Usability Inspection Methods

Edited by

JAKOB NIELSEN
SunSoft

and

ROBERT L. MACK
IBM T. J. Watson Research Center

John Wiley & Sons, Inc.
New York • Chichester • Brisbane • Toronto • Singapore
1994

see book chapter (Lewis & Rieman)

Chapter 5

The Cognitive Walkthrough Method: A Practitioner's Guide

Cathleen Wharton

John Rieman

Clayton Lewis

Peter Polson

University of Colorado at Boulder

The cognitive walkthrough (Lewis et al. 1990; Polson et al. 1992a) is a usability inspection method that focuses on evaluating a design for ease of learning, particularly by exploration. This focus is motivated by the observation that many users prefer to learn software by exploration (Carroll and Rosson 1987; Fischer 1991). Instead of investing time for comprehensive formal training when a software package is first acquired, users prefer to learn about its functionality while they work at their usual tasks, acquiring knowledge of how to use new features only when their work actually requires them. This incremental approach to learning ensures that the cost of learning a new feature is in part determined by the feature's immediate benefit to the user.

The target audience for this chapter is practicing software developers. The goals of the chapter are: to provide a detailed description of how to actually do a cognitive walkthrough, to show how the cognitive walkthrough fits into the development process, and to summarize experiences and evaluations of the method. Readers primarily interested in the details of the theoretical rationale underlying the walkthrough are referred to Polson et al. (1992a).

1. Define inputs to the walkthrough
   Identification of the users
   Sample tasks for evaluation
   Action sequences for completing the tasks
   Description or implementation of the interface
2. Convene the analysts
3. Walk through the action sequences for each task
   Tell a credible story, considering . . .
   Will the user try to achieve right effect?
   Will the user notice that the correct action is available?
   Will the user associate the correct action with the effect that user is trying to achieve?
   If the correct action is performed, will the user see that progress is being made toward solution of the task?
4. Record Critical Information
   User knowledge requirements
   Assumptions about the user population
   Notes about side issues and design changes
   The credible success story
5. Revise the interface to fix the problems

Table 5.1 *Overview of the cognitive walkthrough process.*

5.1 Overview

Brief Description of the Walkthrough Process

The cognitive walkthrough has the same basic organization and rationale as other types of design walkthroughs, such as requirements walkthroughs and code walkthroughs (Yourdon 1989). It is a review process in which the author of one aspect of a design presents a proposed design to a group of peers. The peers then evaluate the solution using criteria appropriate to the design issues.

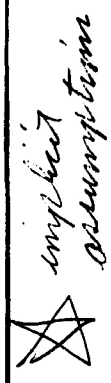
In the cognitive walkthrough, the reviewers evaluate a proposed interface in the context of one or more specific user tasks. The input to a walkthrough session includes an interface's detailed design description (perhaps in the form of a paper mock-up or a working prototype), a task scenario, explicit assumptions about the user population and the context of use, and a sequence of actions that a user should successfully perform to complete the designated task. An overview of the process is presented in Table 5.1.

During the walkthrough process the group considers, in sequence, each of the user actions needed to accomplish the task. For each action, the analysts try to tell a story about a typical user's interaction with the interface. They ask what the user would be trying to do at this point and what actions the interface makes available. If the interface design is a good one, the user's intentions should cause that person to select the appropriate action. Following the action, the interface should present clear feedback indicating that progress is being made toward completing the task.

Scope and Limitations of Method

Cognitive walkthroughs focus on just one attribute of usability, ease of learning. Theories of skill acquisition (e.g., Anderson 1987) predict that facilitating learning by exploration will facilitate skill acquisition. Other attributes of usability like functionality and ease of use are correlated with ease of learning. For example, if an application's functionality is a poor match to a user's needs, and the user is required to perform arcane sequences of actions to complete tasks, then that system will be difficult to learn to use.

Use of the cognitive walkthrough as the only method for evaluating an interface would push design trade-offs in an interface in the direction of ease of learning. For example, the walkthrough process would give a negative evaluation to features intended to enhance productivity if these features make it harder to decide how to perform a task.



Cognitive walkthroughs evaluate each step necessary to perform a task, attempting to uncover design errors that would interfere with learning by exploration. The method finds mismatches between users' and designers' conceptualization of a task, poor choices of wording for menu titles and button labels, and inadequate feedback about the consequences of an action. The procedure uncovers implicit or explicit assumptions made by developers about users' knowledge of the task and the interface conventions. The evaluation procedure takes the form of a series of questions asked about each step in the task that are derived from a theory of learning by exploration (Polson et al. 1992a).

