Iterative Design for Usability

Anthony J. Hornof
University of Oregon

Overheads adapted with permission from materials prepared by David E. Kieras, University of Michigan based on J. Gould's article "How to design usable systems"

Introduction

Usability-driven Iterative Design

Some General Properties of Design For Usability

Overall Steps

Iterative Design 3

Usability-driven Iterative Design

Gould's advice on how to design usable systems
An iterative design process driven completely by usability concerns
It works - it has been demonstrated in practice
• E.g., IBM's Olympic Message System
But won't fit into traditional organizations at all!
Requires special product and organizational situation
Management knows usability is paramount for the product
No serious software design and implementation problems
• Technology is off-the-shelf
Usability team is in charge
Management accepts that design process is evolutionary

Just summary here
If actually developing a system, get a copy of Gould
Many specific suggestions on what to do
Included are checklists to help you keep on track

Reference

**Overall Steps**

1. Define the problem that the customer wants solved.
2. Identify tasks users must perform.
3. Learn user capabilities.
4. Learn hardware/software constraints
5. Set specific usability targets in behavioral terms.
6. Sketch out user scenarios
7. Design and build prototype
8. Test Prototype
9. *Iteratively* incorporate changes and test until:
   - Behavioral targets are met
   - A critical deadline is reached
10. Install system at customer location
11. Measure customer reaction and acceptance

**Four Principles of Usable Design**

- **Principle 1. Early and Continual Focus on Users**
  - The choice of user should drive the design
  - Decide who the users will be and what they will be doing
  - Don't put this off, because "flexibility" is actually "slipperiness"
  - **Designers often mess up here**
    - Don't want such a strong drive to their work
    - E.g. interfaces that are so "powerful" only programmers can use them
    - Research enviroments especially bad: e.g. advance expert system methodology instead of help user
    - Implications not carried through - drive not used
    - E.g. for a system for certain executives, designers never talked to or visited executives, although in same building
  - Don't rely on just descriptive data
    - There is no substitute for direct interaction and testing

- **Principle 2. Integrated Design**
  - All aspects of usability in parallel, under one person
    - Interface
    - Documentation
    - On-line help
    - Training
  - Technical problems of usability are solvable
    - Can measure it, control it, thus can manage it
  - Organizational problems are critical
    - Must manage for usability
    - One group must have resources to drive and control usability
    - Must "sign up" to guarantee usability
  - Hard to do
    - Requires outstanding personnel
    - Have to be given responsibility & accountability
    - Must have good tools
Principle 3. Early and Continual User Testing

Can't tell if it is good for users without testing it
Can't get it right the first time
Plan to throw one away
"The two most important tools an architect has are the eraser in the drawing room and the sledge hammer on the construction site" - Frank Lloyd Wright

With testing, changes will result in an increasingly better system
Can test prior to implementation

Principle 4. Iterative Design

Test the design, make changes, and repeat

Requirements:
- Identification of needed changes
- Results of testing compared to usability goals
- Ability to make the changes
- Good tools, appropriate design important
- Willingness to make the changes
- Organizational support

Methods for Early Focus on Users

Contact with users
Contact directly, don't depend on intermediaries
Users are the experts, not their managers
Make contact early, and continue it
Talk with users
- Identify problems, other user groups
Don't expect users to come up with ideas or designs
- But they can comment on yours, properly presented
Participative design
- Good organizationally, access to expertise
- But users aren't necessarily best designers
- An expert consultant on the design team

Observing users
Observe users working
- "Hang out" - be unobtrusive, ask non-threatening questions
Videotape users working
- Samples of problems especially persuasive
Use think-aloud techniques while user is actually performing job
- Doing it in real time better than reflecting later
Surveys, Questionnaires
- Crude, but can be important
- E.g. low education, high-turnover -> fast learning essential
Methods for Early Focus on Users (continued)

Understanding the task
Learn about work organization
• Watch out for system organization mismatching work organization
Try worker’s job yourself
• Allow for differences in expertise both ways
Task analysis
• Valuable, but should not insulate designers from users
• Informal methods often help
• Focus is on exact steps in activities

Testable Usability Goals
Essential so that management can consider usability along with other system properties
What should users be able to do
• How fast?
• How well?
Specify early

Methods for Early and Continual User Testing

Quick feedback in early stages of design
Printed or video scenarios
• What exactly must user do, what exactly appears on the screen
• Engage details immediately
• Start trying these out on users
Early user manuals
• Try out on users - get reactions, questions
• Spot inconsistencies, ambiguities
• DEC experience with workstation
  Corrected design early enough
  User manual became effective design specification
• Other similar experiences - when tried, it works
Mockups, simulations, early & rapid prototyping
• Often worth the trouble in identifying many design problems
• Can do either low-tech, or with programming
  If planned properly, code can be used in the product
• Prototyping toolkits might help
Early demonstrations
• Get reactions, have others try it out
Hallway and storefront methods
• Put prototype in hall or public-like place
• Have people try it out
• Correct quickly to keep comments coming

After prototype available
Collect think-aloud protocols of prototypes
Feedback about prototype via electronic media - e.g. e-mail
• Respond quickly to keep feedback coming
Formal test of prototype
• Many informal methods - help, but weak
• Formal testing gets numbers recognized as valid by managers, software people
Try-to-destroy-it Contests
• Near end of development to find bugs, problems
Videotapes of problems with prototype can be very persuasive
• Managers, software people often don’t believe there are problems
Follow-up
Field studies
• Make up for lab blind spots
Hot-lines, customer support
• Systematic use of customer problems in revisions
Automatic data collection in software

Methods for Iterative Design

Software tools - UIMS, Toolkits
Speed of change is critical to iterative design
Reduce cost of changes, making iteration possible
Control of changes possible by usability team
• Less dependence on programming expertise
Contribute to usability of tools provide good ways of doing things
Help with consistency
UIMS separates application and interface
• But watch out where application might drive interface poorly
Work Organization
Development group must accept need to make changes
Plan to manage change, not ignore the need
Need strong leader, relatively small group