3.2 How to Conduct a Think–Aloud Usability Test

This module teaches you the details of doing a think–aloud usability study. It includes pragmatic advice on all aspects of this procedure, including how to perform the following steps:

1. Define the study's framework,
2. Choose what to observe,
3. Prepare for the think–aloud usability test,
4. Introduce the participants to the observation procedure,
5. Conduct the observation,
6. Analyze the observation,
7. Find possible redesigns, and
8. Write a report.

You will learn about these steps in the following sections:

Assessments

- Exercise 7

3.2.1 Defining the Study's Framework

Before we get into the pragmatic details, we must cover a somewhat philosophical discussion about the system you are building. The development team must come to a consensus as to the purpose of the system and the usability observation, because this consensus will need to influence choices in the more pragmatic steps.

The development team should ask itself the questions listed below. Most of these questions will have been asked and answered several times earlier in the development process, but right before usability tests are conducted is a good time to ask them again—and summarize the answers.

- What problem is this system trying to solve, or what work is it trying to support? The answer to this will help in choosing the test suite of tasks.
- What level of support will the users have—in terms of training to use this system, documentation, online help, or other resources? The answer to this will help in developing materials, choosing tasks, and in setting up a realistic situation.
- What types of usage do we hope to evaluate in the usability tests? Walk–up–and–use, first–time use? Users skilled in the task domain using this particular system for the first time? Exploratory use with no time pressure or goal–directed use under time pressure? Or, other aspects? The answer to this question will influence choice of tasks, participants, and the definition of what a problem is in this system.
- What usability goals do we have for this system? That 90% of users can accomplish a simple task within 3 minutes with no training? That half of the people trained in using this system will perform with less than one recoverable error per task? The answer to this question will help in choosing tasks and defining the criteria for identifying problems and good features of the system.

For instance, in designing a test of the Date/Time control panel we might propose a framework as follows. Note that this is not the only framework that would be reasonable; it is one of several good approaches.

> The Date/Time control panel supports setting a computer's date, time, and time zone. It is particularly useful to people traveling with laptops. The control panel should require no training or online tutorial: all owners of computers should be able to use it intuitively (a walk–up–and–use situation). *Every* user should be able to complete the tasks of setting the date, time, and time zone. It is not critical that there be *no* errors committed in performing the task, but no complete task should take longer than 3 minutes.

3.2.2 Choosing What to Observe

The first thing to do after deciding to do a think–aloud usability study is to choose the tasks you want to give participants so you can observe their interaction with your system as they perform. The choice of tasks is influenced by several things: the content of the tasks, the need for training to do the tasks, the duration of the tasks, and integration of the tasks.

The Content of the Tasks

You want to pick tasks that reflect actual or expected use of the system. If a similar system is already in place (even a paper–based system), actual tasks can be determined by observing what people already do with the existing system. You may be able to identify many real tasks through observation, interview, or sometimes data collected for other purposes (e.g., a study trying to predict the usability of a new telephone operator workstation looked at routinely–collected billing records for current telephone operators to see what tasks were done and with what frequencies (Gray, John & Atwood, 1993). You will then need to pick several tasks from the many you identify and include them in the test suite.

You usually want to include the most frequent tasks in the test. This is so you get data on how your system supports the tasks users will do on the most regular basis.

You also want to include tasks that cover the range of functionality of the system, to exercise every part of it. If you do not do this, then the entire system is not tested, only the subset represented in the tasks you chose. Make sure you include not only tasks that create things from scratch but also delete or modify existing things. It may also be desirable to include error–recovery tasks (i.e., put the person in a situation where they would be had they made an error and ask them to recover from it).

You may want to include in the test suite very important, though possibly infrequent, tasks. For instance, emergency procedures are (hopefully) used infrequently, but should be tested because they are safety–critical.

The Need for Training to Do the Tasks

If the tasks you choose for the reasons stated above require training to accomplish them, then the training itself must be counted as part of the test suite of tasks. The set becomes "Learn to do Task A" and "Task A," instead of just "Task A." This allows you to test training materials as well as the system, be they paper–based reference materials, Web pages, lectures, or online tutorials.

The Duration of the Tasks

Typically, users cannot be asked to sit in one place doing tasks for more than an hour at a time, or for more than two hours on a single day. More than this is usually too tiring for participants. You need to design your test suite to allow for hourly breaks and a break for the day after two hours. This duration includes the training to do the task. If the training is on one day and the task performance is on the next, you will have to review the training on the second day to refresh the user's memory.

The Integration of Small Tasks

It is important to include tasks in the test suite that are long enough to require the integration of several system features. For instance, do not just have small tasks that create a new object, but start the user out with a system full of objects already created and make them navigate through those objects, modify them, and create new ones, too.

You may also want to include tasks that test the integration of your system with other systems they might use. For instance, how might the users include the results of something your system did into a report written in a word processor? How might they include those results in email? On a Web page? How might someone send them information through email that needs to be used in your system?

Example: Choosing Tasks to Test the Date/Time Control Panel

Given the framework we defined in the last section, we see that the Date/Time control panel can only support a very limited set of tasks.

> The Date & Time control panel supports setting the time, date, and time zone of a person's computer.

Again, from the framework of the previous section, we see that the usability target in this case will not require user training and that the tasks will last no more than a few minutes (the usability target in the framework called for three–minute tasks). These facts make it reasonable to require participants to carry out all tasks in one single test session.

Finally, the self–contained nature of the Date/Time control panel allows for no tasks of any significance that would integrate the use of the control panel with that of other applications.

The suite of tasks to use in testing the usability of the Date/Time control panel, then, would include the following:

- Set the time
- Set the date
- Set the time zone

3.2.3 Preparing for the Observation

Before running any actual think–aloud sessions, there are several preparations you need to make. These include:

- Setting Up a Realistic Situation for Data Collection
- Writing Up the Task Scenarios
- Practicing the Session
- Recruiting Users
- Example: Preparing for a Think–Aloud Usability Test of the Date/Time Control Panel

Setting Up a Realistic Situation for Data Collection

You need to set up as realistic a situation for data collection as you can. This situation will differ depending on the system you are testing. If your system is a desktop system for a PC, an office–like set–up is fine. This is what has traditionally been the set–up for usability labs in large corporations like Microsoft and Sun.

However, with the advent of many more types of computer systems, realistic set–ups take on new characteristics. Testing the interface to a microwave oven is best done in a kitchen–like setting. Testing a PDA (a "personal digital assistant" such as an iPAQ Pocket PC or Palm Handheld) can be done almost anywhere, but you need to pay attention to lighting conditions that will be found in offices, in cars, outdoors, and on public transportation. Navigation devices to be placed in cars need to be tested while driving (or in driving simulators). The variety is endless today with the ubiquity of computer systems. In general, think carefully about how your system will eventually be used and set up the situations (possibly plural!) to match those uses as closely as possible.

