Measuring usability

Michelle Mazurek
500 OOPS: child died
What is usable?

- Intuitive / obvious
- Efficient
- Learnable
- Memorable
- Few errors
- Not annoying
- Status transparent

Image from http://www.xkcd.com
Difficulties

- Many systems and platforms
- Users are different from one another
- Required standards (or no standards)
- Documentation won’t necessarily be read
- Performance
- Legal / time pressures
- Social and external factors
Expert evaluation

• “Discount Evaluation”
• Evaluate BY INSPECTION
• Cognitive walkthrough
  – Lewis, Polton, Wharton, Rieman, 1992
• Heuristic evaluation
  – Nielsen+Molich, 1990 (final list in 1994)
Cognitive walkthrough

• Goal: Imagine thoughts and actions of new user
  – And perhaps more sophisticated user
• First: task analysis

• Then, per task
  – Walk through all required actions
  – “Believable story” for their action/motives
• Start from *CORRECT* list of actions
  – Can user find correct sequence? Not how do you stumble through to a reasonable answer
Cognitive walkthrough

• Do it yourself, or employ an “expert”
• Developing vs. validating
• Some questions:
  – Will user try to achieve current effect / do they know this is part of the overall task?
  – Will user notice correct action is available? (Visible)
  – Will user notice action is the right one? (Is it correct?)
  – Will user understand resulting feedback?
Heuristic evaluation

- *not* task based (pros and cons)
- Start from principles/heuristics
- 2+ passes over all facets of system
- Each evaluator independently identifies problems
  - Then combine
  - Nielsen: 5 evaluators, 75% of problems
Nielsen heuristics

- **Visibility of system status**: The system should always keep users informed about what is going on, through appropriate feedback within reasonable time.

- **Match between system and the real world**: The system should speak the user’s language, with words, phrases and concepts familiar to the user, rather than system-oriented terms. Follow real-world conventions, making information appear in a natural and logical order.

- **User control and freedom**: Users often choose system functions by mistake and will need a clearly marked "emergency exit" to leave the unwanted state without having to go through an extended dialogue. Support undo and redo.

- **Consistency and standards**: Users should not have to wonder whether different words, situations, or actions mean the same thing. Follow platform conventions.

- **Error prevention**: Even better than good error messages is a careful design which prevents a problem from occurring in the first place. Either eliminate error-prone conditions or check for them and present users with a confirmation option before they commit to the action.

- **Recognition rather than recall**: Minimize the user’s memory load by making objects, actions, and options visible. The user should not have to remember information from one part of the dialogue to another. Instructions for use of the system should be visible or easily retrievable whenever appropriate.

- **Flexibility and efficiency of use**: Accelerators—unseen by the novice user—may often speed up the interaction for the expert user such that the system can cater to both inexperienced and experienced users. Allow users to tailor frequent actions.

- **Aesthetic and minimalist design**: Dialogues should not contain information which is irrelevant or rarely needed. Every extra unit of information in a dialogue competes with the relevant units of information and diminishes their relative visibility.

- **Help users recognize, diagnose, and recover from errors**: Error messages should be expressed in plain language (no codes), precisely indicate the problem, and constructively suggest a solution.

- **Help and documentation**: Even though it is better if the system can be used without documentation, it may be necessary to provide help and documentation. Any such information should be easy to search, focused on the user’s task, list concrete steps to be carried out, and not be too large.
Try it! Cog. Walkthrough: IRBNet

• Create and submit a project
• (Don’t really submit it please)
  – Make correct action list
  – Do walkthrough
Usability testing

• In the lab
• A/B testing in the field
Usability of Fruit

- https://www.youtube.com/watch?v=3Qg80qTfzgU
Lab testing

• Mostly covered by experiments, self-reports earlier

• Tricky: task design
  – Realism (E/V)
  – Not cherry picking for your system

• Measurements:
  – Performance as well as self-report
  – Time, accuracy, keystrokes (deletions), mouse movements, eye tracker, etc. etc.
  – “Think aloud” … easy to say, hard to do
Prototype fidelity

High-fidelity, "Wizard of Oz," low-fidelity
System usability scale

• 10 items, widely used and comparable
  – On 5-point SA to SD
  – Validated
  – Easy to administer

• Doesn’t really suggest what the problem is
  – Not diagnostic
SUS questions

1. I think that I would like to use this system frequently.
2. I found the system unnecessarily complex.
3. I thought the system was easy to use.
4. I think that I would need the support of a technical person to be able to use this system.
5. I found the various functions in this system were well integrated.
6. I thought there was too much inconsistency in this system.
7. I would imagine that most people would learn to use this system very quickly.
8. I found the system very cumbersome to use.
9. I felt very confident using the system.
10. I needed to learn a lot of things before I could get going with this system.
Think-aloud example (in pairs)

• Download and install task management software (todoist and friends)
  – Verify that it is installed

• Things you can ask:
  – What are you thinking now?
  – What do you expect to happen if you do X?
  – How did you decide to do that?
Paper prototype example (in groups)

• Draw a paper prototype to improve IRBNet
  – First step: Identify two tasks that you want to make sure are usable
A/B field testing

• Increasingly popular for large-scale web, email, apps, etc.

• Covered in between subjects experiments
  – Vary at most one item per condition?
  – Always test together