Usability Engineering Project

CS 4163

Summer 2002

1 Overview

1.1 Description

For this assignment, you and a team of the students in the class will design and implement an interface for an audio record management system (ARMS). The assignment will consist of a set of phases. Each phase is important and will be graded separately.

All the work you submitted for grading must have a professional, finished appearance.

1.2 Phases of Project and Schedule

Milestone	Deadline	Weighting
User and Feature analyses	27 May	10%
Task analysis	03 June	15%
Test #1	*12 June*	
Design document	17 June	15%
complete prototype		
Testing strategy	26 June	15%
Test #2	*12 July*	
final deadline to get consent form approved	12 July	
begin testing		
collect data		
revise prototype		
Demonstration	17 July	10%
Final revision	24 July	10%
Planned improvements	24 July	10%
Portfolio	24 July	10%
Group leader report		5%
Group work report	29 July	0%
Lessons learned	29 July	

Deadlines are at: the beginning of class on the days when the class meets, and noon on the other days.

1.3 Project Homepage

Any updates to, resources for, and announcements about the project will be available from the project homepage on the World Wide Web (WWW) at ⟨URL:http://www.cs.dal.ca/~jamie/course/CS/4163/project/⟩. Be sure to check it frequently.

A copy of this project description is on the WWW too at ⟨URL:http://www.cs.dal.ca/~jamie/course/CS/4163/project/project.pdf⟩.

2 Groups

As we have seen in our lectures, software with a significant user interface component should be developed in groups. Many of you students are not experts in UI design or the problem domain but you will all benefit from working in a group nonetheless.

2.1 Rules for Groups and Group Dynamics

1. Each group member is expected to make an equal contribution to the project. All group members will receive the same grades for the phases of the project except in the most exceptional circumstances.
2. The products of your group should be high-quality and I expect for the group to produce better products than would result from individual work.
3. For this assignment, your work should be detailed and grammatically correct. Assignments that are too short, filled with spelling errors, or grammatically embarrassing will not receive passing grades.
4. You will need to keep a weekly log of group activities for the group work report.

2.2 Group Leader Reports

You will take turns acting as group leader for phases of the project. You must all work together during each phase and part of the leader's responsibility will be to ensure that the work is shared equally. The group leader will also be responsible for coördinating meetings, activities, and documentation for that phase.

Where there are three members in a team, one of you will be the leader for the task analysis, another one will be a leader for the design document, and the third member will be leader for the testing strategy. Where there are more than three members in a team, the others will lead for the coding phase. No team (for this project) should have more than four members.

Within 48 hours of completing their leadership phase the designated team leader must submit a teamwork report that:

1. lists all the team members and how each one contributed to the phase
2. includes a ranking of the contributions with no ties (someone must have made the biggest contribution, and someone else the least)
3. shows how the leader made sure that the work was fairly divided amongst the team members.

2.3 Final Group Work Reports

After the final project is due all team members must submit their own group evaluation reports. Each team member will write their own report.

In the report you will tell me what you did in the project, what other members of your group did and how your team make sure that the work was divided fairly between you all. You should keep a log of all the activities your group does each week to help you in writing the report.

Both of these reports are mandatory.
You cannot receive a passing grade for the project without submitting both of these reports.

3 Needs and User Analyses (10%)

Due: at start of class on Monday 27 May

To determine the tasks that the user needs to perform with an audio record management system (ARMS) interface, you should interview potential users and evaluate existing software (e.g. shareware). Note in your report if you did or did not interview potential users.

3.1 User Analysis

For your analysis you will need to interview two groups of potential users of your system. Only one of those groups may be similar to students in this class.

Your report should not be longer than 5 pages.

Your report should include the following information for each of the two groups:

1. Characterize the users as a group — who are they?
2. How will they use the system?
    - Who uses the system (will they do it themselves or will someone else do it for them?)
    - What benefit will the users get from the system (why should they use it?)
    - What is the environment in which it is used (e.g. an office, home, car)
3. What skills do they have?
    - computer skills
    - skills in the domain of the system
4. What, if any, other important characteristics do the users have?
5. What interview questions did you use?
6. How did you choose these people to be interviewed?

3.2 Needs Analysis

Your analysis should have the following three parts:

Goal

- a statement of the expected use of the system

Assumptions and Constraints

- what do you need to assume about the available hardware, information, etc. for the system to work
- roughly how much will it cost
- what world-view does this system relate to (e.g. if people are more efficient then the company will make more money)

List of Features

- less than one page long
- specific things a user could do with the system

The entire document should be less than four pages long.

