Who, What, When, Why and How of Prototyping
Administration

- Assignment #1 due today
- Assignment #2a assigned today
Stages of Software Development

1. Study the users and their tasks
2. Study the competition
3. Set usability goals
4. Participatory design
5. Coordination the total interface for consistency
   - Include documentation, help etc.
6. Guidelines for Heuristic Evaluation
   - Evaluate your interface according to guidelines
7. Make prototypes of the systems early and quickly
   - Actually faster to prototype first
8. Empirical testing
9. Iterative Design
10. Collect feedback from field use
Stages of Software Development

1. Study the users and their tasks
2. Study the competition
3. Set usability goals
4. Participatory design
5. Coordination the total interface for consistency
   - Include documentation, help etc.
6. Guidelines for Heuristic Evaluation
   - Evaluate your interface according to guidelines
7. Make prototypes of the systems early and quickly
   - Actually faster to prototype first
8. Empirical testing
9. Iterative Design
10. Collect feedback from field use
Outline

- What is it
- Why do it
- When to do it
- Who does it
- How to do it
- Issues
What?
What is a prototype?

- Prototype = “working” model
- Usually a physical model e.g. cars, airplane, rocket, etc…
- Full-size or to scale
- Fully or partially functioning
- Limited representation of an idea
- Common in almost every field of engineering design.
- In HCI prototype may be a “virtual” model - a “simulation”
What is a prototype?

- Creation of concrete but partial implementations of a system design to explore usability issues.
Why?
Why Prototype?

- Team has expertise in design, programming, etc., so why do it?
Why Prototype?

- To identify user interface and functional requirements
  - Almost impossible to specify in advance
- Can’t always get things right the first time…
Why Prototype?

- To identify user interface and functional requirements
  - Almost impossible to specify in advance
- Can’t always get things right the first time
  - Costly to fix mistakes or start over
- Provides continuous feedback on the current design situation
- In HCI there will never be fully satisfactory design guidelines applicable in all circumstances
  - It is not until users interact with something concrete that their rich expertise about their working environment will surface
Why Prototype?

Reveals problems/prevents gross mistakes
Why Prototype?

Moral: don’t cause your users discomfort because you didn’t prototype your design.
Why Prototype?

- Reveals problems/prevents gross mistakes
Why Prototype?

- Reveals problems/prevents gross mistakes
Why Prototype

- Reveals problems/prevents gross mistakes

For more examples, go to baddesigns.com
Why Prototype?

- Reveals problems/prevents gross mistakes
- Building artifacts immediately leads to costly errors
  - We did not have to consider the sink, forks, teapot, or VCR to see the design flaws, we only considered pictures of them
- Fix problems before they cost you
Why Prototype?

- Enables quick evaluation and feedback throughout design process
- Fosters discussion, reflection and innovative ideas (from designers and users): concrete artifact
- Users enjoy prototyping and feel involved
- Keeps design focused on users
- Results in better usability
- Cheap, easy way to test designs with users
  - Help choose between design alternatives (risky/critical features, go/no-go decisions)
  - Help answer specific questions
When
Prototyping: When/How?

Early Design

- Brainstorm different representations prototypes
- Choose a representation
- Rough out interface Style
- Task centered walkthrough and redesign
- Fine time interface, screen design
- Heuristic evaluation and redesign
- Usability testing and redesign
- Limited field testing

Low fidelity paper

Medium fidelity prototypes

High fidelity prototypes

Working systems
Prototyping the User Interface

- HCI guidelines
- Knowledge of technology

Requirements gathering

Build Prototype

Test

Enhance Design

Design

Classic lifecycle
Who?
Who’s involved in Prototyping?

Role depends on what stage you’re in
How?
What is a prototype?