These situations pose challenges to the data–collection process. In the traditional usability lab, fixed video cameras and microphones could be positioned to capture the actions and voice of the users. In the more mobile situations, more dynamic data–capturing techniques must be employed. You may need wireless microphones and you may need someone to operate a camcorder and follow the user around as they work. Some new software is available to capture and playback the user's actions and voice directly with a portable computer. Since each situation has its own requirements, the solutions will vary. However, at least you will want to capture: what the user does with your system (including observations about what is being displayed to your user) and what the user says as the user thinks aloud, synchronizing what they say with what they do.

When you record the user's actions, you will need to have a record of the passage of time during the session. If you are recording on video, some video recorders can put a timestamp on the signal (visible either on the recording or on the player device). If you are using screen–capture software, you ought to have a system clock within the area of the screen the software is capturing.

Writing Up the Task Scenarios

In the previous section, we spoke about how to choose tasks to test; now, they need to be written into statements of tasks for the participants to perform. Usually, each task is written on a separate sheet of paper and given to the user one task at a time. Depending on the task, these descriptions are either purely verbal, or they include pictures or diagrams to help describe the task. In addition to being described individually, a series of tasks usually is accompanied by a cover story that links the tasks in the series into a meaningful narrative.

Practicing the Session

Do not underestimate the difficulty of making a session run smoothly. A saying among experimental psychologists applies to usability testing: "Subjects are like pancakes; you always have to throw the first few away." That is, it takes a while to get everything to work in concert. The hardware and software must perform without crashing. The written instructions must be clear, or you will see problems comprehending the instructions instead of problems with the system you are testing. Your delivery of verbal instructions must be smooth; otherwise, you will confuse the participant. The data–capture equipment must work. The solution to all these problems is practice, Practice, PRACTICE.

Practice first with yourself, which can at least debug the procedures of what software must be running when, what windows must be open, etc. Walk through the entire session, reading every piece of information, doing every task. Write everything down in a script so you can reproduce the session the next time you run it. Make sure your script includes reminders for things you should do, e.g., to make sure there is a tape in the video camera or that the sound level is adequate. If you find problems with your procedure when you walk through it yourself, fix those problems before bringing in anyone else to practice on.

Practice next with a friend: different eyes will see things you did not catch yourself in your instructions, tasks, or equipment set–up. If your friend has enough time to spare, again, walk through everything in the entire session. Again, fix any problems your friend finds before the next practice session.

Practice finally with someone who has a similar background to the users you will be using in your study. By this third session, you might expect all the software and hardware to work, but don't be surprised if yet another set of hands and eyes finds something else wrong! This is an opportunity for you to practice your delivery of the instructions and to determine whether the instructions themselves are comprehensible to someone who is much like your users.

Since your purpose in conducting these tests is to find problems with your computer system, if you uncover any problems while running these tests of your procedures, write them up in UARs. It is not "cheating" to discover system problems during this preparatory phase of the testing. Don't throw away valuable information just because you weren't expecting to get any during the preparation phase.

Recruiting Users

Just as different systems require different situations in order to provide a realistic test, so will different systems require different types of users to provide generalizable data. The most important consideration is that the participants in your usability tests have the same background knowledge as your eventual users will have. Thus, if your system is for airline pilots to program the destinations of their aircraft, you need to use airline pilots as participants in the usability tests—if current airline pilots are not available, retired airline pilots would be the next best thing. Educational software should be tested by students in the grade where the software fits into the curriculum. Medical systems should be tested by doctors or nurses (whoever will actually be using it). And, the Disney Company tests their new virtual reality games with visitors to their existing theme parks. Obviously, some users are more difficult to obtain than others, but it is crucially important to get at least a handful of participants who represent the actual user group.

You may need to compensate users for their participation. This can be anything from giving them a company T–shirt to promising them a copy of the final report, to actually paying them what their time is worth. For example, if your company designs medical systems for doctors' use, you may need to pay the doctor the equivalent of their consulting fee. Make sure the compensation is clear when you recruit the users.

Depending on the complexity of your system, you will need at least three or four participants to give you a broad test of the system. Each user will uncover different problems, though some problems will be so prevalent that they will bother several users.

A good plan is to do iterative testing. That is, test two or three people, then fix the big problems they find (while preserving the parts that gave them no trouble). Then test two or three more to uncover the next batch of problems. This way, when you inevitably run out of development time, you have progressed through at least one or two iterations and the product has been improved at least a couple of times.

Example: Preparing for a Think–Aloud Usability Test of the Date/Time Control Panel

Since setting the time zone is most appropriate for people traveling with laptops, a realistic hardware setting is to use a laptop. We chose to use a screen–capture program that would allow us to record both the user's interactions with the system and their voice. We made sure that the system clock appeared on the bottom menu bar of the screen, so we would always have a record of the time of the interaction.

We wrote up two of the tasks as follows, one on each page. We did not do the third task, setting the date. You might try that on your friends for practice!

> Imagine the following scenario:
>
> You are a new reporter for the Pittsburgh Post Gazette. There has been some recent unrest in the Philippines, and you have volunteered to go to Manila to cover the story. You are waiting to board your flight at Pittsburgh International Airport.
>
> You have a few minutes to spare before your flight. Using the Date–Time control panel on your laptop computer, adjust the time zone to the correct one for your destination.

We composed a script for the test, which you will see and hear in the next section. Since we were using these recordings for course materials, as well as using them to find usability problems in the Date/Time control panel, our script had to be a little different from what you would normally use: we stated specifically that the recordings would be used in a class. We also composed a consent form specific to this testing session (given below), which also mentioned the classroom use of the recordings. Finally, the analyst rehearsed the script six times before collecting any actual data.

Any potential owner of a computer could be a participant in our study. Thus, we decided to ask participants about their computer background. Questions about computer use were written into the script. We decided to run the observations at the airport, with travelers who had quite some time to wait before their next flights.

Carnegie Mellon University and iCarnegie, Inc.

User Study Participation Consent Form

I agree to participate in the user study conducted by Professor Bonnie E. John or by students or staff under the supervision of Professor Bonnie E. John. I understand that all observations have been reviewed by Professor John and that to the best of her ability she has determined that the

observations involve no invasion of my rights of privacy, nor do they incorporate any procedure or requirements that may be found morally or ethically objectionable. If at any time I wish to terminate my participation in a study, I have the right to do so without penalty. I further have the right to contact the following persons and report my objections, either orally or in writing to:

Bonnie E. John
Associate Professor
School of Computer Science
Carnegie Mellon University
Pittsburgh PA 15213
(999) 999–9999

Allan Fisher
Director of Academic Affairs
iCarnegie, Inc.
4615 Forbes Ave.
Pittsburgh PA 15213
(999) 999–9999

Purpose of the Usability Study

I understand that I will be trying out a piece of software. I understand that the staff is trying to find out how easy the software is to use. They will be recording my voice and my interactions with the computer while I work so they can look for places where the software might be difficult to use.

