CS5714 Usability Engineering

Dr. H. Rex Hartson

Introduction to the Course

Topics

- Motivation
- Objectives of course
- Product and process
- Interaction design vs. software design

Great course coming up!
The Need for Good User Interfaces

- Costs of hardware & software vs. “personware”
- To users, the interface is the system
- Communication vs. computation

Motivation

- The kind of “warranty” we get with most software we buy:

“The software is provided without warranty of any kind. The manufacturer does not warrant that the functions contained in the software will meet your requirements, or that the operation of the software will be uninterrupted or error-free, or that defects in the software will be corrected.”
Is the Fuss Over Usability Real?

What is Usability?

- **Usability** is about the quality of user experience, that includes:
  - How easy to use and
  - How useful that system is

- Usability includes
  - Effectiveness
  - Efficiency
  - Learnability
  - Safety
  - User satisfaction
What Usability Engineering is NOT

- It is NOT “dummy proofing”
- “Doing usability” is NOT (just) usability testing
- Usability is NOT (just) “user friendliness”

What is Usability Engineering?

- *Usability engineering* is a successful, iterative, cost-effective, user-centered development process that ensures a high level of usability
- Creating a complex interactive system requires more than systems and software engineering
How Can You Know if you Have Good Usability?

- Cannot measure usability directly; must measure indicators
  - Speed of user task performance
  - User error rate
  - Subjective user satisfaction
  - Ease of learning
  - Retention over time
  - Usability “in the large”: Ease of use, plus usefulness

Objectives of this course

Course is designed to help you develop more usable interaction designs, including graphical user interfaces (GUIs) and Web applications, by:

- Understanding and applying interaction design guidelines
- Applying an iterative, evaluation-centered usability engineering life cycle
Objectives of this course

- Participating in *systems analysis*, including user, needs, task, and functional analyses
- Doing conceptual and detailed *design*
- Establishing *usability specifications*
- Building *rapid prototypes*
- Performing formative *usability evaluation*
- Iteratively *refining* the interaction design
- Knowing how to get started with these new ideas

Product & Process

People who develop UIs don’t intentionally make user interfaces that others hate and can’t use!

- **High usability requires:**
  - **Product:** content, human factors of an interaction design; design principles and guidelines ("what")
  - **Process:** usability engineering techniques for developing an interaction design ("how")

Significant cause of poor usability in product is the lack of understanding and application of proper development *process*
Interaction Design is Not Software Design

- Developing a user interface involves:
  - **Interaction component** – How a user interface works, its "look and feel" and behavior in response to what a user hears, sees, and does
  - **Interface software component** – Code that implements the interaction component

<table>
<thead>
<tr>
<th>Development of the user interface</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Development of user interaction component</td>
<td>UI software requirements</td>
</tr>
<tr>
<td>Problems, constraints</td>
<td>Development of user interface software component</td>
</tr>
</tbody>
</table>

- Premise: Describing interaction from user’s view should result in more usable design than describing it from software or programmer view
- Inherent conflict of interest: What’s best for the user is seldom easiest for the developer!
- “One head, two hats” to emphasize different roles
Popular Misconceptions About Usability Engineering

- Usability engineer is a building inspector (or UI police)
  - Fear and dread, but no respect
- Usability engineer is a “priest in a parachute”
  - Drop into project, bless it, and leave quickly
- The “Peanut Butter Theory” of usability
  - It can be spread on, after the design is done
Topics

- Connections of usability engineering to software engineering
- Development activities in the usability engineering life cycle
- Usability management
- User interface development team
Typical Software Engineering LC

- The Waterfall Model

The Process of User Interaction Development

- Start with basic software engineering concept:
  - Distinction between design and implementation
The Process of User Interaction Development

- Adding systems analysis, testing, and problem (application) domain design

![Diagram of the process](image)

Main feedback: design flaws, errors, modifications

Major reconsiderations

The Process of User Interaction Development

- Analogous activities for user interface development

![Diagram of the process](image)

Main feedback is due to low usability: design flaws, errors, modifications

Major reconsiderations
The Process of User Interaction Development

- Connecting the processes together and adding rapid prototyping

The big picture
The Process of User Interaction Development

- The rest of this course is about just this part:

Usability Engineering Process

- Must be iterative
  - It’s not possible to get it right the first time (in any complex design domain)
  - Need the artillery approach:

  **Ready, Fire, Aim!**
Usability Engineering Process

- New life cycle concept comes from:
  - The waterfall model – movement toward completion
  - Star (Hartson & Hix, 1989) – evaluation centered
  - LUCID (Cognetics, Inc.) – development activities
  - Boehm’s Spiral Model – iteration
  - Helms & Hartson (2001) – put it together

Usability Engineering Life Cycle

- Intertwined and not really separable
Usability Engineering Life Cycle

**Basic principles**
- Process is product-oriented
- Each part of the cycle is iterative
- Each part of the cycle is evaluation-centered
- Any part of process is instance of what is possible
  - Pick and choose cycles, activities, iterations to meet schedule, budget, management style
  - Remember: Long after schedules are forgotten, usability (or lack of usability) remains

**Integrate with software engineering development process**

**Control mechanism for iteration**
- Evaluating against usability specifications
- Performing impact and cost/benefit analyses
- Deciding on changes to make to interaction design
- Deciding when to stop iterating
Meet the User Interface Development Team

- Roles on user interface development team
  
  *Note: Different roles, but not necessarily different people*
  - User interaction designer (or usability engineer, usability specialist, HF engineer)
  - Evaluator (or facilitator)
  - User and/or user representative
  - Application domain expert (also called subject matter expert)
  - Technical writer
  - Graphic designer
  - Software engineer and/or programmer
CS5714 Usability Engineering

Systems Analysis

Topics

- Ethnographic field studies (field visits, requirements gathering)
- Product concept statement
- Business process model
- Needs analysis
- User class definitions
- Task and work flow analysis
- Usability goals
- Constraints
Introduction to Systems Analysis

- You are here in the usability engineering life cycle

Requirements Gathering

- Note: From here on, “requirements” means interaction design requirements (but cannot separate entirely from system and functional requirements)
Ethnographic Field Visit

- In anthropology and sociology, ethnography is:
  - Participating, “overtly or covertly, in people’s daily lives for an extended period of time, watching what happens, listening to what is said, asking questions”
  
  [Hammersley & Atkinson 1983, as quoted by Shneiderman, p. 107]

Ethnographic Field Visit

- Ethnography for user interaction requirements gathering:
  - Cannot obtain needs information by just brainstorming in your own office
  - User interaction designers limit visit to days or even hours, but have to obtain needed data
  - Cannot substitute market research for ethnographic studies
Field Visits

- Example of using field visit knowledge in marketing: an ad by Lexar (make Compact Flash cards for digital cameras)

“By taking a closer look at how you actually work, we’ve created an entire line of products that actually work better.”

Ethnographic Requirements Gathering

- Process for UI ethnography includes:
  - Preparation for field study
    - Start with “brainstorming” of user task statements
    - Understand organization’s policies & culture
    - Check out their website
    - Know current system & history
    - Prepare script of initial questions for interview
    - Select appropriate users to observe and/or interview
    - Obtain permission to observe and/or interview
Ethnographic Requirements Gathering

- Perform field study
  - Establish rapport with managers and clients
  - Observe and/or interview users in workplace
  - Collect quantitative and qualitative data
  - Collect artifacts (e.g. paper forms) as available
  - Follow leads from visits, if any
  - Document and characterize user classes
  - Document user task & work flow analysis
- Keep focus of activities user centered!

- Explain to users why you are there
- Cannot just ask “What objects do you interact with?”
- Have to “tease out” needed information
- Be ready to modify, explore, branch out
Ethnographic Requirements Gathering

- Seems easy, but it’s not always
  - Hidden traps, surprises (e.g. what to wear, different perceptions of managers vs. users, different use of language/technical terms)

Equally important as data collected:
- Rapport/relationships with client, users established during process
- What if client is reluctant to give access to users?
  - Ask for a couple of hours
  - Establish necessity for usability
Ethnographic Requirements Gathering

- **Caution**: Difficult for users to tell developers what they want or need
  - Do not expect users to do design!
  - Important to observe users in their typical work environment

Developer: I try to tell users what they need, but they don’t want to listen to me

Ethnographic Requirements Gathering – In Sum

- The “User Interface Requirements Detective”
  - Goal of ethnography for UI designer is to discover, extract, and collect “clues” (information) needed to ensure usability of design
Introduction To Example System

- Calendar System
  - Simple automated version of a paper calendar
  - *Goal is to learn the development process, not to produce a marketable calendar product*
  - Working assumptions: some boundaries (e.g., hardware) set by management, customers, marketing, etc., there is a need for this product

Example: System Analysis

- Goal:
  - To make a fast tour through the process of determining basic user and system requirements

- Activities: A sampling of
  - Product concept statement
  - Needs analysis
  - Business process model
  - User class definition
  - Task analysis and work flow analysis
  - Usability goals
  - Constraints
Product Concept Statement

- Brief descriptive summary of product, typically 50-75 words
- Mission statement for a product, to help focus product development
- Writing a good product concept statement is not easy and is not done once, highly iterative

Product Concept Statement

- Answer the following questions:
  - What is the product name?
  - Who are the product users?
  - What will the product do?
  - What problem(s) will the product solve?
Product Concept Statement

- A possible product concept statement for Calendar System:
  - Our calendar will have automated support for scheduling appointments, to improve customer satisfaction.
  - Too vague

Product Concept Statement

- A better product concept statement (47 words):
  - The Calendar System will allow a broad variety of users to schedule and manage appointments. These users can range from professionals using the system to run an office to casual users keeping track of personal information. Automated support will reduce scheduling effort and increase awareness of appointments.
Product Concept Statement

- Example of being more specific
  - ‘Automated support will reduce scheduling effort and increase awareness of appointments.’
    - Reduce scheduling effort by supporting recurring appointments
    - Increase awareness by giving alarm (visual and/or audible)

Team Exercise – Product Concept Statement

- Overall goal: On-going exercise in developing the interaction design for a specific application: *A public ticket kiosk system for selling tickets to entertainment events*
- Exercise goal: Write a concise product concept statement for your ticket kiosk system
Team Exercise – Product Concept Statement

- Activities:
  - Assemble in project teams
  - Talk about your approach to the kiosk
  - Write a product concept statement
  - Iterate and polish it

- Deliverables:
  - Your “final” product concept statement, hand-written on plastic overhead

- Schedule: Due yesterday!

Example: Needs Analysis

- Goal of Calendar System: manage appointments

- Goal details
  - Appointment means information on:
    - Date
    - Time
    - Place
Example: Needs Analysis

- Goal details
  - *Manage* means
    - Add new appointment
    - Delete existing appointment
    - Modify existing appointment
  - Plus, need ability to view/display appointments

(Task=user, function=system)

- Derived from talking with client and users; *not just developers’ ideas*

Example: Needs Analysis

- After observing users, someone thinks of “alarm” idea (the needs don’t come all at once, up front)
  - Do we want to actively inform of appointments (maybe ask or observe users)
  - Decision: Yes, very useful; a way to beat paper
  - Iterate and revise needs

- New feature: Active reminder (alarm)
  - Increased functionality
  - More interaction to manage alarm
Business Process Modeling

- Understand application domain (subject matter), the way your client sees it
- Important for non-UI software, too
- Goal is to capture
  - What gets done to run business
  - Who does what and how it gets done (tasks)
  - How it relates to other things that get done
  - Flow of data and information
  - Work context for users

Business Process Modeling

- How to capture it
  - Look for both computer-supported and non-computer tasks
  - Gather and study work artifacts (e.g., paper work, tickets, slips)
  - Determine if there is a central database
  - Describe work flow, task flow, data & document flow
  - Flow charts are good (e.g., tasks are flow lines to/from people (users) and data objects
Team Exercise – Business Process Model

- **Goal:** a one-page diagram illustrating high-level business process (obviously an oversimplification) for your ticket kiosk operation

- **Activities:**
  - Sketch out a diagram showing business roles, information flow, information repositories, transactions, etc.
  - Label communication, flow lines

- **Deliverables:** one sheet plastic overhead

- **Schedule:** Now!
Two Critical Questions

- Who are our users?
  - User class definitions
- What tasks do they need to do?
  - User task analysis

User Class Definition

- Who are our users?
- “Know thy user”—and it is not you!
  - Important to have representative user(s) on development team and/or have access to representative user(s)
- User classes are about roles, not individuals
- Most of system analysis (e.g., task analysis, usability goals, usability specifications) and design is done for each user class
Some User Class Characteristics

- User knowledge of application/work domain
- User knowledge of computers
- Training and application-related experiences
- Frequency of use
- User goals
- Job- or task-related factors (e.g., job description, location, level of responsibility)
- Come up with your own, specific to situation

User Expertise Level

- Expertise levels don’t necessarily define user classes, but can occur within user classes
  - Novice or first-time user: may know application domain but not specifics of application
  - Intermittent user: uses several systems from time to time; knows application domain but not details of different applications
  - Experienced user: “power” user, probably uses application daily and knows both application and task domain very well

*These are not the specific user class types you should identify for your project!*
User Expertise Level

- Design may have to account for each of these expertise levels
- Remember: experienced users for some systems are novices for others

Example: User Class Definition

- What are characteristics of users of Calendar System?
  - General characteristics?
  - Domain skills?
  - Computer skills?
Example: User Class Definition

- What are *general characteristics* of users of Calendar System?

Example: User Class Definition

- What are *domain skills* of users of Calendar System?
Example: User Class Definition

- What are \textit{computer skills} of users of Calendar System?

