User-Centered Design and Prototyping

Why user-centered design is important
Prototyping and user centered design
Prototyping methods

System Centered Design

What can I easily build on this platform?
What can I create from the available tools?
What do I as a programmer find interesting?

User Centered System Design

Design is based upon a user’s
- abilities and real needs
- context
- work
- tasks
- need for usable and useful product

Golden rule of interface design: Know The User
**User Centered System Design**

... is based on understanding the domain of work or play in which people are engaged and in which they interact with computers...

Assumptions
- The result of a good design is a *satisfied customer*
- The process of design is a *collaboration between designers and customers*. The *design evolves and adapts* to their changing concerns, and the process produces a specification as an important byproduct
- The customer and designer are in *constant communication* during the entire process

*Denning and Dargan, 1996*

**Participatory Design**

**Problem**
- intuitions wrong
- interviews etc not precise
- designer cannot know the user sufficiently well to answer all issues that come up during the design

**Solution**
- designers should have access to representative users
  - END users, not their managers or union reps!

**Participatory Design**

**Users are 1st class members in the design process**
- active collaborators vs passive participants

**Users considered subject matter experts**
- know all about the work context

**Iterative process**
- all design stages subject to revision

**Up side**
- users are excellent at reacting to suggested system designs
  - designs must be concrete and visible
- users bring in important “folk” knowledge of work context
  - knowledge may be otherwise inaccessible to design team
- greater buy-in for the system often results

**Down side**
- hard to get a good pool of end users
  - expensive, reluctance ...
- users are not expert designers
  - don’t expect them to come up with design ideas from scratch
- the user is not always right
  - don’t expect them to know what they want
Methods for involving the user

At the very least, talk to users
– surprising how many designers don’t!

Contextual interviews + site visits
– interview users in their workplace, as they are doing their job
– discover user’s culture, requirements, expectations,…

Methods for involving the user

Explain designs
– describe what you’re going to do
– get input at all design stages
  • all designs subject to revision

Important to have visuals and/or demos
– people react far differently with verbal explanations
– this is why prototypes are critical

UCD principles

User-centered systems design (UCSD) is a process focusing on usability throughout the entire development process and further throughout the system life cycle. It is based on the following key principles.

Source: https://www.it.uu.se/edu/course/homepage/acsd/vt09/Key_principles.pdf

UCD principles (cont.)

User focus – the goals of the activity, the work domain or context of use, the users’ goals, tasks and needs should early guide the development.

Active user involvement – representative users should actively participate, early and continuously throughout the entire development process and throughout the system lifecycle.

Evolutionary systems development – the systems development should be both iterative and incremental.

Simple design representations – the design must be represented in such ways that it can be easily understood by users and all other stakeholders.

Prototyping – early and continuously, prototypes should be used to visualize and evaluate ideas and design solutions in cooperation with the end users.

Evaluate use in context – baseline usability goals and design criteria should control the development.

Explicit and conscious design activities – the development process should contain dedicated design activities.

A professional attitude – the development process should be performed by effective multidisciplinary teams.

Usability champion – usability experts should be involved early and continuously throughout the development lifecycle.

Holistic design – all aspects that influence the future use situation should be developed in parallel.

Processes customization – the user-centered systems design process must be specified, adapted and/or implemented locally in each organization.

A user-centered attitude should always be established.

Source: https://www.it.uu.se/edu/course/homepage/acsd/vt09/Key_principles.pdf
**UCD principles**

![UCD principles Diagram](image)

**Sketching and Prototyping**

Sketches / low / medium / high fidelity prototypes
- as investment in design increases (red arrow), so does the formality of the criteria whereby concepts are reviewed or accepted

From design to evaluation
- similarly, interface design (idea generation) progresses to usability testing (idea debugging and refinement)

**Sketching vs Prototyping**

<table>
<thead>
<tr>
<th>Sketches</th>
<th>Prototype</th>
</tr>
</thead>
<tbody>
<tr>
<td>Invite</td>
<td>Attend</td>
</tr>
<tr>
<td>Suggest</td>
<td>Describe</td>
</tr>
<tr>
<td>Explore</td>
<td>Refine</td>
</tr>
<tr>
<td>Question</td>
<td>Answer</td>
</tr>
<tr>
<td>Propose</td>
<td>Test</td>
</tr>
<tr>
<td>Provoke</td>
<td>Resolve</td>
</tr>
</tbody>
</table>

**Sketching and Prototyping**

Early design
- Brainstorm different representations
- Use sketches for idea generation
- Task centered walkthrough and redesign
- Fine tune interface, screen design

Mid design
- Choose a representation
- Rough out interface style
- Heuristic evaluation and redesign
- Usability testing and redesign

Late design
- Sketches & low fidelity paper prototypes
- Medium fidelity prototypes
- High fidelity prototypes
- Working systems
- Limited field testing
- Alpha/Beta tests
- Working systems
Sketches & Low Fidelity Prototypes

Paper mock-up of the interface look, feel, functionality
- quick and cheap to prepare and modify