I understand that portions of this recording may be used in educational material to teach students how to conduct usability studies and evaluate software.

I understand that the following procedure will be used to maintain anonymity in analysis and reporting. Each participant will be assigned a number. The staff will save the recording by participant number, not by name. I understand that my name will not be revealed in any use of these recordings.

I understand that in signing this consent form, I give iCarnegie, Inc. permission to use these recordings as they see fit. In exchange for the sum of $5.00, I assign all copyrights to these recordings to iCarnegie, Inc.

I am at least eighteen (18) years old.

_____
Signature

_____
Name (Please print)

_____
Date

3.2.4 Introducing Participants to the Procedure

When you and your test participant first meet to do the observation, you must give the participant enough information for him or her to feel comfortable doing the think–aloud so that you can get the highest quality data possible. To set the participant at ease, you will want to describe the purpose of the study, to explain how to think aloud, to provide time to practice the technique, and to explain the rules of the observation.

Describe the Purpose of the Study in General Terms

Give a brief introduction to the entire study with the following elements.

- Introduce yourself (name and title).
- Introduce your organization and its purpose (e.g., XYZ Corp. and you develop information kiosks for the tourist industry).
- State the goal of the particular study (e.g., to test a new kiosk for Pittsburgh's Zoo).
- State that you are testing the computer system; you are *not* testing the user.
- Tell them that their participation is purely voluntary and that it is OK if they want to stop at any time.
- Give the participant the consent form, give them time to read it quietly, and ask for their signature. (This is a good time for you to make sure the computer system is set up correctly for training them in how to think–aloud.)
- Show the participant the equipment in the room, explain what each piece does, and demonstrate the ones they will have to use. This is especially important if there are novel devices in the system, e.g., pointing devices other than a mouse.
- Show the participant how you are going to record their voice and actions. If you are using a video camera, show it to them. If you have a separate microphone, show them how it works. If you are using screen– and audio–capture software, explain that the computer itself will be doing the recording. After doing this, ask them if it is OK to start recording now, and, if so, start the recording device. (If they say "no," ask them what else they need to know, and answer any questions. If they persist in not wanting to be recorded, tell them it is a requirement of being in the study that their actions and voice be recorded. Excuse them if they continue to refuse.)

Train the User to "Think Aloud"

The participants are going to have to be trained to think–aloud as they work. This is a bit awkward at first, so you will have to give them practice doing this on tasks other than the ones you want to observe with your system.

The following instructions, adapted from Ericsson and Simon's *Protocol Analysis* (Revised edition, 1996), have been used successfully to elicit good think–aloud behavior with users. You should adapt these instructions to your task situation, script it, and read it almost verbatim to your participants.

> In this observation, we are interested in what you think about as you perform the tasks we are asking you to do. In order to do this, I am going to ask you to think aloud as you work on the task. What I mean by "think aloud" is that I want you to tell me *everything* you are thinking from the first time you see the statement of the task until you finish the task. I would like you to talk aloud *constantly* from the time I give you the task until you have completed it. I don't want you to try to plan out what you say or try to explain to me what you are saying. Just act as if you are alone, speaking to yourself. It is most important that you keep talking. If you are silent for any long period of time, I will ask you to talk. Do you understand what I want you to do?
>
> Good. Now we will begin with some practice problems. First, I will demonstrate by thinking aloud while I solve a simple problem: How many windows are there in my mother's house? <demonstrate thinking aloud>
>
> Now it's your turn. Please think aloud as you name 20 types of animals.
>
> Good. Now, those problems were solved all in our heads. However, when you are working on the computer, you'll also be looking for things, and seeing things that catch your attention. These things that you are searching for and things that you see are as important for our observation as thoughts you are thinking from memory, so please verbalize these too. For example, listen to the types of things I say as I think–aloud when I disable the screen saver on this computer.
>
> Now it's your turn to think aloud as you are using the computer. Please think aloud as you set the background color to a deep purple.

If during the practice session, the participant stops talking for 5 to 10 seconds (even 10 seconds will sound like a looooooong silence), say "Please keep talking." Do not elaborate on this simple phrase. That is, don't say "What are you thinking" or "Please explain what you are doing" because phrases other than the neutral "Please keep talking" tend to encourage people to switch to Type 3 verbalizations (explaining or filtering what they are thinking), which we discussed earlier as untrustworthy (see "3.1.1 What is Think–Aloud Usability Testing?").

Explain the Rules of the Observation

After the participant is comfortable thinking aloud, you are almost ready to move on to the observation phase of the usability test. Just before doing that, explain the "rules" of the observation.

- You will not be able to answer questions during the observation.

- They should ask their questions anyway. You'll record them and answer them at the end of the observation.
- If they fall silent for any length of time, you will ask them to keep talking.

After explaining these rules, ask the participant if he or she has any questions about the think–aloud procedure or anything else so far.

Example: Introducing Participants to a Think–Aloud Usability Test of the Date/Time Control Panel

In preparation for our test of the Date/Time control panel, we incorporated all the information into a script to which the analyst adhered when testing the software. We have included the script below.

Following the script are several recordings that capture our delivery of this introductory material to a typical user. (Note that this particular user was *not* about to test the Date/Time control panel, but an application called *RealJukebox*: However, the introductory material and script were the same in both cases.)

As with all informal observations, the analyst may not have adhered perfectly to the script. See if you can find instances, in the recordings, of deviations or omissions from the script (we give you some examples in what follows). Typically, experienced analysts are able to follow a script closely without appearing to be unnatural or to lack spontaneity.

Script of the Introductory Material

The actual script is enclosed in quotation marks. Instructions to the analyst about what to do while interacting with the participant are in *italics. Introduce myself.*

> "Hello. My name is <analyst's name>. I'm a master's student at Carnegie Mellon University, and I am conducting a study that looks at how people interact with computers. Do you mind if I talk with you for a few minutes?"

If the person indicates they are interested, sit down.

> "I'm at the airport today talking to people who have about half an hour to spare and are willing to allow me to observe them while you interact with some software that I have here on my laptop. I am offering $5 to people for their time. Are you interested?"

If they are still interested, give the information that is also present in the consent form.

> "I work for a non–profit company affiliated with Carnegie Mellon University. We design courses that teach people how to develop software. I'm here today to record people's interactions with software they are unfamiliar with, to be used as educational material in a new course, which will teach programmers how to design software that is more usable.
>
> "I am interested in finding out how easy the software is for people to use. If you agree to participate, you will be using this laptop computer, and the computer will automatically record your voice and your actions on the screen as you use the software. The recordings will be used later to identify places where the software is difficult to use, and to teach students how to conduct usability studies and evaluate software.

"If we decide to use part of your recording as material for the course, your name will not appear in the recording. In fact, your name will not be associated with the recording in any way."

Give them the consent form.

