User Testing, Discount User Testing

1 INTRODUCTION

Most programmers know from experience that the first version of a new application always contains a large number of syntactic and semantic bugs. The first version of a user interface is similar: No matter how experienced you are, or how careful you have been in your field studies, task analysis and interface design there will always be serious and unforeseen problems in the user interface.

Such interface problems can be found in a user test. Some people confuse user testing (usability testing) with functional testing, which unfortunately is also sometimes called user testing. The purpose of a functional test is to detect functional problems in an application - in other words to check if the application delivers correct results. This chapter deals exclusively with the former kind of testing - usability testing. In the rest of this chapter we will use the term user testing instead of usability testing.

This chapter describes in some detail a test procedure called “Thinking Aloud Testing” [Lewis 1982] that you can use to effectively detect user interface problems.

A Thinking Aloud test has a large scope. It can be used to user test early paper prototypes of a new application for conceptual clarity, and it can be used to test late beta versions of a new application interface for disasters. It can be used to test a novice’s first encounter with an application, and it can be used with expert users who have used an application for a long time. It can be used to test a whole application, and it can be used to fine-tune a particular task that is carried out very often. It can even be used to measure usability as described in section 11.

One important piece of advice before we start with the technicalities of user testing: Before you start to plan and carry out user testing make sure that you have agreed with your management and your developers how the test results will be used. Many technically successful user tests have turned into failures because nobody had made it clear how the results were to be used. In other cases managers had agreed that user testing was to take place without bothering to inform the developers who were supposed to fix the detected usability bugs. Of course, the developers were reluctant to implement changes whose rationale had not been explained to them.

2 OVERVIEW

In a Thinking Aloud test an experiementer gives a typical user a number of realistic tasks to solve in some context. While the user carries out the tasks, he is asked to “think aloud”, that is, he should say what he thinks, what he is in doubt about, what he expects the application to do now, how he interprets a particular message from the application etc.

In a test of a word processing application the user could be a secretary, who is handed a manuscript with hand-written changes. While the secretary corrects the manuscript with the word processor, he is asked to think aloud.

Interesting events during the test are recorded on video tape, audio tape, or paper.

A user test ends with a debriefing, where the test user is asked to comment on the interface and on the test procedure.

3 DEVELOPING THE TEST

3.1 Developing Scenario and Tasks

The Thinking Alound technique requires a number of tasks that the user can solve with the application. The tasks should be based on a hypothetical but realistic scenario (context) in which the test user should imagine that he is working. In an ideal user test the scenario is not hypothetical, that is, the test user is carrying out a set of tasks that he would be solving anyway as part of his ordinary work.

As an example, the scenario could be that the test user is working as a substitute for a secretary who is ill. The tasks could then be to correct a manuscript, and print five copies of it.

The experimenter selects and formulates the scenario and the tasks in close cooperation with a user representative and a software designer. The user representative ensures that tasks are realistic and typical, and that task descriptions are comprehensible. The exprimenter takes care that tasks do not just test interface features that are known to be good. Ideally, the scenario and tasks should be created independently of application design, that is, the experimenter should have little knowledge of the details in the application design (dialog structure, detailed window design etc.) when creating the tasks. Tasks should be based on users' needs and not on what is available in the application menus or what the requirement specification says. One of the main purposes of a user test is to find out if the application meets users' needs.

The scenario and the tasks must be realistic. "Jokes" intended to entertain the test user often fail completely and create embarrassing episodes during the test.

Scenarios and tasks should be given to the user in writing. Oral presentations of tasks almost invariably contain hidden clues.

Here are a number of tips that you may find useful when writing your own scenarios and tasks:

- The first task must be so simple that the test users can solve it within a few minutes. This is a psychological trick, which is intended to calm down test users who may be stressed by the alien surroundings and the unfamiliar technical equipment. Note: It is very difficult to create a task that can be solved within a few minutes! I (the author) have never been able to accomplish this in the first attempt. I have always had to simplify the first task considerably after the pilot test.

- Users are impressed if tasks are specially tailored to them. Scenarios like “Imagine that your name is Joe Smith ...” work but create a distance between the test user and the tasks. Of course, tailoring tasks to each test user requires extra resources and careful preparation.
- Watch out for hidden clues in your task descriptions. For instance, to solve task 4 in table 1 the test user may need to press a tab or a button labelled *Recurring* or *Repeated*; therefore these keywords do not appear in the task description.
- Often, even the very existence of a task gives the test user an important clue, namely the clue that this task can be solved using the application. This is not a natural situation. In many cases where a user tries to carry out a task with an application for the first time, the user takes into consideration that the application may not be able to carry out the task. I have witnessed several tests where test users said “Well, now I continue with this task only because you have indicated (by giving me the task) that it can be done”. You may avoid this situation by including a seemingly reasonable task in the beginning of you task set which cannot be solved with the application. Since this task cannot be solved, and since the test user will try hard to solve it, it is important that the experimenter intervenes after a reasonable period with a careful explanation so the test user does not become uncomfortable or stressed. However, such an incident can be useful because it will mean that the test user from then on will take the possibility into consideration that test tasks may be unsolvable.

Task 1	Install Supercal on your pc.
Task 2	Enter into your calendar: Informal start-up meeting for the customer data base project on Monday, May 20 from 9 a.m. to 12 a.m. The meeting will be held in room 5561. Make a note that you should transfer incoming calls to extension 7991 in the meeting room
Task 3	Register Eric Butler John Wiley & Sons, Inc. 605 Third Avenue New York, NY 10158-0012 Telephone 212.850.6000 (work)
Task 4	Enter into your calendar: Badminton with Ken Thurber every Friday from 3.30 p.m. to 4.30 p.m. from now until the end of the year.
Task 5	Enter into your calendar: The informal start-up meeting for the customer data base project on Monday, May 20 has been moved to the afternoon from 2 p.m. to 5 p.m.
Task 6	You just tried to phone Ken Thurber, but his phone is busy. It is very important that you get in touch with him before he leaves for Europe today at 1 p.m. Make sure that Supercal will give you a reminder to try to call him again in 15 minutes.
Task 7	Laura Smith turns 50 on January 31 next year. Remember her birthday, and remember to send her greetings on future anniversaries.
Task 8	You finally managed to get in touch with Ken Thurber. It turns out that the badminton sessions are every Friday from 4.30 to 5.30 p.m. (instead of 3.30 - 4.30). Change your calendar accordingly.