Example: User Class Definition

- Conclusion
  - Keep it simple
  - Usability as important as functionality (or more)
  - Try to get functionality greater than paper calendar
  - Minimize typing
  - Users must learn quickly

- User class table
  - First, decide on most appropriate set of parameters for your domain and context
### Example: User Class Table

<table>
<thead>
<tr>
<th></th>
<th>User class A</th>
<th>User class B</th>
</tr>
</thead>
<tbody>
<tr>
<td>Computer knowledge</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Domain knowledge</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Complexity of domain content</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Example: User Class Table

<table>
<thead>
<tr>
<th></th>
<th>User class A</th>
<th>User class B</th>
</tr>
</thead>
<tbody>
<tr>
<td>Discretionary or captive</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Receptive or resistant</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Usage frequency, duty cycle</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Training</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Example: User Class Table

<table>
<thead>
<tr>
<th></th>
<th>User class A</th>
<th>User class B</th>
</tr>
</thead>
<tbody>
<tr>
<td>With whom do users interact outside system</td>
<td></td>
<td></td>
</tr>
<tr>
<td>What information is exchanged (outside system)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Work context</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Work Context

- Work context is the overall ambience and environment of user’s work
  - It’s about thought processes and mind set, policies, terminology
  - Example: steel mill floor is about noise, dust, hot temperatures, safety concerns, making iron and steel
  - Doctor’s office is very different culture
Task Analysis

- What tasks do our users need to do?
- What developers observe that users will do, not what developers think users will do
- Gives inventory of user needs
- Feeds scenarios in design
- Intertwines with design
- Probably the most overlooked and shortchanged, but critical, activity in the whole user interaction development process

Hierarchical Task Analysis

- Structured organization and relationships of tasks users perform with system
- Only what users can do, not must do
- Not timing, precedence, order of task performance, work flow, etc.
- Related work flow analysis includes timing, precedence, order of task performance, work flow, etc.
Hierarchical Task Analysis

- Hierarchical task decomposition
  - Task names: <action object>
    - Examples: add appointments, configure parameters
  - User-centered wording, not system centered
    - Example: view appointment, not display appointment

\[\begin{array}{c}
\text{A} \\
\downarrow \\
\text{B}
\end{array}\]

- Hierarchical relationships
  - Means A is a super-task of B, B is a sub-task of A
  - Meaning: Doing B is part of doing A (a "litmus" test for this characteristic)
  - If User is doing B, then also doing A
  - Example: Task A is filling out form; task B is filling out name field
Hierarchical Task Analysis

- Hierarchy does *not* show temporal sequencing
  - *Incorrect* attempt at hierarchical relationship:

```
Drive car

Start engine

Select gear

Press gas pedal
```

Example: Task Analysis

- What tasks will users perform with Calendar System?
  - For highest-level task, start with goal of system: *Manage appointments*
  - Initial list of major sub-tasks?
Example: Task Analysis

- Task analysis iterated
  - As thinking about viewing appointments, realize the need for different levels or scopes of view
    - For example, by month, by week, by day, by hour
    - Implication: add "control view" task to list

Example: Task Analysis

- An example of iteration
  - Also realize need to search appointment database to retrieve by content
    - Implication: Add to needs, tasks, functions, requirements
**Example: Task Analysis**

- Example of possible quasi-hierarchical user task structure for Calendar System
  - Structure diagram is accompanied by brief description of what each box means

**User Activity & Work Flow Analysis**

- Looking at individual tasks not enough; need to look at user activities across tasks
- For complex tasks involving lots of sub-tasks and sequences of inter-related steps
- For user planning, task structure, and organization
- Like scenarios, capture instances of possibilities ("Go paths", representative paths), not complete specification
Work Flow Analysis

- In style of operational-sequence diagrams (OSDs), use parallel “swim lanes” to show:
  - User actions
  - State changes
  - System responses

Work Flow Analysis

- Can include other kinds of information
  - Task/function allocation
  - User information requirements (knowledge needed to perform)
  - Task trigger (system state, condition leading to doing this task)
  - Task transition information (task completion conditions, trigger to next task)
Work Flow Analysis – Printing First Page of Outline in Word Document

<table>
<thead>
<tr>
<th>User action</th>
<th>System state change</th>
<th>Feedback</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;Initial state&gt;</td>
<td>Normal view mode</td>
<td>Normal document view on screen</td>
</tr>
<tr>
<td>View &gt; Outline OR Alt + V + O</td>
<td>Outline view mode</td>
<td>Outline view on screen</td>
</tr>
<tr>
<td>File &gt; Print OR Ctrl-P</td>
<td>Print setup mode</td>
<td>Print dialogue box on screen</td>
</tr>
<tr>
<td>Click ‘Pages’ radio button and type “1”; click Ok</td>
<td>Print setup mode</td>
<td>Print dialogue box</td>
</tr>
<tr>
<td>&lt;final state&gt;</td>
<td>Outline view mode</td>
<td>Outline view on screen</td>
</tr>
</tbody>
</table>

Work Flow Analysis

- Designing to support user work flow
  - Lots of pull-down menus, tool bars just sitting there (“It’s all there for the user”)
  - Better: design to connect task flow for the user
    - Extension of idea of task-thread continuity
    - Especially connections for ‘Go path’ or ‘Happy path’
    - Animated tutorials or Wizards can help
Usability Goals

- Usability evaluation driven by usability goals
- Usability goals driven by business objectives
- Determine usability goals in terms of
  - User classes
  - User task content, special tasks
  - Walk-up-and-use learnability
  - High performance for expert users
  - User errors
  - User satisfaction

Usability Goals

- High-level objectives in terms of usability and design of user interaction
  - Reflect real use of product in real world
  - Determine what is important to organization and to users
    - Example: Learnability for new users, power performance for experts, avoiding errors
  - Usability goals can be market driven
Usability Goals

- Example usability goals for Calendar System
  - Fast walk-up-and-use for simple tasks
  - High learnability for more advanced tasks
  - Low error rate for rescheduling appointments
  - Increased effectiveness of calendar by helping users avoid missed appointments

Constraints

- Cost and budget
- Schedule and development time
  - What restrictions do budget and schedule impose on product scope?
- Size and/or weight
  - Will product be on portable or mobile equipment?
- Integration with existing or other developing systems
- Security or privacy issues
Constraints

- Example constraints for Calendar System:
  - Product will be used in wide variety of environments, from factory floors to open offices to homes
  - Product will run on wide variety of platforms, but mostly PCs, laptops with no special devices
  - Budget is highly limited
  - Schedule is one semester!

Team Exercise – User Class Definition, Task Analysis, Usability Goals

- Application for team exercises: A public ticket kiosk system for selling tickets to entertainment and other events

- Goal: To produce simple user class definitions, task analysis, and usability goals for your ticket kiosk system
Team Exercise – User Class Definition, Task Analysis, Usability Goals

- Activities (on plastic transparencies):
  - Fill out user class table for two user classes
  - Develop a simple hierarchical task analysis
  - Write down a few usability goals for each user class

- Deliverables (on plastic transparencies):
  - User class table
  - Hierarchical task analysis diagram
  - Usability goals statement

- Schedule: Now
Topics

- Usage scenarios
- Conceptual design
- Detailed screen design
- Participatory design
- Interaction flow maps
- Custom style guides
- Team exercise on user interaction design
Introduction to Design

- You are here in the usability engineering life cycle

User Interaction Design

- Approaching user interaction design
  - Do NOT start with screen designs and widgets
  - Do start with user tasks and usage scenarios
  - Get help from users but designers must do the design
Usage Scenarios

- Similar to the software engineering idea of use cases
- Scenarios: Stories about people and their work activities
- Work-oriented: Focus on needs, goals, and concerns of users
- Scenarios make **use** the object of design

Usage Scenarios Are Designs

- Scenarios are envisioned design solutions!
  - Design representation tied to situations of use
- Scenarios evoke thought and discussion about design
Usage Scenarios

● Scenarios reveal requirements
  – Scenarios facilitate agreement on requirements
  – Show how tasks will be carried out and how system will provide functionality to enable tasks

● Where do they come from?
  – Brainstorming
  – Ethnographic field studies
  – Participatory design
  – Reuse of similar designs

Usage Scenarios

● How do scenarios compare to design specifications?
  – Design situations are fluid; written specifications are rigid
  – Customers, users would rather talk about scenarios than formal specifications
  – Scenario-based design is bottom-up (the way people usually think)
    ● Needs to be mixed with top-down structuring of task analysis
Usage Scenarios

- How do scenarios compare to task analysis?
  - Same general goal
  - Complementary in almost every way

<table>
<thead>
<tr>
<th>Task Analysis</th>
<th>Scenarios</th>
</tr>
</thead>
<tbody>
<tr>
<td>General</td>
<td>Specific</td>
</tr>
<tr>
<td>Formal</td>
<td>Informal</td>
</tr>
<tr>
<td>Abstract</td>
<td>Concrete</td>
</tr>
<tr>
<td>Completeness is goal</td>
<td>Deliberately incomplete</td>
</tr>
<tr>
<td>Addresses design structure</td>
<td>Addresses design details</td>
</tr>
</tbody>
</table>

Usage Scenarios

- Scenarios should capture:
  - User tasks and task threads
    - Common, representative
    - Mission critical
    - Error and recovery situations
  - User roles
  - Work flow
  - User planning, thoughts, and reactions to system
  - Environmental and work context (e.g., phone ringing)
Usage Scenarios

- Scenarios go hand-in-hand with screen designs
- But scenarios don’t usually include detailed screen designs, widgets, interaction details
- Don’t design *just* for scenario, but design must *cover* scenario

Usage Scenarios

- How many scenarios should we expect?
  - Number of scenarios can be large, if product is large, complex
  - Need different set of scenarios for each user class
  - May need tools to manage and maintain large set of scenarios and correlate with screen designs
Scenario Creation – Example

- Example of usage scenario for Calendar System:
  Sue, a patient with an existing appointment with Dr. Kevorkian for next Tuesday, calls secretary at the physician’s office. Sue is unable to keep her appointment, and needs to reschedule it. The secretary must locate the current appointment, find an open time slot that also is a time the patient is available, and re-enter patient information into the new time slot. While the secretary is doing this, another phone line is ringing and another patient is standing at the desk waiting to schedule a follow-up appointment with Dr. Kevorkian.

Scenario Creation – Example

- Scenario is just one instance of task thread
  - Temporal order of locating current appointment and finding open time slot could be reversed
  - Don’t let specificity of scenario force rigid design
- Iteratively refine scenarios and screen designs together via design walk-throughs
Scenario Iteration

- Sometimes it takes a large number of iterations to work out a consistent look and feel and to organize the functionality
- Next: as exercise, identify user roles, actions, information objects, object attributes, tasks, work context
  - The idea: extract these to drive design

Marking Scenario Components – Example

- Example of usage scenario for Calendar System:
  Sue, a patient with an existing appointment with Dr. Kevorkian for next Tuesday, calls secretary at the physician’s office. Sue is unable to keep her appointment, and needs to reschedule it. The secretary must locate the current appointment, find an open time slot that also is a time the patient is available, and re-enter patient information into the new time slot. While the secretary is doing this, another phone line is ringing and another patient is standing at the desk waiting to schedule a follow-up appointment with Dr. Kevorkian.
Conceptual Design

- Conceptual design
  - At last: transition from information gathering to design
  - Early sketch of how it all works
  - Where screen designs emerge

- Develop conceptual model/metaphor
  - A chance to be creative with a “theme” for your interaction design
  - Metaphor is analogy with something existing in real world with similarities that can be leveraged for learning new system (e.g., desktop)

Conceptual Design

- Develop some design details
  - Include navigation, screen layout, visual design

- Use early conceptual design to encourage critical comments and iteration
  - Start evaluation this early with team, client, and key users
Example (Together): Conceptual Design

- Goals:
  - To create together a simple conceptual design for Calendar System, from a scenario
  - To develop conceptual screen designs and move toward detailed screen designs
  - To perform an early cognitive-based evaluation of conceptual designs

- Assumptions:
  - Generic desktop platform (not specific to Windows, Mac, etc.)

Example: Conceptual Design

- Activities:
  - Tease out as much conceptual design information as possible, extrapolating from scenario
    - Identify information objects: Appointments
    - Identify information object properties: Date, time, description, length (?), alarm or not
    - Identify information object relationships: Only one object so far
Example: Conceptual Design

- Decide how information objects are represented conceptually in user interaction design
  - By month, week, day, hour, time slots
    - Time slots can be empty or contain appointment
    - Implication: these are all “container objects”
  - Decide how users get at information objects, i.e., access methods
    - Example: Accessing an existing appointment
      - By viewing, possibly preceded by search or navigation through views
      - Got “search” from needs analysis rather than from scenario

Example: Conceptual Design

- Decide how users invoke and carry out operations on information objects
  - Menu? Pull-down?
  - Small, fixed number of commands
  - Implication for interaction style: Buttons or icons?
Example: Conceptual Design

- Conceptual design might lead to something like
- Cognitive/human factors analysis
  - Design does not closely match user’s concept of a calendar
  - Paper calendar not necessarily the criterion; be creative
- Already a good time to involve users in evaluation

Example: Conceptual Design

- Can do better with direct manipulation
  - Eliminate explicit view control by selecting view object
  - Add and modify by typing (editing) directly on text of appointment
Example: Iteration Toward Detailed Screen Design

- Conceptual design revisited
- Access appointment objects by
  - Selection and navigation on desk top
  - Search on content
- Decisions about container objects
- Show several months overlapped; current on top
- In month view select week or day to view
- Try to show as much appointment information as possible at each level (page preview idea)

Example: Iterated Screen Design

<table>
<thead>
<tr>
<th>May 1999</th>
<th>Search</th>
<th>Help</th>
</tr>
</thead>
<tbody>
<tr>
<td>Past Month</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sun</td>
<td>Mon</td>
<td>Tue</td>
</tr>
<tr>
<td>25</td>
<td>26</td>
<td>27</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Future Month</th>
<th>Delete Appointment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Search</td>
<td>Help</td>
</tr>
<tr>
<td>Past Month</td>
<td></td>
</tr>
<tr>
<td>Sun</td>
<td>Mon</td>
</tr>
<tr>
<td>25</td>
<td>26</td>
</tr>
<tr>
<td>SATURDAY MARCH 17, 1999</td>
<td>8:00</td>
</tr>
</tbody>
</table>
Design

State Diagram

Example: Iterated Screen Design

- Day view

No close box; keep on desk top

Click here to edit, type

Scroll up or down to midnight

State Diagram

Example: Iterated Screen Design

- Day view

No close box; keep on desk top

Click here to edit, type

Scroll up or down to midnight
Example: Iterated Screen Design

- Search dialogue box

![Search dialogue box diagram]

Participatory Design

- Active involvement of all roles, especially users, during early design
- People around table with Post-Its™, pencils, paper, plastic
- Democratic – everyone has same say about things
Participatory Design

- Pros: More accurate information about tasks, work context; lets users influence design decisions; builds rapport/buy-in with users
- Cons: Costly; may create antagonism; users not trained designers
- Nevertheless, experiences are usually positive, helped by:
  - Good, experienced leadership
  - Careful selection of users

PICTIVE (Plastic Interface for Collaborative Technology Initiatives through Video Exploration)

- Users actually sketch UI design
- Video recording of scenario walk-through shown to other users, designers, managers
Participatory Design

- What can we get from talking to users?
  - Some of what users need in tasks
  - Usually very incomplete, unintegrated
  - It's still just an input; you are the designer
  - Need to extend to design that works for all users over all tasks
  - You are asking users what they do, not what they think is best for them

Interaction Flow Map

- Similar to wire frames
- Road map of screen transitions
- Thumb-nails of screens connected by user action arrows for a task thread
- Graphical representation of “where you go” when user clicks on buttons, icons, menu choices
- Help designers visualize task flow, organizational structure
- Help illustrate design walk-throughs
Interaction Flow Map

- Interaction flow map is functionally oriented
- Addresses:
  - Where you go if you click here
  - Rather than how you do this task
Custom Style Guide – An Integral Part of Design

- Every project needs one!
- Documented internally within an organization
- Style guide documents visual and other general design decisions that apply in multiple places
- Develop iteratively in parallel with product and maintain throughout usability engineering process

Custom Style Guide

- Must be accepted by team members, not just author(s)
- Very specifically worded
- Describes specific interaction styles, layouts, formats, wordings, button labels, etc.
- Include sample screen sketches; make it visual
- Include templates, style sheets
Custom Style Guide

- Contains details about:
  - Fonts and text usage
  - Color usage, background graphics, other common design elements
  - Icon usage, position, design
  - Widget usage, position, design – dialogue boxes, menus, message windows, toolbars, etc.
  - Formats (e.g., for dates)
  - Consistent use of defaults

Custom Style Guide

- Major advantages of style guides
  - **Consistency** throughout product, across product lines
  - **Re-use** of design decisions

- Major misconception about style guides
  - "We followed the style guide; so we don’t need usability evaluation now.”
Team Exercise – Scenarios

- **Goal:** To develop some usage scenarios and initial screen designs for your ticket kiosk system, for “customer” user class only
- **Before we begin,** move together by team
- **Assumptions:**
  - Specialized remote workstation
    - Generic interaction style
    - Hardened against vandalism, etc.
  - Don't assume much computer or browser knowledge for typical user

Team Exercise – Scenarios

- **Activities – Scenarios:**
  - Write one or two usage scenarios for your kiosk
    - Select one or two good representative tasks for the customer user class
    - Make up the design on the fly
    - Do this quickly; you can clean them up as you go
    - Get detailed and refer to user roles, tasks, actions, and objects
Team Exercise – Screen Designs

- Activities – Screen designs:
  - Begin your screen designs with start-up screen
    - Show broad functionality as user tasks
    - Draw pictures of screens, including menus, buttons, nav bars, icons
    - Add text labels to explain things as appropriate

Team Exercise – Screen Designs

- Activities – Main task thread(s):
  - Design and sketch one or two main task thread over a few more pages/screens
  - Go for some depth
  - Don’t go into much breadth for the whole system yet
Team Exercise – Screen Designs

- Try to capture deep design issues, such as:
  - Information objects, their properties, and relationships among them
  - How objects will be viewed conceptually (not necessarily details of appearance) in interaction design
  - How user will access those objects
  - Operations to be performed on the objects as a result of user tasks
  - How users will invoke and carry out those operations, including navigation

Team Exercise – Screen Designs

- Cautions/Approach
  - Don’t get too involved in human factors issues yet (e.g., icon appearance or menu placement)
    - Control time spent arguing; learn the process!
  - Keep it simple
    - Tons of functionality doesn’t necessarily help learn the process
    - Cut corners (e.g., for Calendar System: number of days in a month, what day each month starts on)
Team Exercise – Screen Designs

- Cautions/Approach
  - Base your screen designs on task analysis we already did
  - Make it different from applications of the same type that you already know

Team Exercise – Scenarios and Screen Designs

- Hint:
  - If team members have different ideas for a feature, consider offering both via "preferences"

- Deliverables:
  - One or two usage scenarios on plastic, to show class
  - A few representative screen designs, sketched on paper, including start-up screen or home page

- Schedule: Due by end of class or bring to next class
CS5714 Usability Engineering

Usability Specifications

Topics

- What are usability specifications?
- Usability specification tables and attributes
- Benchmark task descriptions
- User errors
- Usability specifications and managing the usability engineering process
- Team exercises on usability specifications
Usability Specifications

- You are here in the usability engineering life cycle

Usability specifications – quantitative usability goals against which user interaction design is measured

- Target levels for usability attributes
  - Operationally defined metric for a usable interaction design
  - Establish as early in process as feasible
Usability Specification Data

- Usability specifications based on
  - Objective data: observable user performance
  - Subjective data: user opinion and satisfaction
- Objective and subjective usability specifications can both be quantitative

Usability Specifications

- Usability specifications driven by usability goals
  - Connect usability specifications to usability goals
    - Example: For early usability goal of ‘walk-up-and-use’ for Calendar System, base a usability specification on initial task performance time
  - Quantify usability goals
    - Example:
      - Reduce amount of time for novice user to perform task X in Version 2.0
      - Currently 35 seconds to perform task X the first time; reduce to 25 seconds
Each usability attribute addresses a user class and a usability goal.

