ICS 667
Advanced HCI Design Methods

3. Activity or Conceptual Design

Dan Suthers
Spring 2005

First, a few thoughts on design
Moran’s (and SBD) Levels of Design

• **Conceptual** (Activity)
  - **Task**: what does the user want to do?
  - **Semantic**: what objects, actions and methods are needed?

• **Physical** (Interaction and Information)
  - **Interaction** (Operational): what I/O sequences take place?
  - **Syntactic** (Representational): what should the information displays be like?

Our authors agree ... 

Design the functionality first

- Helps to focus on activities that effectively meet users’ needs and goals
- Helps deal with complexity of design: too many choices to make in interface
Bake the Whole Enchilada

- Scope of Design is entire situation: users’ experiences depend on more than software
- Distributed cognition: task-relevant information is distributed across individuals, artifacts, tools, social structures ...
- Sociotechnical Systems
- Activity Theory

Bill Verplank on Design

“One of the first things you learn in design is to put forward a number of alternatives so that you can then compare them. Having a lot of display space is important for doing this because you can then make them visual. One of the things you can do with visual things is superimpose them, or put them side by side and quite often when you start doing that you like one better than another. Until you’ve made a comparison you have no idea why you prefer one over another. The criteria emerge from the comparison.”

“Part of learning these skills is just looking at creative people, seeing the volume of stuff they do, and realizing the role of chance. ... It’s not just picking the right idea, but recognizing the right idea in all the mess that you produce. Having that rich field of things to compare and contrast that you’ve either generated or collected is something that designers need to know how to produce. Evaluation also comes into brainstorming: when you stop generating ideas you have to start evaluating them.”
SBD Approach to Activity Design

Sources of Design Ideas in SBD

Mix analysis and refinement with generation of new ideas

• **Metaphors:** The activity is like doing ... → implications
• **Tools:** The activity is like using a ...
• **Object Point of View:** I could support the activity by ...
• **Rewrite scenarios with the ideas so generated; re-evaluate claims**
Activity Scenarios

- What do we want the activity to look like (without being too specific about implementation)?
- Work on Activity Scenario and Claims Analysis Together
  - Scenarios are a flexible and inexpensive medium to try out ideas
  - Claims analysis becomes design rationale
  - Maintain or enhance as many positive impacts as you can while removing or minimizing negative impacts

Essential Use Cases

From Usage Centered Design (revisited):

Operational Model (environmental and contextual factors)

User Roles

User Role Map

User Roles (Role1, Role2)

Use Cases

Use Case Map

Tasks

Content Model

Content Map

Context

Navigation Map

Visual and Interaction Design

Domain Model (glossary, data model, or object class model)

Design “from the outside in”
### Use Cases: Various definitions

- The specification of a sequence of actions ... that a system, subsystem or class can perform by interacting with outside actors [Rumbaugh]
- A typical interaction between a user and a computer system that captures some user-visible function and achieves a discrete goal for the user [Fowler]

### Issues in Use Cases

- Typically a description of the interaction between user and system through a particular interface ...
- Inappropriate for design of interface! Assumes the design they are supposed to help you reach
- Hence Essential (abstract) use cases were born ...
C&L’s Definition

• A single, discrete, complete, meaningful and well defined task of interest to an external user in some specific role or roles in relationship to a system,
• comprising the user intentions and system responsibilities in the course of accomplishing that task,
• described in abstract, technology-free implementation-independent terms using the language of the application domain and of external users in role.

Essential Use Case (see paper)

<table>
<thead>
<tr>
<th>Identification</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>Name</td>
<td></td>
</tr>
<tr>
<td>Contextual Purpose</td>
<td>Supported Roles</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Relationships</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Specializes</td>
<td>Extends</td>
<td></td>
</tr>
<tr>
<td>Resembles</td>
<td>Equivalents</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Process</th>
</tr>
</thead>
<tbody>
<tr>
<td>Preconditions</td>
</tr>
<tr>
<td>User Intentions</td>
</tr>
<tr>
<td>Asynchronous Extensions</td>
</tr>
<tr>
<td>(steps)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Post-conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Business Rules</td>
</tr>
</tbody>
</table>
## Comparison to Scenarios and UC