Table 1. A complete set of user tasks. The author has used this set of tasks for comparative user testing of basic features in commercial calendar applications. Supercal is a fictive name.
Test users normally solve these tasks within 40-80 minutes.
Note how the wording of the tasks must be modified as time goes by. The wording shown is for a test conducted in early May 1996.
The original wording of the last paragraph in task 2 was “Transfer incoming calls to extension 7991 in the meeting room”. Tests revealed that this wording mislead a considerable number of test users to start looking for ways to make the transfer through the computer. This is not possible and was not intended by the task author. It is a typical example of a formulation problem in a task.
Tasks 2, 4, 5, 7 and 8 revealed significant usability problems in calendar applications.

Problematic task description	Problem
Play with the application for ten minutes.	Imprecise. Chances are that the user will lose interest in the application and in the test, or that he will go astray in some irrelevant corner of the application.
Enter the address of someone you know.	Better than the previous task, but still too imprecise. You have no control over what kind of address features will be tested (compare to task 3 in table 1).
You find out that you often send e-mail to everyone in your department. Therefore you want to create an ALIAS called “department”.	The wording contains at least the following hidden clues: · The application allows you to send an e-mail to several persons at the same time. · The keyword ALIAS plays a role in carrying out this task · Distribution lists are referred to by name.
Send an invoice for five hours of plumbing work to Richard Milhouse Nixon The Watergate 1972, Impeachment Ave. Anytown, Anystate	The task is silly. Some users may be offended by tasks like this because they feel that the task makes fun of their job.

Table 2. Examples of problematic user tasks.

3.2 Recruiting Test Users

Test users can be colleagues, friends, employees from other companies, people recruited through an employment agency, etc., as long as they belong to the target group for the application to be tested.

For an in-house test, test users are often selected by their managers. Managers have a natural tendency of selecting their most able employees for user testing. In requesting test users, you should therefore explicitly ask your contact persons to provide a broad sample with respect to age, experience, and seniority.

Quite often the results of the test will depend critically on the qualifications of the test user. To ensure that the test user is neither over- nor underqualified for the test, it may be advantageous to ask him a number of specific questions about his qualifications before admitting him to the user test. The questions may be written down in a questionnaire. Such questionnaires are particularly important when recruiting test users through an employment agency. Make sure that the employment agency screeners have the necessary qualifications to evaluate test user candidates.

In some cases people tend to overestimate their qualifications. Just asking “Are you an experienced Windows user?” may produce an affirmative response from a person who has used Windows for several years, but only to start up his favorite word processor. Precise, functional questions are much better, e.g. “Let us assume that you have started up the Microsoft Word word processing application in Windows. Then you have started the Microsoft Excel spreadsheet. Now tell me how you would shift from Excel to Word”. On the other hand it requires much more technical knowledge to evaluate the answers to such functional questions.

Selecting the right test users for a test may take considerable time. Be prepared to reject particpants who do not have the required qualifications for a test, or who are overqualified. Make sure that rejections are non-offensive, e.g. by stating that the prospective test user is overqualified (even if he is not), but that you will keep his name in your database for future tests.

Experience shows that test users sometimes will cancel the appointment at a very short notice. Sometimes they will not even cancel the appointment but just not show up. Be prepared for this situation. Stress how important the test is for you and do all you can to demystify the test situation for the prospective test user (see the following section). Watch out carefully for any sign of hesitation from the test user when speaking to him before the test; if he appears hesitant despite your efforts to “sell” the test, consider dropping him in favor of another test user. For very important tests - for instance tests that will be watched by management - you may want to schedule two test users. If both show up you may run the test using the constructive interaction variant of thinking aloud (see section 9).

When recruiting test users through an employment agency there will normally be a fixed fee that covers both the test user and the overhead from the employment agency. If you recruit test users yourself, you may want to consider reimbursing test users for travel, labor, or both. If you decide not to pay for the labor (for instance to avoid taxation problems) you may want to give the test user a small token of appreciation, e.g. a coffee mug. Never set the compensation so high that test users participate in a test only for the money!

3.3 Informing Test Users

After selecting the test users make sure that each test user receives a written confirmation with exact instructions about how to get to the test site, and that they get sufficient information about what is going to happen at the test. Openness and demystification are some of the keys to a successful user test. You may seek inspiration from the example in figure 1.

If the test will be videotaped or watched by others, e.g. developers, make sure that test users are informed about this in advance. There have been cases where test users refused to take part in a test when they saw the video equipment and when they discovered that others would be watching the test.

If your legal department requires test users to sign a consent form, including perhaps a non-disclosure agreement, include a copy of the form (see the example in figure 2). Make sure that it is short and written in a language that test users will understand - it should definitely not be written in legalese! Professional usability people carry out a user test of the written materials they send out to test users.

The information to the test user may include user documentation for the application to be tested. Experience shows that this will cause some test users to read the documentation even though they would not have read it under more natural circumstances. Many test users will not read the documentation in advance - just as in a natural use situation.

Dear Mr. Smith.

Thank you very much for agreeing to help XYZ Corp. to improve the usability of our applications. XYZ Corp. and our users greatly appreciate your time and effort.

Your usability test is scheduled for February 12, 1997 at 10.00 a.m. It will be finished no later then 11.30 a.m.

The test will take place at our office on 2537 Broadway, 6th floor. I enclose a map that shows you how to reach our office.

During your usability test you will be working with a new version of our Supercal calendar application in an ordinary office room. We will ask you to carry out a number of tasks with Supercal. As I told you on the phone, we will be videotaping the session. The video will be shown only to our usability specialists and to our developers of Supercal so that they may improve the product. At the end of the test I and possibly some of our developers will join you for a brief discussion about your impressions of the new Supercal.

The usability test is solely a test of our Supercal calendar application. We are not testing you in any way. You may experience difficulties in using the application during the test. But remember: If you experience difficulties, then our developers - not you - have a problem. It is their job to make applications that people like you can use.