<table>
<thead>
<tr>
<th>User class</th>
<th>Usability goal</th>
<th>Usability attribute</th>
<th>Measuring instrument</th>
<th>Value to be measured</th>
<th>Baseline level</th>
<th>Target level</th>
</tr>
</thead>
</table>

Usability goal for casual user
- High *walk-up-and-use* performance without training or manuals.

<table>
<thead>
<tr>
<th>User class</th>
<th>Usability goal</th>
<th>Usability attribute</th>
<th>Measuring instrument</th>
<th>Value to be measured</th>
<th>Baseline level</th>
<th>Target level</th>
</tr>
</thead>
<tbody>
<tr>
<td>Casual user, for personal use</td>
<td>Walk-up-and-use for new user</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Usability Specification Attributes

- **Usability attribute** – what general usability characteristic is to be measured
- Some quantitative usability attributes
  - **Objective**
    - Initial performance (on benchmark tasks)
    - Longitudinal (experienced, steady state) performance
    - Learnability
    - Retainability
  - **Subjective**
    - Initial impression (questionnaire score)
    - Longitudinal satisfaction

### Usability Specification Table

- **Usability attribute** for Calendar System
  - We need to look at initial performance in this usability specification

<table>
<thead>
<tr>
<th>User class</th>
<th>Usability goal</th>
<th>Usability attribute</th>
<th>Measuring instrument</th>
<th>Value to be measured</th>
<th>Baseline level</th>
<th>Target level</th>
</tr>
</thead>
<tbody>
<tr>
<td>Casual user, for personal use</td>
<td>Walk-up-and-use for new user</td>
<td>Initial user performance</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Usability Specification Table

- **Measuring instrument** – vehicle by which values are measured for usability attribute

<table>
<thead>
<tr>
<th>User class</th>
<th>Usability goal</th>
<th>Usability attribute</th>
<th>Measuring instrument</th>
<th>Value to be measured</th>
<th>Baseline level</th>
<th>Target level</th>
</tr>
</thead>
<tbody>
<tr>
<td>Casual user, for personal use</td>
<td>Walk-up-and-use for new user</td>
<td>Initial user</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>performance</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Measuring instrument** is how data are generated

- **Benchmark task** generates objective data (e.g., timing with stopwatch)
- **Questionnaire** generates subjective preference data (e.g., with ratings scale)
Benchmark Tasks

- What tasks should be included?
  - Representative, frequently performed tasks
  - Common tasks: 20% that account for 80% of usage
  - Business- or mission-critical tasks: Not frequent, but if you get it wrong, heads can roll
- Clear, precise, repeatable instructions
- **Important:** What task to do, *not how* to do it

Benchmark Task Descriptions

- Example: Schedule a meeting with Dr. Ehrich for four weeks from today at 10 am in 133 McBryde, about the HCI research project
- Adapt scenarios already developed for design
  - Clearly important tasks to evaluate
  - Remove information about how to do it
Benchmark Task Descriptions

- Have clear start and end points for timing (for your timing, not to tell participant)
  - Not good end points (you can’t be sure of what users knows or sees):
    - When users know something they were seeking
    - When users see something they were seeking
    - Example (not to do): View next week’s appointments
  - Better: End with user action confirming end of task
    - Example (to do): Click a Print button to “print out” next week’s appointments

Benchmark Task Descriptions

- Start with fairly simple tasks, then progressively increase difficulty
  - Add an appointment, then add appointment 60 days from now, then move appointment from one month to other, add recurring appointments
- Avoid large amounts of typing (unless typing skill is being evaluated)
- Tasks should include navigation
  - Not: look at today’s appointments
Benchmark Task Descriptions

- Tasks wording should be unambiguous
  - Unless you want to include ambiguity in part of task context
  - Why is this ambiguous? “Schedule a meeting with Mr. Jones for one month from today, at 8 AM.”

- **Important**: Don’t use words in benchmark tasks that appear specifically in interaction design
  - Not: “Find first appointment…” when there is a button labeled “Find”
  - Instead: Use “look for” or “locate”

Benchmark Task Descriptions

- Use work context wording, not system-oriented wording
  - “Find information about xyz” is better than “Submit query about xyz”

- To evaluate error recovery, benchmark task can begin in error state

- Consider tasks to evaluate performance in ‘degraded modes’ (partial equipment failure)
Benchmark Task Descriptions

- Put each benchmark task on a separate sheet of paper
- What is typical number of benchmark tasks? Enough for reasonable, representative coverage
- Example for Calendar System
  - BT1: Add an appointment with Dr. Kevorkian for 4 weeks from today at 9 AM concerning your flu shot (yeah, right)

Benchmark Task Descriptions

- Write a ‘task script’ for each benchmark task
  - Not to give to users, but to help evaluator during evaluation
  - Describe a representative or typical way to do task, so evaluator knows at least one way during evaluation
  - May indicate specific design issues for evaluator to address or watch for during evaluation
### Usability Specification Table

- **Measuring instrument** for Calendar System
  - Benchmark task to add an appointment

<table>
<thead>
<tr>
<th>User class</th>
<th>Usability goal</th>
<th>Usability attribute</th>
<th>Measuring instrument</th>
<th>Value to be measured</th>
<th>Baseline level</th>
<th>Target level</th>
</tr>
</thead>
<tbody>
<tr>
<td>Casual user, for personal use</td>
<td>Walk-up-and-use for new user</td>
<td>Initial user performance</td>
<td>BT1: Add apt</td>
<td>BT1: Add apt</td>
<td>Baseline level</td>
<td>Target level</td>
</tr>
</tbody>
</table>
Examples of *value to be measured*:
- Time to complete task
- Number of errors
- Frequency of help and documentation use
- Time spent in errors and recovery
- Number of repetitions of failed commands
- Number of times user expresses frustration or satisfaction
- Number of commands, mouse-clicks, or other user actions to perform task(s)

<table>
<thead>
<tr>
<th>User class</th>
<th>Usability goal</th>
<th>Usability attribute</th>
<th>Measuring instrument</th>
<th>Value to be measured</th>
<th>Baseline level</th>
<th>Target level</th>
</tr>
</thead>
<tbody>
<tr>
<td>Casual user, for personal use</td>
<td>Walk-up-and-use for new user</td>
<td>Initial user performance</td>
<td>BT1: Add appt</td>
<td>Average time on task</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

*Value to be measured for Calendar System*
**Usability Specification Table**

*Baseline level* – starting point to determine target level

<table>
<thead>
<tr>
<th>User class</th>
<th>Usability goal</th>
<th>Usability attribute</th>
<th>Measuring instrument</th>
<th>Value to be measured</th>
<th>Baseline level</th>
<th>Target level</th>
</tr>
</thead>
<tbody>
<tr>
<td>Casual user, for personal use</td>
<td>Walk-up-and-use for new user</td>
<td>Initial user performance</td>
<td>BT1: Add appt</td>
<td>Average time on task</td>
<td>No time for reqts</td>
<td>No time for reqts</td>
</tr>
</tbody>
</table>

**Baseline level gives**

- Level of performance for current version of system for measuring instrument (when available)
- Basis to help set target level, from:
  - Automated system (existing or prior version)
  - Competitor system
  - Developer performance (for expert, longitudinal use)
  - Trying out some users on early prototype
### Usability Specification Table

#### Baseline level for Calendar System

<table>
<thead>
<tr>
<th>User class</th>
<th>Usability goal</th>
<th>Usability attribute</th>
<th>Measuring instrument</th>
<th>Value to be measured</th>
<th>Baseline level</th>
<th>Target level</th>
</tr>
</thead>
<tbody>
<tr>
<td>Casual user, for personal use</td>
<td>Walk-up-and-use for new user</td>
<td>Initial user performance</td>
<td>BT1: Add appt</td>
<td>Average time on task</td>
<td>20 secs. (competitor system)</td>
<td></td>
</tr>
</tbody>
</table>

### Usability Specification Table

#### Target level – value indicating unquestioned usability success for present version

<table>
<thead>
<tr>
<th>User class</th>
<th>Usability goal</th>
<th>Usability attribute</th>
<th>Measuring instrument</th>
<th>Value to be measured</th>
<th>Baseline level</th>
<th>Target level</th>
</tr>
</thead>
<tbody>
<tr>
<td>Casual user, for personal use</td>
<td>Walk-up-and-use for new user</td>
<td>Initial user performance</td>
<td>BT1: Add appt</td>
<td>Average time on task</td>
<td>20 secs. (competitor system)</td>
<td></td>
</tr>
</tbody>
</table>
**Usability Specification Table**

- *Target level* gives minimum acceptable level of user performance
- Determining target level values
  - Usually an improvement over baseline level
  - Sometimes comes from requirements specifications
  - Value of “0” not realistic for error count (consider fractional value; it’s an average)

**Usability Specification Table**

<table>
<thead>
<tr>
<th>User class</th>
<th>Usability goal</th>
<th>Usability attribute</th>
<th>Measuring instrument</th>
<th>Value to be measured</th>
<th>Baseline level</th>
<th>Target level</th>
</tr>
</thead>
<tbody>
<tr>
<td>Casual user, for personal use</td>
<td>Walk-up-and-use for new user</td>
<td>Initial user performance</td>
<td>BT1: Add appt</td>
<td>Average time on task</td>
<td>20 secs. (competitor system)</td>
<td>15 secs.</td>
</tr>
</tbody>
</table>
## Usability Specification Table

- More example usability specifications

<table>
<thead>
<tr>
<th>User class</th>
<th>Usability goal</th>
<th>Usability attribute</th>
<th>Measuring instrument</th>
<th>Value to be measured</th>
<th>Baseline level</th>
<th>Target level</th>
</tr>
</thead>
<tbody>
<tr>
<td>Casual user, for personal use</td>
<td>Walk-up-and-use for new user</td>
<td>Initial user performance</td>
<td>BT1: Add appt</td>
<td>Average time on task</td>
<td>20 secs. (competitor system)</td>
<td>15 secs.</td>
</tr>
<tr>
<td>Casual user, for personal use</td>
<td>Customer satisfaction</td>
<td>Initial user satisfaction</td>
<td>Qns 1, 2, 7 in questionnaire</td>
<td>Average score over qns (out of 10)</td>
<td>7</td>
<td>8.5</td>
</tr>
<tr>
<td>Professional user for work</td>
<td>Accuracy</td>
<td>Long term use</td>
<td>BT1: Add appt</td>
<td>Average number of errors</td>
<td>2</td>
<td>0.5</td>
</tr>
</tbody>
</table>

## User Errors

- What constitutes a user error?
  - Deviation from any correct path to accomplish task (except, for example, going to Help)
  - Criterion: **Only situations that imply problem in design**
  - Usually not typing errors
  - If user knows right button to click but clicks wrong one?
    - Could be "oops" error (slip), not counted as error or design problem (but watch for it again, to see if design caused it)
    - If can identify design issue that affected user, count as error, because need to fix design (e.g., buttons too close together)
User Errors in Usability Specifications

- Examples of errors
  - Selecting wrong menu, button, icon, etc. when user thought it was the right one
    - Example: Working on wrong month of calendar because they couldn't readily see month's name
  - Double clicking when a single click is needed, and vice versa
  - Using wrong hot key

Creating Usability Specifications

- Design evaluation for ecological validity
  - Usability lab can be "sterile work environment"
  - How can setting be more realistic?
  - What are constraints in user or work context?
Creating Usability Specifications

- Design benchmark tasks for ecological validity
  - Does task involve more than one person or role?
  - Does task require telephone or other physical props?
  - Does task involve background noise?
  - Does task involve interference or interruption?
  - Does user get multiple audio feeds through headsets?

