Prototyping

DECO 1200
Outline

Prototypes
  What is a prototype? Why prototype?

The process of prototyping
  Compromises and issues in prototyping

Conceptual design
  Developing conceptual models based on requirements

Physical design
  Developing final designs based on conceptual designs
What is a Prototype?

A prototype allows stakeholders to interact with an envisioned product, to gain some experience of interacting with it in a realistic setting, and to explore imagined uses.

Prototypes can be anything from a simple storyboard through to a complex piece of software, and from a cardboard mock-up to a plastic replica.
Examples of Physical Prototypes

**PalmPilot**

While developing the idea of the PalmPilot, Jeff Hawkin carved a piece of wood into the proposed shape of the PDA and pretended to enter information into it to see what it would be like to use such a device in context.

**Desktop Printers**

Before laser printers became common in offices, some researchers used cardboard boxes to prototype what it would be like to have a laser printer on a person’s desk so that intended users could imagine what it would be like to have a such a printer on their desk.
Why Prototype?

Prototypes can serve several functions during the design process:

Act as a discussion point with users (stakeholders)
- It is much easier for users to describe what they like and/or need when they have a something tangible

Provide a way to communicate within a design team
- Building prototypes encourages design teams to reflect upon their design ideas as they are developed

Allow a designer to test out ideas early in the process
- Prototypes allow designers to test out the technical feasibility of their ideas before committing resources
Low-Fidelity Prototyping

Low-fidelity prototypes do not look or act very much like the final product

Low-fidelity prototypes often use materials that are very different from those intended for the final version

  e.g. the wooden slab used to prototype the PalmPilot
  or the cardboard boxes used to prototype printers

Low-fidelity prototypes tend to be simple, cheap and quick to produce/modify

Low-fidelity prototypes support the exploration of design alternatives in the early stages of development

Low-fidelity prototypes are not intended to be kept or integrated into the final product
Storyboarding

Storyboarding is a low-fidelity prototyping technique often used with scenarios.

A storyboard consists of a series of sketches showing how a user might use a product to complete a task.

A storyboard may consist of a series of sketched screens for a piece of software or a series of scenes showing a user interacting with a product.

Storyboards allow users to “role-play” and imagine performing task with a product.

Storyboards used in conjunction with scenarios can provide additional details about product interactions.
An Example Storyboard

1. Drive car to gas pump
2. Take nozzle from pump...
3. ... and put it into the car’s gas tank
4. Squeeze trigger on the nozzle until tank is full
5. Replace nozzle when tank is full
6. Pay cashier
Index Cards

Index cards can be a useful alternative to storyboards because they can be reordered.

Each index card can represent a single screen from a software system or a single element of a task.

Using index cards allow a user to evaluate a prototype by stepping through the cards.

Because the index cards can be visited in a variety of orders they are particularly good for prototyping non-linear systems like websites where hyperlinks are modelled as moves from one card to another.
Example of a Index Card Prototype
Wizard of Oz

Wizard of Oz is a low-fidelity prototyping technique for testing ideas for software. The user sits at a computer screen and interacts with a software interface as if interacting with the final product, but instead of running the real software, the responses from the prototype are generated by a person sitting at another machine connected to the prototype.
High-Fidelity Prototyping

High-fidelity prototypes are intended to look and feel much like the final product.

A high-fidelity prototype for a piece of software might be a cut-down version of it created in Visual Basic, for a physical device it might be a “dummy version” with non-functioning buttons, e.g. a dummy mobile phone.

Software prototyping tools are often full-featured development environments, e.g. Visual Basic, Macromedia Director, Smalltalk, etc.
Problems with High-Fidelity Prototyping

Low-fidelity prototype are often preferable to high-fidelity prototypes because:

- High-fidelity prototypes can take too long to build
- Commentators tend to focus on superficial details
- Developers become reluctant to change things
- High-fidelity prototypes can set expectations too high
- One problem with a prototype can halt all testing

High-fidelity prototypes are often best used to market ideas or test out technical issues
Compromises in Prototyping

Prototypes always focus on a subset of a design’s features and involve compromises.

