MIS 44044: System Analysis II

Term Project
Task Centered Design, Prototyping and Usability Testing*
Due December 9th,

This project is a hands-on experience in iterative user-centered system design. Your design process starts with getting to know the intended users, their tasks, and the working context of their actions. Only then do you consider what the actual system design should look like, basing your design on real people, tasks, and needs.

User centered system design is not an academic process where some cookbook formula can be applied. Nor is it an intuitive process where programmers can sit in their offices and think they know the users and their tasks. Rather, it is a hands-on process that requires you to go out and identify actual users, talk to them about what tasks they are trying to do, and understand their work context.

Because your initial designs will be crude and problem-prone, you will have to identify potential usability problems by evaluating your design and by crafting new ones. Once all the input has been processed the revised prototype can be made into the final product.

This user-centered system design consists of six activities:

1. identify tasks (business need) that need to be accomplished
2. generate system requirements
3. create several low-fidelity prototypes based upon system requirements
4. evaluate the prototypes through a task-centered walk-through to create a high-fidelity prototype
5. conduct a usability test
6. revise prototype to incorporate findings from usability test

Deliverables

I. Your group must prepare a system design and discussion portfolio to document the progression of your design. This portfolio should be written for the vice-president of your company and must be neat, well organized, and visually appealing. It should contain titled section separators to separate each section and should be placed in a 1" binder. The cover of the portfolio should include the names of the group members and the project title. Grading will be based upon the sophistication and maturity of the final report, the elegance of the designs, the logic of the presentations, and completeness of the analysis.

   This portfolio should have five sections.

   A. Section 1: User needs: Tasks and requirements (~ 4 pages) Due September 26

      Introduction. Describe in general terms the background for the system. Describe the intended users, their work contexts, and how they will use the envisaged system. For more information refer to Appendix 1.

Concrete task examples. List at least 5-7 concrete task examples (see Appendix 2). These task descriptions should be short and to the point. The class of the intended user (i.e. a typical customer) and the relative importance of the task (frequently done and important, infrequently done but still important) should accompany each task.

System requirements. Identify the *major* system requirements (hardware, software, network, content) and prioritize them into three categories: 1) absolutely must include; 2) should include; and 3) could include.

B. Section 2: References (~ 2 pages) Due October 3, 2005

Provide annotated (1-3 sentences) references to academic papers, industrial reports, and web pages on your topic. Compare your work to existing products.

C. Section 3: First design (~ 4 – 6 pages) Due October 17, 2005
   - Develop at least two very different low-fidelity prototypes that you believe will satisfy the major user requirements
   - Discuss the prototypes with your potential users. The more different you are from the target users, the more important it is for representative users to give you feedback on your design
   - For the prototype design that seems promising, use the tasks from Section 1 to perform a task-centered walkthrough with potential users
   - List the problems and successes for each task
   - Summarize the major design problems that must be corrected, as well as what seems to work well. Use information in Appendix 3 for your analysis.

D. Section 4: Task list and usability questionnaires (~ 2 – 3 pages) Due October 31
   - Prepare the tasks for the usability test
   - Prepare the pre-and post-test questions

E. Section 5: Usability test report (~ 6 – 8 pages) Due November 7
   - Create your high-fidelity prototype
   - Describe the usability test, subjects, and how it went
   - Itemize the problems you identified and give them points in terms of importance (1 is least important; 5 is most important) and effort (1 is least time consuming; 5 is most time consuming).

II. In addition to the progress report you must prepare an **Interface Design Report**. This report, due December 9th, contains the revised prototype, commentaries and the working system that is implemented. The format for this report follows:

A. Title page: Title, Authors, Electronic Mail Addresses, and Date
B. Abstract: 100-150 word overview of project
C. Credits: Indicate which team member is responsible for each part
D. Introduction (2 - 3 single spaced pages)
   1. Overview of the problem
   2. Discussion of previous work including (with references at the end)
      a) commercial systems
      b) previous academic papers

c) relevant web sites

E. Presentation of design (8 – 10 pages)
    a) Give overview of your approach and solution
    b) Show transition diagram for screens in your design
    c) Present all screens you have designed with commentaries to explain (do screen captures of your prototype images)
    d) Present your tutorial/help