Creating Usability Specifications

- Usability specifications must take into account potential trade-offs among user groups
  - Example: Trade-offs between learnability for new users and getting in the way of power performance by experienced users

- **Important**: All project members should agree on usability specification attributes and values
Usability Specifications – Connecting Back to Usability Engineering Process

- Usability specifications help manage the usability engineering process
- Management control of usability engineering life cycle
  - Quantifiable end to iterative process
  - Accountability
  - Indication that development process is converging toward successful design
  - Stop iterating when target level usability specifications are met

Usability Specifications

- Don’t expect to meet all usability target levels on first iteration
  - If usability target levels are met on first iteration, they may have been too lenient
  - DO NOT design usability specifications with the goal of meeting them with your initial design!
  - Goal is to uncover usability problems
Usability Specifications

- Bottom line: This is not an exact science
- **Good engineering judgment is important**
  - For setting levels (especially “target” level)
  - For knowing if specifications are “reasonable”
- You get better at it with experience

Usability Specifications

- Introducing Envision
  - Video-clips to be used as examples of process activities
  - Envision: A digital library of computer science literature
  - Search results are presented in graphical scatterplot

*Video-clip:* Envision prototype

*Video-clip:* Setting usability specifications
Team Exercise – Usability Specifications

● Goal:
  - To gain experience in writing precise benchmark tasks and measurable usability specifications

● Activities:
  - Write, each on a separate sheet of paper (not plastic), two benchmark tasks for kiosk “customer” user class only
    ● Do NOT make the tasks too easy
    ● Make tasks increasingly complex
    ● Include some navigation
    ● Create tasks that you can “implement” in your next team exercise, to build a rapid prototype

Team Exercise – Usability Specifications

● Activities:
  - Write (on plastic transparency table we will give you) three usability specifications: two based on objective measures, one based on subjective measure
  - Cover two usability goals for “customer” user class
Team Exercise – Usability Specifications

- Specifications with objective measures should be evaluable, via each benchmark task, in a later class exercise, on formative evaluation
- Specification with subjective measure should be based on questionnaire supplied; select 3 or 4 items
- Include questionnaire question numbers in subjective specification

Team Exercise – Usability Specifications

- Cautions and hints:
  - Don’t spend any time on design in this exercise; there will be time for detailed design in the next exercise
  - Don’t plan to give users any training
Team Exercise – Usability Specifications

- Deliverables:
  - Two customer benchmark tasks, each on a separate sheet of paper
  - Three usability specifications, in table on plastic transparency
- Schedule: Complete in about 30-40 minutes max.
- Read your benchmark tasks to class for critique and discussion
Topics

- Relation to usability engineering life cycle and iterative refinement
- Advantages and dangers of prototyping
- Low-fidelity paper prototypes
- Prototype evolution
- What to put in prototype
- Team exercise on rapid prototyping
Introduction to Rapid Prototyping

- You are here in the usability engineering life cycle

Rapid Prototyping

- Usability engineering life cycle is evaluation-centered
- Dilemma: Can’t evaluate an interface until it is built, but after building, changes are difficult
- Solution: Rapid prototyping gives you something to evaluate before you have to commit resources to build the real thing
Rapid Prototyping

- Main technique supporting iterative evaluation and refinement
- Don’t wait until first release or field test of product; use a prototype

Advantages of Rapid Prototyping

- Concrete baseline for communication between users and developers
  - Prototype is conversational “prop” to support communication of concepts not easily conveyed verbally (R. Bellamy, Apple Corp.)
- Allows users to “take it for a spin”
- Gives project visibility and buy-in
- Encourages early user participation and involvement
- Allows early observation of user performance
Advantages of Rapid Prototyping

- Low-fidelity prototype is obviously not finished, so users have impression it is easy to change
- Allows immediate observation of consequences of design decisions
- Can help sell management an idea for new product
- Can help effect a paradigm shift from existing system to new system

Example of rapid prototyping advantages (from real world)
- Database program had novice & advanced users
- For novice interface, spent huge effort and large percentage of system code to build in lots of hand-holding for making queries
- When released, found that most users moved rapidly from novice to expert, typing in own SQL
- If could have seen this in advance, could have saved resources and lightened up the application
Dangers of Rapid Prototyping

- Needs cooperation of management, developers, and users
- Management may view as wasteful
- Programmers may lose discipline
- Managers and/or customers and/or marketing may view prototype as final product

Dangers of Rapid Prototyping

- Prototype can be overworked if goal of prototyping is forgotten
- Prototyping tool may influence design
- Possibility of over-promising with prototype
Low-Fidelity Prototyping

- Low-fidelity paper prototyping is bona fide technique on its own
  - Not just a low-tech substitute for computer-based prototype
  - Not just something to do if good software tools are not available
  - Major corporations with extensive resources use paper prototypes routinely for early interaction development (especially design and evaluation)

Low-Fidelity Prototyping

- Paper prototypes can evolve very quickly
- Computer-based prototype can distract from usability focus early on
- People do take paper prototypes seriously
- Low-fidelity prototypes reveal many usability problems, and these are generally the more severe problems

Sample lo-fi prototype, from M. Rettig, CACM, April 1994
Why Not Just Program a Low-Fidelity Prototype?

- Need team member who knows VisualBasic (or Java or whatever tool) and is available to code
- Paper has much broader visual bandwidth; often need multiple screens visible at once
- Major reorganization of sequencing is faster and easier by shuffling paper on table
- Often useful to write on pages during evaluation
  - Natural, fast, and less intrusive (than reprogramming) for editing and modifying design

High- & Low-Fidelity Prototyping

- User interaction design has two parts
  - Look & feel: objects
  - Sequencing: behavior, including changes to object behavior

<table>
<thead>
<tr>
<th>Type of prototype</th>
<th>&quot;Strength&quot;</th>
<th>When in life cycle to apply &quot;strength&quot;</th>
<th>Cost to fix look &amp; feel</th>
<th>Cost to fix sequencing</th>
</tr>
</thead>
<tbody>
<tr>
<td>Paper (lo-fi)</td>
<td>Flexibility; easy to change sequencing, overall behavior</td>
<td>Early</td>
<td>Almost none</td>
<td>Low</td>
</tr>
<tr>
<td>Computer (hi-fi)</td>
<td>Fidelity of look &amp; feel</td>
<td>Later</td>
<td>Low</td>
<td>High</td>
</tr>
</tbody>
</table>
Prototype Evolution Within A Project

- Low-fidelity prototype – Hand-made paper prototype, used to evaluate conceptual model and early screen design ideas
- Mid-fidelity prototype – Computer-printed paper prototype or on-line (VB-like) mockup
- High-fidelity prototype – Computer-based with some working functionality (e.g., database functions)

What To Put In A Prototype

- Purpose of early prototypes – To evaluate usability of overall interaction design concept
  - Low-fidelity
    - Start with several representative sample screens
    - Mock-up a few representative tasks
    - Follow a few representative task threads
  - Learn a great deal from incomplete design and from a single brand new user
Fleshing Out A Prototype

- Purpose of later prototypes – To evaluate usability of more detailed designs
  - High-fidelity
    - More specific, refined screens
    - More complete task threads

Effort To Put Into A Prototype

- Invest *just enough* effort in a rapid prototype
  - To achieve formative evaluation goal, but no more
- What to retain from final prototype for real system
  - Keep details (e.g., code when possible) of user interaction design objects, and look and feel
  - Discard any functional or other code holding prototype together
    - E.g., sequencing code; never meant to be product code

*Example:* Lo-fi prototype
Team Exercise – Rapid Prototyping

Almost all you ever wanted to know about rapid prototyping you learned in Kindergarten!

- **Goals:**
  - To obtain experience with rapid construction of a low-fidelity prototype for early stages of user interaction design
  - To have something to generate lots of critical incidents in the evaluation exercise

- **Activities:**
  - Draw screens in more detail and make your prototype manually “executable”
  - Start with an interaction flow map, if helpful

Team Exercise – Rapid Prototyping

- **Activities:**
  - Start with simplest possible background for each screen in pencil or pen on full size paper, as base for all moving parts
  - Include only parts that never change (e.g., for Calendar System: monthly “grid”, no month name)
Team Exercise – Rapid Prototyping

- **Activities:**
  - Draw everything else (e.g., interaction objects) in pencil or pen on **smaller pieces of paper**, cut them out, and tape (in aligned position relative to other objects) onto separate blank plastic sheets.
  - **Do not write or draw on plastic**, except for transparent objects such as highlights.

- During “execution” most dynamics will be created by adding and removing various paper objects on plastic sheets to/from the easel.
  - Prototype will be “executed” on the easel, usually taped to tabletop for stability.
  - Use “easel” to register each sheet of plastic with other sheets.
Team Exercise – Rapid Prototyping

- **Activities:**
  - Prototype at least all benchmark tasks from your usability specifications, since this prototype will be used in the formative evaluation exercise
    - If users can do all benchmark tasks without problems, tasks and/or prototype are too simple
    - Include lots of buttons, etc. not needed for your benchmark tasks, even if they don’t do anything (so participants see more than just ‘happy path’)
  - Remember: **You’re learning the process**, not creating an initially perfect kiosk!

Team Exercise – Rapid Prototyping

- **IMPORTANT:** Get **everyone** on your team involved in drawing, cutting, taping, etc., not just one or two people
  - You’ll be done much faster if everyone pitches in. However, this is not art class, so do not worry too much about straight lines, exact details, etc.
Team Exercise – Rapid Prototyping

- Don’t draw *anything* twice; make it modular to reuse
  - The less you put on each layer, the more modular
  - Build up interface design in layers
  - We’ll be suspicious of a lot of writing/drawing on one piece of paper
- Whatever changes when user gives input should go on separate paper-on-plastic sheet
- If user will type in values (e.g., credit card number), use clear sheet on top and marking pen

Team Exercise – Rapid Prototyping

- Make a highlight for major selectable objects
  - Use plastic square or rectangle with handle; color with marking pen
- Make a “This feature not yet implemented” message
- Fasten some objects (e.g., pull-down lists) to top or side of easel with tape hinges, so they can “flap down” to overlay the screen
- Use any creative techniques to demonstrate motion, dynamics, feedback
  - E.g., scrolling can be done with paper through slits cut in larger paper (all taped to plastic sheet)
Team Exercise – Rapid Prototyping

- Final activity:
  - Pilot testing: Be sure your prototype will support all your benchmark tasks by having one member of your team “execute” the prototype while another member plays “user” and tries out all benchmark tasks
  - When you think your prototype is ready, call us and we will help you pilot test

Team Exercise – Rapid Prototyping

- Deliverables:
  - An “executable” version of your prototype, constructed of paper taped in registration to plastic sheets
  - Pilot test complete

- Schedule: Complete by end of class (today!)
Formative Evaluation of User Interaction: Introduction

Topics

- What it is: Formative vs. summative evaluation
- Types of formative usability evaluation
- Steps in formative usability evaluation
  - Last piece of the puzzle
You are here in the usability engineering life cycle

- Systems analysis
- Interaction design requirements, tasks, user classes
- Formative user-based evaluation
- Prototypes
- Usability specifications
- Rapid prototyping
- Usability specs
- Scenarios, screens

Users will evaluate the user interaction design sooner or later...

Let’s have them do it sooner, so we can make changes to improve usability!
Formative vs. Summative Evaluation

- In simplest terms:
  - **Summative** evaluation helps you *sum up* the design
  - **Formative** evaluation helps you *form* the design

- "When the cook tastes the soup, that's formative; when the guests taste the soup, that's summative."  (Robert Stakes)

Summative Usability Evaluation

- **Summative** evaluation
  - Formal summative
    - Supports *science*
    - Uses inferential statistics (traditional human factors testing with rigorous statistics, significant results)
    - Evaluation of an interaction design for *scientific* assessment of usability level, *after development is done*
    - Used for comparison (e.g., with another system)
    - Used to make *public* claims, scientific evidence
    - Done with randomly selected participants
Summative Usability Evaluation

- **Summative** evaluation
  - Informal summative
    - Supports engineering
    - Uses descriptive statistics (not significant results)
      - In support of formative evaluation
      - For internal use only! (For convincing team; no public claims)
    - Can be used for informal comparison (e.g., previous version)
    - Evaluation of an interaction design for engineering assessment usability level

Formative Usability Evaluation

- Formative usability evaluation is:
  - Diagnostic!
  - Center of usability engineering process
  - Evaluation of the interaction design, *as it is being developed*, for the purpose of improving usability
  - Done with carefully selected participants
- Our (usability engineering) focus: formative evaluation
Formative Usability Evaluation

- Formative usability evaluation should be:
  - For evaluating and improving interaction design and user experience
  - An ego-free process
    - You are improving designs, not judging designers
  - We call it usability evaluation or usability testing, but it is NOT “user testing”!

- Done by non-designers, if possible
- Designers should observe
- Early and continual
  - "Those user interface people don't know what they're doing; they can't get it right the first time; they keep changing their minds"
  - Should have something to evaluate no later than 10% into a project schedule
Matching Evaluation Technique to Stage of Development

<table>
<thead>
<tr>
<th>Stage of development</th>
<th>Evaluation approach</th>
</tr>
</thead>
<tbody>
<tr>
<td>Scenarios &amp; screen designs</td>
<td>Design Walk-Throughs</td>
</tr>
<tr>
<td>Low-fidelity prototype</td>
<td>Usability inspection</td>
</tr>
<tr>
<td>High-fidelity prototype</td>
<td>Lab-based usability testing</td>
</tr>
<tr>
<td>Operational product</td>
<td>Remote usability evaluation</td>
</tr>
</tbody>
</table>

Formative Usability Evaluation Methods

- Walk-throughs: very informal, you do the ‘driving’, important early feedback
- Usability inspection
- Lab-based usability evaluation
  - Remember, we do not do user testing!
- Remote usability evaluation methods good for deployed system in the field (e.g., user self-reporting of usability issues)
- Overall state-of-the-art in usability evaluation is some art, some engineering, and some repeatable science (in summative evaluation)
Formative Usability Evaluation Data

- Types of empirical usability evaluation data are same as types of data in usability specifications
  - **Objective**: Directly measured from observed user performance
  - **Subjective**: Based on user opinion

Formative Usability Evaluation

- Types of empirical usability evaluation data
  - **Quantitative** — numeric data and results
    - For assessing usability achievements
    - To monitor converge of design toward usability specifications via iterative cycles
    - To know when to stop iterating
  - **Qualitative** — non-numeric data and results
    - Main input to formative component
    - Most important for finding and fixing usability problems
Phases in Formative Usability Evaluation

- **Prepare (Before evaluation session)**
  - Develop tasks and protocols for experiment
  - Set up equipment and materials
  - Select participants
- **Collect (During evaluation session)**
  - Generate and collect data
- **Analyze and Report (After evaluation session)**
  - Analyze data
  - Draw conclusions
  - Report results

Redesign and iterate
Topics

- Facilities and equipment
- Developing the tasks
- Evaluation protocol
- Participant selection
- Preparing for participants
Facilities and Equipment

- Usability lab
  - Formal lab is nice
    - One-way glass
    - Inter-com
    - Sound isolation
    - Low background noise (e.g., air-conditioning)
    - Controlled lighting levels
    - Digital video and audio equipment, if necessary
    - Disability access
    - Emergency exits
    - Reception room/area

Facilities and Equipment

- Formal lab is not essential
  - Conference room works well
  - Evaluator sits with participant
  - You can still get a lot of useful usability data

- Observation area
  - Pipe video and audio into conference room
  - Better than one-way glass
  - Separates developers/clients from users
  - Gives developers team feeling
  - Afterward can sit and talk about it; can bring users in, too
  - A big way for developers to get the “usability religion”
Facilities and Equipment

- Video of participant (e.g., face and hands) used less often these days (unless important, as for virtual reality)
- Video useful with paper prototypes, too

Facilities and Equipment

- De facto standard: Video of screen action
- Usability data capture software tools (e.g., Morae by TechSmith)
  - Full-resolution video of screen action
  - Audio of user and/or facilitator
- For remote usability evaluation: TechSmith is working on the “Astoria Project”, to be released in 2006
Facilities and Equipment

- High audio quality (on video) is essential and hard to get
  - No battery microphones; risk data loss if batteries go out during session
  - Need high-quality external (read: expensive) mike, not built into camcorder
    - Place on participant's lapel or on top of monitor
    - Need separate amplifier/mixer

- Audio recording backup & note taking
  - Effective if used selectively for note taking, if not too distracting
  - Can be used to capture dialogue with participant (more agile than video taping)
Developing the Tasks

- **Benchmark tasks**
  - *Representative, frequent, and critical tasks for user class* represented by participant
  - Written out in detail, one per sheet, for participant
  - Usually take metrics during participant performance of task (for usability specifications)

- **Unmeasured tasks**
  - Exploratory tasks, also written out in detail; no metrics taken

- **Unstructured tasks** – “Free play” for participant

Evaluation Protocol

- **Training materials**
  - Only when appropriate
  - Create as early as possible
  - None should be needed for a calendar or a ticket kiosk

- **Props and task aids**
  - Have them ready (e.g., telephone)
Evaluation Protocol

- Usability lab vs. field evaluation: Tradeoffs
  - Alpha and beta testing are NOT usability testing
- Typical length of time of evaluation session for one participant: 30 minutes to 4 hours, average 2 hours or less

Evaluation Protocol

- Expected number of full usability engineering cycle iterations per version or release is 3
  - Resource constraints often limit to fewer iterations
  - Any iterations are better than none
- In subsequent cycles of evaluation, typically keep “best” participant from previous cycle, add 2 (or more) new participants
  - A kind of empowerment, to make a difference
Participant Selection

- Representative users
  - Must match target user class for associated benchmark task
  - Knowledgeable of target system domain
  - Know what they don’t like, and user expectations are increasing with usability knowledge

Participant Selection

- User interaction design expert
  - Broadly usability knowledge
  - Can find subtle problems
  - Can offer suggestions for solutions
Participant Selection

