CS 66-460
Introduction to Graphical Human Machine Interfaces

The Usability Engineering Lifecycle

MONDAY, FEBRUARY 24, 1997

Instructor: G. Bowden Wise

Rensselaer Polytechnic Institute
Spring 1997

Usability Engineering

- like software engineering, is a process for developing software to ensure high quality
- user interface activities cannot be just tacked on at the end of design
  - UI guys try to fix up the interface, "beautify" the screens
  - does not work because usability impacts overall design
- instead, usability engineering
  - is a set of activities that ideally take place throughout the lifecycle of the product
  - consists of significant activities happening at early stages before the user interface has ever been designed

Lifecylce Stages

1. Know the user
   (a) individual user characteristics
   (b) the user's current and desired tasks
   (c) functional analysis
   (d) the evolution of the user and the job
2. Competitive analysis
3. Setting usability goals
   (a) financial impact analysis
4. Parallel Design
5. Participatory Design
6. Coordinated design of the total interface
7. Apply guidelines and heuristic analysis
8. Prototyping
9. Empirical testing
10. Iterative design
    (a) capture design rationale
11. Collect feedback from field use

Lifecycle Model

- not all steps may be necessary in all projects
- usability activities can be prioritized subject to varying levels of resource constraints
- emphasizes that one should not rush into design
- do as much possible before design is started
- won't have to change the design to comply with the usability requirements

What is Design?

- Analysis of alternative activities to see if they are practical
- Conducting investigations to fill in gaps in knowledge
- Drawing up draft specifications
- Making performance and cost estimates
- Building and testing prototypes
- Modifying design and revising the prototype as necessary

Know The User

- study the intended users and the use of the product
  - best if developers interview/observe real users, visit customer sites
- the concept of "user" is defined to include everybody whose work is affected by the product in some way, including the users of the system's end product or output even if they never see a screen
- users can include installers, maintainers, system administrators, and other support staff in addition to the people who sit at the keyboard
- often difficult to get access to users
  - the development company wants to hide developers
  - reluctance of sales reps to let anyone talk to "their" customers
  - reluctance of users to spend time or be observed
- your goal is to get direct access to users and avoid indirect interaction and heresay

Know The User: Iindividual user characteristics

- class of people
  - a select group (physicists)
  - a broad class of people (e.g., ATM)
  - entire population
- work experience
- education
- age
- gender
- novice vs expert computer users
- work environment
- social context of use
- time for learning, training
- user profiles may also come from market analysis or observational studies one may conduct as part of task analysis
- don't rely totally on written information ... observe and talk to real users

Know The User: Task Analysis (1)

- Very important!
- What tasks will the users do? How do they do it? What do they need?
- study
  - the user's overall goals
  - how users currently approach the task
  - how users deal with exceptional circumstances or emergencies
- information needs: what does the user need to know or view to do this task?
  - Includes what needs to be on the screen.
  - Both: What does the system need to show? What does the user need to know?
- interview and observe users, can also interview user's clients
- Look for problems in the current mechanisms for achieving the tasks that are susceptible to automation.
- identify user's model of the task, since it can be a source for metaphors for the user interface
- identify weaknesses of the current situation

Know The User: Task Analysis (2)

- Ask users to show concrete examples of work products
- Observe users working on real problems,
- Larger tasks and goals often broken down into sub-tasks and sub-goals
- when a user says "...then I do this..." ask
  - "Why do you do that?" to relate the activity to larger goals
  - "How do you do that?" to decompose the activity into subtasks
- other good questions
  - "Why do you not do it in such a such a manner?"
  - "Do errors ever occur when doing this?"
  - "How do you discover and correct these errors?"
- Users should be asked to describe
  - exceptions, emergencies
  - notable successes
  - what they like/dislike
  - what changes they would like
  - what would they improve

Know The User: Task Analysis (3)

- Build up scenarios of typical uses from the task analysis:
  - Specific example of how a user might use the system.
  - One scenario for each major class of users doing each kind of important task
  - Will want to make those tasks efficient and easy
  - What is important to optimize?
  - Will significantly affect the design
  - Try to include lots of exceptional cases
  - Shows how the interface will be used
  - Uses:
    * Refine the interface
    * Demonstrate to management, marketing, customers what your concept is
    * Can replace much textual specification

Know The User: Task Analysis (4)

- Output of task analysis can include
  - scenarios to be used during design
  - a list of all the things the user wants to accomplish (goals)
  - information they will need to achieve these goals (preconditions)
  - the steps that need to be performed
  - interdependencies between the steps
  - all the various outcomes and reports that need to be produced
  - the criteria used to determine the quality and acceptance of these results
  - the communication needs of the users as they exchange information with others while performing the task or preparing to do so

Know The User: Functional analysis

- Analyze not just the way users are currently do a task, but also the underlying functional reason for the task:
- What is it that really needs to be done,
- What are merely surface procedures which can, and perhaps should, be changed

  e.g., design of on-line documentation based on analysis of reading of printed manuals

  users turn pages frequently

  naive interpretation: system need fast scrolling/paging

  functional analysis: users turn pages a lot because it is hard to find what they want
- limit to how drastically one can change how users currently approach their task, so functional analysis should be coordinated with task analysis

Know The user: The Evolution of the User

- Users will not stay the same
- As users gain more experience with the system they will use the system in new ways e.g., spreadsheets (ledgers ... databases)
- Impossible to forecast these changes completely
- A flexible design will accommodate these new uses
- Experts want interaction shortcuts (accelerators)
- Important not to design a system just for the way the user will use the system in the first short period after its release.