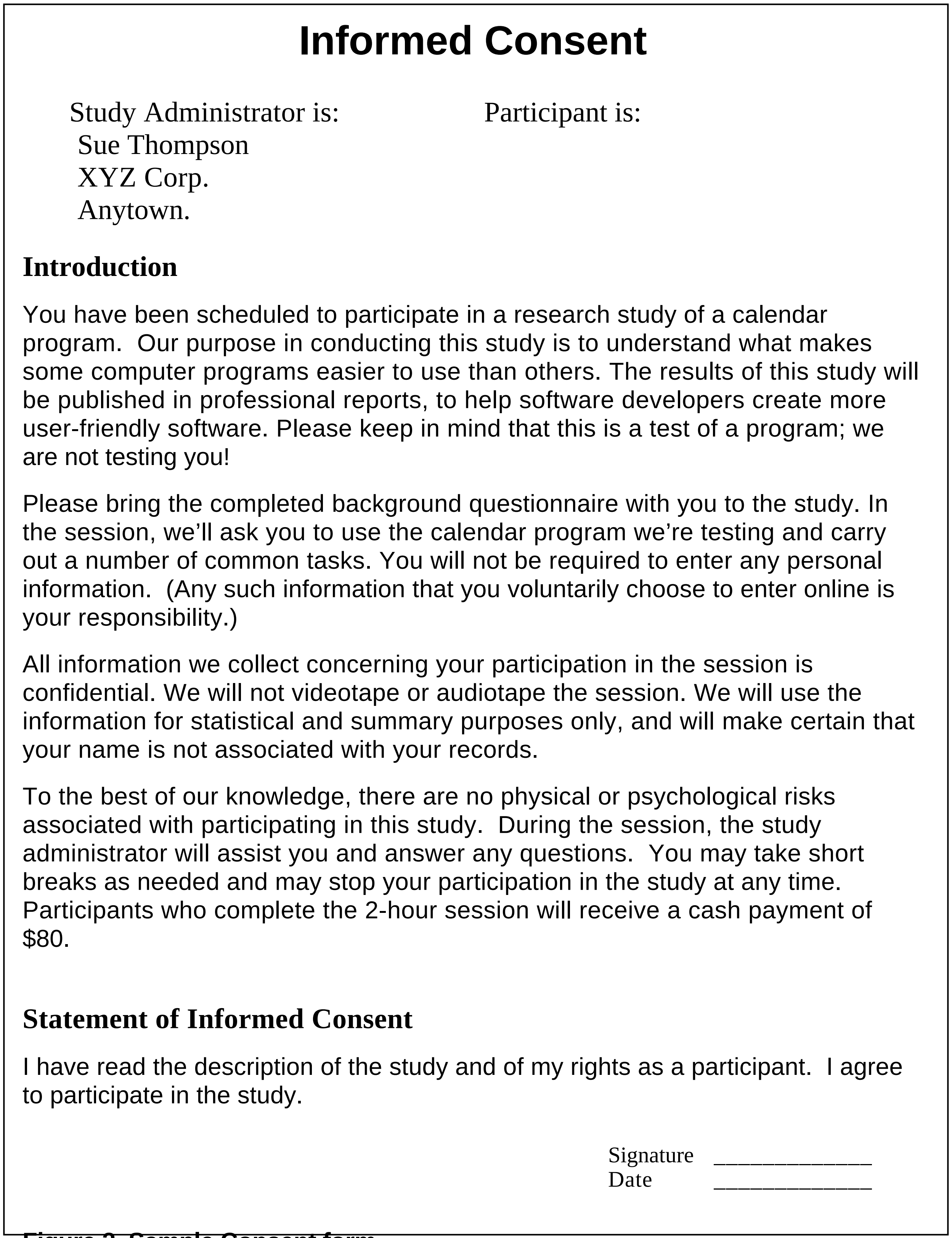
We are investing a lot of resources in this test. Therefore, if for some reason you are unable to meet the appointment, please inform us immediately on telephone 123-4567.

If you have any questions, please do not hesitate to call me on 123-4567.

Best regards,

Sue Thompson
XYZ Corp.

Figure 1. Sample Confirmation letter to test user.



Informed Consent

Study Administrator is:
Sue Thompson
XYZ Corp.
Anytown.

Participant is:

Introduction

You have been scheduled to participate in a research study of a calendar program. Our purpose in conducting this study is to understand what makes some computer programs easier to use than others. The results of this study will be published in professional reports, to help software developers create more user-friendly software. Please keep in mind that this is a test of a program; we are not testing you!

Please bring the completed background questionnaire with you to the study. In the session, we'll ask you to use the calendar program we're testing and carry out a number of common tasks. You will not be required to enter any personal information. (Any such information that you voluntarily choose to enter online is your responsibility.)

All information we collect concerning your participation in the session is confidential. We will not videotape or audiotape the session. We will use the information for statistical and summary purposes only, and will make certain that your name is not associated with your records.

To the best of our knowledge, there are no physical or psychological risks associated with participating in this study. During the session, the study administrator will assist you and answer any questions. You may take short breaks as needed and may stop your participation in the study at any time. Participants who complete the 2-hour session will receive a cash payment of $80.

Statement of Informed Consent

I have read the description of the study and of my rights as a participant. I agree to participate in the study.

Signature ______________
Date ______________

Figure 2. Sample Consent form.

3.4 Selecting a Test Location

A user test can be conducted in two different places:

- At the test user's normal workplace.
  Advantage: Natural work situation. Gives a realistic impression of how the application fits into the test user's actual working conditions. If for example the application requires the test user to concentrate, then the test will indicate whether the surroundings will actually allow the test user to concentrate (noisy environment, constant phone calls, interruptions from colleagues).
  Disadvantage: Video or audio recording may be difficult to carry out. The test user's colleagues may interfere with the solution of the tasks.
- In some location that is specially equipped for running user tests ("Usability Laboratory", see chapter 21).
  Advantage: Video or audio recording is simple to carry out.
  Disadvantage: The test user is in an unfamiliar place surrounded by alien technical equipment.

