Course Overview

- Introduction
- Understanding Users and Their Tasks
- Principles and Guidelines
- Interacting With Devices
- Interaction Styles
- UI Design Elements
- Visual Design Guidelines
- UI Development Tools
- **Iterative Design**
- Project Presentations and Selected Topics
- Case Studies
- Recent Developments in HCID
- Conclusions

Chapter Overview Iterative Design and Prototyping

- Motivation
- Objectives
- Iterative Design
  - Software Engineering Life Cycle
- Prototyping
  - Prototypes
  - Prototyping Techniques
  - Benefits and Drawbacks
- Important Concepts and Terms
- Chapter Summary

Motivation

- the design of the user interface influences the overall design and development cycle considerably
- the user interface determines the impressions of the users about the overall product
- iterative design and prototyping allows evaluations to be done as early as possible
  - mock-ups, scenarios, prototypes, …
- prototyping, testing and evaluation can be expensive
  - correcting errors late in the development process is even more expensive
  - for many software systems, modifications based on dissatisfied users are a very large part of the overall costs

Objectives

- to understand how the design and development of the user interface fits into the overall software development cycle
- to know the important methods for the development of user interface prototypes
- to understand the importance of prototypes for early evaluation

Prototypes

- simulate the structure, functionality, or operations of another system
  - represent a model of the application, service, or product to be built
  - may or may not have any real functionality
  - can be either paper based or computer based

 [Mustillo]

Paper-based Prototypes

- cheap
- low fidelity
- can often be useful to demonstrate a concept
  - e.g., a back-of-the-envelope sketch
- can not show functionality so that users can actually interact with them

 [Mustillo]

Computer-based Prototypes

- higher fidelity than paper based
- can demonstrate some aspect with varying degrees of functionality
- can offer valuable insights into how the final product or application may look like

Why Prototype?

- part of the iterative nature of UI design
- 20%-40% of all system problems can be traced to problems in the design process
- 60%-80% can be traced to inaccurate requirements definitions
- cost of correcting a problem increases dramatically as the software life cycle progresses

Prototyping Techniques

- low-fidelity prototypes
- high-fidelity prototypes

Low-fidelity Prototypes

- low-fidelity prototypes
  - cheap, rapid versions of the final system
  - limited functionality and/or interactivity
  - depict concepts, designs, alternatives, and screen layouts rather than model user interaction with a system
    - e.g. storyboard presentations, proof-of-concept prototypes
  - demonstrate the general ‘feel and look’ of the UI
    - their purpose is not to show in detail how the application operates
  - are often used early in the design cycle
    - to show general conceptual approaches without investing too much time or effort

High-fidelity Prototypes

- high-fidelity prototypes
  - fully interactive
    - users can enter data into entry fields, respond to messages, select icons to open windows, and interact with the UI
  - represent the core functionality of the product’s UI
  - typically built with 4GLs such as Smalltalk or Visual Basic
    - can simulate much of the functionality of the final system
  - trade off speed for accuracy
    - not as quick and easy to create as low-fidelity prototypes
    - faithfully represent the UI to be implemented in the product
    - can be almost identical in appearance to the actual product

Comparison

Type	***Advantages***	***Disadvantages***
Low-Fidelity Prototyping	Lower development cost Evaluate different design concepts Useful communication vehicle Addresses screen layout issues Useful for identifying market requirements Proof of concept	Limited error checking Poor detailed specification for coding Facilitator driven Limited usefulness after requirements established Limitations in usability testing Navigational & flow limitations
High-Fidelity Prototyping	High degree of functionality Fully interactive User driven Defines navigational scheme Useful for exploration & testing Look and feel of final product Serves as a living specification Marketing and sales tool	More expensive to develop Time consuming to build Inefficient for proof of concept designs Not effective for requirements gathering

Fidelity Requirements

- recent study by Cantani and Biers (1998) investigated the effect of prototype fidelity on the information obtained from performance test
  - 3 levels of prototypes:
    - paper - low fidelity
    - screen shots - medium fidelity
    - interactive Visual Basic - high fidelity

Case Study (cont.)

- 30 university students performed 4 typical library search tasks using one of the prototypes
  - total of 99 usability problems were uncovered
- no significant difference in the number and severity of problems identified, and a high degree of commonality in the specific problems uncovered by users using the 3 prototypes
  - Catani, M.B., And Biers, D.W. (1998). Usability Evaluation and Prototype Fidelity: Users and Usability Professionals. Proceedings of the Human Factors and Ergonomic Society, 42nd Annual Meeting, 1331-1336.