Competitive Analysis

- "Know the competition"
- Competing products can serve as "prototypes" of your own product
- Analyze existing products heuristically according to established usability guidelines
- Perform empirical user tests with these products.

  Learn how well its functionality and interaction techniques support the kinds of tasks the planned new product is expected to support based on the initial analysis of the intended users
- Perform comparative analyses of features of several competing products, such as, various user interface design issues for the kind of product being studied

  Will provide ideas for the new design and give a list of ad hoc guidelines for approaches that seem to work and those that should be avoided.
- Can also read trade press reviews of competitors products
- Can also study non-computer interfaces for insights
- Importance of various features, issues

Goal Setting

- What does it mean to be “easy to use”
- Some proposed definitions:
  - “I like it”
  - “I always do it that way”
  - “That is the way the xxx system does it”
  - “It is easy to implement”
- Much better:
  - Can be learned in less than 20 minutes.
  - User will perform 30 error-free operations per minute.
  - The error rate will be lower than 1 per 40 operations.
  - Tasks will be performed in 30% of the time it took before the system was used.
  - Users will have a high satisfaction with the system as measured by a survey.
- Explicit, specific, measurable metrics.
  - Tradeoffs, so have to pick relevant metrics.

Goal Setting (2)

- Usability has many different aspects
  - learnability: time to learn to do specific tasks (specific proficiency)
  - efficiency: (expert) time to execute benchmark typical tasks. Throughput
  - errors: error rate per task; time spent on errors; error severity
  - subjective satisfaction: questionnaire
  - memorability: casual users
- Normally not all usability aspects can be given equal weight
- So we have to prioritize by setting goals for each aspect
- Pick levels for each aspect:
  - minimum acceptable level
  - desired (planned) level
  - theoretical best level
  - current level or competitor’s level

Financial impact analysis

- Analyze the financial impact of the usability of the system.

```
cost of system per hour =
   number of users * loaded salary per hour * hours using system
```

- Estimate savings of reduced training, error time, need for support staff, etc.
- Tells how much time to spend on usability
- Example:
  - F. Montaniz and G. Kissel, "Reversing the Charges," ACM interactions, vol. 2, no. 3, July 1995, pp. 29-33.

  - Cost of Usability Specialists to identify bad error messages = \$533
  - Savings from improving the error messages = \$411,918
    From reducing calls to support person-el over 6 months

Parallel Initial Design

- Have several designers work out preliminary designs
- The goal is to explore different design alternatives before one settles on a single approach that can be developed in further detail
- Each designer works independently
- Generate new combined designs by incorporating the best features from the independent designs
- Diversified parallel design
  - each designer concentrates on a different aspect of the design problem e.g., novice vs expert users
- Important to perform parallel design in novel systems in which their is little information about what interface approaches work best
- Parallel design is a very cheap way of exploring the design space because most of the ideas will not need to be implemented, which might not be the cause if the design was explored later during iterative design

Iterative Design Process

- Participatory Design
  - Users involved in the design process through regular meetings.
  - Users are good at reacting to concrete designs and prototypes.
  - Specifications must be in a form the user can understand (e.g., paper mock ups, verbal discussions)
- Coordinating the Total Interface for Consistency
  - Include documentation, help, etc.
  - May need a centralized authority on each project to coordinate various aspects of the interface.
  - Interface standards can also be used to promote consistency.
  - Consistency can be increased through code re-use and or through a constraining development environment
- Guidelines and Heuristic Evaluation
  - Evaluate your interface according to the guidelines.

Iterative Design Process (2)

- Make a Prototype or partial implementation of the system early and quickly
  - Actually is faster to prototype first
- Redesign the interface based on feedback from evaluation
- New design may be worse or may break something else.
- Keep track of reasons for design decisions
  - Called "Design Rationale"
  - So don't need to keep revisiting the same decisions
  - When future conditions suggest changing a decision will remember why made that way and what implications for change are.
  - Instead of arguing about a design feature, figure out what information would tell you which way to go
  - Experiment, marketing data, etc.