"I have this consent form here, which the University requires me to use. If you sign it, it means you are giving my employer permission to use your recording as part of their course material, and it tells you whom to contact if you want to report any objections. I'll give you two copies—one is for you to keep, and the other is for you to sign and return to me."

Screen them for computer knowledge.

"Before we begin, I have a couple of questions to ask you about how familiar you are with using computers, so I can choose an appropriate task for you to work on with the software.

- Do you have a computer at home?
- Have you ever used Microsoft Windows?
- How often do you use a computer? Every day, once a week, once a month?
- In the last couple of <time> for what kinds of things did you use computers?
- Have you ever used <software>?"

Study the instructions.

"The point of today's study is to discover where people have problems with software that is supposed to be "walk up and use"—it does not come with a user's manual to tell a new user what to do. I'm going to ask you to perform some tasks with <whatever software this person will be using>.

"I'm testing the software; I'm not testing you. I'm looking for places where the software might be difficult to use, so if you can't do some things please don't feel bad. That is exactly what we are looking for.

"Remember, this is completely voluntary. Although I don't know why this would happen, if you become uncomfortable in any way feel free to stop.

"In this observation, I am interested in what you think about as you perform the tasks I will be asking you to do. I'm going to ask that you 'think aloud' while you are using the software. What I mean by 'think aloud' is that I want you to tell me ***everything*** that you are thinking from the first time that you see the statement of the task until you finish the task. I would like you to talk aloud ***constantly*** from the time I give you the task until you have completed it. I don't want you to try to plan out what you say or try to explain to me what you are saying. Just act as if you are alone, speaking to yourself—just a little louder."

Instruct them on how to think aloud: doing a non–computer task.

"Let me demonstrate thinking–aloud for you as I try to multiply 42 x 22 in my head. <Illustrate thinking aloud>.

"Now, you try thinking aloud. Here's a problem: please think aloud while you answer the question, 'How many windows are there in your mother's house?'

"Good!"

Instruct them on how to think aloud: disabling the screen saver.

"Now, those problems were solved entirely in our heads. However, when you are working on the computer, you'll also be looking for things, and seeing things that catch your attention. These things you are searching for and things that you see are as important for our observation as thoughts you are thinking from memory, so please verbalize these too. For example, listen to the types of things I say as I think–aloud when I disable the screen saver on this computer.

<Illustrate thinking aloud>.

"Now it's your turn to think aloud as you are using the computer. Please think aloud as you set the background color to blue.

"Great! I'm just going to give you some final instructions now."

Give them some final instructions.

"As you're doing the tasks, I won't be able to answer any questions. But, if you do have questions, go ahead and ask them anyway so that I can learn more about what kinds of questions the <software> brings up. I'll answer your questions after the session. Also, if you forget to think aloud, I'll say, 'Please keep talking.'

"Do you have any questions about thinking aloud?

"Now, I have some tasks printed out for you. I'm going to go over them with you and see if you have any questions before we start."

Hand them the task and ask if they have any questions.

"Here is the task you will be working on. Why don't you read it aloud, just so you can get comfortable with speaking your thoughts?

"Do you have any questions about the task?"

Tell them they may begin.

"You may begin."

Recordings of Participant Briefings

1. The first recording is of the analyst describing the purpose of the study in general terms. Refer to that subsection of this module, the script above, and the consent from, as you view it. Think about what the analyst is doing according to the procedures presented in this module and what she is doing that deviates from those procedures.

    - ♦ Click Introduction & Consent to view the recording.
    - ♦ After you have viewed the recording, click Intro Critique for our critique of the recording.

2. The second recording is of the analyst introducing the think–aloud technique. Refer to the relevant previous subsection, and to the script above, as you view it. Think about how the analyst is following the procedures presented in this section and what she is doing that deviates from those procedures.

- ♦ Click How–to–TA to view the recording.
- ♦ After you have viewed the recording, click TA Critique for our critique of the recording.

3. The third recording is of the analyst demonstrating the think–aloud technique and giving the participant a chance to practice. Refer to that subsection of this section and the script above as you view it. Think about where the analyst follows and deviates from the procedures presented in this section.

   - ♦ Click Practice to view the recording.
   - ♦ After you have viewed the recording, click Practice Critique for our critique of the recording.

4. The last recording is of the analyst explaining the rules of the observation and giving the participant the task. Refer to that subsection of this section and the script above, as you view the recording. Think about where the analyst follows and deviates from the procedures presented in this section.

   - ♦ Click Rules & Tasks to view the recording.
   - ♦ After you have viewed the recording, click Rules–&–Tasks Critique for our critique of the recording.

3.2.5 Conducting the Observation

After you have introduced the participant to the general purpose and procedures of the study and have trained him or her in thinking–aloud, you are ready to conduct the observation.

Introduce the Observation Phase

First, describe the system you are going to test. Tell the participant as much about the system as you expect real users will know before they are going to use the system. That is, if they will only have seen it through trade–magazine advertising or TV commercials, give them that level of overview. If they will have gone through an hour's training, this is the time to do that training.

Then tell the participant about the tasks you wish him or her to do. If there is more than one task, it is best to do one at a time rather than describe all of them and then set the user free to do them all on their own. Have the tasks written on a sheet of paper, which you will give to the participant, but also tell them about it verbally so they get the information twice. If diagrams or pictures might make the task easier to understand, take the time to prepare these ahead of time and walk through them with the user.

At this point, ask the participant if she or he has any questions about the goals of the study, the procedures, the product, or the task. Answer any questions that clarify what the user has to do, but not any that solve some of the problems they might encounter in doing the task. That is, don't answer questions like "How do I do that task?" Instead, if the participant asks such a question, say, "That's exactly the sort of thing I need to observe. I would like to see how the system helps you figure that out."

Begin the Observation

After all questions are answered, check that your recording devices are still working, and let the user work on the tasks while thinking aloud. If you are conducting the test in a usability lab, then you can probably leave the room and still monitor their progress from another room. If you are running the test in the field (or any environment where progress cannot be monitored if you leave the area), then just move out of the user's line of sight and sit quietly.

As the user performs the task, actively monitor their progress. If they fall silent for 5 or 10 seconds, say, "Please keep talking. Do not elaborate on this simple phrase. That is, don't say, "What are you thinking?" or "Please explain what you are doing" because phrases other than the neutral "Please keep talking" tend to encourage people to switch to Type 3 verbalizations (explaining or filtering what they are thinking), which we discussed earlier as untrustworthy (See Section 3.1).

If the user slips into explaining their procedures rather than just reporting the contents of working memory, then stop him or her and say, "Please don't try to explain to me what you are doing. Just act as if you are alone, speaking to yourself as you solve the problem." This usually works well to get a user simply to report their thoughts.