F. Report on development process (8 - 10 pages)
    1. Show a few of your low-fidelity prototype screens
    2. Describe your process for arriving at the high fidelity prototype
    3. Describe your usability testing process (subjects, tasks, results)

G. Conclusions (1 - 3 pages)
    1. Describe the final status (what was implemented)
    2. Future work possibilities (what needs to be done to complete & refine)
    3. Recommendations to future developers of your idea
    4. Acknowledgements (a few sentences)
        a) Thanks to teachers, bosses, organizations, or friends who have helped you

H. References (5-20 references)
    1. in a neat standard format, alphabetical by last name of first author

Editing Guidelines:

Use Times Roman New font 12 point for text throughout, bold
for section titles. On title page use 14 point bold for the title (capitalize first letter of each word), the rest in 12 point not bold, all left justified.
In body of paper, use single spacing; skip two lines before and one after each section title.
Number each figure and give it a caption.
Cite references by authors and date, for example, (Smith and Jones, 1999).

Appendix 1.
Methods for Task Analysis

Step 1. Generate a list of expected users, and an initial list of tasks. Interview knowledgeable people about their real-world tasks and observe them doing their tasks. Your goal is to generate an initial list of concrete task descriptions. You may or may not be able to access your real clients in the short time available for this exercise. Consequently, each team should select the approach below that best fits their constraints and team membership.

i. *The ideal: Interviewing the client.* Get in touch with current or potential users. These users may now be using paper methods, competing systems, or antiquated systems for doing their tasks. Interview them about why and how they do their work activities, and what they expect out of a system. Ideally, this interview will occur while you are observing them do their work activities. These interviews and observations will give you some contact with customers and give you a feel for the real situation. This is more important than you think, for it will make you realize that 'the user' is not an abstract notion, but real people with real needs and concerns. It will help you put a face on the faceless, and will help you understand where they are coming from.

ii. *A reasonable alternative: Interviewing the client representative.* When you cannot get in direct contact with end users, you can use customer representatives instead. These will be people who have the most knowledge of the clients' needs. Examples are help desk people, or a worker's manager. However, it is crucial that the client representative has a deep and real (rather than idealized) understanding of what the workers actually do. People who work "in the trenches" with the staff are the best bet.

iii. *When all else fails: Making your beliefs of the task space explicit.* If you cannot get in touch with either real end users or representatives, use your team members to articulate expected tasks. While this runs the risk of producing tasks that bear no relation to reality, at least you will get a diverse list of tasks out (because you have a diverse team), and it will put your beliefs and assumptions on the table. You can always show these to clients later, to see if these tasks indeed reflect what they do!

Regardless of the approach you chose, do the following steps. If you have a client and/or representative, you would do it with them. If you are "making it up", try to imagine as realistic a situation as possible.

1. Have the client/representative/team recount a few (3-4) stories that typify the actual use of their system and/or process. Where possible, describe the people, the particular problems they wanted to solve, what information they brought into the meeting, the steps taken in the actual process, the constraints they had (e.g., time), what was produced, and whether they were satisfied with the outcome. All details are relevant. Alternatively, the task could be derived from direct observation of them doing their work.
2. On a more general and less detailed level, list as many related tasks and their variations as possible.
3. There will be many task variations in the list. Identify (with the user, if possible) which tasks are frequent, which are infrequent but still important, and which are rarer and not very important.

At this point, you will have a set of concrete, detailed examples of tasks that people now perform, or would want to perform on your system. Each task description should have the attributes described in the appendix and the second reading.

Step 2. Validating the tasks. The next step is to get a reality check of your task list. Have end-users and/or client representatives review your tasks. They should check to see if the set of people are representative of potential end-users of your product, if tasks capture the variations of those done by real people, and if details are realistic (they will be, if they are based on real customers!). You should ask for details that were left out of the original task description, get corrections, clarifications, and suggestions, and then re-write the task descriptions.
Note: This step is critical if you used a client representative or just yourselves instead of a real client. While it may not be possible for you to interview and observe many real clients, you can probably get one to comment on a compiled list of prototypical tasks.

Step 3. Deciding upon key users and a tentative list of requirements. The task examples will provide clues on specific system requirements that you could use to design your system as well as your target users. Because it is unrealistic to meet all requirements and address all users, it is your job to prioritize them. From the task examples (and possibly by further discussion with end users), decide upon the major system requirements and prioritize them into a) absolutely must include; b) should include; and c) could include. Similarly, decide upon what kind of users you must address, up to those you will exclude.

