Human–Computer Interactions

This slide set contains several examples of good interaction design and bad interaction design. The slide title of these examples indicates the main point of the example, and if it is a bad design the issues and how they might be addressed are included as bulleted points on the slide.

These slides also contain definitions and some guidelines. Please pay particular attention to the definitions of user interactions, usability versus user experience goals, and the iterative process of interaction design.

The in-class activity will be to take a user environment design diagram, create a paper prototype for a portion of it, and evaluate this prototype. A description of this diagram, how it can be created from Use Cases, and how it can be used to design a User Interaction is included in these slides.

The next slide is an example of an unfortunate interaction – think about how you might re-design it to make a better experience for the user.
An ATM Encounter

The goal: to withdraw some cash from one account and deposit it to another. How hard can it be? Isn't ATM user interface design an established, known thing? Apparently not, as I found out...

I walked up (with a helper friend) to a cashpoint (the "hole in the wall") which had a big colour screen with buttons [on] either side of it and more buttons in a keypad below. The screen rotated through some adverts and included no instructions as to what [to] do: I inserted my card to make a withdrawal. Nothing happened. We both stared at the screen and start pressing some buttons – nothing. Eventually she went in to the bank to ask for instructions while I continued tapping on the screen and buttons for a little longer. I got bored so withdrew my card... and after a couple of seconds, guess what? It asks me for the PIN number! Wow! You have to pull the card out in order for it to operate... good thing I didn't walk away immediately as the next person could have come along and played "guess the PIN" without ever getting hold of my card! Anyway, in the end the withdrawal succeeded and I had $500 in $20 denominations which is what it gave out. Next step: deposit that cash into another account.

I insert the other card, nothing happens but now I know the trick so I pull the card out and I get prompted for the PIN (after a brief delay). It asks how much I want to deposit. I enter the amount of $500. It dispenses an envelope and instructs me to insert the cash in the envelope and stick it back in the slot; it states that the envelope must not contain more than 10 banknotes! How am I to deposit $500 that it just gave me in 20s by using only 10 notes?! If that is its limitation why didn't it state that before asking me how much I wanted to deposit? This is yet another example of getting the steps in the wrong order. Anyway, since I couldn't find a cancel button (or some other Ctrl+Z option) I put $200 in the envelope and stuffed it in the slot. It gave me a receipt thanking me for depositing $500 :-) By this stage, the person working at the bank comes out and after hearing the story says: "Yeah that happens all the time, I'll correct it on the computer". Sigh.
Don't Redefine Well-Known Concepts

http://www.componenthouse.com/article-22
Confusing

Issues:
• Simplicity is best – the fix: **Cancel E-mail Subscription**
• Incomplete instruction label – the fix: **Enter e-mail address to remove**
• Already have e-mail
• 1\(^{st}\) button unnecessary, awkward
• 2\(^{nd}\) button label cumbersome – the fix: **OK**
What is Interaction Design?

Designing interactive systems that help people interact and communicate with the system and with other people as they perform an activity or a job.

Key things to keep in mind:
• Who will be using the system?
• How is it going to be used?
• Where is it going to be used?
• What kind of activities are people doing when they interact with it?
Recall Iterative Development:

Small complete development cycles that extend and deliver functional systems.
Interaction design activities:

Requirements:
- Usability & User experience goals
- Model!

Identify needs:
- goals
- models
- scenarios

Develop designs:
- low fidelity
- storyboards

Design:
- paper prototypes
- wireframe diagrams

Develop prototypes

Implementation:
- interactive versions

Evaluate what is built and its associated user experience; there will always be a gap – find gaps early so they can be more easily fixed.
Visual Elements

Whenever your local SMS Administrator sends you an actual software Package, the SMS Package Command Manager will appear (usually at network logon time) displaying the available Package(s). The following screenshots display scenes similar to what you will see when you receive an actual SMS Package.

To start the demonstration, click the "CLICK HERE TO ACCESS" button of the screen.

Issues:
• (above) font, color, background make text almost completely unreadable
• (right) dockable menu in Office ‘97

http://homepage.mac.com/bradster/iarchitect/
Design Principles

- **Visibility**
  - make functions/controls visible

- **Feedback**
  - action taken & results

- **Constraints**
  - restrict actions depending on situation, e.g., grayed out menu items

- **Consistency**
  - similar operations, similar interface elements, e.g. always left mouse click (input operation) to select objects

- **Affordances**
  - an attribute of an object tells people how to use it, e.g. push a button, turn a knob

ATM Example

Two equivalent interfaces:
(a), (c) Japanese
(b), (d) Israeli

(a), (b) rated more usable because:

