The User Interface Is Important

- The user interface is the means by which the software presents itself to the world.
- User interface strongly affects perception of software:
  - Usable software sells better
  - Unusable web sites are abandoned
- Perception is sometimes superficial:
  - Users blame themselves for UI failings
- User interfaces are hard to design:
  - Most SW engineer is not the user
  - The user is always right
    - Consistent user’s faults are the system’s problem
- User interface accounts for ~50% of the SE life cycle.
Usability Defined

- Usability: how well users can use the system’s functionality

Dimensions of usability
- Learnability: is it easy to learn?
- Efficiency: once learned, is it fast to use?
- Memorability: is it easy to remember what you learned?
- Errors: are errors few and recoverable?
- Satisfaction: is it enjoyable to use?

Usability depends on the user
- Novice users need learnability
- Infrequent users need memorability
- Experts need efficiency
Usability Is Only One Attribute of a System

Software designers have a lot to worry about:

- Functionality
- **Usability**
- Performance
- Size
- Cost
- Reliability
- Security
- Standards

Many design decisions involve tradeoffs among different attributes.
Usability Engineering
- Iterative Design

The current best-practice process for developing user interfaces
Iterative Design of User Interfaces

**Design**
- Focus on users and tasks
  - user analysis: who the users are
  - task analysis: what they need to do

**Implement**
- Prototyping

**Constant evaluation**
- Users are involved in every iteration
- Every prototype is evaluated somehow
Why Waterfall Model is Bad for UI Design

- UI design is hard
- Users are not involved in validation until acceptance testing
- UI flaws often cause changes in requirements and design
Iterative design may release too many version and customers do not want to pay the evaluation costs.
Design in Spiral Model

1. Task analysis: collecting the requirements for the UI
2. Design sketches: paper sketches of UI design
3. Paper prototype: an interactive prototype
4. In-class user testing
5. Computer prototype: an interactive software prototype
6. Heuristic evaluation: usability exports evaluation
7. Implementation: the real implementation
8. User testing: users test & refine
Usability Guidelines

Fitts’s Law

- A fundamental law of the human sensory-motor system
- The time for pointing a target from the current position
  \[ T = a + b \log(2D/S) \]
  - \( a \): reaction time
  - \( b \): index of difficulty
  - Target size \( S \) at distance \( D \) away
- Size and proximity of controls should relate to their importance
- Tiny controls are hard to hit
- Targets at screen edge are easy to hit
  - Provides infinite size
Usability Guidelines (cont’)

- Working memory
  - Small capacity: golden number 7
  - Fast decay: 7 sec.

- Color Blindness guidelines
  - Don’t depend solely on color distinctions (color blindness)
  - Avoid red on blue text (chromatic aberration)
  - Avoid small blue details
Direct manipulation Cues

Norman’s principles of direct manipulation

- Affordances: perceived and actual properties of a thing that determine how the thing could be used
  - A button is for pushing
  - Chair is for sitting
- Natural mapping: physical arrangement of controls should match arrangement function
  - The orientation of turn switches
- Visibility: relevant parts of a system should be visible
  - show important control menus
- Feedback: action should be immediate, visible effects
  - Push buttons
  - Scrollbars
  - Drag & drop
Usability Guidelines (“Heuristics”)

Plenty to choose from

- Nielsen’s 10 principles
  - One version in his book
  - A more recent version on his website
- Tognazzini’s 16 principles
- Norman’s rules from Design of Everyday Things
- Mac, Windows, Gnome, KDE guidelines
  - Platform-specific guidelines give consistency but limited

Heuristics can be used for

- Help designers choose design alternatives
- Help evaluators find problems in interfaces
Nielsen’s 10 principles

1. Match the Real World

- Use common words, not techie jargon
  - But use domain-specific terms where appropriate
  - What is the meaning of “Type mismatches”
- Don’t put limits on user defined names
- Allow aliases/synonyms in command languages
- Metaphors are useful but may mislead
  - Word processor vs. typewriter
Nielsen’s 10 principles