Step 4. Develop low fidelity prototypes. From the task examples and requirements, your team should sketch out several competing interfaces. Discuss and choose the most promising of these, and develop a low-fidelity prototype (using storyboards) that demonstrates how the interface fulfills the requirements.

Specifically, use the key users, their tasks, and the prioritized requirements as a type of requirements document to help you brainstorm prototypes that illustrate how your system would appear to the customer. You should be creating low fidelity prototypes e.g., paper sketches, storyboards, cut-and-paste mockups. You should not be concentrating on prettiness or completeness; rather, you are trying to show the overall interaction style of your system. Each prototype should contain the core screens that illustrate how the system will work as a whole, including (perhaps) a sample interaction based upon some of the key tasks.
Hint: To get diversity, each group member may want to try to create a few rough sketches before gathering as a group. You should also realize that some people may be better at this than others; this is not supposed to be a competition!

Step 5. Task-centered walkthrough. Test the prototype for usability bugs (problems, faults, weaknesses) by performing a task-centered walkthrough, as described in Appendix 1 and the readings.

Appendix 2.
Getting to Know Users and Their Tasks

Read: Lewis, C. and Rieman, J. (1993) Task Centered User Interface Design
http://www.hcibib.org/tcuid/
Chapter 2: Getting to know users and their tasks is the source for the exercise and for the following notes.

Good task examples:

- Says what the user wants to do but does not say how the user would do it
  - you are not to make any assumptions about the system interface
  - we will eventually use this to compare different interface design alternatives in a fair way
- Are very specific
  - says exactly what the user wants to do
  - we will eventually use this to specify the actual information the user would want to input to a system, and what information they will want out of it
- Describes a complete job
  - should list all aspects of the task, from beginning to end
  - this forces designer to consider how interface features will work together
  - we will eventually use this to contrast how information input and output is carried through the dialog, i.e.:
    - where does information come from?
    - where does it go?
    - what has to happen next?
- Says who the users are
  - use particular people, if possible
  - reflects real interests of real users
  - the success of a design is strongly influenced by what users know and their real work context; we will eventually use this information to see if people realistically have the desire, knowledge and/or capabilities to accomplish their task with the system.

Example task description for a clerk in a video store, including discussion. The eventual system will assist the clerks to perform their tasks.

> Mary Farness, an experienced full-time clerk at the video store, opens the store in the morning. She begins the day by checking in all the videos returned in the night video slot, which typically number between 90 to 150 videos. She pauses her task whenever customers ask for her services. She usually checks in ten videos, and then reshelves them before going onto the next ten.
>
> *Discussion.* In this case, the "user" is the full time person who normally carries out this task. We expect them to be typical of an experienced clerk who will know

the process well, and who will become well practiced at using the target system. The task is routine and frequently done.

George Marlay, a regular video store customer, approaches Mary and asks if they have the Frankenstein comedy video. She asks if he means "Young Frankenstein" by Mel Brooks, and he say “yes”. She then directs him to the shelf where the video is expected to be. George retrieves the video card and brings it to the front desk. Mary asks for George's membership card, but George has forgotten it. Mary then looks up his membership number. Mary checks out the video, but reminds George that he has not yet returned the video "Brazil", which is now a day late. George says that he will bring it in later today, and leaves with the video.

Discussion. This task contains many typical clerk activities, which deals with vague requests about video titles, the location of the video in the store, forgotten membership cards, the video checkout activity, as well as reminders to customers about late videos. Most these tasks are frequently done, and important.

Anil, a part time clerk who works the telephone, comes in for an hour every third evening. His job is to search the rental records to find customers who are at least one day late on their video returns. For example, he phones Bob Jakobs, who is two days late. Bob answers, and Anil identifies himself, tells him that he still has the video "Volcano", and reminds him to return the video. Bob says he will bring it back in an hour or so, and Anil crosses his name off the list. He then phones Ania Sliven, and says (more or less) the same thing. However, Ania says that she has already returned the video the day before. Anil puts her on hold, runs to check the shelf and finds the video there. He apologizes and hangs up. He then phones Ang Lee, but there is no answer. He notes on another list that he should try this person again later. He continues in this manner. When he has finished the list, he starts again on those who have not answered.