1-9 buttons on keypad go from top to bottom, the amount is more visible on the top of the screen, symmetric keypad layout is more usable

http://www.sigchi.org/chi97/proceedings/paper/nt.htm
Visual Elements

Issues:
• arrange options vertically (easy scanning)
• fields need to be long enough for info/input
• left-align labels in a group
• use significant letters for mnemonics (‘Wait’)

Use position, alignment, and grouping to provide information flow. This means the brain doesn’t have to apply order in addition to finding something.
Usability: How well interactions are optimized to help people do their work

- Framed as **measurable** questions.
- How useful/productive a system is from its own perspective.
- **Goals:**
  - Effectiveness – how well does the system do what it is supposed to do?
  - Efficiency – how are users supported in doing their tasks?
  - Safety – protect from dangerous/undesirable situations.
  - Utility – is the right functionality provided?
  - Learnability – how easy is the system to learn to use?
  - Memorability – how easy is it to remember how to do things once you’ve learned them?

User Experience: Subjective qualities

- Positive or negative qualities.
- From the user’s point of view.

Qualities:

- satisfying
- enjoyable
- engaging
- exciting
- challenging
- boring
- cutesy
- frustrating
- pleasureable
- entertaining
- helpful
- motivating
- aesthetically pleasing
- supporting creativity
- cognitively stimulating
- rewarding
- fun
- provocative
- surprising
- emotionally fulfilling
- enhancing sociability
- annoying

Tabs

Issues:
- number of tabs
- icons as labels
- icon colors
- nested tabs

http://homepage.mac.com/bradster/iarchitect/
Unnoticeable

Issues:
• easy to misread
• easy to miss severity

Alternative: critical message box (has X in it): not successful

http://www.buigallery.com/search/label/Confusing
Review Your Icons

http://www.componenthouse.com/article-21
<table>
<thead>
<tr>
<th>Technique</th>
<th>Purpose</th>
<th>Data</th>
<th>Advantages</th>
<th>Disadvantages</th>
</tr>
</thead>
</table>
| Interviews                 | Exploring                   | Mostly qualitative, some quantitative     | • Can guide interviewee  
• Encourages interaction between developers, users | • Can take a long time  
• Can intimidate interviewee |
| Focus groups               | Collect multiple viewpoints  | Some quantitative, mostly qualitative      | • Finds consensus, conflict  
• Encourages interaction | • Dominant personalities can be a problem |
| Questionnaires             | Specific question answers   | Quantitative and qualitative              | • Many people  
• Doesn’t take many resources | • Design crucial  
• Response can be low |
| Direct observation in field| Understand context          | Mostly qualitative                        | • Can find insights not otherwise possible | • Very time consuming  
• Lots of data |

17
Unrelated information shouldn’t be displayed together
Questionnaires

- Ordering of questions can influence response
- May need different versions for different user types/populations
- Provide instructions!
- Balance length with white space (crowding issues)

Response types:
- Check boxes (ranges)
- Likert scales
  - Measure opinions, attitudes, beliefs
  - Short statements: e.g. “home page instructions are clear”
  - Decide on scale: e.g. “strongly agree, agree, don’t know, disagree, strongly disagree”
- Semantic differential scales
  - Explore range of bi-polar attitudes: “clear ... OK ... confusing”
Remove Conflicting Ideas

Erlebnisweg is the name of a forest or nature trail

http://www.componenthouse.com/article-21
How do you feel about the MS Paperclip?
- Qualitative information: helpful, annoying, ...
  - Answers not homogeneous, must be treated individually
  - Good starting point for questions on future evaluations

In your opinion, is the MS Paperclip amusing, irritating, or neither?
- Quantitative: n out of m respondents find the paperclip amusing, ...

In your opinion, the MS Paperclip is amusing:
  Likert scale:
  - [ ] strongly agree
  - [ ] agree
  - [ ] neither
  - [ ] disagree
  - [ ] strongly disagree

- Quantitative: n out of m respondents agreed with the statement that the paperclip is amusing
Accessibility should work in practice

http://www.componenthouse.com/article-21
Goal: **Keep work coherent** to support work and fit users’ expectations; enable transformation to work practices in new system.