Be sensitive to a severe desire to quit on the user's part. If the user gets very upset performing the task, he or she may or may not want to tell you directly they want to quit. However, if the user starts saying things like

"I'm so stupid, I'll NEVER finish this" or "This seems like it will go on forever, stupid &%@#$* computer!!!", then gently ask "Would you like to stop now?" Do *not* wait for tears or for the user to throw the equipment across the room!

In the introduction phase, you told the user you would not be able to answer questions about the product while they were working through the tasks, but they should speak their questions aloud and you would answer them later. Therefore, jot down any questions you hear them ask so you can answer them if they are still pertinent after the tasks are over.

Conclude the Observation

When the user has finished all the tasks, go to your notes and answer any questions you jotted down that are still relevant (the user might have answered many of the questions themselves in the course of doing the task). Ask the participant if he or she has any more questions about the system, the study, or the organization. Answer any questions you can right then, or put the user in touch with someone who can answer the questions.

Ask the user if he or she has any opinions about the product they just tested. Ask if they have any suggestions about what the company could do to improve the product. Although the think–aloud test will yield more reliable data, it is a good idea to ask the user these questions, because (1) it gives them a chance to express their opinion, which people often want to do and, (2) users some times have great ideas that you can only collect by asking.

Thank the participant for participating. Reiterate that their participation will help you identify problems with the system so your company can fix them. Give the participant whatever compensation you had promised them when they were recruited, or arrange for the compensation to be sent to them (e.g., fill out any necessary paperwork to pay them or get their address to send them a copy of the final report). Thank the participant again as you part company.

Example: Conducting an Observation of the Date/Time Control Panel

In conducting an observation of the Date/Time control panel, we decided not to give any training or introduction other than the name of the panel. This was to simulate how real users would be introduced to this control panel; it would just come installed on their machine, ostensibly without help or documentation.

We gave the descriptions of the tasks to the participant, each task on a separate sheet of paper (as shown in "3.2.3 Preparing for the Observation"). We have included a recording of a new participant performing a think–aloud on the Date/Time control panel. To give you another example of a good practice session, this recording also includes the practice of think–aloud when solving a computer–based task, much like that we have seen in recordings included within the previous section. Refer to this module and to the task write–ups from "3.2.3 Preparing for the Observation," as you view the recording. Think about how the analyst follows and deviates from the procedures presented in the course material. We will analyze this recording in the following section.

Click the following link to view the DateTime3 recording.

Both the practice session setting the background color of the display and the Date/Time tasks are great think–aloud sessions. The participant starts by reading the task aloud, but also reporting his thoughts about the task as he reads it (e.g., "*That's a long set up for a straight–forward problem*"). The participant talks constantly, never pausing long enough for the analyst to have to remind him to talk. He is reporting his

thoughts, not explaining them, except in one place, where he summarizes the problem he faces (when he says, "*So basically the problem here is, that I can see the map, but I don't know where it is on the map. And I can see the names of these cities, but none of them is ringing any bells.*"). But, even in this event, he immediately goes back to straight reporting, and the analyst does not have to correct him.

At one point, the participant says, "*I give up*" but then he continues to look for an answer, so the analyst correctly lets him continue for some time. He tries some ideas, summarizes the problem, tries some more, and, finally pauses for a length of time and gives the analyst an imploring look (which is not visible on this recording). The analyst is sensitive to the participant's frustration, and decides to suggest that the participant move on to the next task. He gratefully accepts and continues to "think–aloud" very well.

To demonstrate a conclusion for a study, we have included another recording of a different participant. Here is the script the analyst used to conduct and conclude this observation.

During the data–collection phase, if they stop talking say:

"Please keep talking."

If they start explaining rather than reporting their thoughts, say:

"Please don't try to explain what you are thinking. Just act as if you are alone, speaking to yourself—just a little louder."

After the tasks are complete, conclude the observation.

"Thank you very much, that was great."

"Let me answer the questions you had during the session."

Answer the questions the participant voiced as he or she worked.

"Do you have any additional questions you'd like to ask about the product, our company, or this observation?"

Answer the questions or make a note to contact a person who can answer the questions.

"Do you have any opinions or suggestions to make about the product you tested?"

Make a note of the participant's opinions or suggestions.

"Thank you again for your help. Here is your five dollars. Here is also the URL of a page where you can download the software you just tested in case you'd like to continue using it. Thank you very much. Good–bye."

Refer to the previous material in this section as well as to this script as you view the recording. Think about where the analyst is following and diverging from the procedures presented in the course material.

- Click the following link to view the Conclusion recording.
- After you have viewed the recording, click this link to hear our critique.

The recordings we have provided in this module represent quite good Type–2 verbalizations (see "3.1.1 What is Think–Aloud Usability Testing?" for the discussion of the Type 1, Type 2, and Type 3 verbalizations). For contrast, we have included an example of a think–aloud session where the participant took it upon herself to *explain* her thoughts rather than merely report them (despite hearing the explicit instructions you are now familiar with). The analyst should have interrupted the participant to ask her not to explain (as the script earlier requires), but failed to do so. When you view this recording, notice how much slower and less natural the process seems.

Click the following link to view the DateTime2 recording.

3.2.6 Analyzing the Observation

Having collected think–aloud usability data from several participants, you now must analyze the data to find good features that you want to preserve in future versions of your system, problems with the system you want to fix, and possible solutions to those problems. To do that, you will first establish criteria for what observable behaviors indicate critical incidents pertaining to usability, study all such incidents in the recorded behaviors, and then describe them in usability aspect reports (UARs).

Establish Criteria for Critical Incidents

It is important to think hard about what behaviors should be considered critical incidents for the system you are designing. You must ask, "What is a problem for this system?" Note, this question doesn't ask, "What are the problems?", but instead more generally, "What is a problem?" Likewise, you must ask yourself what observable behaviors indicate that a feature is so well designed that it should be preserved in future redesigns.

Remember from the definitions of critical incident analysis (see "3.1.1 What is Think–Aloud Usability Testing?") that critical incidents "are defined as extreme behavior, either outstandingly effective or ineffective with respect to attaining the general aims of the activity" (Flanagan, p. 338). Thus, not everything a user does will be "critical"; not everything is worth thinking whether it should be fixed or preserved in the next version of the software.

Different design situations will generate different criteria for criticality. For example, in a walk–up–and–use kiosk at the Olympics, a criterion for identifying a problem might be that a user has to try three different actions (e.g., clicking on three different buttons) before finding the action that actually accomplishes the user's goal. This limit–of–three criteria may seem rather harsh, because, after all, the participant in the usability test *did* accomplish the goal, but when you think about the eventual real–world use of the system, it seems more reasonable. A user at the Olympics will probably be more in a hurry than the user in your test, and going down three dead–ends may frustrate the real–world user so much that he or she will not use the kiosk again. Also, going down three dead–ends will take time, making the lines at kiosks longer than they would be if those dead–ends could be eliminated. On the other hand, if you were designing a data–analysis tool for highly trained engineers, trying three different things may not be as problematic. As long as you have also tested whether the engineers could find the correct action easily if they used the help system or documentation, it is their choice whether to explore the functionality of the system.