Someone old
Been around, knows how it’s done

Someone new
From outside

Someone borrowed
From a different department

Someone blue
Never likes anything; always wants it different

Participant Selection

- How many participants is enough?
  - Focus not on large number of experiments with large number of users, but rather on extracting as much information as possible from every user
  - 1 participant is too few, more than 10 not worth it
  - Optimum number of participants is 3 to 5 per user class per major version/iteration (empirically-based rule-of-thumb)
Participant Selection

- Where does the ‘3 – 5 users’ rule come from?
- Discovery likelihood = 1 - (1 - p)**n
  - Think of as % of problems found as function of # of users (n)
  - Each curve is for a given individual detection rate (p)
Participant Selection

- Then look at the expected number of new problems found for each added participant
- Cost is linear per added participant used
- Total # problems found per unit cost (per user) peaks out around 3-5 users
  - Based on lots of assumptions, such as average detection rate
  - Your smileage can vary

- How many participants is enough?
  - More severe usability problems are typically detected by the first few participants
  - Sometimes could need more participants
    - In a very large application or Web site different users evaluate different parts
Preparing for Participants

- Institutional Review Board (IRB)*
  - To protect the rights of people participating in experiments
  - To protect your organization against liabilities (legally responsible for welfare of all human subjects)
  - Must approve ALL empirical studies using human subjects (even usability testing)
    - When absolutely needed? If in doubt, protect yourself
  - Procedure varies by organization

*Thanks to Dr. Robert Beaton

Preparing for Participants

- Informed consent form
  - Legal requirements
    - Permission must be obtained prior to participation
    - Written in clear, understandable language
    - Voluntary, signed without duress or stress
    - Can withdraw anytime
    - Copy given to participant
Preparing for Participants

- Informed consent form
  - Content includes:
    - Research purposes, procedures, duration of participation
    - Foreseeable risks or discomforts
    - Benefits to participants (e.g., payment or education)
    - Statement of confidentiality (anonymity of data)
    - Contact person and information

- Signed consent forms must be retained for 3 years following IRB approval

Preparing for Participants

- Coverage
  - **ALL** empirical studies using human subjects (even usability testing you do for this class) conducted in the name of university (or any organization) **MUST** be reported to and approved by the IRB
  - Most research in HCI does not put participants at risk and approval is given with 'exempt' status
Preparing for Participants

- Typically, principle investigator (team leader) must submit
  - “Request for exemption” cover letter
  - Statement of complete protocol
  - Written subject instructions
  - Informed consent form
  - Standard IRB forms
- I will submit a blanket application for the class
- You must remember to do this in the future

Preparing for Participants

- IRB approval
  - Normally takes 1-2 weeks
  - May require changes in documents
  - Evaluates ethical and legal issues, not quality of the research
Preparing for Participants

- **Materials for participants**
  - Develop instructions \(\text{[see p.299-300 in book]}\)
    
    “You are helping us evaluate the system-- we are not evaluating you!”
  - Get IRB approval
  - Develop informed consent form and non-disclosure agreements \(\text{[see p. 300 in book]}\)

Preparing for Participants

- **Pilot testing – final shakedown**
  - Pilot testing and rehearsal are essential
  - Design should not have known “show stopper” usability problems

- **Establish evaluator roles**
  - Facilitator – to keep evaluation session going
  - Observer(s) – to help collect data
  - “Executor” – to run paper prototype
Making It Really Work

- Bottom-line for developing the experiment:
  - Creativity rules
  - Ecological validity is important
  - Many variations on the theme
  - Do what is necessary to make it work, to discover usability problems
CS5714 Usability Engineering

Usability Evaluation:
Collect
(During Evaluation Session)

Copyright © 2006 H. Rex Hartson and Deborah Hix

Topics

- Preliminaries with participants
- Generating and collecting quantitative data
- Generating and collecting qualitative data
- Observational data generation/collection techniques
- Forms, variations, attitudes, tools
Preliminaries with Participants

- Welcome and thank participant
- Explain protocol to participant, including any compensation
- Show participant the lab and experimental set-up if they are interested
- Have participant sign informed consent form (and non-disclosure agreement, if desired)

Quantitative Techniques

- Quantitative data are collected to assess usability levels
- Benchmark tasks
  - Measuring time on task, number of errors, etc.
  - Quantitative measures such as timing can be valuable even with paper prototype, though not very precise
- User satisfaction scores
Qualitative Techniques

- Qualitative data are collected to identify usability problems
- Verbal protocol taking
  - Participant “thinks aloud”, talks while performing tasks
    - Can be intrusive, but effective
    - Facilitator sits in room with participant to collect this kind of data; observer(s) can also collect it
  - Can be used for both timed and untimed tasks
    - Studies show it can be done with minimal effect on performance time

Qualitative Techniques

- Verbal protocol taking
  - Some participants not good at talking
  - Facilitator may need to prod participant who stops talking
  - Typically avoid discussion during timed tasks
  - Answer participant questions about what to do with a hint, not a direct answer
    - Ask them what they expected to see, happen
Qualitative Techniques

- Critical incident taking
  - *Critical incident*: Something that happens while participant is working that has significant effect on task performance or user satisfaction
  - Critical incidents are usually indicators of usability problems
  - Later analyze the problem and cause within the interaction design

Qualitative Techniques

- Critical incident taking
  - Arguably single most important kind of formative evaluation data
  - Although participant indicates critical incidents, facilitator and/or observers are responsible for recognizing, identifying, and recording critical incidents
  - In most situations, finding usability problems is bad, but in formative evaluation, finding usability problems is good!
Qualitative Techniques

- Critical incident taking
  - Pay attention to detailed participant behavior!
    - It’s easy to miss critical incidents! It’s a skill; takes experience
  - Example: User wasn’t sure what the alarm clock icon meant
    - Could have had to do with time of day. Redesign suggestion: Show clock “ringing” to emphasize alarm feature
  - Example: User confused by “cancel” label on button in dialogue box showing appointments
    - Subtle: Normally understand “cancel”, but in calendar domain, cancel has meaning of removing appointment

Observational Data Collection Techniques

- Structured interviews
  - Post-session questioning, debriefing
  - Typically obtain general information
- Co-discovery
  - More than one participant, using system together, thinking aloud together
  - Can lead to rich verbal protocol from conversations among participants
Observational Data Collection Techniques

- Primary technique is note taking
  - Comprehensive, real-time notes (e.g., pencil and paper, on-line) during each session
  - Nothing beats this for effective data gathering
  - Do not depend on audio, video recording as primary method; only as backup
- Software tools for critical incident recording

Data Collection Forms

- Form for collecting both quantitative and qualitative data during session

<table>
<thead>
<tr>
<th>TASK NAME:</th>
<th>PARTICIPANT ID:</th>
<th>Task start time:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>No. of errors:</td>
<td>Task end time:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Time to perform task:</td>
</tr>
</tbody>
</table>

Critical Incident Description

1. 
2. 
3. 

Link to screen video sequence

Evaluator’s Comments
Adopt the Right Attitude

- Evolution of developers' attitude as they watch user evaluating a product

  “Stupid user!”
  “Let me at him/her!”
  “It’s his/her (another developer’s) fault!”
  “I’m mud!”
  “Let’s fix it!”

Variations

- Variations on the theme
  - Major point: No rules; do what works best in your situation
  - Evaluator sitting with participant (cf. in separate room)
  - Abandon verbal protocol if it doesn’t work for a participant
  - Try co-discovery with two participants

- Video-clip: Formative evaluation of Envision
Team Exercise – Formative Usability Evaluation

- **Goal:** To perform the data collection part of a very simple formative usability evaluation of your ticket kiosk system
- **Preparation activities:**
  - Assemble in teams (do this now, before we proceed)

---

Team Exercise – Formative Usability Evaluation

- Decide roles for team members (do now):
  - **Prototype executor,** to move transparencies, provide feedback (person who knows design best and who can "play computer")
  - **Evaluation facilitator,** to keep experiment moving, to interact with participants, and to record critical incidents (qualitative data)
  - **User performance timer,** to time participants performing tasks and/or count errors (quantitative data)
  - **Two participants,** to trade to another team
  - **Anyone else can help record critical incidents**
Team Exercise – Formative Usability Evaluation

- Make the switch (do now):
  - Trade your two participants to another team, getting two new participants from a different team (we'll help make this work in a "cycle" among the teams)
  - Your new participants are now permanently on your team (for the rest of these exercises)
  - Sit together in your newly formed teams (now)
  - On a sheet of paper: write team number and names of the two new participants
  - We will now pass out and explain the forms you will use
  - Find your plastic usability specifications form
  - Circle the questionnaire questions you will use

Team Exercise – Formative Usability Evaluation

- Cautions and restrictions
  - Team members must not coach participants as they perform tasks.
  - Person playing computer must not anticipate user actions, especially do not give the correct computer response for a wrong user action! Respond only to what user actually does!
  - Person playing computer may not speak, make gestures, etc.
  - You may not change the design on the fly!
Team Exercise – Formative Usability Evaluation

- How much data to collect?
  - Collect a dozen or more critical incidents in this exercise
  - If you do not get at least a half dozen from each participant, continue with that participant doing exploratory use of your prototype until you get enough critical incidents
    - For example, have them browse through each screen, looking at each object (button, menu, etc.) commenting on and giving their opinion about usability of various features

Team Exercise – Formative Usability Evaluation

- Conduct evaluation session
  - Perform first task with first participant
    - Bring first participant into the “lab”, greet participant, and explain evaluation session
    - Have first participant use your prototype to perform your first benchmark task for your objective usability specifications
    - That is, have participant read first task aloud, then perform that task while thinking aloud
Team Exercise – Formative Usability Evaluation

- Perform first task with first participant (continued)
  - *Executor* moves transparencies in response to participant actions
  - *Facilitator* records critical incidents and keeps session moving
  - *Timer(s)* record timing and error count data as indicated in usability specifications, as user performs task
    - Don’t count participant’s reading aloud of task in task timing

- Perform second task with first participant
  - Have participant read second task aloud and perform it while thinking aloud
  - Have this participant complete questionnaire, and then give them their “reward”

Team Exercise – Formative Usability Evaluation Sessions

- Perform session with second participant
  - Bring second participant into the "lab", greet participant, and explain evaluation session
  - Have the second participant perform all tasks, complete questionnaire, and receive "reward"
  - First participant should stay and help with observations
    - If you still don’t have a list of at least a half dozen usability problems, you need to generate them for the exercise
      - Exploratory use
      - Group discussion
Team Exercise – Now Start Formative Usability Evaluation

- Have your new participants leave the room temporarily
- Get your prototype “booted up”
- Bring first participant into "lab“ and get started on your evaluation sessions
- Go for it!

- Schedule: Complete by end of class
- Deliverables: none (yet), just keep your data
CS5714 Usability Engineering

Usability Evaluation:
Analyze & Report
(After Evaluation Session)

Copyright © 2006 H. Rex Hartson and Deborah Hix

Topics

- Analyzing the data
- Cost-importance analysis
- Drawing conclusions
- Connecting back to usability engineering life cycle
- Team exercise on formative evaluation
Analyzing Qualitative Data

- Identify usability problems & causes (interaction design flaws)
- While participant still present
  - Clarify any ambiguities
  - Interact with participant as doctor does in diagnosing patient
- Create usability problems list
  - Get at real problems and causes in design
  - Include designer knowledge

Analyzing Quantitative Data

- Compare observed results to usability specifications
  - Formative, not summative (statistical significance is not an issue)
- Major decision
  - If usability specifications are met:
    - Accept as is and STOP iterating
    - Keep usability problems data for next time
  - If usability specifications are not met
    - Solve usability problems in order of cost and effect on usability, to iterate this time
Grouping of Related Problem Reports

- Merge together multiple reports for same problem (from different evaluators or evaluation sessions)
  - Get best of each report
- Group together related reports for fixing together
  - Problems that can share common solution
  - Represent in single problem report, but keep original constituent problem reports (in case you want to ungroup later)

Cost-Importance Analysis – Grouping Example

<table>
<thead>
<tr>
<th>Problem group</th>
<th>Problems</th>
<th>Imp.</th>
<th>Problem solution</th>
<th>Group solution</th>
<th>Single cost</th>
<th>Group cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>Purchase Ticket Problem</td>
<td>7. Wanted to enter or choose date and venue first and then “Purchase Ticket”</td>
<td>3</td>
<td>Change to select the date and venue first, and then purchase ticket</td>
<td>More comprehensive change of flow logic and labeling to encompass both solutions</td>
<td>3</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td>17. “Purchase Tickets” took user to page to select tickets and commit to them, but then there is another button there that says the same thing, “Purchase Tickets”, but really means “Pay for Purchase”</td>
<td></td>
<td>Change the label to “Pay for Purchase”</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Eval after
Cost-Importance Analysis

- Best to use a spreadsheet
- **Problem** - Something that has an impact on usability; observed as user interacts
  - Deduced from critical incident data

<table>
<thead>
<tr>
<th>Problem</th>
<th>Imp</th>
<th>Solution(s)</th>
<th>Cost Prio ratio</th>
<th>Prio rank</th>
<th>Cuml cost</th>
<th>Reso lution</th>
</tr>
</thead>
</table>

User didn't know to select appt before trying to delete
Cost-Importance Analysis

- Importance to fix—effect on usability (independent of cost), based on best engineering guess (include risk of not fixing)
  - Importance=M: must fix, regardless
  - Importance=5: If interaction feature is mission critical or usability problem has major impact on task performance or user satisfaction (e.g., user cannot complete key task), expected to occur frequently, could cause costly errors, or major source of dissatisfaction

- Importance=3: If user can complete task, but with difficulty (e.g., caused confusion and required extra effort), or problem was a source of dissatisfaction
- Importance=1: If problem did not impact task performance or dissatisfaction much (e.g., irritant or cosmetic), but is still worth listing
Cost-Importance Analysis

- Adjustment factors for *Importance*
  - Probability of occurrence
    - Over all affected user classes, how often will user encounter this problem?
    - Example: if task cannot be completed (e.g., Importance = 5) but usability problem represents situation that will arise rarely and not critical task, downgrade importance to 4 or 3
    - Example: If impact is moderately significant (3), but occurs frequently, upgrade to 4
  - Learnability
    - If users can learn to solve it immediately, it won’t affect subsequent usage (reduce Importance by 1)

<table>
<thead>
<tr>
<th>Problem</th>
<th>Imp</th>
<th>Solution(s)</th>
<th>Cost</th>
<th>Prio ratio</th>
<th>Prio rank</th>
<th>Cuml cost</th>
<th>Resolution</th>
</tr>
</thead>
<tbody>
<tr>
<td>User didn’t know to select appt before trying to delete</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Cost-Importance Analysis

- Solution(s)--proposed changes to solve problem
  - Design principles and guidelines, brainstorming
  - Study other similar designs
  - Solutions suggested by users and experts
  - One problem can have several solutions
  - Not a good solution: More training or documentation
    - Adjust the design, not the user!
    - You cannot “train-in” usability
    - Don’t say “they’ll figure it out”; your evaluation showed that is not true

Cost-Importance Analysis

- Solution for Calendar System

<table>
<thead>
<tr>
<th>Problem</th>
<th>Imp</th>
<th>Solution(s)</th>
<th>Cost</th>
<th>Prio ratio</th>
<th>Prio rank</th>
<th>Cuml cost</th>
<th>Reso lution</th>
</tr>
</thead>
<tbody>
<tr>
<td>User didn’t know to select appt before trying to delete</td>
<td>3</td>
<td>Allow user to click in time slot then press delete; add pop-up hint ‘Click in appointment to select for deletion’</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Cost-Importance Analysis