- **EUC** are the most abstract: Completely separate problem description from situation or design solutions
  - More difficult for users to understand?
  - But not difficult! Designed to be accessible
  - More robust: remain valid in spite of implementation changes ...
  - ... or do they? Does the notation force any implementation decisions?
- **Scenarios** mix user and system, internal and external perspectives
  - This may be useful very early in design
- **Conventional UC** mix design solutions with requirements

## Question of granularity

- **Convincing arguments for fine granularity**
  - Each part easier to understand
  - Overall model simplified
  - Encourages reuse in both model and implementation
  - Encourages functionalities that are recomposable by the user
- **Yet I found coarse granularity to be useful early in design (NetLearn example)**
NetLearn Examples: Fine grained

Learner or Coach
Submit Work

<table>
<thead>
<tr>
<th>USER INTENTION</th>
<th>SYSTEM RESPONSIBILITY</th>
</tr>
</thead>
<tbody>
<tr>
<td>&gt; Enter study-group</td>
<td>List work and options</td>
</tr>
<tr>
<td>Select assignment</td>
<td></td>
</tr>
<tr>
<td>Select work to upload</td>
<td>Upload</td>
</tr>
<tr>
<td>Add explanation and indicate completion</td>
<td></td>
</tr>
</tbody>
</table>

Learner or Coach
Annotate Work

<table>
<thead>
<tr>
<th>USER INTENTION</th>
<th>SYSTEM RESPONSIBILITY</th>
</tr>
</thead>
<tbody>
<tr>
<td>Select link type</td>
<td>Indicate readiness</td>
</tr>
<tr>
<td>Indicate completion and attachment point</td>
<td></td>
</tr>
<tr>
<td>Optionally add comment/pointer</td>
<td>Add link to display</td>
</tr>
<tr>
<td>Criteria are lists provided by cate?</td>
<td>Attach comment to link</td>
</tr>
</tbody>
</table>

Learner or Coach
Review Work

<table>
<thead>
<tr>
<th>USER INTENTION</th>
<th>SYSTEM RESPONSIBILITY</th>
</tr>
</thead>
<tbody>
<tr>
<td>&gt; Enter study-group</td>
<td>List work</td>
</tr>
<tr>
<td>Select work collection</td>
<td></td>
</tr>
<tr>
<td>Link work to criteria</td>
<td>Display work details</td>
</tr>
<tr>
<td>Display work details and links</td>
<td></td>
</tr>
</tbody>
</table>

Learner or Coach
Annotate Work

<table>
<thead>
<tr>
<th>USER INTENTION</th>
<th>SYSTEM RESPONSIBILITY</th>
</tr>
</thead>
<tbody>
<tr>
<td>&gt; Enter study-group</td>
<td>List work</td>
</tr>
<tr>
<td>Select work</td>
<td></td>
</tr>
<tr>
<td>Select link type</td>
<td>Indicate readiness</td>
</tr>
<tr>
<td>Indicate completion and attachment point</td>
<td></td>
</tr>
<tr>
<td>Optionally add comment/pointer</td>
<td>Add link to display</td>
</tr>
<tr>
<td>Criteria are lists provided by cate?</td>
<td>Attach comment to link</td>
</tr>
</tbody>
</table>

Big Picture: Assessing Work with respect to Criteria

Purpose: Evaluate whether work meets criteria; evaluate skills in doing this evaluation.

<table>
<thead>
<tr>
<th>USER INTENTION</th>
<th>SYSTEM RESPONSIBILITY</th>
</tr>
</thead>
<tbody>
<tr>
<td>Learner+ &gt; Submit Work</td>
<td>Records work in coach area</td>
</tr>
<tr>
<td>Coach &gt; Edit Work Collection+</td>
<td>Moves anonymous work to work area</td>
</tr>
<tr>
<td>Learner*, Coach* &gt; (Review Work) Annotate Work+</td>
<td>Records and displays links</td>
</tr>
<tr>
<td>Learner+ &gt; Review Annotations</td>
<td>Enables easy browsing of links and attached comments</td>
</tr>
</tbody>
</table>

Coach > Review Use of Criteria

<table>
<thead>
<tr>
<th>USER INTENTION</th>
<th>SYSTEM RESPONSIBILITY</th>
</tr>
</thead>
<tbody>
<tr>
<td>Coach &gt; Review Use of Criteria</td>
<td>Enables summary display of which criteria each person used and which they missed</td>
</tr>
</tbody>
</table>

Coach > Create Public Archive of Annotated Work

<table>
<thead>
<tr>
<th>USER INTENTION</th>
<th>SYSTEM RESPONSIBILITY</th>
</tr>
</thead>
<tbody>
<tr>
<td>Coach &gt; Create Public Archive of Annotated Work</td>
<td>Moves anonymous work with links/comments to public (visible) area. Makes printable form available</td>
</tr>
</tbody>
</table>

Comments: Might start with only coaches doing the annotation, but then gradually draw the learners in. Might start with individual feedback of coach to learners, albeit in public space; but gradually other learners start commenting on each other’s work. (Need to select display of links by particular persons.)