Low-fidelity Prototyping

- identify key market and user requirements
- provide a very high-level view of the proposed UI and service concept
- provide a common language or vision
- develop a common understanding with others
- investigate early concepts and ideas independently of platform, technology, and other issues
- evaluate design alternatives
- get customer support during requirements gathering
- elicit user input prior to selecting a design

High-fidelity Prototyping

- create a living specification for programmers and customers
- make an impression with customers to show how well the product, service, or application will operate
  - prior to the code being fully developed
- test UI issues prior to committing to a final development plan
  - e.g., error handling, instructions

Software Prototypes

- actually work to some degree
  - not an idea or drawing
- must be built quickly and cheaply
  - throw-away - thrown away or discarded immediately after use
  - incremental - separate components, added to the system
  - evolutionary - may eventually evolve into the final system
- may serve many different purposes
  - elicit user reactions, serve as a test bed
- integral part of an iterative process
  - includes modification and evaluation

Levels of Prototyping

- full prototype
- horizontal prototype
- vertical prototype
- scenarios

Full Prototype

- contains complete functionality
- lower performance than the final system
  - e.g. trial system with a limited number of simultaneous users
  - may be non-networked, not fully scalable, ...

Horizontal Prototype

- demonstrate the operational aspects of a system
  - do not provide full functionality
  - e.g. users can execute all navigation and search commands, but without retrieving any real information as a result of their commands
- reduced level of functionality
  - all of the features present

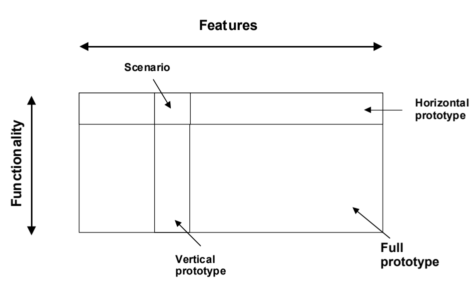
Vertical Prototype

- contain full functionality, but only for a restricted part of the system
  - e.g., full functionality in one or two modules, but not entire system
  - e.g. in an airline flight information system, users can access a database with some real data from the information providers, but not the entire data
    - in other words, they can play with a part of the system
- reduced number of features, but with full functionality

Scenarios

- both the level of functionality and the number of features are reduced
- very cheap to design and implement
- but, only able to simulate the UI as long as the test user follows a previously plan test
- small, can be changed frequently and re-tested
- reduced level of functionality and reduced number of features

Diagram Levels



Levels of prototyping.

Chauffeured Prototyping

- involves the user watching while another person 'drives' the system
  - usually a member of the development team
- the system may not yet be complete enough for the user to test it
  - it is nevertheless important to establish whether a sequence of actions is correct

Wizard of Oz

- a person hidden to the user provides feedback for the system
- user is unaware that he/she is interacting with another user who is acting as the system
- usually conducted very early in development
  - to gain an understanding of the user's expectations

Testing of Prototypes

- structured observation
  - observe typical users attempting to execute typical tasks on a prototype system
  - note number of errors and where they occur, confusions, frustrations, and complaints
- benchmarking
  - oriented toward testing the prototype UI or system against any pre-established performance goals
    - example: error-free performance in less than 30 min

Testing of Prototypes (cont.)

- experimentation
  - two or more UI design (prototype) alternatives with the same functionality are directly compared
  - the one that leads to the best results is selected for the final product

Benefits of Prototyping

- integral part of the iterative design process
- permits proof of concept/design validation
- raises issues not usually considered until development
- provides a means for testing product- or application-specific questions that cannot be answered by generic research or existing guidelines
- permits valuable user feedback to be obtained early in the design process

Benefits of Prototyping (cont.)

- qualitative and quantitative human performance data can be collected within the context of the specific application
- provides a relatively cheap and easy way to test designs early in the design cycle
- permits iterative evaluation and evolving understanding of a system, from design to the final product
- improves the quality and completeness of a system's functional specification
- substantially reduces the total development cost for the product or system

Drawbacks

- inadequate analysis
  - inadequate understanding of the underlying problem
  - the lack of a thorough understanding of the application, service, or product being developed
- the prototype may look like a completed system
  - customers may get the mistaken idea that the system is almost finished, even when they are told very clearly that it is only a prototype
- unattainable expectations
  - unrealistic expectations with respect to actual product performance
- ignoring reality
  - limitations and constraints that apply to the real product may often be ignored within the prototyping process
    - e.g., network constraints

Drawbacks (cont.)

- users that are never satisfied
  - users can ask for things that are beyond the scope of the project
- viewing the prototype as an exercise
  - developers may develop the wrong thing
  - at great effort and expense
- the trap of over-design or under-design
  - "just one more feature ..."
  - "this is just the prototype, we'll fix it when we develop the product"

 [Mustillo]

Important Concepts and Terms

- active intervention
- analytic evaluation
- benchmarking
- co-discovery
- cognitive walkthrough
- contextual inquiry
- evaluation
- experimental evaluation
- expert evaluation
- focus group
- formative evaluation
- heuristic evaluation
- horizontal prototype
- human factors engineering
- interview
- prototype
- questionnaire
- rapid prototyping
- scenario
- summative evaluation
- survey
- testing
- usability
- user interface design
- user observation
- user requirements
- vertical prototype
- walkthrough

Chapter Summary

- user interface design is an integral part of the overall development process, and determines to a large degree the impression of the system on the user
- prototyping allows the testing and evaluation of important aspects in early stages of the development cycle
- testing and evaluation are important activities to be performed as early as possible, and throughout the development cycle
- the emphasis should be on the user
  - user-centered design and evaluation