- A series of screen sketches (paper)
- A storyboard, i.e. a cartoon-like series of scenes illustrating key points of a scenario
- A PowerPoint Slide show
- A video simulating the use of a system
- A piece of software with limited functionality written in the target language or in another language
- An electronic mock-up (hardware)
- A cardboard mock-up
- A lump of wood (e.g. Palm Pilot), foam core, cardboard: fabricated with simulated controls/displays
- “Wizard-of-Oz”
Types of Prototyping

Software Life Cycle

- Exploratory
- Experimental
- Rapid
- Evolutionary
- Incremental

- Throw it away prototypes
- Unstructured development
- Section-at-a-time

Horizontal  Vertical  Full
Fragmentary --------- Complete
Divergent ---------- Convergent
Compromises in Prototyping

- All prototypes involve compromises
- For software-based prototyping maybe there is a slow response? Sketchy icons? Limited functionality?
- Two common types of compromise: horizontal vs. vertical
Compromises in Prototyping

- All prototypes involve compromises
- For software-based prototyping maybe there is a slow response? Sketchy icons? Limited functionality?
- Two common types of compromise: horizontal vs. vertical
- Be sure to identify the compromises being made
- Prototypes are representations of the final object
Representations

- Designers create representations of application at *multiple levels of detail*
- Example: Web sites are iteratively refined at all levels of detail

Site Maps  Storyboards  Schematics  Mock-ups
Fidelity in Prototyping

- Fidelity refers to the level of detail
- High Fidelity
  - Prototypes that look like the final product
- Low Fidelity
  - Artist’s renditions with many details missing
Why Use Low-Fi Prototypes

- Traditional methods take too long
  - Sketches -> prototype -> evaluate -> iterate

- Can simulate the prototype
  - Sketches -> evaluate -> iterate
  - Sketches act as prototypes
    - Designer “plays computer”
    - Other design team members observe and record

- “Kindergarten” implementation skills.
  - Allows non-programmers to participate
Low-Fidelity Prototyping

- Uses a medium which is unlike the final medium, e.g. paper, cardboard for software
- Is quick, cheap and easily changed
- Support the exploration of alternative designs and ideas
- Best in early stage of developments and not intended to be final product
- For exploration ONLY: concepts, look and feel, iron out usability issues early on
- Examples
  - Sketches of screens, task sequences, etc
  - “Post - it” notes
  - Storyboards
  - “Wizard - of - Oz”
Low Fidelity Prototypes

- Paper Prototypes
  - Paper mock-up of the interface look, feel, functionality
  - Quick and cheap to prepare and modify

- Purpose
  - Brainstorm competing representations
  - Elicit user reactions
  - Elicit user modifications / suggestions
Sketches

- Drawing the outward appearance of the intended system
- Crudity means people concentrate on high level concepts but hard to envision a dialog progression
Sketching

- Sketching is important to low fidelity prototyping
- Don’t be inhibited about drawing ability, practice simple symbols
- Develop your own symbols and icons
How to paper prototype

- Set deadline (don’t want to use too much time)
- Draw window on large paper
- Draw different screen regions on index cards and then put on the window paper
- Ready responses for any user action (“magic mouse” can make anything a link)
- Photocopy to make test version
- Test and iterate
Using index cards

- Index cards (3”x5”)
- Each card represents one screen or one element of a task
- Often used in website development
- Used in user evaluations
A card-based prototype for a shared calendar system

- Include enough detail for users to interact with the prototype
Durable case—the tough plastic exterior enables complete protection of the device if dropped, and the rubberized outer casing lessens the impacts of shocks. In addition, the exterior is lightweight and makes the design ideal for use in virtually any environment.

Battery Indicator shows amount of battery left before recharging is required.

Communication keys—these are sensitive touch-panel buttons. On being triggered, a recorded message related to that key is output from the speaker.

Amplified speaker provides excellent output.

Ring attachment for belt/straps. This enables the device to hang from a person's trouser belt in a similar way to a key ring.

Figure 8.1 A paper-based prototype of a handheld device to support an autistic child.
Storyboard

- A series of “key” frames
  - Originally a term from film: used to get the idea of a scene
  - Snapshots of the interface at particular points in the interaction
- Often used with scenarios (from last time), bringing more detail, and a chance to role play
- Used in early design, when users can evaluate quickly the direction the interface is heading
- They indicate the flow from one state or screen to the next.
Storyboard: Image digitizing
Storyboard: GUI-based software system
Storyboard: Ordering Example

**What to do**
Touch a different color, or scan another item.

**What you selected**
JPG Stroller
For children between 1-3 years old...$98.
- Green
- Blue (not selected)
- Red (out of stock)