There are more activities to be sorted but this is a starting point that covers a lot of ground.
On Notation

“Notation is, in truth, the user-interface of models.”

“... the visual form of notation can significantly increase or decrease the effectiveness of those who create and interpret the models” [C&L]

My research has shown similar influences of notations used during learning.

How do SBD and UCD notations compare?

Activity

• First we’ll catch up from last week.

• Then we’ll try the following two methods on the address book problem:
  - Activity Scenario and Claims Analysis
  - Essential Use Cases
Assignment 3

- Apply these methods to the calendar problem
  - Metaphors
  - Activity scenario and claims analysis (together)
  - Essential Use Cases

Extras

There’s some good stuff here, but I moved it back here so I would not talk too much in class.
Supporting Design: Techniques

- Understand the problem domain
- Identify previous successful solutions to similar problems
- Decompose into less complex, better defined sub-problems
- Generate and evaluate alternative designs
- Simulate partial designs with someone not on the team
- Opportunistic reasoning, “jump around” but have well organized places to put the ideas

Supporting Design: Tools

- Post-its, paper, posters, markers, ...
- Shared representations, common language
- Software
  - Communication tools supporting negotiation about artifacts
  - Diagramming facilities (dataflow, ER, UML)
  - Storyboarding facilities
  - Visual GUI builders
- Design Room
**Effectiveness**

- Innovation is good, but how much is too much?
  - build on what is already working well
  - engage stakeholders in cooperative design
- What parts of a task to support via technology?
  - leverage other aspects of the work context, both people and things (distributed cognition)
- Balance tendency toward general solutions with the needs of specific tasks
  - predict and support exceptions
  - provide special cases for common or critical tasks

**Comprehension**

- Cannot directly observe comprehension
  - must rely on users’ behaviors, reactions, comments
  - make inferences about their mental models
- Metaphors play a crucial role in this
  - designers explore metaphors to get new ideas
  - users evoke metaphors to understand new concepts
- Try to leverage users’ existing knowledge
  - anticipate and support analogical reasoning
  - but look for ways to “break” current understandings
Satisfaction

• Automate tedious tasks, but try not to remove sources of reward or accomplishment
  - carefully examine sources of reward, maintain or enhance opportunities for feelings of achievement
  - use the computer to make tasks more personal, more stimulating, more “fun”
• Balance the needs of individuals with those of the groups they work with
  - the people who do the most “work” when using a system may not be those who get the most “benefit”

Metaphors

• Understanding a system by analogy to a more familiar system
• Two ways that metaphors can be useful:
  - To generate ideas in design
  - As a basis for the interface itself (to make it more comprehensible to users)
• A little early to decide on the latter!
Metaphors in the Interface

- **Verbal metaphors** (given in instructions: “this is like a file system”) can enhance learning
- **Visual metaphors** make the interface look like the physical world
  - Original: Xerox Star desktop metaphor
  - Objective is to make the interface behave like the physical world so actions have the expected effect

Examples of Metaphors

- OS as desktop
- Spreadsheet as ledger book
- Hypertext/Hypermedia as notecards
- Hypertext/Hypermedia as rooms
- Software as physical world (object oriented)
- Software as service industry (agents)
- Learning activity as travel (tours and navigation)
Issues with Metaphors

• "Like" or "Is"?
  - Visual metaphors lead to mental models based on the metaphor, not the system
  - Functional rather than structural model
• Mismatch in behavior: interface object may behave differently than real world object
  - "Drag the disk into the trash so you can take it with you"
• Incorporating functionality not in the metaphor world ...

Composite Metaphors

• How to extend the metaphor’s functionality?
• “mixed metaphors”
  - “Copy the text you want: it will be on the clipboard. Then go to the desktop and open the window you want to put it in, scroll the window with the mouse to where you want it, and paste the text.”
• People are ok with it!
• The problems are in the parts