- **Cost** – resources (time, money) needed for each proposed solution
  - Should include cost of redesign
  - Changes to paper prototype are minimal cost, but the redesign can have substantial cost
  - Cost more significant in computer-based prototypes and versions of real product
  - Usually units of cost are in person-hours (round up fractional values)
  - For problems with multiple possible solutions, give cost for each on separate line (presumably importance is same)

### Cost-Importance Analysis

<table>
<thead>
<tr>
<th>Problem</th>
<th>Imp</th>
<th>Solution(s)</th>
<th>Cost</th>
<th>Prio rank</th>
<th>Cuml cost</th>
<th>Reso lution</th>
</tr>
</thead>
<tbody>
<tr>
<td>User didn’t know to select appt before trying to delete</td>
<td>3</td>
<td>Allow user to click in time slot then press delete; add pop-up hint ‘Click in appointment to select for deletion’</td>
<td>4 person-hours</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Cost-Importance Analysis

- **Priority ratio** – metric to establish priorities
  - To compute: \( \frac{\text{importance}}{\text{cost}} \times 1000 \)
  - Intuitively, high priority means high importance, low cost
- Compute priority ratios for each usability problem, including groups as single problems

Compute priority ratios for the Calendar System

<table>
<thead>
<tr>
<th>Problem</th>
<th>Imp</th>
<th>Solution(s)</th>
<th>Cost</th>
<th>Prio ratio</th>
<th>Prio rank</th>
<th>Cuml cost</th>
<th>Reso lution</th>
</tr>
</thead>
<tbody>
<tr>
<td>User didn’t know to select appt before trying to delete</td>
<td>3</td>
<td>Allow user to click in time slot then press delete; add pop-up hint ‘Click in appointment to select for deletion’</td>
<td>4</td>
<td>750</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Cost-Importance Analysis

- Filling in more rows for Calendar System

<table>
<thead>
<tr>
<th>Problem</th>
<th>Imp</th>
<th>Solution(s)</th>
<th>Cost</th>
<th>Prio ratio</th>
<th>Prio rank</th>
<th>Cuml cost</th>
<th>Reso lution</th>
</tr>
</thead>
<tbody>
<tr>
<td>User didn’t know to select appt before trying to delete</td>
<td>3</td>
<td>Allow user to click in time slot then press delete; add pop-up hint ‘Click in appointment to select for deletion’</td>
<td>4</td>
<td>750</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>abc</td>
<td>5</td>
<td>Yada yada</td>
<td>5</td>
<td>1000</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>pqr</td>
<td>M</td>
<td>Blah blah</td>
<td>20</td>
<td>M</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>xyz</td>
<td>1</td>
<td>Sure, right</td>
<td>3</td>
<td>333</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

19 Eval after

Cost-Importance Analysis

- Move all “must fix regardless of cost” problems to top of table
- Sort rest of table in descending order by priority ratio
  - “Must fix” problems first
  - High importance, low cost problems next
  - These are the problems that, when fixed, give biggest improvement in usability for least cost

20 Eval after
### Cost-Importance Analysis

- Listing M problems at top, sorting rest by Priority ratio to give **Priority rank**, and computing **Cumulative cost**

<table>
<thead>
<tr>
<th>Problem</th>
<th>Imp</th>
<th>Solution(s)</th>
<th>Cost</th>
<th>Prio ratio</th>
<th>Prio rank</th>
<th>Cuml cost</th>
<th>Reso lution</th>
</tr>
</thead>
<tbody>
<tr>
<td>pqr</td>
<td>M</td>
<td>Blah blah</td>
<td>20</td>
<td>M</td>
<td>1</td>
<td>20</td>
<td></td>
</tr>
<tr>
<td>abc</td>
<td>5</td>
<td>Yada yada</td>
<td>5</td>
<td>1000</td>
<td>2</td>
<td>25</td>
<td></td>
</tr>
<tr>
<td>User didn’t know to select appt before trying to delete</td>
<td>3</td>
<td>Allow user to click in time slot then press delete; add pop-up hint ‘Click in appointment to select for deletion’</td>
<td>4</td>
<td>750</td>
<td>3</td>
<td>29</td>
<td></td>
</tr>
<tr>
<td>xyz</td>
<td>1</td>
<td>Sure, right</td>
<td>3</td>
<td>333</td>
<td>4</td>
<td>32</td>
<td></td>
</tr>
</tbody>
</table>

### Cost-Importance Analysis

- Draw the “line of affordability”: Assume we have 26 person-hours available for changes; draw line just before **Cuml cost** exceeds

<table>
<thead>
<tr>
<th>Problem</th>
<th>Imp</th>
<th>Solution(s)</th>
<th>Cost</th>
<th>Prio ratio</th>
<th>Prio rank</th>
<th>Cuml cost</th>
<th>Reso lution</th>
</tr>
</thead>
<tbody>
<tr>
<td>pqr</td>
<td>M</td>
<td>Blah blah</td>
<td>20</td>
<td>M</td>
<td>1</td>
<td>20</td>
<td></td>
</tr>
<tr>
<td>abc</td>
<td>5</td>
<td>Yada yada</td>
<td>5</td>
<td>1000</td>
<td>2</td>
<td>25</td>
<td></td>
</tr>
<tr>
<td>User didn’t know to select appt before trying to delete</td>
<td>3</td>
<td>Allow user to click in time slot then press delete; add pop-up hint ‘Click in appointment to select for deletion’</td>
<td>4</td>
<td>750</td>
<td>3</td>
<td>29</td>
<td></td>
</tr>
<tr>
<td>xyz</td>
<td>1</td>
<td>Sure, right</td>
<td>3</td>
<td>333</td>
<td>4</td>
<td>32</td>
<td></td>
</tr>
</tbody>
</table>
Drawing Conclusions

- Deal with “Must fix” problems
  - If enough resources, fix
  - Otherwise, someone (e.g., project manager) must decide
    - Sometimes fixing “must fix” problems means no resources left for any other problems
    - Extreme cost of a “must fix” problem could make it infeasible to fix in current version
    - Exceptions (with cost overruns) can be dictated by corporate policy, management decision, marketing, etc.
  - In our example, we have (barely) enough resources

---

### Resolution for Calendar System

<table>
<thead>
<tr>
<th>Problem</th>
<th>Imp</th>
<th>Solution(s)</th>
<th>Cost</th>
<th>Prio ratio</th>
<th>Prio rank</th>
<th>Cum cost</th>
<th>Resolution</th>
</tr>
</thead>
<tbody>
<tr>
<td>pqru</td>
<td>M</td>
<td>Blah blah</td>
<td>20 M</td>
<td>1</td>
<td>20</td>
<td>Fix</td>
<td></td>
</tr>
<tr>
<td>abc</td>
<td>5</td>
<td>Yada yada</td>
<td>5 1000</td>
<td>2</td>
<td>25</td>
<td>Fix, if time</td>
<td></td>
</tr>
<tr>
<td>User didn’t know to select appt before trying to delete</td>
<td>3</td>
<td>Allow user to click in time slot then press delete; add pop-up hint ‘Click in appointment to select for deletion’</td>
<td>4 750</td>
<td>3</td>
<td>29</td>
<td>Table until next version</td>
<td></td>
</tr>
<tr>
<td>xyz</td>
<td>1</td>
<td>Sure, right</td>
<td>3 333</td>
<td>4</td>
<td>32</td>
<td>Probably never</td>
<td></td>
</tr>
</tbody>
</table>

Eval after
Drawing Conclusions

- Sometimes get ties on Priority Ratios
  - Must break ties by Importance, personal preference, etc.
  - If Importance (which reflects severity, impact on user, etc.) is more significant factor in your environment, weight it a little more in the formula (for Priority Ratio)

Connection Back to Usability Engineering Lifecycle

- Implement chosen design solutions
- Realize benefits of improved usability
- Cycle back through life cycle process and evaluate again
- Usability testing controls iterative process
  - Stop when achieve usability specifications
  - Goal is not perfection

Stop!
Connection Back to Usability Engineering Lifecycle

- **Cost-effectiveness**
  - If schedule for first release is too tight for testing, use:
    - Usability inspection
    - Isolate most severe problems, "show stoppers"
    - Leave rest until next release
  - BUT don’t let tight production schedules force release of something that could embarrass your organization
  - Quality is remembered long after schedules are forgotten!

*Video-clip:* Analysis of Envision evaluation sessions

Team Exercise – Usability Data Analysis

- **Goal:** To perform the analysis part of a very simple formative usability evaluation

**Activities:**
- Assemble in same teams (including your new participants)
- Fill in "Observed results" column on plastic usability specification table
- Together, team compiles results to determine whether usability specifications were met
Team Exercise – Usability Data Analysis

- Organize usability problem list and perform cost-importance analysis
  - Using paper cost-importance table (or laptop spreadsheet), list a dozen or more usability “Problems” from critical incidents.
  - Assign an "Importance" (to fix) rating, 1 through 5, to each observed problem
  - Propose "Solutions" (without doing all the work of re-design)

Team Exercise – Usability Data Analysis

- Continue Cost-Importance Analysis
  - Group together any related problems and list as single problem
  - Assign “Cost” values (in person-hours) to each solution
    - For cost estimates, pretend your prototype is on a computer (not paper)
  - Compute “Priority ratios"
Team Exercise – Usability Data Analysis

- Analyzing results:
  - Using paper cost-importance table, list “Must fix” problems to top
  - Sort remaining problems by decreasing Priority Ratios to determine Priority Rank of usability problems
  - Fill in Cumulative Cost column
  - Assume hypothetical value for available time resources (something to make this exercise work)
  - Draw the cutoff, line of affordability
  - Finalize your “management” decisions (Resolution) about which changes to make in the next version

Team Exercise – Usability Data Analysis

- Deliverables
  - Summary of quantitative results, written in “Observed results” column on plastic usability specification table form (for comparison)
  - List of raw critical incidents
  - Plastic cost-importance table form containing 3 usability problems selected as interesting to present to class (complete across all 3 rows)
  - Choose someone to give brief report on evaluation results.

- Schedule: Complete by end of class
CS5714 Usability Engineering

User Interaction Design Guidelines & Principles

Copyright © 2006 H. Rex Hartson and Deborah Hix.

Topics

- Affordances
- Organizing usability issues: The User Interaction Cycle
- The User Action Framework
- Selected user interaction design guidelines
- In-class exercises on guidelines
Affordances

- **Affordances*** – an essential concept
  - An affordance gives or provides something that helps a user do something
  - In interaction design, affordances are characteristics of user interface objects that help users perform tasks


Cognitive affordance

- A design feature that helps, aids, supports, facilitates, or enables thinking and/or knowing about something
- Often depend on shared conventions
- Example: Clear and precise words in button label enabling users to understand meaning of button in terms of functionality behind it and consequences of clicking on it
- Plays starring role in interaction design for less experienced users
Affordances

- Physical affordance
  - A design feature that helps, aids, supports, facilitates, or enables physically doing something
  - Example: Adequate button size and easy-to-access location enable users to click easily on the button
  - Plays starring role in interaction design for experienced (power) users

Affordances

- Sensory affordance
  - A design feature that helps, aids, supports, facilitates, or enables user in sensing (e.g., seeing, hearing, feeling) something
  - Includes design features or devices associated with visual, auditory, haptic/tactile, or other sensations
  - Plays critical supporting role to cognitive affordance and physical affordance
  - Example: Large enough font in button label text and appropriate color contrast to support legibility
Affordances

- Functional affordance
  - Functionality of non-UI software
  - Adds purpose for physical affordance
  - Adds sense and goal orientation to design discussion
  - Connects usability with usefulness (of system functionality)
  - About higher-level user enablement in the work domain

Affordances work together in design
- Example: Devices for opening doors (round doorknobs and lever-type door handles)
- Sensory design clearly shows cognitive and physical affordances
- Visual design of each conveys cognitive affordance via implied message “this is what you use to open a door”
- Doorknob and lever suggest, each in own way, physical grasping and rotating for door operation
- But message is understood only because of shared cultural conventions
Affordances

- Affordances work together in design
  - Door operating devices also provide physical affordance, to help users open and close doors
  - Some devices work better than others as physical affordances
  - Push-bar on double doors is another type of physical affordance for doors

Affordances

- False cognitive affordances misinform and mislead
  - Example: Web page links that only look like buttons
  - The “booby-trapped” X in a pop-up advertisement
  - Horizontal line in Web page that falls at bottom of screen
Affordances

- Trails of user-made artifacts
  - Tape added to shovel handle, Post It note added to monitor or keyboard
The User Interaction Cycle

- Framework based on sequence of sensory, cognitive, and physical actions by user during interaction with a machine
  - Adapted and Extended Norman’s “stages of action”

- All about what users think, do and see during cycle of interaction with computer
  - THINK – represents all cognitive actions
  - DO – represents all physical actions
  - SEE – represents all sensory actions

The User Interaction Cycle

- Organizes usability concepts in terms of user actions during task performance
- Works for interaction with any kind of machine and any style of interaction
- Puts usability problems in context of user actions
Norman’s ‘Stages of Action’ Model

- User interaction with any machine

Transition to Interaction Cycle
**Transition to User Action Framework (UAF) – Interaction Knowledge Base**

- Interaction Cycle is highest level of categories in UAF

**Diagram:**
- Interaction Cycle
  - Planning
  - Translation
  - Physical Actions
  - Assessment
  - Outcomes

Hierarchically structured knowledge base of usability issues, concepts, and guidelines

**Norman vs. UAF**

- Norman’s model and our Interaction Cycle are both about what users do within cycle of interaction with computer or other machine
- The UAF is about design, about how interaction designs support users in performing sensory, cognitive, and physical actions during interaction with a machine
The User Action Framework

- Hierarchical knowledge base of usability concepts and issues organized on Interaction Cycle
  - Organizes usability concepts in terms of user actions during task performance
  - Works for interaction with any kind of machine, any style of interaction
  - Puts usability problems in context of relevant design guidelines and principles
  - Not claimed complete, but self-extending

The User Action Framework

- Integrated framework for UE tools for:
  - Usability inspection
  - Usability problem analysis (problem extraction and diagnosis)
  - Usability problem reporting
  - Usability data management
  - Design guidelines
Organizing Design Guidelines by Interaction Cycle

- Simplest view of the Interaction Cycle
  - **Planning** (What to do)
  - **Translation** to determine actions (How to do it)
  - **Physical actions** (Doing it)
  - **Assessment** of outcome (Did it turn out right?)