<table>
<thead>
<tr>
<th>Item</th>
<th>Style</th>
<th>Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>JPG Stroller</td>
<td>Green</td>
<td>$98.00</td>
</tr>
</tbody>
</table>

**Tax:** 6.98
**Total:** $104.98

**All done?**
- Place your order
- Print this list
- Throw this list away
1. Initial Screen

2. Scan the stroller

3. Change the color

4. Place the order
5. Alternate Path…

6. Scan the shirt->

7. Touch Previous Item ->

8. Delete that item->
Storyboard: Tutorial Manuals

- Write them in advance of the system
- A step by step storyboard walkthrough with detailed explanations

Apple’s Tutorial Guide To the Macintosh Finder
Making Your Own Low-fi Prototype

- This example gives an overview of the layout without any detail - a good starting point
- Numerous alternatives can be quickly created without worrying about details
- Should be produced in pencil (so you can easily change it)
- Should be hand-drawn (rulers take too much effort)
Making Your Own Low-fi Prototype

- It might be tempting to draw more “tidy” pictures like this example
- But it is difficult to change, even if in a drawing package
- And there is no benefit over the hand-drawn sketches
- It is highly unlikely that the first (or 2nd, or 3rd, or 4th) designs will be completely correct
- But if they are hard to amend, they won’t be amended.
Making Your Own Low-fi Prototype

- Once you are happy with your overview of the layout (for multiple windows/dialogs if necessary) you can start to focus on details of the design:
  - example data values, menu content, buttons, labels, etc
  - More specific layout of interface objects
Making Your Own Low-fi Prototype

- Now you can choose to return to the storyboard and provide some detail
Making Your Own Low-fi Prototype

- Once you are happy with those details you can create your interface “toolkit”
  - Cut out each of the components (windows, dialogs, menus, etc) into its own window
- These can be used to dynamically simulate the interface following the storyboard
- Next, evaluate with users.
Summary: Low-fi

- Traditional methods take too long, can do in a few hours
- No expensive equipment required
- Can simulate almost anything
- Anyone can implement
- Fast iterations: # iterations has impact on quality of final design.
Summary: Low-fi Prototyping

- Advantages
  - Low development cost
  - Can evaluate multiple design concepts quickly
  - Useful communication device
  - Good for considering screen layout issues and user navigation through the interface
  - Anyone can use it to quickly simulate anything
Summary: Low-fi Prototyping

- Disadvantages
  - Not detailed enough to implement from
  - Need to be facilitated when presented to users
  - Does not address issues that arise from implementations
Medium Fidelity Prototypes

- Prototyping with a computer
  - Simulate some but not all features of the interface

- Purpose
  - Provides sophisticated but limited scenario for the user to try
  - Can test more subtle, sophisticated design issues

- Dangers
  - User’s reactions often “in the small” (details)
  - Users reluctant to challenge designer
  - Users reluctant to change the design
  - Management may think it is real!
Medium Fidelity

- Wizard of Oz (WoZ)

- Can test ideas that would be hard to build prototypes of e.g. complex human-like reasoning
  - Artificial intelligence systems
  - Medical diagnosis could be simulated with a doctor “wizard”
Wizard of Oz

- A method of testing a system that does not exist yet
- Example: a system that transcribes text.

What the user sees

The Wizard
Wizard of OZ

- Human “Wizard” simulates system response
  - Interprets user input according to an algorithm
  - Controls computer to simulate appropriate output
  - Uses real or mock interface
  - Wizard sometimes visible, sometimes hidden
    - “pay no attention to the man behind the curtain”

- Good for
  - Adding simulated and complex vertical functionality
  - Testing futuristic ideas
Wizard of Oz Examples

- IBM: an imperfect listing typewriter using continuous speech recognition
  - Secretary trained to:
    - Understand key words as “commands”
    - To type responses on screen as the system would
    - Manipulating graphic images through gesture and speech

- Intelligent Agents
  - Person trained to listen to mobile user and wait for them to ask a question, when question is asked, they do a google search, and give them result
  - Shows us how people would use the system, and helps understand what the intelligent agent should do.
Medium Fidelity: Horizontal vs. Vertical