4 Task Analysis Exercise (15%)

Due: at start of class on Monday 03 June

For this exercise, we will be performing a task analysis for an audio recording management system. A task analysis is a hierarchical decomposition of the tasks that users need to complete to achieve their goals. As you know, goals are what the users want (or need to do) and tasks are what the users think they need to do to reach those goals.

If you have not already identified users goals through the user analysis and initial feature list then you will need to do it before you can develop a complete task analysis.

Sections 20.2 and 20.3 (pp. 413 – 419) in Preece give examples of a hierarchical task analyses. The goal of this exercise is to produce at least the first four levels of a tasks hierarchy. The first level is one task name that describes the use of the whole system, from the users view. The second level is a list of the major subtasks that users perform. At levels 3 and 4, the major subtasks are decomposed.

Note that you may be making some design decisions by your decomposition. But the task analysis is *not* the same as the design. The tasks must all be actions! E.g. 'Select albums' or 'Manage albums' rather than 'Albums'. Some of the tasks will have sub-tasks and others will be categories of tasks that are grouped together.

Do not make any of your tasks specific to any one implementation.*

4.1 Hand in:

A graphical diagram of your analysis, in the style of *Preece et al.*'s[9] Figure 20.2. You should use a computerized drawing package but you may draw this by hand if you are extremely neat.

For each task, give a written description of its function. Include the following details:

Identification

- What is the name of this task?
- What is the goal of this task?

Location in Hierarchy

- What sub-tasks define this task?
- Is this task a subunit of a larger task?

Input/Output

- What kinds of inputs or actions does this task require from the user?
- What kind of outputs or results occur when this task is performed?

Non-interface Aspects

- What non-interface functions does this task require? or
- What automatic actions does this task expect from the system?

Other

- What special characteristics of this task should we record?

*If you follow the examples in *Preece et al.*[9] then watch out for the implementation-specific parts at the bottom left of Figure 20.1 and the lower levels of Figure 20.2.

4.2 Suggestions

4.2.1 Typical Errors to Avoid

- Lack of detail;
- Tasks which refer to implementation details rather than to the user's goals (e.g. selecting from a menu, or sorting recipes);
- Describing a hierarchical menu structure not a hierarchical task analysis (writing the description from the programmer's perspective, not the users);
- Inconsistencies between the items in the chart and the written descriptions;
- Spelling and grammatical errors.

4.2.2 Best Practices

There are many ways to make good hierarchical task decomposition charts however the best ones tend to have these properties:

- numbered tasks to make finding the corresponding description in the text easier. The numbers correspond to the hierarchy in the chart, e.g. task 1.2.3 is part of subtask 1.2 which is part of task 1. Task 1 is part of the top-level task, which is not numbered.
- where special symbols are used (e.g. $\star$ for tasks that are defined elsewhere, slashed corners for optional tasks, boxes that contain tasks to indicate that they must be performed in order, dashed and solid lines) there is a legend explaining their meaning.

The best task analyses are easy to read and refer to. I suggest that you make sure that yours include these properties:

- the pages and tasks are numbered; and
- each of the tasks is described on one page, that is the task description is not split across a page boundary. Often three tasks can fit comfortably on one page.

4.3 Approximate Grading Scheme

Professional appearance	15%
Format	15%
Breadth of analysis	25%
basic tasks and parts	
non-essential tasks and parts	
Depth of analysis	20%
accurate descriptions of tasks	
sufficient details in tasks	
Organization of analysis	25%
independent of implementation	
tasks relate to users' goals not to system actions	
structure of tasks and subtasks reflect users' goals	

If your task analysis is not acceptable then you will need to do it again before you can move on to the next phase.

5 Design Document (15%)

Due: at start of class on Monday 17 June

For this exercise, your group will develop a design document for your interface. The design document will be a record of the decisions you made about what the users will need to accomplish their goals. As with all parts of the project it is subject to change later, but you must make a detailed record of your decisions so that you can refer to them later. In a commercial venture the design document would also be used to communicate with clients.

Chapters 21 – 23 of *Preece et al.* discuss some specific activities in the development of interface design documents. Chapter 22 is of particular interest.

The design document will be in two parts: a conceptual design and a concrete design.

5.1 Conceptual Design

The conceptual design is about the things that the user will use and the actions the user will take with those things. We call those things *objects*. The objects are very much like data structures in program code, but I don't want to see the syntactic details of program code from you at this stage. I do want to see descriptions of the code that you would write but I do not want you encumbered by program syntax.