Iterative Design Process (3)

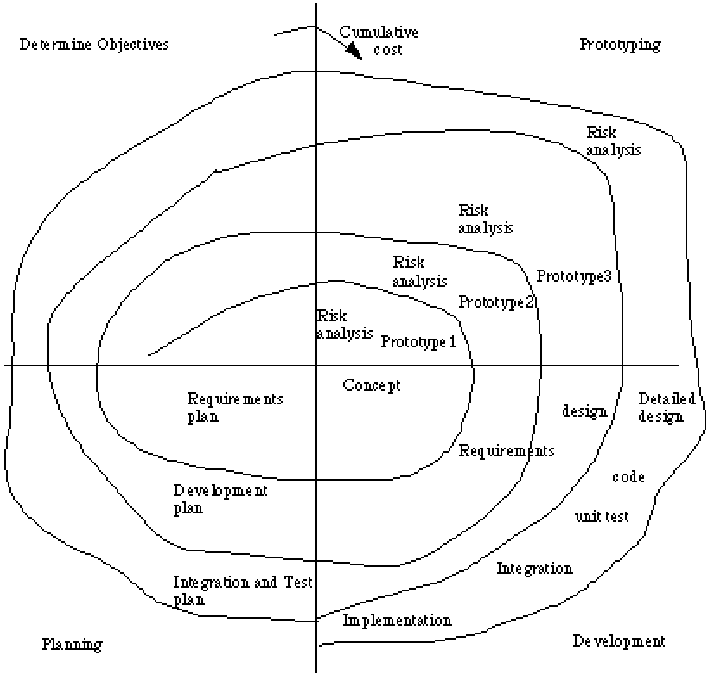
- Implementation Phase:
  - Waterfall model: 1970:

```
Requirements -> Design -> Detailed Design -> Code -> Integration
             -> Implementation -> Test
```

    fully elaborated documents as completion criteria for stages
  - “Spiral” model: 1988
    Barry Boehm, ”A Spiral Model of Software Development and Enhancement”, IEEE Computer, 12(5), May, 1988, pp. 61-72

  - Follow-up after release
    * For the next version
    * From bug reports, trainers, initial experiences

The Spiral Model



Prototyping

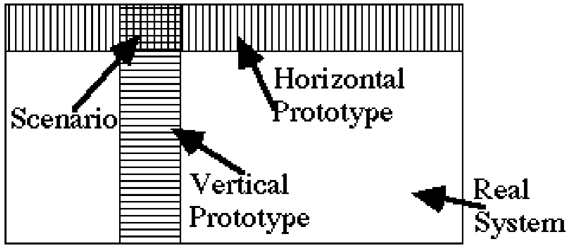
- Prototypes are cheaper because
  - Not worried about efficiency
  - Accept less reliable code
  - Use simplified algorithms
  - Fake data
  - Might not need to implement anything, fake the system

Prototyping (2)

- Uses of Prototypes:
  - What questions will the prototype help you answer?
  - Is this approach a good idea?
    - Usually only need to test a few people for test:
    - Most results with first 3 people
    - Can refine interface after each test
  - Look what a cool design we have!
  - Transfer design from UI specialists to programmers
    - Often better than written specifications
  - Design A versus Design B
    - Rare, except in academic environments
  - What are the real requirements and specifications?

Prototyping(3)

- Types of Prototypes
  - vertical: fewer features, but realistic on part
  - horizontal: overview of complete system, but shallow functionality



Prototype Methods

- paper mock-ups
  - designer plays "computer"
  - also called low fidelity prototypes
- wizard of oz
  - Subject at a computer, but person is at another computer operating the interface.
  - "Pay no attention to that man behind the curtain"
  - Especially for AI and other hard-to-implement systems
- storyboarding
  - depict sequences or snapshots of the interface
  - might be a sequence of drawings or paper mock ups or Director animations

Prototype Methods (2)

- scenarios
  - ultimate "minimalist" prototype
  - describe a single interaction session without any flexibility for the user
  - combine limitations of both
    * horizontal: users cannot interact with real data
    * vertical: users cannot move freely through the system
  - a scenario is an encapsulated description of
    * an individual user
    * using a specific set of computing facilities
    * to achieve a specific outcome
    * under specified circumstances
    * over a certain time interval
  - main uses
    * used during design to understand how users will interact with a future system
    * used during early evaluation to get user feedback without the expense of constructing a running prototype
    * may also be used for testing if developed properly

Prototype Methods (3)

- PICTIVE = plastic interface for collaborative technology initiatives through video exploration
  - designs are put together using multiple layers of sticky notes and overlays that can be changed by simple colored pens
  - use video to convey the result of the sequence of overlays
  - good for use in participatory design since the low tech nature of the materials make them equally accessible to users and to developers.
- interactive prototyping using prototyping tools
  - HyperCard, etc.
  - Visual Basic
  - MacroMind Director
  - Landay's SILK
  - Interface Builders and other tools for "Real" code
    * C++

```
Use of the Prototype                 Tools
==================================   ================================
Working out initial ideas            Paper, SILK
Demonstration of the concepts        Paper, Director, Hypercard, VB
Experiments with users               Paper, ~Director, Hypercard, VB
Release "experimental" version       ~Hypercard, VB, C++
Real, delivered system               C++
```