- Next: Selected design guidelines for each part of the Interaction Cycle

Planning – User Model of System

- Provide clear model of how users view system in terms of tasks
  - Help users with system model, metaphors, work context
  - Help users plan goals, tasks
    - Help users decompose tasks logically
Planning - User Awareness

- Make clear all possibilities for what users can do at every point
- Keep users aware of system state for planning next task
- Keep users aware of task progress (what’s been done and what’s left to do)

Translation

- Issues include:
  - Existence (of cognitive affordance)
  - Presentation (of cognitive affordance)
  - Content and meaning (of cognitive affordance)
Translation – Affordance Existence

- Provide effective cognitive affordances – cues (e.g., in labels, data field formats, icons) that help users get access to system functionality
  - Help users know/learn what action are needed to carry out intentions
  - How to do something at action/object level
    - Users get knowledge from experience, training, AND: cognitive affordances in design
  - Help users predict outcome of actions
  - Help users determine what to do to get started

Translation – Affordance Presentation

- Support user with effective sensory affordances in presentation of cognitive affordances
  - Make noticeable
    - Object contrast, size, layout complexity, location with respect to user focus
  - Make legible, readable (discernable)
    - Font size, font type, color, and contrast
  - Control complexity with effective layout, organization, and grouping
    - Avoid screen clutter
Translation – Content/ Meaning

- Help user determine actions with effective content/meaning in cognitive affordances
  - Design for clarity
    - Use precise wording in labels, menu titles, menu choices, icons, data fields
    - Example: “adjust speed” or “clockwise to increase speed” rather than “adjust” or “speed”
    - Use dynamically changing labels when toggling (e.g., Play/Pause, Partial view/Full view)

Translation – Content/ Meaning

- Design for clarity
  - Provide clearly marked exits
    - Example: “Return to XYZ” instead of Cancel or Exit
  - Provide clear “Do It” mechanism
    - Example: for add-record task, use “Add Record” instead of just Ok or Return
  - Be predictable; help users predict outcome of actions (feedforward)
  - Predictability helps learning and error avoidance
Translation – Content/meaning

- Design for clarity
  - Be consistent (a guideline with interpretation difficulties)
    - Consistency: Similar semantics \(\leftrightarrow\) similar syntax (wording or user actions)
    - Use consistent wording in labels for menus, buttons, icons, fields
    - Use consistent layout/location for objects across screens
    - Custom style guides help consistency
  - Use appropriate layout and grouping by function to convey content and meaning

- Design for cognitive directness
  - Furnish useful defaults (e.g., most likely values, cursor position)
  - Support human memory limits with recognition over recall
  - Minimize mental transformations
  - Examples
    - Dreamweaver ftp function
    - Others [thanks to Paul Kemmerling]
Translation – Content/Meaning

- Design for completeness
  - Be complete; include enough information for users to determine correct action
  - Use enough words for unambiguous labels
    - Long labels are not necessarily bad
  - Give enough information for users to make confident decisions
    - Prevent loss of productivity due to hesitation, pondering
  - Give enough alternatives for user needs

- Help users avoid errors
  - Example
    - Disable buttons, menu choices to make inappropriate choices unavailable
    - Gray out to make inappropriate choices appear unavailable
  - Offer constructive help for error recovery
    - “To err is human; forgive by design”
    - Provide clear way to undo (multiple levels) and reverse actions
Translation – Content/Meaning

- Design carefully for modes
  - Modes are states where actions have different meanings
  - Distinguish modes clearly
  - Avoid confusing modalities
    - Users cannot easily shift focus
    - Even works against expert users

Translation – Task Structure and Control

- Support user with effective task structure and interaction control
  - Keep users in control
    - Avoid feeling of loss of control (e.g., bossy attitude projected to users)
      - Example: “You need to answer your mail” or “Enter next command” vs. “Ready for next command”
Translation – Task Structure and Control

- Design task structure for flexibility and efficiency
  - Provide alternative ways to perform task
  - Provide shortcuts (e.g., hot keys)
  - Make the most of user's work
    - Examples:
      - User fills out part of form & goes away; don’t let them return to empty form
      - Retain user preferences; retain navigation through directory structures
      - Avoid requirement to retype or copy from one place to another

Translation – Task Structure and Control

- Design natural interaction control
  - Give direct manipulation support
    - Example: Direct editing of text object instead of requiring user to ask system to put it away
  - Anticipate likely related tasks; support task thread continuity ('happy path')
    - Example: if message suggests something, offer an easy way to do it
  - Always provide a way to 'bail out'
    - Example: Error message box has buttons for Task A, Task B (not enough: needs Cancel, too)
Translation – Different User Classes

- Accommodate different levels of expertise/experience with preferences

<table>
<thead>
<tr>
<th>Lead</th>
<th>Novice</th>
</tr>
</thead>
<tbody>
<tr>
<td>Follow</td>
<td>Intermittent</td>
</tr>
<tr>
<td>Get out of the way</td>
<td>Expert</td>
</tr>
</tbody>
</table>

- Don’t let affordances for new users be performance barriers to experienced users

Physical Actions – Sensory and Physical Affordances

- Support user with effective sensory affordances for **sensing** physical affordances (e.g., help in **seeing** objects to manipulate)
  - Make objects discernable, legible, noticeable
- Support user with effective physical affordances for manipulating objects – help in **doing** actions
Physical Actions – Doing Actions

- Avoid physical awkwardness
  - Example: Time-consuming switches between multiple input devices (e.g., mouse and keyboard, touchscreen)
  - Accommodate physical disabilities–limited motion, motor control, vision, hearing

- Design layout to support manual dexterity (e.g., Fitts’ law)
  - Support hand-eye coordination limits by making selectable objects large enough
  - Locate related clickable objects close together
    - Avoid fatigue, slow performance
    - But not too close
    - Avoid erroneous selection
The System’s Turn – Outcome and System Response

- Outcome is internal computation or state change
  - Not directly visible to user
  - Interaction designer must make visible via feedback in system response
- System response is only way user knows about outcome of actions
  - System response can contain:
    - Feedback – Information about course of interaction so far
    - Information display – Results of outcome computation
    - Feed-forward – Information about what to do next

System Response

- System response example
  - “The value you entered for your name was not accepted by the system.” (feedback → Assessment)
  - “Please try again using only alphabetic characters.” (feed-forward → Translation)
Outcomes

- Includes all issues about system (non-user-interface) functionality
  - Missing features
  - Non-user-interface software bugs
- Avoid too much automation and *real* loss of control
  - Example: Changing folder name “IRS” to “Irs”
  - Unnecessarily preemptive dialogue box interrupts normal planning

Assessment

- Assessment issues are similar, parallel to those for Translation, except for feedback
  - Existence (of feedback)
  - Presentation (of feedback)
  - Content, meaning (of feedback)
Assessment - Existence

- Provide feedback
  - No news is no news!
  - Feedback keeps users on track
- Provide progress report on long operations (e.g., percent-done indicator)
- Request confirmation as a kind of intervening feedback, to prevent errors (especially for potentially destructive actions)
  - But don’t overuse and annoy

Assessment – Presentation

- Support user with effective sensory affordances in presentation of feedback
  - Make feedback noticeable
    - Locate feedback within user focus of attention
    - Make large enough to see
    - Present feedback promptly
    - Make feedback persistent (avoid flashing)
**Assessment – Presentation**

- Use most effective presentation medium
  - Consider audio as alternative channel
    - To get attention if heavy task or sensory work load
    - For vision impaired users

**Assessment – Content, Meaning**

- Support user with effective **content/meaning** in feedback
  - Design for clarity
    - Support clear understanding of outcome (system state change), so users can assess effect of actions
    - Give clear indication of error conditions
Assessment – Content, Meaning

- Design for completeness
  - Provide enough feedback so users can be either confident their command worked or certain about why it didn’t
- Help users understand what the real error is
- Provide helpful, informative error messages, not “cute” unhelpful messages

Assessment – Content, Meaning

- Design feedback wording (especially error messages) for positive psychological impact
  - Make system take blame for errors
  - Be positive, to encourage
  - Avoid violent, negative, demeaning terms
  - Avoid use of “illegal”, “invalid”
- Employ user-centered wording (language of user and work context) in displays, messages, other feedback

Unsupported

Psychological

Try again

Illegal

Gobbledygook
Assessment – Content, Meaning

- Design for consistency
  - Label outcome (e.g., title of new screen or dialogue box) consistently with starting point and action (e.g., button label or menu choice)
  - Organize feedback for ease of understanding
    - Provide user control over amount and detail of feedback
    - Give only most important information; more on demand

Assessment – Information Displays

- Organize information displays for ease of understanding
  - Eliminate unnecessary words
  - Group related information
  - Control density of displays; use white space to set off
  - Columns are easier to read than wide rows
  - Use abstraction per Shneiderman’s “mantra”: Overview first; zoom and filter; details on demand
Overall

- Overarching issues, not just in one part of User Interaction Cycle (e.g., global wording, style, color)
- Examples of overall style issues
  - Use user-centered (language of user and work context) wording
  - Avoid anthropomorphism-attributing human characteristics to non-human objects
  - Avoid poor attempts at humor
    - Easy to do badly
    - Easily misinterpreted

Overall

- Examples of overall style issues
  - Avoid irritation in displays (e.g., bright color, blinking, audio, offensive messages)
    - Use pastels, not bright colors
    - Be aware of color conventions (e.g., avoid red, except for urgency)
    - Allow user settings, preferences (e.g., sounds levels, blinking, color)
    - Watch out for focusing problem with red and blue

Red and blue
Overall

- Examples of overall style issues
  - Make presentation of text legible
    - Make font size large enough for all users
    - Use good contrast (color and intensity) with background
    - Use mixed case for extensive text
    - Avoid too many different fonts, sizes
    - Use legible fonts (e.g., Ariel, sans serif Verdana, or Georgia for online reading)
    - Use color other than blue for text
    - Accommodate sensory disabilities and limitations (e.g., visually challenged, color blind)

Design Guidelines: Conclusions

- Be cautious using guidelines
  - Need careful thought and interpretation
  - In application, they can conflict and overlap
  - They do not guarantee usability
  - Using guidelines does NOT eliminate need for usability testing

- Design by guidelines, not by politics or personal opinion

Dr. J. Foley: “The only correct answer to any UI design question is: It depends”
What’s Wrong With This Picture?

What interaction design guideline(s) does each of these situations violate?

- This message appears after the user issues a web search command (assume quick communications performance):
What’s Wrong With This Picture?

- A field in a form for an inventory control system has this label
  - This is just part of the form; control buttons are elsewhere

What’s Wrong With This Picture?

- The user clicks on the sort button and gets this message:
What’s Wrong With This Picture?

- This is a message I got when I tried to print a document:

![Printers Folder]

There was an error writing to LPT1. for the printer [HP LaserJet 6P/6MP - PostScript].
The printer is out of paper. Add more paper.
To continue printing, click retry.
Windows will automatically retry after 5 seconds.

What’s Wrong With This Picture?

- A “help” screen contains this information on the system’s “help” facility:

The comma key on the minikeypad is the HELP key for forms. While in the ABC-Style Editor and Calendar, use PF2 for HELP; use "H" for HELP while in the Desk Calculator; use the "gold" key plus "H" key while using the XYZ-Style Editor. ...By the way, if you need help creating a document, it is better to be in the Word Processing Menu when you press HELP rather than in the main menu.... It is a good idea to remember the location and purpose of each key mentioned above.
What’s Wrong With This Picture?

- This form is used by the Human Resources Department.

What’s Wrong With This Picture?

- The following dialogue box appears:

![Dialogue Box Example](image)
What’s Wrong With This Picture?

- The following dialogue box appears:

![Work-in-progress Dialog](image)

What’s Wrong With This Picture?

- The following dialogue box appears:

![Action Required!](image)
What’s Wrong With This Picture?

- The following dialogue box appears:

```
Completely erase disk named "document 1" (Internal drive)?

Cancel  Eject  Initialize
```
CS5714 Usability Engineering

Usability Inspection

Copyright © 2006 H. Rex Hartson and Deborah Hix

Topics

- Heuristic, or expert, inspection
- Advantages and disadvantages
- Web usability inspection
- Usability inspection process
Usability Inspection

- **Usability inspection** is a design guidelines-based usability evaluation technique
- Especially suitable for early evaluation of an evolving design

Usability Inspection

- Done by one or more usability experts as inspectors in role of “usability problem detective”
  - Not on interaction design team
  - “Double expert” is best
  - Does not use actual users
  - Inspectors ask themselves questions about what would cause users problems
  - Inspector gives expert opinion predicting usability problems
Usability Inspection

- Advantages
  - Some think inspection is single most cost-effective method to improve usability
    - Our experience supports this
  - Provides design team with independent (of team) perspectives and experience
  - Complements usability testing with users

- Disadvantages
  - Does not use real users
  - Not a substitute for lab-based usability testing
  - Experts may not know system in depth
  - May find “false positives”
  - May find higher proportion of lower severity problems
Assessing a Web Design Via Inspection

- Web inspection is same as for GUI, but based on Web-specific design guidelines (next section in this course)
- Examples
  - Availability and accessibility of needed information
    - Information is there AND user can easily find it
  - Navigation
    - Helps user avoid getting ‘lost in space’
  - Scanability of links
    - Users can follow sequences of link without having to read much text

Usability Inspection

- Usability inspectors assess specific user interaction design by determining guidelines both violated and supported
  - Inspector or inspection team selects small, tractable set (about 10) of most important guidelines as basis for inspection questions
    - Example guideline: Use “natural” language of user
    - Example question: Does design use “natural” language (application- and domain-context) of user?
Usability Inspection

- Typically 2 or 3 inspectors
  - Gives diversity of opinion
  - Typically, one inspector finds only about 30% of problems
  - Can have up to 5 inspectors for important part of design
- Each inspector separately assesses interaction design
- Also effective to work together, for co-discovery
- Based on questions, each inspector gives expert opinion predicting usability problems (not just own usability incidents)

- Then, all inspectors work together
  - Merge problem lists
  - Select most important to fix
  - Brainstorm suggested solutions
- Based on findings, experts recommend modifications to improve usability of site
- Realistic variation: Usability design inspection not specifically guideline-driven, but using expert experience and knowledge
Usability Inspection

- Reporting results
  - Give overview of evaluated system
  - Give overview explanation of inspection process
  - List inspection questions based on guidelines used
    - Example: Does design give good feedback throughout?
  - For each guideline
    - Give specific examples of design violations and examples of support
    - Give explanation and screen image (if available), and, if Web site, give URL
    - Give suggestion(s) for redesign (if appropriate)

- List “Top 3” (or 4 or 5) suggestions for modifications
  - Prioritization of suggestions, to give biggest improvement in usability
  - Based on most frequently visited screens, screens with most usability problems, guidelines most often violated, minimum resources to make changes
Topics

- Why Web usability?
- Web user behavior
- Web usability design guidelines
Why Web Usability?

- For many, your Web site is you
- But: Users/customers won’t tell you if your site isn’t usable!

How is Web Development Different (from GUI development)?

- Everyone wants a Web presence
- “We’re working in Internet/Web time”
  - Meaning: Have to do it fast; no time to do it right
- Anyone can be both Web author and publisher, with no review process!
- “How hard can this be? We don’t need to bother with a real development process!”
Web User Behavior – Web Design Influences

- Most users
  - Don’t want to read lots of text on-line
  - Scan first for key words and links
  - Are "casual" users, browsing a site
  - Don’t browse aimlessly
  - Are interested in small part of a site
  - Are put off by sloppy layout, poor grammar, misspellings

- Many users
  - Are annoyed by flashing, banners, etc.
  - May not be English-speaking or US citizens/residents
  - May be over 40!
  - Will get lost on the Web!