Your document must answer these questions:

- What things will the user work with?*
- How will the user get access to and use those things?†

5.1.1 Objects

You need to specify:

- What objects to use
- What the properties of those objects are
- How those objects and properties will be represented conceptually
- How users will get access to those objects

5.1.2 Operations

We call the actions that users can do with your objects 'operations'. You need to specify:

- What are the necessary operations
- How will users select those operations (through your system)
- How will users perform those operations

Remember that operations are often about how a user can get to use instances of an object.

* *Hix & Hartson*[2] (Table 1.1, p. 7) call these 'interaction components of the interface'.
† *Hix & Hartson* would call these parts of the 'interface software (to support interaction)'.

5.1.3 Presentation

You may present these details in any way that you think is meaningful. I have found that some of the clearest presentations have used a hierarchical format such as*:

- Object: `Recipe`
  - Components
    - Ingredients
      - Amounts
      - Name
    - Instructions
    - Yield
    - Comments
    - ...
- Manipulations:
  - create
  - edit
  - delete
  - ...

- Object: `Text`
  - Properties:
    - contents
    - ...
  - Manipulations:
    - create
    - edit
    - delete
    - ...
  - Instances:
    - recipe title
    - ingredient
    - comment
    - ...

Note that some objects have components and others have properties. Objects with components have many parts. Objects with properties have many aspects or attributes, but these are not the same as parts.

5.2 Concrete Design

In the concrete design you show how you think you will put your design into practice. This is typically done with drawings or sketches that give the impression of the overall interface and some important parts in more detail. If you choose to do drawings or images from screen building software (e.g. Visual Basic) then you should include some text to annotate your figures.

Your interface should be original and support users doing the tasks you specified in your task and user analyses.

Make your design detailed but don't try to make it perfect. It needs to be detailed enough for use but if it looks too much like a finished product it can be too hard to change later. Remember iteration is the key — you will need to change your design later.

5.3 Hand in:

Turn in a document of *at least* 4 pages that includes both stages. I need to see evidence that you have considered the implications of your design, used your (modified) task analysis, and are making good progress towards the final deadline.

Hand-in your (modified) task analysis along with the design document. If I told you that you had to modify your task analysis before moving on to the design document, or if you received a grade of B^- or below for the task analysis, then you *must* redo your task analysis before I will grade your design document.

*This example is specifically for a cookbook system.

5.4 Typical Errors to Avoid

- Lack of detail;
- Inconsistency between the design and task analysis;
- Icons or command names that are not relevant to the user;
- Spelling or grammatical errors.

5.5 Approximate Grading Scheme*

Extent of design	70%
coherence	
completeness (breadth and depth)	
Overall quality of design	15%
quality of proposed interface	
functionality beyond basic requirements	
Appropriate innovation	15%
appropriateness of interface to users and task	
originality of interface	
Lacking necessary parts	−75%
interface type	
input/output devices	
concrete design (by description, diagram or both)	
conceptual design objects	
Lacking professional appearance	−10%
clarity	
legibility	
spelling, grammar	

5.6 Related Readings

- Chapters 21 – 23 of *Preece et al.*[9] discuss some specific activities in the development of interface design documents. Chapter 22 is of particular interest.
- The terminology from Hix & Hartson[2] is described on pages 7, 132 – 144.

*Negative numbers indicate maximum deductions.

6 Testing Strategy (15%)

Due: at start of class on Wednesday 26 June

You can start this phase *without* feedback about your design document

For this exercise your group will develop the documents and plans needed to evaluate your prototype system with actual users.

You cannot perform any testing without approval of your methodology (including the consent form) by the Human Research Ethics authority. You will need to develop your testing methodology and documents before you can get your consent form approved. You can download a boilerplate consent form (in plain text format) from the project homepage. Your consent form must be on Dalhousie letterhead.

You cannot perform any testing without approval for your methodology (including the consent form). You will need to develop your testing methodology and documents before you can get your consent form approved. You can download a boilerplate consent form (in plain text format) from the project homepage.

6.1 Hand in

- scenarios of activities for users to perform (these should be based on your task analysis and should include a mixture of representative and critical tasks)
- a list of measures you plan to use in your testing (where the reason for a measure is not obvious you should include a justification for how it will be used later)
- a description of the protocol you will use when interacting with the users (the instructions you will give to users, the forms you will use to record the test, how you will perform the test, a copy of the consent form you will use, etc.)

6.2 Suggestions

- Plan to have all of your experimental participants perform tasks in the scenarios. You will need *at least* five participants plus pilot testers. Serious testing of earliest prototypes requires at least eight participants.
- Plan to take careful notes of any comments, problems, and other noteworthy events that occur during testing. Plan to record what users do with the system as well as taking notes yourselves.