- Compromises in low-fidelity prototypes are often obvious, e.g., the prototype does not function.
- Compromises in high-fidelity prototypes can be hidden from evaluators, e.g., “quick-and-dirty” code.
Approaches to Prototyping

Two common compromises are breadth vs depth of the functionality prototyped

- These compromises lead to two different approaches in the ways that prototypes are used

Horizontal Prototyping

- Produce lots of prototypes to try out lots of ideas

Vertical Prototyping

- Produce a few prototypes to lots of design details
Evolutionary vs Throw-Away Prototyping

Low-fidelity prototypes are always intended to be thrown away at the end of testing. Often low-fidelity prototypes can’t be developed into a final design because of the choice of material etc.

Software prototypes can be developed to be evolved over time into the final product. Some application development environments have been developed to support the evolution of products from early prototypes to final products, e.g. Ruby-on-Rails. Rapid evolution of products from early prototypes to final products can be particularly important for start-up companies that must get their product to market as quickly as possible and can fix problems later.
Conceptual Designing

Conceptual design transforms user needs and requirements into a conceptual model. A conceptual model is a description of a product in terms of a set of ideas and concepts about what it should do, how it should behave, and what it should look like, that will be understandable to potential users.

Prototypes can often play an important role in the conceptual design process.
Guidelines for Conceptual Designing

Keep an open mind but never forget the users and their context

Discuss ideas with other stakeholders as much as possible

Use low-fidelity prototyping techniques to get rapid feedback

Iterate, iterate, iterate
Understanding the User Experience

Designers can go to extremes in an attempt to understand the user’s experience.

While trying to design a chest-implanted defibrillator, a group of designers tried to understand the disruption to a person’s life that might occur as the result of an event like an unexpected defibrillation by using pagers that would go off at random times of the day.
Ford’s Third Age Suit

Designing for an ageing population
A group of researchers at Loughborough University developed the “Third Age Suit” to simulate the reduced mobility and additional effort typically experienced by people over 55, so that designers could get a first-hand experience of what it was like to be in that age group when designing new products.
Interaction Modes

Which interaction mode is most suitable for supporting user activities?

Instructing
The user issues commands to the system

Conversing
The user enters into a dialogue with the system

Manipulating & Navigating
The user manipulates objects in the system’s interface

Exploring & Browsing
The user explores the contents held in the system

Conceptual models will typically involve a combination of multiple interaction modes
Process-Oriented vs Product-Oriented

Conceptual models for software products are often process- or product-oriented

Process-oriented software supports the monitoring and maintenance of a process
  e.g. control software for industrial plants, financial management packages, web browser

Product-oriented software supports the creation of works by users
  e.g. Microsoft Word, Adobe Photoshop, Macromedia Flash, etc.
Interaction Metaphors

Is there a suitable interface metaphor that will help users understand the product?

Interface metaphors should be familiar to the users of a product and support their understanding.

Three steps to choosing interface metaphors

Understand what the system will do

- Identify functional requirements and develop partial conceptual models to try them out on users

Understand the potential problems in the system

- Identify which tasks or sub-tasks cause problems, are complicated, or are complex

Generate appropriate interface metaphors

- Look for metaphors in user descriptions of tasks
Evaluating Interface Metaphors

How much structure does the metaphor provide?
Good metaphors provide structure for thinking about systems.

How much of the metaphor is relevant?
Users can apply inappropriate elements of a metaphor.

Is the metaphor easy to represent?
Good metaphors can be represented in multiple ways.

Will users understand the metaphor?
Is the metaphor something that users are familiar with?

Can the metaphor be extended if needed?
Does a metaphor have aspects that could be useful later on?
Interaction Paradigms

Which interaction paradigms help designers think about the product being developed?