- **Vertical Approach**
  - Restricted to a subset of the system
  - But the subset offers the interface *and* functionality
  - Can undertake realistic testing

- **Horizontal approach**
  - User interface is fully present
  - But no, or limited, functionality behind it
  - Fast implementation
  - Full extent of interface can be tested, but not in a real-use context

- **Slide/video simulation**
  - Simple
  - Not flexible, or interactive
Painting / Drawing Packages

- Draw each storyboard scene on computer
  - Very thin horizontal prototype
  - Does not capture the interaction “feel”

Next Drawing
For shut down condition
Scripted Simulations

- Create storyboard with media tools
  - Scene transition activated by simple user inputs
  - A simple vertical prototype
- User given a very tight script/task to follow
  - Appears to behave as a real application
  - Script deviations blow the simple

![Control panel for pump 2]

DANGER!

- coolant flow 0%
- retardant 20%
- speed 100%

Shut Down
What to do
Find the item you want in the catalog and scan the bar code next to it.

What you selected

<table>
<thead>
<tr>
<th>Item</th>
<th>Style</th>
<th>Cost</th>
</tr>
</thead>
</table>

All done?
- Place your order
- Print this list
- Throw this list away

Total: $0.00

tax:
What to do
Touch a different color, or scan another item.

What you selected
JPG Stroller
For children between 1-3 years old ...$98.

Item
JPG Stroller

Style
Green

Cost
98.00

Blue
Red (out of stock)

tax: 6.98

Total: $104.98

All done?

Place your order
Print this list
Throw this list away
What to do
Touch a different color or scan another item

What you selected
JPG Stroller
For children between 1-3 years old ...$98.

- Green
- Blue
- Red (out of stock)

Item
JPG Stroller

Style
Blue

Cost
98.00

Delete

tax: 6.98

Total: $104.98

All done?
Place your order
Print this list
Throw this list away
What to do
To get your items, bring your printout to the front counter.

<table>
<thead>
<tr>
<th>Item</th>
<th>Style</th>
<th>Cost</th>
<th>tax: 6.98</th>
<th>Total: $104.98</th>
</tr>
</thead>
<tbody>
<tr>
<td>JPG Stroller</td>
<td>Green</td>
<td>98.00</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Interface Builders

- Design tools for laying out common widgets
  - Ex: Flex, VB.Net
- Excellent for showing look and feel
  - A broader horizontal prototype
  - Alas, it is constrained to a widget library
- Vertical functionality added selectively through programming
High Fidelity Prototyping

- Uses materials that you would expect to be in the final product
- Prototype looks more like the final version than a low-fidelity version
- Need software tools
- Danger that users think they have the full system
- Useful for selling ideas
High Fidelity

- **Advantages**
  - User-driven
  - Provide look and feel
  - Can be used as a specification for final implementation

- **Disadvantages**
  - Expensive and time-consuming to develop
  - Not good for gathering requirements
  - Not good for proof-of-concept designs
Hi-Fi Prototypes Can Warp ...

- Perceptions of the tester/reviewer?
  - Formal representations indicates “finished” nature
    • Will receive comments on fonts, color, and alignment

- Time?
  - Encourage precision
    • Specifying details takes more time

- Creativity
  - Lose track of the big picture
High vs. Low Prototyping
## High vs. Low Prototyping

<table>
<thead>
<tr>
<th>Type</th>
<th>Advantages</th>
<th>Disadvantages</th>
</tr>
</thead>
<tbody>
<tr>
<td>Low-fidelity prototype</td>
<td>• Lower development cost.</td>
<td>• Limited error checking.</td>
</tr>
<tr>
<td></td>
<td>• Evaluate multiple design concepts.</td>
<td>• Poor detailed specification to code to.</td>
</tr>
<tr>
<td></td>
<td>• Useful communication device.</td>
<td>• Facilitator-driven.</td>
</tr>
<tr>
<td></td>
<td>• Address screen layout issues.</td>
<td>• Limited utility after requirements established.</td>
</tr>
<tr>
<td></td>
<td>• Useful for identifying market requirements.</td>
<td>• Limited usefulness for usability tests.</td>
</tr>
<tr>
<td></td>
<td>• Proof-of-concept.</td>
<td>• Navigational and flow limitations.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>High-fidelity prototype</td>
<td>• Complete functionality.</td>
<td>• More expensive to develop.</td>
</tr>
<tr>
<td></td>
<td>• Fully interactive.</td>
<td>• Time-consuming to create.</td>
</tr>
<tr>
<td></td>
<td>• User-driven.</td>
<td>• Inefficient for proof-of-concept designs.</td>
</tr>
<tr>
<td></td>
<td>• Clearly defines navigational scheme.</td>
<td>• Not effective for requirements gathering.</td>
</tr>
<tr>
<td></td>
<td>• Use for exploration and test.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Look and feel of final product.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Serves as a living specification.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Marketing and sales tool.</td>
<td></td>
</tr>
</tbody>
</table>
Other Issues