Discussion. This task identifies a specific activity that is less frequently done but still quite important. It also indicates that a non-regular staff member may be doing this task.

Why these are good examples:

- They say what the person wants to do, but does not say how it will be done. For later exercises, we can use these examples to see if a particular system would allow the person to accomplish their task.
- They are specific. They say exactly what type of information a person is bringing in to the task, exactly what information the person wants out of it, and how the person will use it.
- They describe a complete job. When we test a system, we can actually follow this sequence of events and see the amount of work that has to be done to get from one step to the next.
- They say who the user is. In this case, they identify particular people, their experience, and what knowledge/experience they have in their head. Again, this has implications to eventual use of a system.

Appendix 3:
Developing and Evaluating Initial Prototypes

For more information, read:

- Rettig, M. (1994) *Prototyping for tiny fingers* . Communications of the ACM, 37(4). http://www.cs.umd.edu/~bederson/classes/cmsc434/p21-rettig.pdf
- Logical User Centered Interaction Design (LUCID), http://www.cognetics.com/lucid
-

Step 1. Develop low fidelity prototypes

Use the key users, tasks, and system requirements generated previously as a type of requirements document to help you brainstorm prototypes that illustrate how your system would appear to the customer. You should be creating several *low fidelity* prototypes e.g., paper sketches, storyboards, or physical mockups. You should also be thinking about what *conceptual model* the system will portray to its customers. You should not be concentrating on prettiness or completeness; rather, you are trying to show the overall representation and interaction style of your system. Each prototype should contain the core screens that illustrate how the system will work as a whole, including (perhaps) a sample interaction based upon some of the key tasks.

You should use the ideas of "psychology of everyday things" and "beyond screen design" to help you, as well as your general knowledge of other systems (there is nothing wrong with copying ideas, providing its legal!).

Hints: To get diversity, each of you may want to try to create a few rough sketches before gathering as a group. You may also want to talk to the systems people in your group, because there may be opportunities that already exist with the current way the information is stored and presented (e.g., if customers are already using software), or constraints that limit what you can do (e.g., if the system must be delivered as a Windows ’95 application). You should also realize that some people may be better at this than others; this is not supposed to be a competition!

Step 2. Evaluate the prototypes

- Discuss each prototype to see whether it is a possibility in principle (e.g., are there obvious problems with the conceptual model? Can it be implemented?)
- Do a task-centered system walkthrough for each of your key tasks, and each of your user types.
- From the ones that are left, elicit reactions and further discussion from customers / counter people / appraisers. You may find that your end-users will tell you about further tasks and task details that were not thought about before!
-

Step 3. Reconsider priorities and make a preliminary decision

Based on the prototype and evaluation exercise, you may wish to reconsider what customers you will address as well as what tasks and requirements you will support. It could be that you were wildly optimistic about what you could do!

At this point, you should have a reasonable idea of which prototype styles are worth pursuing, or whether you should start again. Make your decision on what direction(s) to follow. If you have more than one direction, you may want to continue developing both a bit further. If you have no worthy candidates, return to step 2.

Hint. Don't feel committed to any prototype. This is the time where prototypes are quick to generate and cheap to discard. Use this time to explore the design space. While you may want to just get on with it, a bad design choice now can have disastrous and expensive repercussions later.

Step 4. Refinement and evaluation

Refine your prototype by considering the nuances of each task, the functions that must be incorporated, and the expected reaction of each user. You may want to start considering the more subtle aspects of interface design at this point (e.g., psychology of everyday things, principles of graphical screen design, design principles). You should be continually evaluating these prototypes as appropriate. Real users should be commenting on them. You should be using walk-throughs, heuristic evaluation, and various observational methods.

If you follow this process, your prototypes will evolve from a competing series of design ideas, to a crude single design representation, to more realistic looking screen designs, to functioning systems. As your design is refined, you will move from very low fidelity prototypes done on paper to medium fidelity prototypes done on-line, likely through an interface builder. You will have detected and corrected the major interface design problems, and will be concentrating on fine-grained details. Eventually, you will create vertical prototypes with back-ends that fake some of the system functionality. To the user, however, it will look like the real system.

*These specifications are based on those created by Dr. Ben Shneiderman for a team project for University of Maryland's course, CMSC 434, taught in Spring 2004 and retrieved from http://www.cs.umd.edu/class/spring2004/cmsc434/project.html.