2. Consistency and Standards

- Principle of Least Surprise
  - Similar things should look and act similar
  - Different things should look different

- Other properties
  - Size, location, color, wording, ordering, …

- Command/argument order
  - Noun-verb order: first select the command then invoke the command
  - Verb-noun order: first invoke the command then select command

- Follow platform standards
Nielsen’s 10 principles

3. Help and Documentation

Users don’t read manuals
- Prefer to spend time working toward their task goals, not learning about your system

But manuals and online help are vital
- Usually when user is frustrated or in crisis

Help should be:
- Searchable
- Context-sensitive
- Task-oriented
- Concrete
- Short
Nielsen’s 10 principles

4. User Control and Freedom

- Provide undo
- Long operations should be cancelable
- All dialogs should have a cancel button
Nielsen’s 10 principles

5. Visibility of System Status

- Keep user informed of system state
  - Cursor change: hand over a hyperlink, drag/drop
  - Selection highlight
  - Status bar
  - Don’t overdo it…

- Response time
  - < 0.1 s: seems instantaneous
  - 0.1-1 s: user notices, but no feedback needed
  - 1-5 s: display busy cursor
  - > 1-5 s: display progress bar
Nielsen’s 10 principles

6. Flexibility and Efficiency

- Provide easily-learned shortcuts for frequent operations
  - Keyboard accelerators
  - Command abbreviations
  - Styles
  - Bookmarks
  - History

Good for repeated operations
Nielsen’s 10 principles
7. Error Prevention

- Selection is less error-prone than typing
  - But don’t go overboard…

An example of overboard

- Disable illegal commands
Nielsen’s 10 principles
7. Error Prevention (cont’)

- **Description Error:** Intended action is replaced by another action with many features in common
  - Pouring orange juice into your cereal
  - Do not locate similar action buttons at the same position
- **Capture Error:** A sequence of actions is replaced by another sequence that starts the same way
- **Modes Error:** states in which actions have different meanings
  - E.g., Caps lock
Nielsen’s 10 principles

8. Recognition, Not Recall

- Minimize the user’s memory load
- Use menus, not command languages
- Use combo boxes, not textboxes
- Use generic commands where possible (Open, Save, Copy, Paste)
- All needed information should be visible
- “selection is better than typing”
Nielsen’s 10 principles
9. Error Reporting, Diagnosis, Recovery

- Be precise=exact; restate user’s input
  - Not “Cannot open file”,
    but “Cannot open file named paper.doc”

- Give constructive help
  - why error occurred and how to fix it

- Be polite and non blaming
  - Not “fatal error”, not “illegal”

- Hide technical details (stack trace) until requested
Nielsen’s 10 principles

10. Aesthetic and Minimalist Design

“Less is More”

- Omit extraneous information, graphics, features
Nielsen’s 10 principles

10. Aesthetic and Minimalist Design (cont’)

Good graphic design

- Few, well-chosen colors and fonts
- Group with white space
- Align controls sensibly

Use concise language

- Choose labels carefully
Nielsen’s Heuristics

- **Meet expectations**
  1. Match the real world
  2. Consistency & standards
  3. Help & documentation

- **User is boss**
  4. User control & freedom
  5. Visibility of system status
  6. Flexibility & efficiency

- **Errors**
  7. Error prevention
  8. Recognition, not recall
  9. Error reporting, diagnosis, and recovery

- **Keep it simple**
  10. Aesthetic & minimalist design
How To Do Heuristic Evaluation

- One application of Nielson’s 10 heuristic
- Justify every problem with a heuristic
  - “Too many choices on the home page”
  - Can’t just say “I don’t like the colors”
- List every problem
  - Even if an interface element has multiple problems
- Go through the interface at least twice
  - Once to get the feel of the system
  - Again to focus on particular interface elements
- Don’t limit yourself to the 10 heuristics
  - We’ve seen others: affordances, visibility, Fitts’s law, perceptual fusion, color principles
  - But the 10 heuristics are easier to compare against
1. Training
   - Meeting for design team & evaluator
   - Introduce application
   - Explain user population, domain, scenarios