Some criteria that are useful for systems and are easily prototyped in Visual Basic are listed in the following table. You should also list your own criteria for problems and good features before you view your recorded data and include these in a similar table, using it as a reference as you review the data. Also, a good rule of thumb is to have no more than about 10 criteria, because keeping more than that in mind as you review the recordings can get unmanageable.

Possible Criteria for Problems	Possible Criteria for Good Features
The user articulates a goal and cannot succeed in attaining that goal within three minutes (then the experimenter steps in and shows the user what to do).	The user describes something as a positive effect or says that something is really easy.
The user articulates a goal, tries several things, and then explicitly gives up.	The user expresses happy surprise.
The user articulates a goal and has to try more than three things to find the solution.	Some previous analysis (e.g., a Heuristic Evaluation) has predicted a usability problem, but this user has no difficulty with that aspect of the system.
The user does not succeed in a task. That is, there is a difference between what you asked the user to do in the task and what they actually did.	
The user expresses distressed surprise.	
The user describes something as a negative effect or says that something is a problem.	
The user makes a design suggestion.	

View the Recorded Behavior and Write UARs

As you view the recorded behavior and you hear evidence that an aspect of the interface has met one of the criteria, start a new UAR as you did with HE UARs. That is, give it a unique UAR identifier and state whether it is a problem or a good feature. Then tentatively give it a short name (you might find that after you fill in the entire UAR, you understand more about the incident and want to change this short name). As with any UAR, when you are finished writing it up, examine the other UARs you have and find relationships, if any.

Evidence for the Aspect

Evidence for a critical incident in a think–aloud usability test includes the ***actual behavior*** of the participant. That is, it includes what the user ***said*** and what the user ***did*** (typed words, clicked on buttons, etc.). You should also include what the user could have seen, that is, what was on the screen at the time of the incident. (Note, that something was on the screen does NOT in itself mean that the user actually noticed it.) Thus, "evidence" includes "the facts, just the facts," NOT an interpretation of the facts—because there may be more than one interpretation of the same facts. Interpretation will appear in the "explanation" slot of the UAR, not in the "evidence" slot. Make sure you include a pointer to that part of the recording so you can replay this incident anytime you need to see it again (e.g., if some development team member asks to see the evidence). This pointer can be a tape–counter for a videotape or a time (if there is a time stamp on the videotape) or the time shown on a clock that was displayed on a computer screen.

Remember from the definitions of critical incident analysis that a critical incident is complete enough to permit inferences about the person performing the act—that is, what they knew, what they paid attention to, how they approached a problem. Therefore, your evidence is going to have to include enough context to permit those inferences. Further, both the purpose of the user's actions and the consequences of those actions should be fairly clear.

Typically, your evidence will start with the user's statement of a goal. It is much easier to understand how the interface supported or failed to support a user if you have evidence indicating what the user was trying to do!

Often a user will tell you what he or she is trying to accomplish by saying something like "I wanna find..." or "Now, let's try to..." or "Gotta go do..." Sometimes this goal statement will occur well before the critical incident—but you'll include it in your evidence section even if it isn't contiguous with the incident itself, because it sets the framework for the complete activity. This framework is necessary to understand the incident.

Sometimes the user will not explicitly voice the goal, but other actions can be considered evidence for that goal. For instance, if the user picks up the written task sheet you gave her, rereads a portion of it, puts it down, and then starts moving the mouse through menu items—it is not a bad assumption that the user's goal is to accomplish the part of the task she just read. Therefore, the act of reading the task description would be evidence for that goal.

Sometimes the system sets the user's goal. For instance, if the system presents a moded dialog box that the user must dismiss before she can do anything else, the only goal the user *can* have at the moment she discovers that the dialog box is blocking progress on anything else is to dismiss the dialog box. Therefore, the appearance of a dialog box and the fact that it is moded should be recorded as evidence for the user's goal.

Always include the effects of the user's actions in the evidence slot. These are usually screen shots or descriptions of what happened on the screen and with the system. Sometimes user actions have side effects that have no visible effect on the screen. If so, your evidence must include a factual description of those side effects. Sometimes the consequence of a user's actions will not be seen until much later in the recording. If so, the evidence slot must include the evidence for this effect, even if it is not contiguous with the incident itself. Thus, just as the goal may be stated minutes before the incident, its effect may not be realized until minutes afterwards. The evidence you include in the report must be complete enough to describe both the goal and the effects, no matter how separated they are by time.

In this section, we have discussed only written or graphic evidence of critical incidents. However, as video technology improves, it will become increasingly easy to include actual clips of the recording in on–line UARs. Paper UARs will continue to be needed for many reasons (e.g., easier to carry and hand out copies at a meeting)—so even when you have the capability to include dynamic evidence, consider using it as a back–up to static evidence.

Explanation of the Aspect

In a UAR written from a think–aloud usability test, the explanation is your hypothesis about what the user was seeing, interpreting, understanding, or guessing as he or she was performing the acts you record in the evidence slot. You take your inspiration for this explanation from the observed behavior you include in the evidence slot, your understanding of the user's background knowledge, and your understanding of how the system actually works. You should make sure your explanation is consistent both with the evidence and with how the system actually works. This explanation is what will inspire, in turn, the possible fixes for any problems you find.

Sometimes there will be more than one plausible explanation for the evidence. In that case, record all the plausible explanations in the explanation slot. When you are proposing solutions for a problem, and different explanations point to different solutions, you must then look for more evidence that either confirms or disconfirms the alternative explanations. Sometimes you will have to try a few more users, but on a shorter, more focused task, just gather evidence that will distinguish between these explanations. Luckily, many times a single solution will cover all of the alternative explanations, and you will not need to do more investigation.

Severity of the Problem or Benefit of the Good Feature

In a UAR written from a think–aloud usability test, the severity can be related to the criteria used to identify the incident as critical. That is, the user explicitly giving up on a task is probably more severe than the user expressing distressed surprise and moving on. Just as there are no standard criteria for identifying critical incidents because what counts as critical varies from one design situation to the next, what counts as severe also varies with the design situation. You will have to establish criteria for measuring severity as each design situation dictates.

Possible Solution, Including Possible Trade–Offs

In general, a solution to a usability problem comes from supporting the user's goals more directly. That is one reason why it is so important to try to find evidence for the user's goals. Ask yourself if the goal the user articulated is supported by the system, and, if not, why not? Is it a reasonable goal, and, if so, how can you support it? If it isn't a reasonable goal, ask yourself what it was about the system that guided the user to form an unreasonable goal. Asking yourself these questions in light of a problematic critical incident can lead to inspiration for a solution.