Although important, we will not address internationalization, disabilities, etc.
Guidelines for Web Design

- Guidelines for Web design relate to:
  - Organization and structure
  - Navigation
  - Format, content, and appearance
    - Content and navigation have to be done together
  - Housekeeping
    - Of site and of individual pages

Organization and Structure

- Make important information easy to find
  - What are most likely and/or important tasks a user will do at the site?
  - What information are they most likely to want to find most frequently?
Organization and Structure

- Users tend not to form mental model of sites
  - Sites usually designed toward moving forward, not backing out
  - E.g., people use Back button far less than one would expect!
  - E.g., when users get lost, they don’t back out to familiar territory; just keep going
    - Don’t think much about where they have been
  - Users tend to come back to Home page to go somewhere else, even though needed link might be on page where they were

Organization and Structure

- Use “site bites”: Organize information so each piece fits on a single screen
- Organize site, when possible, for speed of presentation of information
  - Common Web user complaint is slow downloads!
Organization and Structure

- Make site as browser-, platform-, and resolution-independent as possible
- Take great care with home page design
  - First impression very important
  - Gets seen more often than any other page
- Use frames with caution
  - Disorienting, restrict amount of presented information, necessitate more scrolling

Organization and Structure

- Allow user to adjust frame boundaries
- Generally use to hold navigation options if main area of site changes frequently
  - Frames as tables of contents (TOC) with links can help user performance
    - User clicks on link in TOC frame and regular frame changes content
  - Separate frame can also be used effectively for glossary, index, help
Navigation

- Two key user questions:
  - Where am I?
  - How do I get to X?
  - This is as far as most users go towards a mental model of a site

- Keep navigation as simple as possible
  - E.g., by following hierarchical organization

- Show site map for larger sites
  - Make sure user can always determine current location within site

Navigation – Search

- Have good “search” feature in your site
  - About one third of users use site search as initial strategy
    - Similar use of FAQs
  - Problems to watch for:
    - User does not understand scope of search
      - Whole site or just part
      - How to limit search to part
Navigation – Search

- Problems to watch for:
  - User not able to interpret search results
  - Organization and display of results (e.g., by relevance measure)
  - Often result in large lists to scan
  - Not enough information in result of list items, not descriptive of content

- Design to help users with searching
  - Help users browse results
    - Previous and Next buttons
      - Allow sideways movement through list without coming back up to list
      - But limit to sequential viewing
  - Help users choose from among results
Navigation

- Use navigation options consistently throughout site
  - Navigation bars
  - Links at top and bottom
  - Redundant navigation options
- Put navigation options where they can’t disappear
- Same for other kinds of controls
  - Make relation of controls to pages, frames, boxes, etc. clear

- Provide short-cuts for most likely task paths
- Never make a use scroll horizontally (from left to right)
- Minimize the need to scroll vertically (from top to bottom)
Navigation

- Use links liberally but appropriately
  - Text links are vital
    - Downloading delays can mean text links are visible first
    - Users may look at text links before trying image links
  - Use meaningful link content—perhaps single most important factor in Web site
    - Use precise, unambiguous wording

- Avoid “click here” or “here” or “go to”
- Long (several words) links can be better for precision and differentiation; short wording often too vague
- Link predictiveness: Be clear where link will take user
  - May be single most important guideline of entire course! You might improve the average site’s usability by 50% by following this one guideline.
- Roll-overs (fly-overs) may help explain link
Navigation

- Corollary is *distinguishability*: How distinguishable is each link from the others?
- Users may select correct link by eliminating ones they do not want
  - Text link layout
    - Affects user performance
    - Links embedded in text don’t always work well especially for scanning
      - Studies found strong negative effect
      - Goes against intuition - text should add explanation

Navigation

- Wrapped links cause confusion about what is a link
  - Happens especially in multi-column format or narrow frames
  - Clarify by adding bullets or button outline to set off items
  - Be consistent in style of links
Navigation

- Image (graphical) links
  - Often do not help performance much (over text links)
  - Often do not look selectable
  - Users will look for them with cursor
  - Do not change color to indicate already used
    - Use border or outline box to show selectable and to show visited (e.g., border color change)

- Image (graphical) links
  - Interfere with process of elimination in finding right link
  - Interfere with scanning (one of the most important user information seeking behaviors)
Navigation

- Link destination
  - Be consistent in wording among links
  - Default user model of links takes user to another page in same site
    - Confusion can arise when link takes user to another site (especially one with totally different style)
    - User may find no links back into original site
      - Open new window with these links
  - Avoid dead end pages (use the power of hypermedia)

Format, Content, & Appearance

- Sloppy appearance implies questionable content!
  - Spell check, grammar, inconsistencies
- Don’t simply translate existing printed matter into hypermedia
Format, Content, & Appearance

- Keep background images simple, light, and non-distracting
- Generally make selectable items look selectable
  - Another potential problem with graphical links
- Don’t overly use graphics, animation, and other distractions “just because you can”
  - “Gratuitous graphics” - costly in terms of downloading time, but distracting
    - E.g., blinking something clickable

Format, Content, & Appearance

- Graphic design doesn’t automatically make a site “better”
  - Depends on how well graphics convey information
  - Graphics don’t necessarily make sites more interesting (“sticky”) to users
    - Study: Did not spend more time
  - Often not as important as some think in helping find information on Web
  - Some sites with best user performance have almost no graphics
Format, Content, & Appearance

- Information seeking is different than surfing
  - Cool stuff attracts surfers and correlates with user preferences and satisfaction
  - But animation, movement, blinking, flashing, zooming distracts users and hurts performance
    - People seen covering up animation with hand while trying to read, think

- Content and descriptive links drive information seeking
- Content and links correlate positively with use performance
- Minimize reading, especially to get to content users want
Format, Content, & Appearance

- Maximize readability - fonts, color, contrast, layout, etc.
  - Old standards for books (typography) and GUI screens do not necessarily work on web
  - Primary design requirement: scanning
  - People skim and scan; people don’t read all information

- Scanability: use bullets, less grammatical baggage, remove extra word
- Less reading to access content (which is then read)
- Too much white space is not necessarily good (despite what is true for GUIs and books)
- Web pages have much lower visual bandwidth than books, newspapers
- A little white space helps organization, but . . .
- White space spreads out information and works against scanning
Housekeeping

- Perform thorough usability evaluation of site before each release
  - Technical quality
  - Structure
  - Navigation
  - Readability

Housekeeping

- Before each new release, view site with:
  - Numerous browsers
  - Different platforms
  - Images turned off
  - Different connection speeds
  - Different monitor sizes and resolutions
  - And at time of expected peak usage
Housekeeping

- At least monthly, perform cobweb search: avoid link rot
  - Broken, stale links

- At least monthly, update site
  - Outdated, incorrect information

- Have contact information (e.g., to Webmaster) at least on home page
Usability Problem Analysis

- Formative evaluation – identify critical incidents = raw usability data
- Usability problem analysis:
  - Problem extraction – isolate individual usability problems
  - Identify effects on user (the real usability problem)
  - Diagnosis each usability problem in terms of problem type and causes within interaction design
What the Whole Process Looks Like to the Usability Engineer

Usability testing

Usability inspection

Remote CI Reporting

New version of system

Implement changes

Critical incident descriptions

UP Extraction

Possibly multiple usability problems

Analysis, Reporting

Usability problem reports in database

Design

Effect on user = the problem

Interaction design flaw = the cause

Priority rankings, resolution decisions

C/I Analysis

Redesign solution suggestions

Effect on user = the problem

Interaction design flaw = the cause

New version of system

Effective Analysis – Key to Return on Investment in Usability Process

- Quality usability problem analysis and reporting is key to return on investment in usability testing
  - Currently: Ad hoc usability problem reporting
    - Evaluators write down what they believe salient based on what they notice, what are they thinking at the time
    - Laundry list of raw usability problems
    - Inconsistent quality, vague, incomplete descriptions
Fixing the Real Problems

- Usually can’t fix all the problems
- Usability problems can seem the same on surface but have different underlying causes
- Need good diagnosis to fix the right problems
  - Fixing wrong problem can create more problems
  - Failure to fix right problem: missed opportunity (that you paid for)

Usability Problem Diagnosis Within the UAF

- Gives precision and completeness
- Means:
  - Determining the interaction design flaw that caused the problem with the user
  - Locating it in the space of interaction design issues within the User Action Framework
  - Putting usability problem in context of applicable usability principles and guidelines
  - Specificity of problem description often yields self-suggesting solution (e.g., label font too small)
UAF Diagnosis Path is Diagnosis Encoding

- Each decision point (UAF node) in structure is about a different usability attribute
- Decision path represents set of usability attributes (coordinates within usability problem/design space)
- Decision path is
  - ‘Encoding’ of problem description
  - Basis for diagnosis

Learning Problem Diagnosis is Difficult

- It is known that diagnosis is difficult to learn
- VT Vet School had trouble with clinical diagnosis course
  - Determined need for structure and guidance
  - To head off jumping to conclusions without following a logical process of building the case
  - We have same problems with new practitioners diagnosing usability problems
No Jumping to Conclusions

- Jumping to conclusions does not work with diagnosis (using UAF or otherwise)
  - UAF is organized on Interaction Cycle, sequence of different kinds of user actions
  - UAF is not organized as a *taxonomy* of usability concepts
  - Thus, high-level concepts (e.g., consistency) are *distributed*
  - Thus, there might be more than one node that appears to match
  - Diagnosis must match an entire classification path

Not About User Coping

- Key point for usability problem diagnosis
  - It's about how design fails to support user, not about how user fails to cope with the design
  - Example, if a user tries to do something the system can’t do, it is not poor planning on the part of the user for planning to use a feature that didn’t exist
Usability Problem Analysis Exercises

- Next, some in-class exercises to diagnose usability problems
Human-to-Human Communication as Model for Human-Computer Interaction

A model for human-computer interaction?

Important scope limitation: problem solving

Why look at human-human interface?
  - The human-computer interface is really a human-human interface because a computer’s intellectual substance is wholly derived from people!
Human-to-Human Communication

- Computer messages are messages from human designers to human users:
  - Offset in time
  - Sent through a "mask"
    - Lowers inhibitions
    - Can bring out hostility or misdirected sense of humor
    - A phenomenon observed by sociologists in actors

Chapanis did study of nature of human-human communication for problem solving
- Source and seeker of information
- Neither can solve problem alone
- Must work together
- Problem: assemble a trash toter
Human-to-Human Communication

- Communication channels
  - Voice
  - Hand-writing
  - Typing
  - Video
- Communication mode – combination of channels
- The experimental setup

Excerpt from typed communication

Excerpt from voice communication
Human-to-Human Communication

Results

- Order of efficiency (time to solve)
  - Communication-rich mode (all channels)
  - Voice (but cannot see each other)
  - Hand-writing (including drawing sketches)
  - Typing (experienced)
  - Typing (inexperienced)

- Time to complete
  - From less than 30 minutes (communication-rich mode) to over 60 minutes (factor of 2.5)

Results

- Unexpected aspects
  - Small difference between experienced and inexperienced typists
    - Only about 1/3 of time on average used in typing
    - Different kind of typing (cf. business)
  - Small difference between communication-rich and voice only
    - In spite of all the information-bearing gestures
  - Voice was most important channel
    - All modes with voice did substantially better
Human-to-Human Communication

- Observations
  - Communication was very ungrammatical
    - A big barrier to voice I/O in HCI
    - Almost unintelligible
    - Expected, but surprising in extent that it was true
    - Communication success is tribute to human ability

- Voice channels are very verbose
  - Higher bandwidth got used, but not efficiently
  - Allowing interruption
    - Did not shorten solution time
    - Does cause different message packaging
Human-to-Human Communication

- In-class exercise
  - Everyone: as you observe, write down important characteristics of human-human-communication for problem-solving
  - Are these not also important in human-computer interaction?
Wrap up

Topics

- Review
- Selling these new techniques
- Cost justification
- Getting started
- Parting words
Wrap Up

- We have addressed how to design the content of quality user interaction, and presented a process by which usability can be ensured in user interaction.
- You have seen how to
  - Apply interaction design guidelines
  - Use an iterative, evaluation-centered usability engineering life cycle for user interaction development
  - Participate in systems analysis, including user, needs, task, and functional analyses

What you can now do:
  - Perform conceptual and detailed user interaction design
  - Establish usability specifications
  - Build rapid prototypes
  - Perform formative usability evaluation
  - Iteratively refine an interaction design
  - Know how to get started with these new ideas
Your biggest challenge may be:
- Not technical!
- Selling this to management

By necessity, the usability engineering process has changed from linear to iterative, which in turn changes at least:
- Control
- Scheduling
- Organizational roles
- Territoriality
- Project management
- Communication and skills
- Test facilities and tools

What we’ve presented is the basis for controllability, accountability, and quantitative methods that are so important (rightfully) to management.
Wrap Up

- Selling these techniques to management
  - They may not be aware that there is a problem
  - They may view these techniques as a solution to a non-existent problem
    - “The product is selling well”
    - “We’re getting lots of interested inquiries”
    - “Users don’t complain”
  - “We’ve never done it this way before”
  - Seek out corporate mission statement and show how usability engineering supports
  - Remember the “personware” factor

Successful interaction designs are being developed using these techniques, because they’ve been shown to work!

- Resources needed: Minimum of 15% of entire development effort!
- “You have to keep running just to stay in the same place!”
Cost Justification of Usability

Can we afford to include usability engineering in our system development process?

- Answer: Usability engineering does not add overall cost, for two reasons
  - Costs are added only to a limited part of total development process
  - Usability saves on many other costs

Cost Justification of Usability

- Added costs are confined
  - Reality: Interaction development process must be iterative—cannot get it right the first time
  - But interaction development is small part of overall system development
  - Rest of development – user interface software and other application software – is not necessarily iterative
Poor usability is costly; good usability is all about saving costs
- Costs of hardware and software vs. costs of “personware”
- Usability is about good business, not about “being nice”
- Costs of development vs. cost of operation
  - Development costs are mostly one time; operational costs accrue for years
  - Cost/benefit scope must be broad enough to include usage, as well as training, help desk support, etc.

Scope issue: one group pays for development cost and another group gets benefits.

- Development savings from usability in process
  - High software maintenance costs – trying to get it right after release
  - “Pay me now or pay me more later”
  - Implementation Costs

- Usage savings; even more significant if users are your employees!
Cost Justification of Usability

- Usage savings can include costs of:
  - Operational productivity
  - User training and on-line help
  - User errors
  - Database corruption
  - Help desk and user support operations
  - Employee dissatisfaction
- **Point:** Not more resources to ensure usability, but different resources with different distribution during life cycle

Cost Justification of Usability

- Beyond cost savings: In the e-commerce world of the Web, good usability can mean increased revenue!
  - Can market your organization as having a focus on usability; competitive edge
  - Huge need to avoid releasing something that will embarrass you and the organization (despite the pressure of “Internet time”)
Cost Justification: A Simple Example

- For a large distributed system:
  - Users: 75,000
  - Average transactions/user a day for one task type: 20
  - Transactions/day: 1,500,000
  - User time per transaction: 5-20 minutes
  - Average time saved per transaction, due to improved usability: 30 seconds
  - Average fully-loaded hourly rate: $25.00

Saved per year

\[
\text{Saved per year} = 75,000 \text{ users} \times 20 \text{ trans/user-day} \times 0.5 \text{ min/transaction} \times 230 \text{ days/yr} \times \frac{25 \text{ dollars}}{\text{hr}} \times \frac{1 \text{ hr}}{60 \text{ min}}
\]

\[
= 71,875,000.00
\]

- Other savings: user training, help desk
- Regardless of what usability engineering cost for this product, payback is enormous
Cost Justification of Usability

- But won’t it be nice when we no longer have to justify “costs” of usability?
- When have you heard anyone ask: Can we afford costs of designing data structures, implementing algorithms, doing quality assurance, etc……….?!

Getting Started

- Some ideas for selling these techniques to management:
  - **Start small**
    - Try the process on a small part of a project
    - Try a few usability specifications
    - Set up a small usability lab somewhere, anywhere – and use it
    - Develop at least a minimal customized style guide
Getting Started

- Tell management exactly what you intend to try and hope to accomplish, and within what time frame
- Expect some rough spots in initial stages
- Get appropriate resources lined up
  - Get buy-in from management

Getting Started

- Get at least one person with appropriate skills on the user interface development team, and give them a title, responsibility, and authority
- Give appropriate training to team members
- Get commitment from team members to try new these new techniques
- Find someone you can apprentice with
- Get consulting help when needed, especially during start-up
Getting Started

- **Professional preparation**
  - Subscribe to HCI publications
  - Join Usability Professionals’ Association (UPA)
  - Join Special Interest Group on CHI (SIGCHI) – local and/or national
  - Go to appropriate conferences – e.g., Computer-Human Interaction (CHI); Human Factors and Ergonomics Society (HFES); User Interface Software and Technology (UIST); National Institutes of Standards & Technology (NIST)
  - Start a “brown bag” user interface lunch bunch

Getting Started

- Try the process all the way through once
- Generate a failure story
- Better: Generate a success story
  - E.g., Video-clips for “before and after”
Parting Words

- Encourage focus on the process, rather than just the product
  - Make customized process guide
  - Operationalize the process organization-wide
- Ensure usability “by practice” rather than “by decree” or “by politics”
- Characteristics needed by user interaction/interface developers:
  - Dedication: to the cause of quality interfaces
  - Daring: to do things differently

Congratulations, you made it!!!!