3.5 Testing the Test

A new user test should be tested in one or two pilot tests (dress rehearsals). Real users or colleagues can be used as test users in a pilot test. Some of the typical problems that a pilot test can reveal are: First task takes too long to solve; the test user does not understand the description of a task; one or more tasks are atypical; there are too many tasks.

After the pilot tests - but before the real user tests - the experimenter should critically evaluate the tasks, the test procedure and his own role in the user test. During debriefing the pilot test user may provide important help to perform this evaluation.

The problems that are found during the pilot test should of course be corrected. Try to avoid radical changes in the tasks or test procedure since that will require another pilot test. However, if the pilot test reveals a large number of problems or if it reveals problems that cause major changes in tasks and scenarios you should consider running another pilot test.

4 CONDUCTING THE TEST

During the test the test user attempts to solve the given tasks with the application while thinking aloud.

4.1 Welcoming the Test User

Right from the moment he arrives at the test site, the test user should feel important and safe.

The experimenter can make the test user feel important by ensuring that everything is ready when the test user arrives, and by welcoming him personally instead of sending a secretary to escort him from the reception to the test location. Some companies have specially designated, attractive (shadowy) parking places reserved for test users.

The experimenter can make the test user feel safe by appearing to have complete control over the user test. Everything and everyone should be ready to start when the test user arrives. Stressed programmers performing panicky last second changes to a prototype on the test user's pc when he arrives do not signal safety.

Always remember to tell the test user that you are testing the application, not the test user. Many test users explicitly state during debriefing that they appreciate that this is mentioned clearly - even if it is also stated clearly in the written confirmation that they receive before the test.

In some cases a test user may have to be instructed before he can use the application. Make sure that he receives an unbiased instruction, that is, an instruction that does not contain important, unconscious clues to using the application, based for instance on the instructor's knowledge of problems that arose during previous tests. You can minimize the risk of such hidden clues by giving the instruction either in writing, on video or through a computerized introduction.

4.2 Running the Test

The key points of the test are to make the test user feel as comfortable as possible, and to establish a work context that is as natural as possible.

You can run the test in different ways:

- You can let the test user solve his tasks without any interference from the experimenter. In this case the test user decides when to seek help, for instance by explicitly addressing the experimenter or calling a hot-line help using a phone.

Advantage: More natural work situation.

Disadvantage: In Western culture it is considered unnatural to think aloud when there is no one present to listen.

- You can let the test user solve his tasks while smalltalking with the experimenter, commenting on his progress or lack of progress. The conversation can occur directly with the experimenter sitting next to the test user, or over an intercom.

Advantages: The test user feels more comfortable. The experimenter can prompt the test user if he has difficulties in thinking aloud.

Disadvantages: Any direct communication between the test user and the experimenter may give the test user clues to solving the task. For instance, the experimenter's tone of voice often tells the test user that he is close to finishing a task. To prevent this problem the experimenter should insist that the test user explicitly signals when he considers a task to be solved, for instance by saying so explicitly, or by handing back the task description to the experimenter.

Have coffee, tea, soft drinks, beer, snacks, doughnuts, fruit etc. in the test room, and take a break if the atmosphere should become tense.

A test should run for 60-90 minutes. When you approach the predetermined time limit, stop the test as soon as the user completes a task. Never stop the test in the middle of a task; if necessary provide sufficient help so the user can finish the current task quickly. It is simpler to stop the test if you have not told the user in advance how many tasks there are; that is, consider carefully in advance whether you want to hand out all tasks to the user at once at the start of the test, or if you want to hand out the tasks one by one.

Test users often completely lose their sense of time during a test; a common reaction after a test is "Wow, did I really spend 90 minutes on this? I thought we just started five minutes ago!". Do not let this mislead you to continue the test. The user often will be stressed after a long test even if he says that he would not mind to continue.

4.3 The Role of the Experimenter

The main task of the experimenter during a test is to make sure that the test user feels comfortable.

The experimenter should be neutral and friendly. He should insist that the test user acts as if he was not present. If for instance the test user asks “May I press the F3-button?” the experimenter should respond in a neutral tone “You can try” and neither “Yes” nor “No”.

If the test user is silent for more than 30-60 seconds the experimenter may prompt him to tell about his thoughts and considerations using questions like “What are you thinking of now?” or “How do you interpret that message?” or “What do you think is happening?”.

The experimenter must be ready to assist whenever assistance is required. It makes no sense to let the test user stumble around for more than at most a few minutes in a dead end. The main principle is: Help the test user as soon as the problem is evident.

On the other hand, the most common error in user testing is helping too early. The experimenter most often feels sorry for the user, and wants to help him out. However, just like ordinary users, test users are often ingenious and after a short while they will find a way out from a situation that looks hopeless. If the experimenter helps too early, a reasonably functioning interface may be classified as faulty, and the test user can feel a little bit irritated that he was not allowed to sort out the problems on his own. Helpful remarks from the experimenter may also inadvertently contain important clues to strategies and features that the test user had not discovered or understood himself.

Who can be an experimenter? My experience is that it is difficult to answer this question clearly. It depends on the attitude.I have seen developers and super-users with little experience in usability run almost perfect user tests at the end of a two-day crash-course on usability. I have seen usability professionals violate basic test rules which invalidated the test results and (worse) made the test users feel bad. Table 3 points out a number of important attitudes that a successful experimenter shoud have.

It should be a requirement for any experimenter that he has tried to be a test user himself during an earlier user test. Only in this way will he be able to acquire the necessary humility and respect for the strong emotional forces that may be at work during a user test.

Desirable attitude	**Problematic attitude**
Direct contact with end users is important. Design questions should be resolved in close contact with users as equal partners.	"I can resolve any design question on my own because I am a user myself". "Users really don't know what they want, or what technology has to offer for them"
Frank discussions with users.	User contact should be used to sell own ideas to users.
Invite comments and criticism by making design visible early, e.g. through paper prototypes. Insist on frequent and frank reviews	Design is never written down; no reviews; criticical comments on design work are considered a personal offense.
Listen to suggestions, even if they appear unreasonable or technically impossible at the first glance	Suggestions are turned down immediately: "We already tried that but ..."
Appreciate that usability problems are found and corrected	Usability problems are mostly caused by lack of training or by incompetent users.
Seek inspiration from outside sources such as other software, experienced colleagues, or consultants. Insist that true usability requires a multi-disciplinary team.	"We can solve any problem without outside help."

