What is a prototype?

- Definition: A concrete but partial implementation of a design
- Can take many forms
  - Sketches, set of screens, day-in-the-life video, virtual environment, limited-function system implementation, early-version of final implementation
- Is used for many purposes
  - See if your ideas look as good as they sound
  - Show design concepts to users
  - Convince management/sponsors to invest
  - Contrast two (or more) technical alternatives
  - Early and continuing evaluation
Is a scenario a prototype?

<table>
<thead>
<tr>
<th>Approach</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Storyboard</td>
<td>Sketches or screenshots illustrating key points in a usage narrative</td>
</tr>
<tr>
<td>Mock-up (e.g., cardboard)</td>
<td>Fabricated devices with simulated controls or display elements</td>
</tr>
<tr>
<td>Wizard of Oz</td>
<td>Invisible human assistant simulates input, output, or processing functionality not yet available</td>
</tr>
<tr>
<td>Video prototype</td>
<td>Video recording of persons enacting one or more envisioned tasks</td>
</tr>
<tr>
<td>Computer animation</td>
<td>Screen transitions that illustrate a series of input and output events</td>
</tr>
<tr>
<td>Scenario machine</td>
<td>Interactive system implementing a specific scenario's event stream</td>
</tr>
<tr>
<td>Rapid prototype</td>
<td>Interactive system created with special-purpose prototyping tools</td>
</tr>
<tr>
<td>Working partial system</td>
<td>Executable version of a system with a subset of intended functionality</td>
</tr>
</tbody>
</table>
Why build a prototype?

- Make design concepts concrete so that they can be
  - Experienced (more vividly)
  - Discussed (more precisely)
  - Critiqued (more technically)
  - Evaluated (in the sense of “formative” evaluation)
  - Engineered (systematically improved)
- Facilitate early and continuing participation by users and other non-developer stakeholders
- Specific technical issues (choose among risky or critical features, early and continuing evaluation)
- Enables evolutionary software development

Prototyping Tradeoffs

- Glitzy vs premature commitment
- Expensive but realistic (wrt timing, content) vs early availability or throw-away efforts
- Constant “hill-climbing” iteration to local optima vs radical change and/or re-factoring of a design
- Dynamic (highly malleable) platforms vs organized, well-structured code base
- Low-fidelity vs high-fidelity
Lo-Fi/Hi-Fi tradeoffs in Prototyping

**Low Fidelity**
- Fast/easy to create/change
- Costs less
- “Roughness” helps to control premature commitment and invite participation
- Cannot investigate/guide detailed technical issues
- Less eyeball appeal

**High Fidelity**
- Can slow things down, delay testing, distract effort
- Expensive
- Can test detailed performance issues
- Assess aesthetics in detail
- Impress clients/managers
- Provide guidance to technical writers
- May oversimplify complex implementation issues

When would you prototype?

- Continually
- The specify-build-deploy waterfall is history
- All software development incorporates prototyping
  - Prototype to identify, analyze, validate requirements
  - Prototype to develop and specify design
  - Prototype to implement and integrate system
  - Prototype to enable formative and summative testing
Boehm’s Spiral Model

Examples of prototyping through (or instead of) the development process

- PICTIVE = Plastic Interface for Collaborative Technology Initiatives through Video Exploration
- Wizard of Oz
  - SmartHelp
- Scenario Machines
- Off-the-shelf prototypes
- Evolutionary prototypes
Low-Fidelity Participatory Design

Wizard of Oz Prototyping

Test participant thinks he is working with actual system, pursuing prescribed tasks

Experimenter monitors, responds as if system following detailed script
“Off-the-Shelf” Prototyping

- 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
Evolutionary prototype development

- Fred Brooks: throw one away
- The prototype evolves into the final system
  - take the first one that comes along and keep changing it until it works
  - could be trial and error, or dangerously conservative
- New ways to create tighter direct linkage between “HCI issues” and implemented designs
  - OO modeling enable simultaneous attention to user task and software design issues
  - Via “use cases” (scenarios)
Agile Methods and XP

- Make evolutionary development fast and personal
  - no one method effective across all projects
  - for a given context, identify “lightest” method possible
  - particularly for Web-based business applications
- Extreme Programming: pair-based coding teams
  - begin by writing “user stories” (scenarios again)
  - then just start coding, with partner providing critique
    (real users may be present or even participate)
  - highly incremental, relies heavily on testing