Sometimes users will generate solutions themselves (e.g., one of the criteria listed above is if the user makes a design suggestion). Record these suggestions and give them consideration, much as you would any design suggestion from the development team. But, do not give them more importance just because they come from a user—users are notorious for not knowing what features would actually serve them in the long run. Any suggestion arising in a usability test is likely to be a reaction to a local difficulty and may not take into account enough understanding of the whole system to be truly insightful.

Example: Analyze the Observations of the Date/Time Control Panel

We analyzed for you the think–aloud recordings that you viewed in "3.2.5 Conducting the Observation." We used the criteria for identifying problems and good features presented in the earlier table.

We identified three critical incidents on which three UARs are based, two problems and one good feature. These UARs are presented below, and again in "Appendix A. Date/Time Control Panel UARs." You will notice that the UARs for the think–aloud seem to be bigger in scope than the UARs resulting from Heuristic Evaluation. This is because the critical incident technique requires incidents to be "complete," which means that they encompass the setting of a goal, the problem–solving required to accomplish that goal, and its final resolution (whether it is success, abandonment, or a change of goals). It is natural that think–aloud UARs will be "bigger" than HE UARs because HE UARs can identify problems in the small steps towards a goal (e.g., understanding a label, or searching through a cluttered screen), whereas TA UARs require a complete chain of events towards a goal. As we will see in "3.3.1 Comparing Heuristic Evaluation with Think–Aloud Usability Testing," there is often evidence supporting the HE UARs *within* the TA UARs, but they will not be identical in scope.

Here you can find three Think–Aloud UARs generated from observation of the critical incidents apparent in the recordings you viewed in "3.2.5 Conducting the Observation":

- TA UAR 1
- TA UAR 2
- TA UAR 3

3.2.7 Finding Possible Redesigns

After writing up UARs from several usability tests, it is time to step back and generate ideas about how to redesign the system to fix any problems you have found. The first step is to try to find UARs that relate to each other—in order that several problems might be solved with a common redesign and that solving one problem won't make another problem worse or destroy a good usability aspect.

Relate Different Usability Aspects

As you look across the UARs, look for users' similar goals. They can be the same goal that arose for different users or under different circumstances, or slightly different goals that arose with the same user (e.g., the goal to delete a file and the goal to delete a folder are similar goals that one user might have). Goals can often be expressed as an action on an object. For example, deleting a file and deleting a folder are both expressed as an action (delete) on an object (file or folder). "Similar" goals usually share an action or an object. Look for repeating the same action on different objects, like deleting `file1`, then deleting `file2`, `file3`, and `folderA`, which contained them. Likewise, a string of different actions on the same object may relate several UARs, like copying `file1` to another folder, then deleting `file1` (which has the effect of moving the file). Such strings of related goals often signal a larger goal that is not being supported well in the system.

Another clue to related aspects is offered by a system feature about which many problem UARs have been written. For instance, if the user has many problems with the spelling checker, perhaps the operation of that entire feature should be rethought.

Often an application that was designed to be used by itself is in reality used in conjunction with other applications on a single computer. This integration with other applications (or lack thereof) is often a source of usability problems. Therefore, examine UARs that relate to other parts of the computer system outside the system you are designing (of course, the tasks you gave the user must have afforded an opportunity to work outside the system, or such problems will not surface).

There are many other ways that UARs can relate to each other that will depend very heavily on the particular system you are designing. Finding these commonalities is a place for your problem–solving talents to come into play. Always stop to think about how to relate the UARs you found, because these patterns have the potential to inspire radically better designs. But, don't get bogged down in trying to impose patterns where they do not jump to mind (e.g., don't spend more than a day thinking about this). Sometimes there really are no big patterns, and the right thing to do is just to go about fixing the little things.

Determine Possible Solutions

As with the solutions for individual UARs, the solution to most usability problems comes from supporting the user's goals more directly. If your examination of the set of UARs suggests that the system's support for a larger (and possibly unarticulated) user goal seems to be problematic, then ask yourself the same questions you asked concerning individual UARs. Is this larger goal intended to be supported by the system, and, if not, why not? Is it a reasonable goal, and, if so, how can you support it? If it isn't a reasonable goal, what is it about the system that guided the user to form an unreasonable goal?

If a set of UARs clusters around a specific system feature, ask yourself what user goal the feature was designed to support. Is this a reasonable goal? Was there any direct evidence that users would form that goal (i.e., did anyone articulate that goal)? If not, what other goals did the users express in place of that goal? Also, if they *didn't* voice the goal the feature was designed to support, what goals were they working on when they *did* use that feature?

If there are problem UARs concerning the integration of your system with other applications on the computer, make sure you list the goals the users had when they ran into those problems. These goals are likely to be only a subset of what people will actually want to do with your system once it is deployed—so use this list as a starting point, and generate other possible integration requirements.

Besides the general idea that the system redesign should support users' goals more directly, there are no hard and fast rules about how to generate new designs. This is where your problem–solving and creative talents should be brought to bear. Use the UARs and their relationships as inspiration for a redesign, then check your ideas with a quick Heuristic Evaluation to make sure you are not violating heuristics (unless they conflict—then refer back to "2.2.2 HE: Consistency and Standards"). Of course, as time allows, prototype the new design and iteratively user test it again.

Example: Looking For Possible Redesigns of the Date/Time Control Panel

It is not very fruitful to find relationships between the think–aloud UARs we saw in "3.2.6 Analyzing the Observation" because there are only three of them from this short think–aloud usability test of this small application. TA2 and TA3 testify to the difficulty experienced by participants in finding a city or country that is not on the ListBox control within the Time zone tab. TA1 is an unrelated good feature praising the time zone label on the tab. If the application were more complex, or if we had more users in our usability test, it might be more fruitful to look for relationships. To allow us to demonstrate the power of UAR relationship analysis, we consider here the three think–aloud UARs in combination with the 28 UARs based on Heuristic Evaluation that we developed in "Unit 2. Interfaces: Creating with Visual Basic, Evaluating with Usability Heuristics." The combined set can be found in "Appendix A. Date/Time Control Panel UARs."

We proceeded by printing these 31 UARs and spreading them out on a big table (the floor might work just as well.) We then scanned them and put them into piles that seemed to go together, making copies as necessary when a UAR seemed to belong in more than one pile. This sort of loose grouping activity is sometimes referred to as building an "affinity diagram."

The Date/Time control panel UARs in question seemed to fall into two clusters: UARs pertaining to the "OK", "Apply", and "Cancel" buttons, and those pertaining to setting the time zone. The rest seemed to be isolated problems. We recorded these relationships in the *Relationships* slot at the bottom of each UAR, which you can see in the completed set in "Appendix A. Date/Time Control Panel UARs."

We then looked at each of the two clusters in turn. We looked at the *good–feature* UARs and at the preliminary solutions recorded in their *Solution/Tradeoff* slots. In the context of the whole UAR set, we then looked to see if any of those solutions satisfied all needs. In this case, the solution suggested for HE8 (suggested again in HE14) appeared to solve all the problems without destroying the good features and had no obvious negative side effects. We thus wrote a UAR to resolve the main issue of this UAR cluster: the behavior of the "OK", "Apply", and "Cancel" buttons. We gave this UAR the name HE17.