Table 3. Attitudes to look for when recruiting experimenters.

4.4 Debriefing the Test User

After the user has carried out the final task, the experimenter should not just see him to the door. An informal debriefing is important, partly to answer questions from the user and partly to determine the correctness of the mental model that the user has built of the application. During debriefing the experimenter may for instance want to find out if the test user's understanding of the application model will enable him to work with parts of the application that were not covered by the user test.

The experimenter should prepare a checklist of the topics that he wants to discuss with the user during debriefing. The debriefing should take 10-20 minutes.

The purpose of the debriefing is to listen to the test user - the purpose is not to demonstrate the cleverness of the application design. Application features should be demonstrated only if the user explicitly asks for it.

At the time of debriefing the experimenter will have some idea of where the critical problems in the interface are. Focus on these problems during debriefing. In some cases the test user may have a good suggestion for solving the problem.

The test user should feel safe and appreciated during debriefing. The experimenter should treat any comment or suggestion from the user with respect even if at first glance it collides with his model of the application or with what seems to be technically possible. Repeated use of unsympathetic phrases like “We already thought of that, but ...” tend to quieten users quickly.

Debriefing is also well suited for finding out whether the test tasks are representative and well phrased. The experimenter may quantify this by asking the user to point out the two most and the two least representative tasks.

It sometimes happens that developers who have watched a user test want to ask questions to the user after the test. Of course they should be allowed to ask these questions directly to the user since the developers often have a better understanding of the application area than the experiementer. However, to avoid unpleasant incidents the experimenter should be present during the discussion and intervene immediately if feelings start to surface. It is my experience that there are developers that you may not want to allow to discuss directly with users. It requires a skilled experimenter to sort out the developers who sincerely want to improve their application from the few developers who feel offended by the user test and want to defend their design decisions.

5 COLLECTING AND ANALYZING DATA

A user test does not have to require a lot of technical equipment. It depends on the amount of data that you want to collect.

In one extreme you may collect data on the fly by just sitting next to the user taking notes of the problems that he runs into.

In the other extreme you may conduct the test in a usability lab where you videotape the user and the display screen. A computer system or a human assistant logs what happens on the display and on the keyboard.

The choice between these two techniques is yours. However, there are two important, basic considerations that you may want to make when making the choice or compromise:

- Do not collect data unless you know beforehand what kind of useful information you will extract from it. In other words: Do not use for instance video taping unless you know how to extract useful information from the videotapes in a cost-effective way.
- There are several more or less advanced data analysis techniques around. When evaluating these techniques consider this basic question: Are we better off using resources to analyze data from tests in detail with this technique, or using a comparable amount of resources to conduct another user test?

5.1 The Simple Pen-and-Paper Approach

With this approach the experimenter collects data by sitting next to the user taking notes on paper of the problems that he runs into. Immediately after each test the experimenter uses 15-30 minutes to consolidate his notes. A hint: Use hardcopies of windows for noting usability problems.

Advantages:

- The test may be conducted at the user's normal workplace where the environment is as natural as possible and where the user feels safe.

Disadvantages:

- There is little documentation afterwards of exactly what happened during the test.
- Some users are distracted when the experimenter starts writing notes during the test ("What did I do now that made him write a note?").
- Developers cannot watch the test, or see it later on video.

5.2 The Advanced Video Approach

With this approach the experimenter conducts the test in a usability lab where he videotapes the user and the display screen. A computer system or a human assistant logs what happens on the display and on the keyboard. Chapter 21 contains detailed information about how to set up and run a usability lab.

Advantages:

- Developers can watch the test using a tv-monitor or through one-way windows to the usability lab.
- Developers can see what happened during the test afterwards on video.
- It is possible to create a 5-20 minute video summary of what happened during the test to show to doubting developers. Note, however, that is is very time consuming to produce such a summary.
- The considerable investment in hardware for user testing is a tangible proof of management’s serious commitment to usability.

Disadvantages:

- The surroundings are unnatural.
- Users may be intimidated by the technical equipment.

6 COMMUNICATING TEST RESULTS TO DEVELOPERS

Designers and programmers are users, too. They are users of the test results that come out of user testing. It is important that test results are communicated well to the people who will actually implement the test results. Otherwise the results may be neglected. Building a usable application is teamwork, which among others involves analysts, designers, programmers, and usability specialists. All must work together to deliver the best possible application to users within given resouce limits.

Here are a few tricks to ensure the best possible cooperation between developers and usability specialists:

- Invite developers to come and watch the user tests. Advertise the times when user tests start. Provide free food and soft drinks (or beer, if company rules allow it) to attract as many prosepctive users of the test results as possible.
- Communicate test results quickly. It is a good idea to send out a brief summary of the most important test results in a brief e-mail immediately after the last user test. It often happens that developers who have watched one user test go back and immediately start implementing changes based on their experience from that test, which may have been atypical.
- Make the test report short. If you limit yourself to one or two pages you have a chance that some developers will actually read your report. Formal reports of more than 20 pages do not communicate well. Some developers have the attitude that "real developers do not read reports". You can write the full report for your own documentation purposes at some later time.
- Invite developers to see your equipment and hear a brief explanation of your test methods. Make a big point of the quantitative improvements in usability that you have obtained and documented.
- Give designers and developers a chance to comment on the written user test results before they are made available to others. This gives your colleagues a chance to correct misunderstandings before they are seen by outsiders.
- Point out the things that work well in the interface. Ideally you should have one positive remark for each problem you point out. This is difficult to accomplish in practice; usually there are 2-4 problems for each positive feature. Be as explicit with the positive features as you are with the problems. Avoid statements like "There are a lot of positive things to be said about this interface, but I don't have time to write them down".

After all, it would be a pity if the developers removed some of the positive features in the interface just because no one had bothered to tell them that these features actually worked well.

When planning how to communicate with your developers please remember: Your success primarily depends on the measurable improvement in the usability of your company's products that results from your user tests. In a business environment your success is not measured by the scientific correctness or the appearance or the length of your test reports.