- Other prototyping methods
- Testing your prototype
- Prototyping systems
“Off-the-shelf” Prototyping

- Useful to jump-start the design and iteration process
  - Recruit existing tools and devices
  - Integrate into approximation of a “system”
- Example as used in virtual school project
  - Telephone for audio conferencing
  - Netmeeting for video conferencing, chat
  - Web pages for project questions and answers
  - Email for interaction with mentors
- Can be very useful in requirements exploration and in activity-oriented feasibility studies.
Prototyping with PowerPoint

- Create general look-and-feel of interface with essential functionality

- Generate interface widgets using Visual Basic Macros
  - Available through toolbar that can be turned on
  - Must set security level to “Low”
  - Actual control functions can only be tested in “slideshow mode”

- Supports creation of an output file for testing

Testing Prototype

- Decide on testing goals for each task being tested
- Run test
  - Practice to avoid bugs
  - Test with “n” participants
  - Multiple testers: greeter/observer, facilitator, computer
  - Debrief users
- Analyze results and iterate
Desirable Features for Rapid Prototyping Tool

- Graphical construction of presentation layer
  - Develop and modify screens
  - Import text, graphics and other media in various formats
- Dialogue interaction
  - Menu building
  - Link screens
  - Support a variety of I/O devices
  - Support a range of types of user interfaces
Desirable Features for Rapid Prototyping Tool (Cont)

- Functional aspects
  - Immediate “test drive”
  - Allow calling of procedures and programs (either internal or external)
  - High level language (to control sequencing)
  - Suspend time editing (stop/resume)

- Good usability
- Good vendor support
Videos

- **SILK, DENIM, SUEDE**
  - Designer sketches with mouse
  - Components are recognized
  - Test interface
  - System produces finished UI

- **Phidgets**
  - [http://grouplab.cpsc.ucalgary.ca/phidgets/gallery](http://grouplab.cpsc.ucalgary.ca/phidgets/gallery)
What is SILK?

__Sketching
__Interfaces
__Like
__Krazy
Designing Interfaces with SILK

1. Designer sketches ideas rapidly with electronic pad and pen
   - SILK recognizes widgets
   - Easily editing with gestures
2. Designer or end-user tests interface
   - Widgets behave
   - Specify additional behavior visually
3. Automatically transforms to a “finished” UI
Specifying Behaviors

- Sequencing behavior *between* widgets

- Storyboards
  - Series of rough sketches depicting changes in response to end-user interaction
  - Expresses many common behaviors
SILK Storyboards

- Copy sketches to storyboard window
- Draw arrows from objects to screens

- Switch to run mode to test
  - SILK changes screens on mouse clicks
SILK Storyboards
Prototyping on its own: Possible Limitations

- No coherent conceptual model -> user’s feel system has unpredictable components
- Uneven appreciation of various user groups
- Lack of task analysis -> lack of breadth of task support
- Failure to fully comply with a style guide -> lack of internal and external consistency
- Lack of usability evaluations
  - Users involved with prototype development may not represent cross-section of users
  - Not easy to learn or intuitive for newcomers

See Redmond-Pyle and Moore, 1995
Some Key Tradeoffs

- Quality vs. premature commitment
- Special purpose systems vs. scheduling and resource management
- Realism (e.g. timing, content) vs. early availability or throw-away efforts
- Constant iteration vs. radical change and/or refactoring of a design
- Dynamic (highly malleable) platforms vs. organized, well structured code base
Questions?