Turning next to the second cluster of UARs that were related to setting the time zone, we scanned them again. And, when we did, we discovered that one user goal that is not supported by this interface is the finding of an

obscure city or country when the user's knowledge of geography (in long term memory) is not sufficient to identify the place in a map. We judged this to be a reasonable goal, especially because current events in several continents have recently changed radically the geography of many countries. There is ample evidence in UARs TA2 and TA3 that our user had this goal in the absence of sufficient geographic knowledge.

Looking at the solution slots of the UARs already generated, we see several solutions that might address this new user goal. For example, HE20 suggests making sure that every time–zone *name* is represented (for example, Mid–Atlantic, Eastern Europe). HE21 and HE27 suggest having a search function and a bigger database. HE28 (the aggregate UAR of HE2, 18–21, 25–7) proposes the searchable database plus time zone lines on the map and the ability to click on the map to narrow the search to approximately that area. TA2 suggests rollover country and city names to let the user explore the map, and TA3 details an interaction with that interface. There is no lack of possible solutions here.

The next step is to examine these possible solutions, and any more you can invent and suggest, and see which of them solve the most of the related problems without destroying the good features. Combining solutions is also a possibility. For the time zone setting problem, we favor combining the solutions in HE28 and TA2 (detailed in TA3). That is,

- Label the map with time zone lines.
- Provide rollovers of cities and countries.
- Allow the user to click on the map to select a city or country to narrow the search.
- Provide a larger database.
- Allow searching.

The exact details of the resulting overall interaction are still to be worked out by the entire development team.

3.2.8 Writing a Summarizing Report

Summarizing Reports

Sometimes, a good set of UARs is all the documentation that is required. This might be the case, for example, if you are evaluating the few usability problems of a very small application, or if only two or three developers are working on the application. In such cases, everyone involved will be able to keep in mind all the usability issues as they design and write program code. However, such cases are the exception rather than the rule. Most real applications are large, involve the participation of many people, and require documentation beyond UARs. In most cases, you will need to write a report that communicates with clarity and brevity the results of the usability analysis.

The sad fact is that almost no one will read all the UARs you prepare on the usability tests you conduct. Developers will ask you to tell them just what to do or what the problems are, and they *won't* want a list of all 100–plus problems, but the equivalent of a "top ten list." Therefore, you will have to prepare an "executive summary" of your findings—which will include a reference pointing to the volume where the complete set of UARs resides. This summary should rarely be more than two or three pages long.

The content of the report depends to a large extent on the results of your usability tests. If you found one or several patterns of problems with the system that indicate a conceptual redesign is in order, it is best to concentrate on explaining those key conceptual changes and to justify them with examples from your data.

If you found no radical conceptual problems, but many smaller problems, it is best to present a ranked list. The ranking should include both how severe you've estimated the problem to be and the effort you believe it would take to fix the problem. Rank first in order of decreasing severity, with three levels of severity as follows:

- This usability aspect MUST be fixed before the product is shipped, or the product will be unusable for the customers.
  Comment: This category should be reserved for uncontroversial problems like system crashes or aspects that actually prevent the users from accomplishing their tasks.

- This usability problem SHOULD be fixed—even if it means expending some resources to do so.
  Comment: This category should include those problems that slow the user down but do not prevent goals from being accomplished, or for those that, evidence suggests, users will find particularly annoying.

- It is DESIRABLE that this usability problem be fixed before the product is released—but only if the effort to do so is trivial (that is, if the fix costs practically nothing).
  Comment: This category should include aspects that are mildly annoying and aspects that were surprising to the users but didn't seem to slow them down or prevent them from accomplishing tasks.

Practically speaking, it is very difficult to predict severity, so applying a scale that is finer than these three categories is not likely to be worth the effort. Also, for each item in the list, cite the specific UARs where the development team can find more details about possible solutions and design trade–offs. Within a level of

severity, list the problems in order of increasing cost, so that the least expensive problems appear higher on the list and, therefore, are fixed first: this way, more can be fixed before the product is shipped.

It is often compelling to include a "highlights" videotape to support your report. This tape should include specific, targeted segments from the usability tests for each of your conclusions. Including snippets of several users having the *same problem* is a good way to get the point across. Sometimes the development team can be convinced of the severity of a problem by watching the painful experience of an obviously intelligent and motivated user failing time and time again to discover the correct path to achieve a goal with the system. Videotapes are especially useful in organizations where the results of usability tests are new to the development process (and less useful in organizations that are already convinced of the worth of usability tests).

Example: A Summarizing Report for the Date/Time Control Panel

If the think–aloud data were all you had to report on, a summarizing report would not be necessary, because the three UARs themselves would be sufficiently short for everyone to read. So, for illustrative purposes again, we will include the 28 UARs that came from the Heuristic Evaluation and the process of finding possible redesigns as well. A report follows.

Report on Evaluation and Testing of the Date/Time Control Panel
A Heuristic Evaluation and a Think–Aloud Usability Test were conducted on the Date/Time Control Panel, resulting in 31 Usability Aspect Reports (Appendix A), and the following conclusions.
Remove the ListBox as the mechanism for setting the time zone. The current ListBox mechanism for setting the time zone is not usable and MUST be changed. It is too limiting to restrict the user to a small number of countries or cities, and it does not support users who do not know sufficient geography (which we judge to be a major fraction of potential users). If it is not changed, many users will not be able to complete the time–zone–setting task for a city or country not on the list. A solution should include more information about time zones in the control panel and interactivity of the map. For instance, • Label the map with time zone lines. • Provide rollovers of cities and countries. • Allow the user to click on the map to select a city or country to narrow the search. • Provide a larger database. • Allow searching of that database. The exact details of the total interaction are still to be worked out by the entire development team.
The behavior of OK, Apply, and Cancel buttons SHOULD be changed slightly. Change the behavior of all three buttons so that it gives correct feedback about what actions are available or not. Change the "help" information to be accurate about what each button does. The current design violates several heuristics of visibility of system status and at least one user displayed confusion about these functions.

Condition	Appearance of Buttons and Effect They Have
A change has been made to the property sheet since it was opened or since the last time Apply was clicked.	**OK** – black (available), makes changes and closes the window **Apply** – black (available), makes changes and leaves the window open **Cancel** – black (available), removes changes and closes the window
No changes have been made to the property sheet since it was opened or the last time Apply was clicked.	**OK** – black (available), closes the window **Apply** – gray (not available), no effect **Cancel** – gray (not available), no effect

Help Boxes

It is DESIRABLE that one other problem be fixed: Allow help boxes to stay up while the user interacts with the control panel. This way, the help they have requested is available to them while they are interacting; they do not have to memorize it.