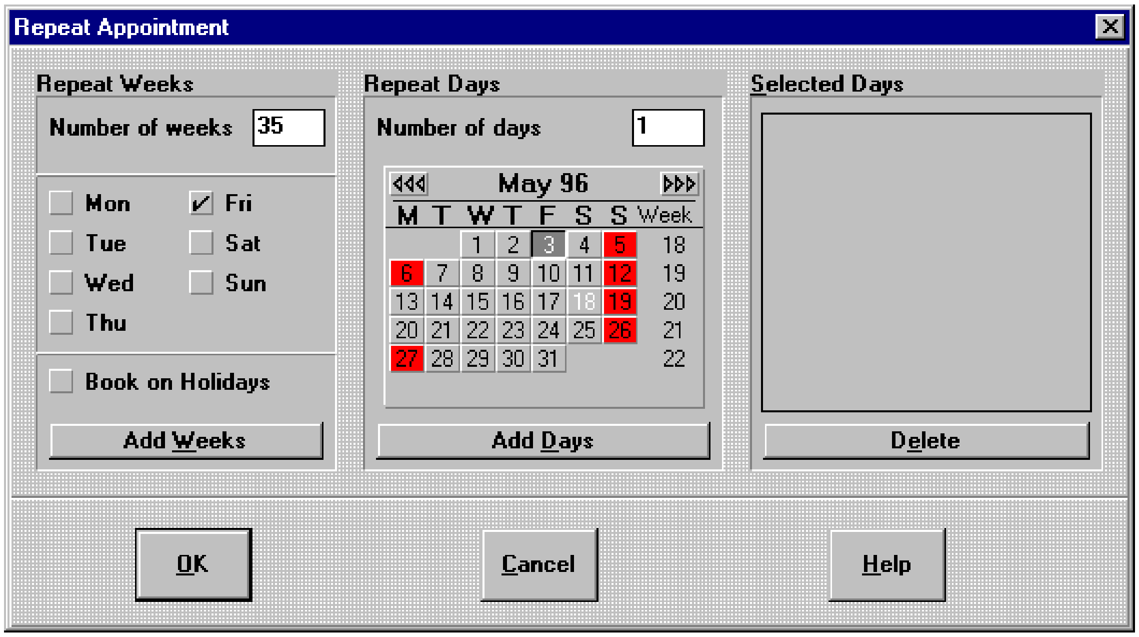
6.1 The Test Report

A full report from a user test could contain the following information:

1. Executive summary.
2. Short functional description of the tested application.
3. List of test users. Preserve a reasonable level of anonymity by providing no names. Provide brief information about sex, approximate age, professional experience and computer experience.
4. Short description of test procedure.
5. Information on how many minutes each test user worked on each task.
6. List of good features.
7. List of problems (refer also to the example in figure 3). For each problem provide:
   - Problem number for reference purposes.
   - Problem description.
   - Suggestions for solving the problem (optional).
   - Problem classification (see section 6.2).
   - How many test users encountered the problem? How many of these test users were able to solve the problem on their own, and how many required assistance from the experimenter or from the hot-line?
8. Conclusion. Has the minimum usability goal for the application been reached? Or would you recommend that the application is subjected to another usability test after the most serious problems have been corrected, even if this means the application delivery may be delayed?
9. Appendix: Scenario and tasks.
10. Appendix: Screen shots. The screen shots are important for readers who want to understand the usability problems but do not know the application well, or when re-reading the report some time after the user

test was conducted when the actual appearance of the application may have changed.

The length of a usability report strongly depends on the complexity of the application. Normally they are from 20 to 100 pages including appendices. If you find more than approximately 30 usability problems you should seriously consider to describe only the 30 most important ones in your report. Otherwise the developers may get so depressed by the report that they will not make any changes at all.



Number of weeks: 35

Mon [ ] Tue [ ] Wed [ ] Thu [ ] [✓ Fri] Sat [ ] Sun [ ]

Book on Holidays [ ]

Number of days: 1

May 96

M	T	W	T	F	S	S	Week
		1	2	3	4	5	18
6	7	8	9	10	11	12	19
13	14	15	16	17	18	19	20
20	21	22	23	24	25	26	21
27	28	29	30	31			22

The above window is taken from the commercial calendar application Task Timer for Windows version 2 (now outdated). The window appears when the user wants to make a repeated appointment, for instance in response to task 4 in table 1. User testing of this window revealed a number of usability problems. Here are two of the problem descriptions:

1. Severity 1, Critical. About half of the users filled out the “Number of weeks” field and then pressed the OK-button. The application, however, expects the user to press the Add Weeks button first, and then press the OK-button. Unfortunately, pressing the OK-button directly does not cause any warning or error message. All users were convinced that they had created the repeated appointments that the task asked for. When the experimenter asked the users to show him some of the created appointments, the users were surprised that no appointments had been created.
2. Severity 2, Serious. Often, the user knows only the starting and ending date for a repeated appointment. All users had difficulties in computing the exact number of weeks to enter in the field “Number of weeks”.

Figure 3. Sample problem descriptions.

6.2 Classification of Problems

Usability problems should be classified in at least two dimensions: Severity and Time to correct.

The following scale may be used for classifying severity:

1. Critical problem: Causes frequent disasters (see section 11.1). It is strongly recommended that this problem should be fixed before the application is released.
2. Serious problem. Causes some disasters. Should be fixed in the next release.
3. Cosmetic problem. Should be corrected when time permits.

Some problems take just a few minutes to correct, like inserting a line with a textual instruction in a window. Other problems take longer to correct. You can classify the time it takes to correct a problem roughly on the following scale:

1. Corrected in less than one to two hours.
2. Corrected in less than one to two days.
3. Corrected in less than one to two weeks.
4. Indefinite time to correct; at least two weeks.

The two scales may be combined to yield a reasonable compromise between severity and resources when determining which usability problems to correct: Add the Severity score (1-3) and the Time to correct score (1-4) and correct all problems that score less than some fixed number, e.g. 3.

7 ETHICAL CONSIDERATIONS

In a Thinking Aloud test you work with human beings. If you are not very skilled in human relations, chances are that you may inadvertently create a situation which makes the test user feel uncomfortable. The following table, which is based on considerable experience, describes a number of potential problems that you should consider carefully to ensure that both the test user and the experimenter will consider the user test a positive and rewarding experience.