- Keep work orderly and natural
  - Problem: PowerPoint – can’t change slide notes and slide content at the same time – have to switch between areas. **Result: user work is disrupted.**
  - Problem: Email system – search address book via separate query window but search sent mail via text field – inconsistent structures for similar work. **Result: user work is disrupted.**
  - OK: Multiple tasks – see all emails for scanning purposes, then click to view one in depth; structurally supports a change in intent: from scanning to reading. **Result: user work flows naturally.**
Metaphors: “Stop Light”

Issues:
• stoplight labels/corresponding tab labels
• distance between stoplight and tab
• too much detail
System Work Model to Structure

- Use case steps capture a single task, coherently.
- A full system has to support all the use cases in a coherent way, and this coherence starts with a system work model that takes all the system sequential use cases into account, and provides a system structure to support this model.
- Move from sequential thinking about the system work model to structural thinking – what parts or places in the system are needed to support the sequences of tasks?
- The email example:
  - one place to scan messages
  - another place to read a message in full
- The intents of the tasks (*scan* versus *read*) imply structure – a part or place in the system that supports scanning and other that supports reading.
- Keep going back and forth between details of the system work model and overall system structure to keep the system coherent.

A complete system work model design means that each part of the system structure is NOT self-contained from the perspective of the user interaction.

Example: In one module of a system an engineer includes a way to scan and search without the user having to enter text. It is guaranteed that users will expect this in any other place they can scan/search.

User interactions cannot be consistent unless they are understood in the context of the entire system work model.

Creativity saves the day

http://www.componenthouse.com/article-21
Make up the structure of the system.
Each place has functions that allow the work in the place to occur.
Each place has links to move from place to place.

Example: Mail system

**Place:** scan all emails

**Functions:** sort emails by recipient, sort emails by date received, delete emails, move emails to folders, etc.

**Links:** to reading one email by itself, to creating a new email, etc.

System Structure as Places

- Work goes on in a *place*, it moves to a new location with a new *intent* and continues on till finished. This is similar to a floor plan of a house – each room has a different intent, and you can do things in it, and you can move to different rooms when you want to do something different.

- Represent the System Work Model as **places** with **functions** and **links** to other places in a User Environment Design. The functions and links in places can be mapped to user interaction/user interface elements. The user environment design provides a validation point for the user interaction.

Use the right punctuation or appropriate separators

http://www.componenthouse.com/article-21
Undergraduate Advisor Support System User Environment Design diagram

### Find plan
Purpose: Find a student’s approved plan, a persisted plan that is not approved, a working plan created during this session, or a plan that has been recommended to the advisor.

- **Functions:**
  - If user is an advisor, then find plans associated with a particular student, otherwise only find plans associated with the user
  - View plan
    - Entire plan or any combination of:
      - Classes taken to date
      - Classes planned for entire future
      - Classes planned for next semester
      - GPA
      - Estimated graduation
      - Applicable rule violations

- **Links:**
  - Recommend plan
  - View rules
  - Home

### Manage plan - Advisor
Purpose: Handle proposed plans and deal with progress notifications.

- **Functions:**
  - Receive progress notification
    - Add notes to related plan
    - Request meeting with student
  - Receive plan proposed for approval
    - View plan
    - View changes from existing plan
    - Add notes
    - Request meeting with student
    - Approve or deny plan
    - Defer plan decision
  - View history of prior approved plans

- **Links:**
  - External: email system
  - Edit plan
  - Print plan
  - View rules
  - Home

### Edit plan
Purpose: Create a new plan or change an existing plan.

- **Functions:**
  - Choose starting plan
    - New plan
    - Current approved plan
    - Persisted working plan
    - Working plan created during this session
  - Set major/minor
    - Choose major
    - Change major
    - Choose a 2nd major
    - Change minor
    - Add a minor
    - Delete a minor
  - View choices
    - Propose to satisfy the most requirements
    - Select choices
  - Show changes with respect to chosen approved or working or persisted plan
  - Optimize for earliest graduation
  - View plan

- **Links:**
  - Persist plan
  - View requirements
  - View rules
  - Print plan
  - Home

### Recommend plan
Purpose: Forward plan to advisor for approval.

- **Functions:**
  - Forward plan
    - Choose advisor

- **Links:**
  - Home

### View course info
Purpose: View complete information about courses.

- **Functions:**
  - Choose department
    - View all courses in department
  - Choose course
    - View all info about course
    - View description
    - View credit hours
    - View pre-, co-requisites
    - View general offerings (e.g. spring, fall, summer, all)
    - View detailed offerings for next semester

- **Links:**
  - External: University General Catalog
  - Home

### Persist plan
Purpose: Save a plan for up to a month.

- **Functions:**
  - Select to save plan
    - Name plan being saved

- **Links:**
  - Find plan
  - Edit plan
  - Home

### Home
Purpose: Access Undergrad Advisor Support System

- **Functions:**
  - Login/Logout

- **Links:**
  - Find plan

---

**NOTE:** Double line arrow means work done in the area pointed TO may be needed by work done in the pointing FROM area, and switching back and forth can occur.
UIs should be easy to read