  Remember that if you use a paper prototype then one of your team will not be able to take any notes so plan accordingly.
- Use a modified version of QUIS or SUMI to collect post hoc impressions.
- Maner's notes about formative evaluation and scenario development are recommended.

6.3 Optional

- Include scripts which describe what the user should do with the interface. These can be used to test if the interface works the way you think it should and also to ensure that users understand the interfaces parts.
- Do not delay submitting your testing strategy because your consent form has not been approved. You must have an approved consent form to conduct the testing but I can give you useful feedback even if the form has not yet been approved.
- You might use a table with headings such as those below to help you record events during the testing.

Time	Problem	Effect on Performance

6.4 Grading Principles

Your strategy and supporting documents will be evaluated using three criteria: completeness of your plan (type of data to be collected, how data is to be collected, range of tasks used, etc.), extent of preparation, and professional appearance. A perfect strategy is one which shows me that you are prepared to begin serious testing immediately. Strategies which are disorganized, untidy, or do not appear professional for other reasons will be penalized as much as 10%.

Typical Errors to Avoid

- Lack of detail, specifically:
  - pilot testing not mentioned
  - cognitive walkthroughs (for baseline performance) mentioned but not included
  - number of test participants not included
- Recording made by hand and only by one person;
- Inconsistency between description and forms;
- Long documents without tables of contents;
- Texts to be read to users that are in the passive voice.

6.5 Related Readings

6.5.1 Highly Recommended

★ *Dix et al.*'s[1] Chapter 11 is an excellent overview of testing and evaluation

★ Maner's notes (at ⟨`URL:http://www.cs.dal.ca/~jamie/teach/WaltManer/README.html`⟩) about formative evaluation[5] and scenario development[6]

★ Part VI of *Preece et al.*[9] discuss evaluation and testing. Pay particular attention to Chapters 30 and 31.

6.5.2 Other Readings

- For initial testing and test design see Sections 4.1 and 4.3 of *Lewis & Reiman*[4]
- For user testing see Chapter 5 until Section 5.6 of *Lewis & Reiman*[4] (also complete excercise 5.1)
- There is a copy of part of QUIS in *Shneiderman*[11] on pp. 136 – 143.
- *Rubin*[10] has some very straightforward practical advice
- Information about SUMI is available online at ⟨`URL:http://www.ucc.ie/hfrg/questionnaires/sumi/index.html`⟩.)

7 Demonstration to Professor & Class (10%)

In class on Wednesday 17 July

I will be examining your interface to see how much you have accomplished by this time. I will be determining how many of the features you specified in your task analysis and described in your design document you have implemented and how well they have been implemented.

As a class, we will be using the first demonstration as an excercise in applying heurstic analysis. Only constructive comments will be allowed.

8 Final Version and Portfolio (total of 30%)

Due: at start of class on Wednesday 27 July

8.1 Demonstration of Final Project to Class

In class on Wednesday 27 July

This will be an opportunity for the class to appreciate the great projects you all will have produced.

8.2 Final Version (10%)

Due: at start of class on Wednesday 27 July

Turn in a copy of your prototype (electronic media or paper prototype)

Approximate Grading Scheme*

Extent of implementation	70%
coherence	
completeness (breadth and depth)	
Overall quality of implementation	15%
quality of interface	
functionality beyond basic requirements	
Appropriate innovation	15%
appropriateness of interface to users and task	
originality of interface	
Not consistent with (updated) design document	−20%
Shabby appearance	−10%
clarity	
legibility	
spelling	

*Negative numbers indicate maximum deductions.

8.3 Planned Improvements (10%)

Due: at start of class on Wednesday 27 July

This must be included in your portfolio (see Section 8.4, below).

Based on your users experiences with your system, give a detailed list of the improvements that you plan to make in the next version of your user interface. For each improvement, specify why your improvement should be made. You must address each item of user feedback.

If you have not already turned in proof that your consent form has been approved then you must include that proof with this report. If you don't have such proof then you will get no grade for this part of the project.

8.3.1 Suggestions

A summary table with headings such as in the one below can help to organize the parts you will need.

Problem	Effect on Performance	Importance	Frequency of Occurrence	Proposed Solution	Cost to Fix	Resolution

8.3.2 Grading Principles

When grading your report I will be considering: (1) the thoroughness of your analysis of the problems in your interface, (2) how specific you are about solutions, and (3) the rationality of your suggested changes.