- Desktop computing
  - Classic applications with graphical user interfaces
- Ubiquitous computing
  - Interfaces embedded into familiar objects
- Pervasive computing
  - Interfaces designed to be accessible from everywhere
- Wearable computing
  - Interfaces embedded into wearable objects
- Tangible bits
  - Real objects that represent digital information
- Attentive environments
  - Environments that monitor and interact with visitors
Expanding the Conceptual Model

What functions will the product perform?
How should the functions required to complete a task be broken down between users and the system?

How are the functions related?
Do some functions/tasks need to be completed before others can be started? This information can be used to possible inform which functions should be available at different stages in the completion of a task.

What information needs to be available?
What data is required to perform a task? How does the system transform the data? How will the system be able to access the data it requires?
Using Scenarios in Conceptual Design

Scenarios can be used to describe existing work situations and practices

Users of existing systems can write scenarios that explain how they currently work to help identify challenges and opportunities for a new system.

Scenarios can be used to express proposed or imagined situations

Users and other stakeholders can write scenarios to describe how they would like a new system to work without having to understand the technical details.
Roles for Scenarios

As a basis for the overall design of a system
Scenarios can be used to describe the complete functionality of some systems

To specify a technical implementation
Scenarios can be used to describe the technical requirements of a system

To support co-operation in design teams
Scenarios can be used to provide a simple way for team members to communicate possible situations

To assist communication across disciplines
Scenarios can be used to describe a system without using discipline specific jargon
Using Prototypes in Conceptual Design

Producing prototypes often requires some consideration of the details of the design. Producing a prototype will often force a designer to consider the practicalities of implementing an idea.

Prototyping is often used to get feedback on an emerging design from users and designers. Low-fidelity prototypes are most often used in the early conceptual stages of design and high-fidelity prototypes are used later in the design process when some of the details have been worked out based on earlier feedback.
Physical Design

Physical (or detailed) design involves the development of detailed elements
  e.g. screen layout, icon design, menu structure, etc.

The detailed design stage will involve many system specific decisions
Guidelines for Physical Design

Be Consistent
   Design interfaces to be consistent
Shortcuts
   Allow experts to use shortcuts, e.g. accelerators
Provide Feedback
   Update users on progress/completion of tasks
Avoid Errors
   Prevent users from making common errors
Reversible Actions
   Allow user actions to be easily reversed
User Control
   Let the user stay in control of the system
Reduce Memory Load
   Reduce the amount that users have to remember
Guidelines for Designing Interfaces

Design Principles
Principles that embody design-related information derived from theory that can be used to guide design

Rules
Rules provide more detailed guidance than design principles that can be applied in specific situations

Style Guides
Styles guides often combine rules and principles and are published to encourage consistent look and feel

Standards
Standards are maintained and published by international organisations often to promote interoperability
Design Patterns

Design patterns are used to capture design practices and common design solutions

Design patterns were first introduced in architecture and have become very popular in software engineering

Design patterns can provide a vocabulary for designers to communicate the designs

e.g. using the Factory design pattern is a useful way to construct user interface screens as required
Designing for the Web

The design of web pages should make the answers to the following questions clear:

Where am I?
Where is the page located on the web? Where is the page located within a web site? Who owns the page?

What’s here?
What is the main content of the web page? What are the most important points made in the content?

Where can I go?
Where can a visitor go from the current page to get to other potentially interesting pages on the web site?
Tool Support

Ideally interface design software tools should support the following tasks:

- Designing user interfaces based on user tasks
- Implementing user interfaces based on a design
- Creating consistent easy-to-use user interfaces
- Rapidly exploring different design alternatives
- Allowing non-programmers to design user interfaces
- Evaluating a user interface and suggest improvements
- Allowing a user to customise a user interface
- Deploying user interfaces across multiple platforms
Summary

Prototypes
Low-fidelity and high-fidelity prototyping

Compromises and issues in prototyping
e.g. evolutionary vs throw-away prototyping

Conceptual design
Developing conceptual models based on requirements

Physical design
Developing final designs based on conceptual models