2. Evaluation
   - Evaluators work separately
   - Generate written report, or oral recorded by an observer
   - Focus on generating problems, not on ranking their severity yet
   - 1-2 hours per evaluator
3. Severity Rating
   - Evaluators prioritize all problems found (not just their own)
   - Take the mean of the evaluators’ rating

4. Debriefing
   - Evaluators & design team discuss results, brainstorm solutions
Kinds of User Tests

Formative evaluation
- Find problems for next iteration of design
- Evaluates prototype or implementation, in lab, on chosen tasks
- Qualitative observation (usability)

Field study
- Find problems in context
- Evaluates working implementation, in real context, on real tasks
- Mostly qualitative observations

Controlled experiment
- Test a hypothesis (interface A is faster than interface B)
- Evaluates working implementation, in controlled lab environment, on chosen tasks
- Mostly quantitative observation (time, error rate, satisfaction)
Recording Observations

- Pen & paper notes
- Audio recording
- Video recording
- Screen capture & event logging
How Many Users?

How many users do you need for **formative evaluation**?

Landauer-Niesen model

- Every tester finds a fraction $L$ of usability problems (typical $L=31\%$)
- $n$ users will find a fraction $1-(1-L)^n$ so 5 users will find 85% of problems
- *But $L$ may be much smaller and variant."

Which is better?

- 15 users to find 99% of problems with one design
- 5 users to find 85% of problems with each of three design iterations, this is better

For multiple use classes, get 3-5 users from each class
Controlled Experiment

- Start with a testable **hypothesis**
  - Interface A is faster than interface B
- Manipulate **independent variables**
  - different interfaces, user classes, tasks
- Measure **dependent variables**
  - times, errors, satisfaction
- Use statistical techniques to accept or reject the hypothesis
Controlled Experiment Example

- Fitts’s law for menu bar
  1. hypothesis: Mac menu bar is faster than Windows menu bar
  2. Independent: position of menu bar (whatever the height of the title bar is)
  3. dependent: time to reach menu bar (how long the user to move the mouse up to the menu bar and click on a particular target menu)
Concerns Driving Experiment Design

- **Internal validity**
  - Are observed results actually caused by the independent variables?

- **External validity**
  - Can observed results be generalized to the world outside the lab?

- **Reliability**
  - Will consistent results be obtained by repeating the experiment?
Threats to internal validity

Ordering effects
- First people learn, second people get tired or bored.
- Don’t present tasks or interfaces in same order for all users.
- Solution: Randomize or counterbalance the ordering

Selection effects
- Randomly assign users to independent variables
- Unknown variables are kept the constant states

Experimental bias
Threats to external validity

- **Population**
  - Draw a random sample from your real target population

- **Ecological validity**
  - Make lab conditions as realistic as possible in important respects

- **Training validity**
  - Training should mimic how real interface would be encountered and learned

- **Task validity**
  - Base your task on task analysis
Threats to Reliability

- **Uncontrolled variation**
  - Previous experience: novices and experts
  - User differences: fastest users are 10 times faster
  - Task design: population’s problems (questions, distractions, coughing)
  - Measurement error

- **Solution**
  - Eliminate uncontrolled variation
  - Repetition: many users $n$ many trials
**Between Subjects**

- Users are divided into two groups:
  - One sees only interface A
  - Other sees only interface B

- Results are compared between different groups:
  - Is \( \text{mean}(A) > \text{mean}(B) \) ?

- No variation due to ordering effects:
  - User only uses one interface
Within Subjects

- Each user sees both interface A and B (in random order)
- Results are compared within each user
  - For user $i$, compute the difference $a_i - b_i$
  - Is mean($a_i - b_i$) > 0?
- Eliminates variation due to user difference, but may have reliability problem due to ordering effects
Usability Engineering

Design
(task analysis)

Evaluate
(heuristic evaluation)

Implement
(prototyping)

Release