Ethical problem	**Prevention**
The test user feels that he and not the application is being tested.	· Ensure that the user will experience an initial quick success. Ensure through pilot testing that the first task takes less than three minutes to carry out. · Tell the user before the test that the application, not the user, is being tested
The test user is afraid that the test results may be used against him	· Do not allow the test user's manager or colleagues to attend the test. · Carefully explain how the test result will be used, and that they are confidential.
Tasks are atypical	· Develop tasks in close cooperation with typical users. · Ask test users how relevant the tasks are.
Tasks are silly	· Avoid "funny" names of persons and places in tasks. · Avoid unusual tasks. Focus on ordinary ("boring") tasks. · Develop tasks in close cooperation with typical users
Tense atmosphere during test	· Interrupt the test to take a coffee break
The test user feels intimidated by the alien equipment	· Explain and demonstrate the purpose of the equipment · Put flowers on the desktop and tasteful paintings or reproductions on the wall · Conduct the test in a room with a window so the user can see daylight.

Table 4. Some potential ethical problems in a Thinking Aloud test, and how to prevent them.

8 DISCOUNT USER TESTING

The previous sections have described how full-blown user tests can be carried out using the Thinking Aloud method. Sometimes the term “DeLuxe usability testing” is used to describe the full-blown way of doing user testing. However, while DeLuxe usability testing often detects important interface problems, using it rigorously during design and implementation can be time consuming. It may be difficult to convince a cost conscious management that the benefit of applying DeLuxe usability testing indeed matches the cost. The problem is that conducting and analysing each user test is a time consuming and thus costly task.

Usability specialists are therefore often faced with a choice between doing more limited user tesing, or doing no user testing at all. The aim of “Discount user testing” is to obtain the best possible cost/benefit ratio. In other words, to get as much information about the usability of an application with as few resources as possible. Discount user testing permits you to discover the majority of the truly serious interface problems as quickly as possible - but with a limited risk of missing a few serious problems. Discount user testing is particularly well suited for applications that are intended for casual users; for such systems the presence of just one serious usability problem may jeopardize the success of the whole system.

A discount user test is conducted as a Thinking Aloud test at the user's ordinary workplace. Neither audio nor video recording is used. The main purpose of the test is to detect user interface disasters, which - by definition - require no analysis (see section 11.1). Therefore little or no analysis is required after the user test. Serious interface problems are recorded with paper and pencil on the fly. Other usability problems are recorded if time permits. Eliminating audio and video recording has the additional advantage of making the test situation more natural for the user.

The main differences between DeLuxe user testing and Discount user testing are summarized in table 5.

	Discount User Testing	**Deluxe User Testing**
Basic method	Thinking Aloud testing.	Thinking Aloud testing.
Main scope	Detect user interface disasters (see section 11.1)	Detect all kinds of usability problems.
Data collection equipment	Little; perhaps an audio recorder	Videotaping of test user and computer screen.
Number of test users	3-5	5-10
Test site	User's workplace	Usability lab
Can developers witness the test?	Difficult	Yes, both during the test and afterwards on video.
Data analysis	Usability problems are detected on the fly, i.e. during the user test. After the test notes are consolidated into a short report.	After the user test. Rerun videotape, possibly with test user. A highlights tape may be prepared.
Reporting	Brief report focusing on disasters, 1-5 pages. Possibly e-mail message.	Detailed report, 20-100 pages
Usability staff requirements	One person	Two persons (experimenter and logger)
Resource requirements for one series of tests	With three test users: Around 25 person hours [Molich 92].	With six test users: 100-150 person hours (one person month)

Table 5. The most important differences between Discount user testing and DeLuxe user testing.

9 OTHER USER TEST METHODS

There are a few variants of the Thinking Aloud method. The most important is Constructive Interaction. In a constructive interaction test two test users carry out the tasks together. The usability problems in the application will then appear from their conversation.

Constructive interaction makes it more natural for test users to think aloud. Carrying out a task while discussing it with another user is a natural situation for many users. In addition, most users are accustomed to learning from other users.

Constructive interaction has the disadvantage that it may be difficult to separate the different problem solving strategies applied by each of the two test users. One test user may have one perception of the system and one strategy; the other user may have a completely different perception and strategy. These strategies may be intermixed to the extent where it is difficult afterwards to determine how each test user has perceived the application.

It also sometimes happens that the two test users cannot work together. In todays highly competitive job market some users do not like that colleagues detect holes in their professional knowledge, which may happen when constructive interaction is used. It usually matters less if a stranger (the experimenter) sees such holes.

Another problem with constructive interaction is that each user test requires two users. Unless you are lucky and have easy access to test users, consider carefully what will tell you most about the usability of your application: One constructive interaction test, or two ordinary thinking aloud tests.

10 COMMON PITFALLS

Experience shows that some Thinking Aloud tests are not as successful as they could be because experimenters make errors in running the test. These errors are tricky because the experimenter will not get any indication of the error - yet the error may invalidate the results of the user test. Many of these errors can be avoided for instance by regular peer review or by hiring a skilled consultant to review the experimenter's performance.

Here are the ten most important pitfalls that I have seen in practice. Each of these problems reduces the value of a Thinking Aloud test considerably.

Problem	**Prevention**
Preparations for the test are not complete when the test user arrives	Start early; it is better that the experimenter has to wait 15 minutes for the test user than that the test user has to wait two minutes while the experimenter completes the preparations.
The first task is too difficult	The first task must be so simple that the user can solve it within at most three minutes.
Tasks are atypical or silly	Do a pilot test. During debriefing of both pilot test and ordinary test, ask the user if the tasks are typical. Ask the user to point out the two most typical and the two least typical tasks. If several users point out the same tasks as being atypical, analyse and revise those tasks.
Task descriptions contain hidden clues	Review the tasks with experienced colleagues.
The experimenter helps too early (the user test turns into a demonstration of the application)	The test user should be given a reasonable amount of time to solve the task without help. Ask the user to tell the experimenter explicitly when he would normally give up trying to find a solution by himself.
The experimenter does not help even though the interface problem is evident	Experimenters must be properly trained, possibly by an outside consultant.
The experimenter defends the application	Be careful when you are testing applications in which you hold a personal stake. This also applies when you are testing applications containing usability recommendations that you have made.
The experimenter provides unconscious help	Experimenters must be properly trained, possibly by an outside consultant.
The experimenter shows no appreciation for suggestions from the test user (e.g. “We already thought of that”)	Experimenters must be properly trained, possibly by an outside consultant. The experimenter’s attitude to users may not be optimal (see section 4.3).
No debriefing	Use at least five minutes for debriefing. Make a list of topics that you want to discuss with the user during debriefing.

