
  - Users may have under- or over-constrained conceptions of what is possible
- Good fit to Scenario based Design
  - Helps communicate and evaluate Information and Interaction Scenarios
- Prototyping forms a concrete basis for discussion and/or evaluation

Arguments Against Prototyping

- Premature commitment to specific design
- May be mistaken for a working product
- May require a lot of work (resulting in reluctance or lack of time to change and iterate)
Key Tradeoffs

- Quality *versus* premature commitment
- Realism *versus* early availability or throw-away efforts
- Constant iteration *versus* radical change or refactoring
- Dynamic malleable platforms *versus* well structured code base

Some choices: Coverage

- **Horizontal**: all of interface, little or no functionality beyond navigation
- **Vertical**: full interface and functionality only for restricted part (scenario or collection of related use cases)
- **Deep and Wide**: combines both (horizontal for navigation, vertical for one scenario or collection of use cases)
### Some choices: Fidelity

- **Low fidelity** may better support consideration of alternatives
  - Unpolished look → criticisms less inhibited
  - Ambiguity → open to interpretation and discussion
- **High fidelity** ...
  - Good for selling the idea
  - Can expose more subtle design issues

### Methods

- **Storyboarding**: sketches or screenshots illustrating key points in a usage narrative
- **Paper Mockup**: fabricated devices with simulated controls and displays
- **Scenario Machine**: Interactive system implementing a specific scenario (example tool: Dreamweaver)
- **Computer Animation**: screen transitions that illustrate a scenario (example tool: Director)
- **Rapid Prototype**: working system created with special purpose tools (example tool: Visual Basic)
- **Wizard of OZ**: invisible human simulates functionality
- **Video Prototype**: persons enacting one or more envisioned tasks
Tools: Be open to all possibilities

- Paper, markers, Post-its ("equal opportunity")
- Whiteboards, Smartboards, Mimeo
- Sketch, Painting, and Drawing tools
- Multimedia Authoring
  - Macromedia Director
- Hypermedia Authoring
  - HyperCard, Dreamweaver
- Integrated Development Environments
  - JBuilder, Kawa, etc.
- Graphical User Interface Toolkits
  - Easy to prototype but limited control

Prototyping and Design Stages

- Product Conceptualization
  - Rapid sketching of alternatives
  - Low fidelity paper prototypes are best
- Screen Design
  - Test comprehension and aesthetics
  - Transition from paper to software prototype
- Task Level Prototyping
  - Test suitability of support for specific tasks
  - Need full or vertical functionality
  - Software prototypes may be best
  - Need not have polished interface
Examples

- **NetLearn design sketches:**
  - [http://lilt.ics.hawaii.edu/netlearn/design/](http://lilt.ics.hawaii.edu/netlearn/design/)
  - Paper and software (html) based
  - Low to medium fidelity
  - Mixture of horizontal and vertical functionality
  - Product conceptualization and screen design

- **Video Prototype: Apple’s Knowledge Navigator**
  - [http://www.billzarchy.com/clips/clips_apple_nav.htm](http://www.billzarchy.com/clips/clips_apple_nav.htm)
  - Completely fake rather than implemented
  - Intended to convey vision and inspire

Examples: Wizard of OZ

- My student used WOZ to prototype an automated coach for collaborative learning
- Experimenter monitors, responds as if system following detailed script
- Test participant thinks he is working with actual system, pursuing prescribed tasks
Low-Fidelity Participatory Design

Prototyping’s Role in Process Models
Traditional SE Model

Problems with linear models

Requirements are unclear or may change
Specification gap: always ambiguous, always interpreted

Three Types of Solutions

- Test and refine abstract prototypes
  - Not really solving the problem no matter how much you test the abstractions

- Allow flexible movement between specification and implementation
  - Bouncing between abstract and concrete

- Iterative prototyping: build and throw away until you get it right
  - Giving up on abstractions, work with the implementation
Iterative Models

"Plan to throw one away: you will, anyway" -- Brooks

W Model:

Star Model

- Move flexibly between aspects of design
- Evaluation is central
View design as an inquiry process

Not a deterministic derivation, but rather a search in design space

Agile Methods: Motivation

- Take the idea of iterative prototyping to its extreme: tight integration of requirements and design
- The code is the design representation
- Prototype becomes the product
- Claim that you can flatten the cost curve

(cost to change)

(time or project maturity)
Example: Extreme Programming

- Begin with “user stories”
  - Written by the customer
  - A few sentences expressing users’ needs (not implementation)
- User stories drive process:
  - Users specify acceptance tests
  - Development team specifies programming tasks
  - Development team estimates time to implement each story
  - Customer prioritizes development order

Extreme Programming Discipline

- “On-site customer” (same as a “user”?)
- Code Review is good, so do it continuously! Pair Programming
- Integration and testing is good, so do it every few hours!
- Collective ownership of code
- 40 hour work week
- Start with a minimal working system, add functionality as needed
- Refactor code as needed as it grows …
- And more …
Evaluation of Agile Methods

• Pros
  - Easier to learn
  - Less of a documentation/design burden
  - Includes practices of proven value

• Issues
  - Does it scale without a planned architecture?
  - Does your organization have the right culture and personalities?
  - Difficult to do regular (daily) testing on user interface! Hard to automate
  - Will users tolerate refactoring of a GUI that is in use?

Discussion:

How may agile methods and UCD/SBD be profitably combined?
Projects: Topics

- Wide range of applications are suitable, as long as there is a human-computer interface to design
- You can redesign an existing application
- The design must be under your control
- See examples on prior class web sites
- See also CHI videos for inspiration
- Keep in mind you have 9 weeks

Projects: Expectations

- Groups of 2-3 (will consider 1 or 4 with good justification)
- Due Wednesday: letter to “boss” proposing the project and advocating usability
- Then do one cycle, depending on project, for example:
  - requirements, design, prototyping, evaluation
  - evaluation, redesign, prototyping, evaluation
- Project evaluated on portfolio and peer assessment of contribution
- 50% of grade
Projects in DisCourse

- You will be given your own workspaces (called “sections”)
- Use as you wish to coordinate your work
- Come to agreement before posting assignments to the discussion