Purpose
- brainstorm competing representations
- elicit user reactions
- elicit user modifications / suggestions

Sketches
- drawing of the outward appearance of the intended system
- crudity means people concentrate on high level concepts
- but hard to envision a dialog’s progression

![Sketches example images]
The attributes of sketches

**Quick**
- to make

**Timely**
- provided when needed

**Disposable**
- investment in the concept, not the execution

**Plentiful**
- they make sense in a collection or series of ideas

**Clear vocabulary**
- rendering & style indicates it’s a sketch, not an implementation

**Constrained resolution**
- doesn’t inhibit concept exploration

**Consistency with state**
- refinement of rendering matches the actual state of development of the concept

**Suggest & explore rather than confirm**
- value lies in suggesting and provoking what could be i.e., they are the catalyst to conversation and interaction

From Design for the Wild, Bill Buxton (in press) with permission

---

**Storyboarding**

- a series of key frames as sketches
  - originally from film; used to get the idea of a scene
  - snapshots of the interface at particular points in the interaction
  - users can evaluate quickly the direction the interface is heading

Excerpts from Disney’s Robin Hood storyboard, www.animart.com/Cellar/disneyart/90robin/ro20storyboard.jpg.html

From www.michaelborkowski.com/storyboards/images/big_biggu1.gif

note how each scene in this storyboard is annotated
Storyboarding

**Spotlight: an interactive foam core and paper sketch/storyboard**

Credit: Sue-Tze Tan, Dept Industrial Design, University of Washington

From Design for the Wild, Bill Buxton (in press) with permission

---

**Tutorials as Storyboards**

- a step by step storyboard walkthrough with detailed explanations
- written in advance of the system implementation
- also serves as an interface specification for programmers

---

**Pictive** plastic interface for collaborative technology initiatives through video exploration

Designing with office supplies
- multiple layers of sticky notes and plastic overlays
- different sized stickies represent icons, menus, windows etc.

interaction demonstrated by manipulating notes
- new interfaces built on the fly

session videotaped for later analysis
- usually end up with mess of paper and plastic!
Pictive

Can pre-make paper interface components

Buttons
Menu
Alert box
Combo box
Tabs
List box
Entries

Medium fidelity prototypes

Prototyping with a computer
- simulate some but not all features of the interface
  - engaging for end users

purpose
- provides sophisticated but limited scenario for the user to try
- can test more subtle design issues

dangers
- user's reactions often "in the small"
- users reluctant to challenge designer
- users reluctant to touch the design
- management may think its real!

Limiting prototype functionality

Vertical prototypes
- includes in-depth functionality for only a few selected features
- common design ideas can be tested in depth

Horizontal prototypes
- the entire surface interface with no underlying functionality
- a simulation; no real work can be performed

Scenario
- scripts of particular fixed uses of the system; no deviation allowed

Integrating prototypes and products

Throw-away
- prototype only serves to elicit user reaction
- creating prototype must be rapid, otherwise too expensive

Incremental
- product built as separate components (modules)
- each component prototyped & tested, then added to the final system

Evolutionary
- prototype altered to incorporate design changes
- eventually becomes the final product

Painting/drawing packages

draw each storyboard scene on computer
- very thin horizontal prototype
- does not capture the interaction “feel”

Scripted simulations

create storyboard with media tools
- scene transition activated by simple user inputs
- a simple vertical prototype

user given a very tight script/task to follow
- appears to behave as a real system
- script deviations blow the simulation
Interface builders

Design tools for laying out common widgets
excellent for showing look and feel
- a broader horizontal prototype
- but constrained to widget library
vertical functionality added selectively
- through programming

Wizard of Oz

A method of testing a system that does not exist
- the listening typewriter, IBM 1984

**Wizard of Oz**

A method of testing a system that does not exist
- the listening typewriter, IBM 1984

What the user sees
- Speech
- Computer

The wizard
- What you now know

**User centered + participatory design**
- based upon a user’s real needs, tasks, and work context
- bring end-user in as a first class citizen into the design process

**Prototyping**
- allows users to react to the design and suggest changes
- sketching / low-fidelity vs medium-fidelity

**Prototyping methods**
- vertical, horizontal and scenario prototyping
- sketches, storyboarding, pictive
- scripted simulations, Wizard of Oz

**Storyboard of a computer telephone**

Human ‘wizard’ simulates system response
- interprets user input according to an algorithm
- controls computer to simulate appropriate output
- uses real or mock interface
- wizard sometimes visible, sometimes hidden
  - “pay no attention to the man behind the curtain!”

**good for:**
- adding simulated and complex vertical functionality
- testing futuristic ideas
**Wizard of Oz Examples**

IBM: an imperfect listening typewriter using continuous speech recognition
- secretary trained to:
  - understand key words as "commands"
  - to type responses on screen as the system would
  - manipulating graphic images through gesture and speech

Intelligent Agents / Programming by demonstration
- person trained to mimic "learning agent"
  - user provides examples of task they are trying to do
  - computer learns from them
- shows how people specify their tasks

In both cases, system very hard to implement, even harder to change!