Table 6. Important pitfalls in user testing.

11 PUTTING USER TESTING IN CONTEXT

Carrying out a user test will not in itself improve the usability of your products. User testing is only one small, albeit important, step in the development lifecycle.

Here are some of the reasons why even a successful user test may not make any significant difference: Often, user test results are not communicated well to the developers, who then tend to ignore them in favor of seemingly more important problems. Or, the results are communicated well, but no changes are made because the necessary resources are lacking. Or, changes are made, but they do not improve usability. Or, usability is improved but the improvements are so small that they do not justify the cost. Or, significant usability improvements are made, but they are not made visible and therefore remain unknown to customers and management.

True improvement of usability requires an iterative process where user testing alternates with usability problem correction. Several cycles are always required because usability problems are often hard to correct (solutions may turn out to be wrong), or because the correct solution to a usability problem reveals another usability problem that was previously hidden by the solved one. It is my experience that three to five test cycles are typically required before you can say with just some certainty that your application is usable.

Section 11.1 provides a measurable goal for when an application is acceptable with respect to usability. The goal may be used to end the usability test cycle. Section 11.2 provides some more detail as to how you can carry out iteration in practice.

11.1 A Usability Goal

In order to describe the usability goal, we first define that a **Serious Interface Problem** exists if one or more of the following occurs:

1. The user is unable to proceed without help from a human being, e.g. from a super-user or from a telephone consultant.
2. The user experiences annoying, irrational behavior from the application. Example: The application requires the user to perform tasks that he considers unreasonable or incomprehensible
3. There is a critical difference between what the user thinks the application does and what the application actually does.

Problems of type 1 are easily recognized when they occur. Problems of type 2 and 3 are determined by explicitly asking the user.

A serious interface problem becomes a **User Interface Disaster (UID)** if during user testing at least two motivated and typical users of the application run into it.

Serious interface problems that are encountered by only one user should normally not be considered disasters since such problems are often of a personal and temporary nature.

A usability goal could be to eliminate as many user interface disasters as resources permit. The usability goal could for instance be attained when a series of at least three user tests, each covering a number of major features of the application, does not reveal any new User Interface Disaster.

11.2 User Testing in the Development Cycle

The overall development of the application is carried out in accordance with the traditional waterfall model with usability extensions such as involving the users, coordinating all aspects of usability, and iterative design [Gould et al. 1985], [Mantei et al. 1988].

The user test cycle runs as follows:

1. Inspection: A number of usability specialists or skilled designers perform an inspection of the interface as described in chapter 22. Inspection is a cost-effective way of eliminating the most apparent usability problems. The more expensive user tests can then find the tricky usability problems, which inspection techniques overlook.
2. Functional check: A usability specialist or an experienced tester checks that the application is ready for user testing.
   The check is a traditional black-box test. Each input field on each display is tested with normal input values, extreme but permissible input values, and qualified guesses at probable, incorrect input values. In addition, each display is scanned for compliance with the relevant dialog standard and for potential usability problems. The test also includes the execution of a number of typical user tasks on the application including the ones that are later used in the user tests.
3. Selection of Test users: Three or more users are selected who have not used the application before.

4. User Testing: A usability specialist performs a user test with each of the test users selected in step 3.
   Each test ends with a debriefing, which addresses the test user's understanding (mental model) of the application in order to identify usability problems of type 2 and 3, with particular focus on critical differences between what the user thinks the application does and what the application actually does. The debriefing also addresses user suggestions for eliminating recognized UIDs.

4. Consolidation of Problem List: Problems found during the user tests are added to the problem list.
   The problem reports resulting from the tests are handled by the project team.

5. The problem list is scanned for new UIDs.
   The UID criteria defined in the previous section are applied to the problem list to see if any new UIDs can be identified. Note that a UID exists even if the two problem occurrences required to identify a UID are detected in different user test cycles.

6a. If the user tests reveal one or more new UIDs,
   then a new user test cycle is started by repeating the process from step 3 after as many usability problems as possible have been corrected by the project team.

6b. If the user tests reveal no new UIDs,
   then the usability goal is fulfilled, and from a usability viewpoint the application can be delivered to the users.

The number of users selected in step 3 depends on a cost/benefit consideration. Three users are the minimum. More users mean increased costs, but also increase the number of UIDs that will be found. Example:

· A serious interface problem that affects 50% of all users is introduced just before the final user test cycle.
· With three users in the final test cycle there is a 50% probability that this problem will be recognized as a UID (50% = 1 - $0{,}5^3$ * (1 + 3)).
· With five users in the final test cycle, the probability increases to 81% (= 1 - $0{,}5^5$ * (1 + 5)).

ACKNOWLEDGMENTS

Many of the ideas in the important section about Ethical Considerations are adapted from a chapter in [Molich 86] written by Anker Helms Jørgensen of the University of Copenhagen.

REFERENCES

GOULD, J.D., and LEWIS, C. 1985, Designing for usability: Key principles and what designers think. Communications of the ACM, 28, 300-311.

LEWIS, C. 1982, Using the 'thinking-aloud method in cognitive interface design. Research report RC-9265. Yorktown Heights, N.Y.: IBM T.J. Watson Research Center

MANTEI, M.M., AND TEOREY, T.J. 1988, Cost/benefit analysis for incorporating human factors in the software lifecycle. Communications of the ACM, 31, 428-439.

MOLICH, R. (ed.) 1986, Brugervenlige edb-systemer (in Danish; “Usable Computer Systems”). Teknisk Forlag, Copenhagen, Denmark.