8.3.3 Related Readings

- *Dix et al.*'s[1] Chapter 11 has some good advice about analysing test data
- *Lewis & Reiman*[4] from Section 5.5.5 to the end of Chapter 5 is about evaluating data collected during testing

8.4 Portfolio (10%)

Due: at start of class on Wednesday 27 July

Turn in a neat and well-organized portfolio of your project, including

- all analyses,
- user assessments,
- planned improvements (see Section 8.3 above),
- source code (where applicable), and
- anything else to show how your project evolved.

The **signed consent forms** must be included, but should be in a separate section or envelope so that they can be removed from the portfolio after grading.

8.4.1 Suggestions

I expect a detailed and well-organized document. This should be something that can be referred to again. Consider including a table of contents, indexing tabs, or both.

8.4.2 Approximate Grading Scheme

professional appearance	40%
completeness of document consistency of documents	60%

9 Fairness Evaluation (0% — Required, Max. deduction 50%)

Due: in my mailbox or under my door, 10:00a.m. Monday 29 July

It is essential that the group work is done fairly. Each of the group members must do this assignment on their own. Refer to your weekly log of group activities to help you complete this report.

Every student must tell me (briefly):

1. What work each team member did to complete the project; and
2. What steps the team took to ensure that the work was divided fairly.

10 Project Assessment (0% — Optional)

Due: in my mailbox or under my door, 10:00a.m. Monday 29 July

I will appreciate it if you would use a few minutes to write me briefly:

- What you learned (about yourself, working in groups, and human-computer interaction) by working on this project; as well as
- If you had do the project again, how would you do it differently?

If you prefer, you can submit these 'lessons learned' anonymously. I want to know what you truly think about your experience with this project.

The following are references to materials used to create this project.

Bibliography

[1] Alan Dix, Janet Finlay, Gregory Abowd, and Russell Beale. *Human-Computer Interaction*. Prentice Hall Europe, second edition, 1998. ISBN 0-13-239864-8.

[2] Deborah Hix and H. Rex Hartson. *Developing User Interface Ensuring Usability Through Product & Process*. Wiley Professional Computing. John Wiley & Sons, Inc., 1993. ISBN 0-471-57813-4.

[3] B. Kirwan and L.K. Ainsworth (editors). *A Guide To Task Analysis*. Taylor & Francis Inc., 1992. ISBN 0-7484-0057-5(cloth)/0-7484-0058-3(paper).

[4] Clayton Lewis and John Rieman. Task-centered user interface design: A practical introduction, 1993, 1994. Available at ⟨URL:ftp://ftp.cs.colorado.edu⟩ and ⟨URL:http://hcibib.org/tcuid/⟩.

[5] Walt Maner. Formative usability evaluation, 15 March 1997.
⟨URL:http://csweb.cs.bgsu.edu/maner/domains/Formeval.htm⟩.
There is also a copy at ⟨URL:http://www.cs.dal.ca/~jamie/teach/WaltManer/Formeval.htm⟩.

[6] Walt Maner. Scenario method, 15 March 1997.
⟨URL:http://csweb.cs.bgsu.edu/maner/domains/Scenario.htm⟩.
There is also a copy at ⟨URL:http://www.cs.dal.ca/~jamie/teach/WaltManer/Scenario.htm⟩.

[7] Deborah J. Mayhew. *The Usability Engineering Lifecycle: A practioner's handbook for user interface design*. Morgan Kaufmann Publishers, Inc., 1999. ISBN 1-55860-561-4.

[8] Jakob Nielsen and Thomas K. Landauer. A mathematical model of the finding of usability problems. In Stacey Ashlund, Kevin Mullet, Austin Henderson, Erik Hollnagel, and Ted White (editors), *INTERCHI '93 Conference on Human factors in Computing Systems INTERACT '93 and CHI '93 Bridges Between Worlds*, pages 206 – 213. Amsterdam, The Netherlands, 24 – 29 April 1993. See also Spool and Schroeder's [12] article.

[9] Jenny Preece, Y. Rogers, H. Sharp, D. Benyon, S. Holland, and T. Carey. *Human-Computer Interaction*. Prentice-Hall Europe, 1994. ISBN 0-201-62769-8.

[10] Jeffrey Rubin. *Handbook of Usability Testing: How to Plan, Design, and Conduct Effective Tests*. John Wiley & Sons, Inc., 1994. ISBN 0-471-59403-2.

[11] Ben Shneiderman. *Designing the User Interface: Strategies for Effective Human-Computer Interaction*. Addison-Wesley, third edition, 1998. ISBN 0-201-69497-2.

[12] Jared Spool and Will Schroeder. Testing web sites: Five users is nowhere near enough. In *Proceedings of the SIGCHI Conference on Human factors in Computing Systems*. 2001.
⟨URL:http://www.winwriters.com/chi01.htm\#chi01_ts1⟩.