Later on in this chapter, we review some comparisons and evaluations of the cognitive walkthrough method as related to alternative methods (Cuomo and Bowen 1992; Desurvire et al. 1992; Karat et al. 1992; Wharton et al. 1992; Jeffries et al. 1991). (See also Mack and Montaniz, Chapter 12; Kahn and

Prail, Chapter 6; Karat, Chapter 8; Desurvire, Chapter 7; and Bias, Chapter 3 for related discussions.) These studies show that the method is narrowly focused. We argue that this is a trade-off. In the interest of acquiring a great deal of information about ease of learning, the method sacrifices obtaining valid information about other important usability attributes such as global consistency or the relative ease with which a user might be able to make catastrophic errors. All methods have their own strengths and weaknesses, and several methods will need to be used by the developer/evaluator to ensure good interface coverage. The methods presented in this volume complement each other and collectively serve to provide a reasonable suite of usability inspection tools.

5.2 *Detailed Description of the Walkthrough Procedure*

The cognitive walkthrough analysis has two phases: a preparatory phase and an analysis phase. In the preparatory phase, the analysts agree on the input conditions for the walkthrough: the tasks, action sequences for each task, user population, and the interface that will be subjected to analysis. The main analytical work takes place during the second phase, during which the analysts work through each action of every task being analyzed. The details of both phases, in particular the recording requirements, depend heavily on how the walkthrough is to be used within the evaluators' own development process.

How the Walkthrough Fits into the Development Process

Formally, the cognitive walkthrough is a usability inspection method to evaluate a design for ease of learning by exploration. It can be performed after specification of a relatively detailed design of the user interface, which occurs after requirements analysis and definition of functionality of an application. A walkthrough can also be performed on a paper simulation of the interface, or on a minimal prototype constructed with HyperCard, Visual Basic, Toolbook, or other similar tool, or on a full functioning prototype of the design.

The walkthrough can be an individual or group process. For a group evaluation, the designer presents the design to a group of peers, typically after an intermediate milestone such as prototype creation, and then uses the feedback to modify or strengthen the next revision. The peers may include other designers, software engineers, and representatives from other organizational units such as marketing, documentation, and training development organizations. Additionally, in attendance may be an interface evaluation specialist. Each member of the evaluation team has a specified role: One person should assume the role of recorder or scribe, another act as facilitator, and the rest contribute various kinds of expertise, such as knowledge of the potential market, user needs analyses, and so forth. Everyone is generally responsible for carrying out the evaluation.

An individual may also use the cognitive walkthrough to evaluate a personal design. Since the walkthrough is based on an explicit model of the process of learning by exploration, developers participating in their own (or other group) walkthroughs also have the opportunity to internalize knowledge of the processes associated with the underlying theoretical model. This knowledge can then influence the later design decisions that a developer makes when designing new or follow-on products. Thus, the evaluation processes can be used at the earliest phases of the design cycle by individual designers, developers, or groups of designers.

The cognitive walkthrough can have a beneficial impact on all phases of the design and development process. Marketing studies and related inputs to a requirements analysis can be influenced by the resulting interface evaluation. Too, the selection and evaluation of core user tasks will be of use when benchmark tasks are later selected, tested, and advertised.

Defining the Inputs to the Walkthrough

Before the walkthrough analysis begins, four areas must be agreed upon.

Who will be the users of the system? This may be a simple, general description, such as, "people who use existing ATM machines." But the walkthrough may be more revealing if the description includes specific background experience or technical knowledge that could influence users as they attempt to deal with a new interface. For example, users might be "Macintosh users who have worked with MacPaint." The users' knowledge of the task and of the interface should both be considered.

What task (or tasks) will be analyzed? The walkthrough involves detailed analyses of a suite of tasks. It is possible to do an analysis of all important tasks for a system with simple functionality, such as a basic voice mail system or other consumer-oriented, walk-up-and-use applications. Too, for existing systems, a single task may be evaluated, such as one that has proven problematic in a previous release. In general, for systems of any complexity, the analyses should be limited to a reasonable but representative collection of benchmark tasks.

A critical question is how to select these representative tasks. Task selection should be based on the results of marketing studies, needs analyses, concept testing, and requirements analyses. Some benchmark tasks should be sampled from the core functionality of the application, that is, the basic operations that the system is intended to support. In addition, some tasks should be included that require combinations of these basic functions.

The benchmark tasks should be made as concrete and realistic as possible. The task descriptions must include the necessary context, such as the contents of databases that users are expected to be using. This context should reflect typical conditions under which the systems will be applied. For example, in a database retrieval task, the sample database should be large or small with respect to the expected use of the system. More discussion regarding these and other related recommendations for task selection, complexity, variants, identical subtasks, task/application boundaries, and overall interface coverage can be found in Wharton et al. (1992).

What is the correct action sequence for each task and how is it described? For each task, there must be a description of how the user is expected to view the task before learning the interface. There must also be a description of the sequence of actions that should accomplish the task with the current definition of the interface. These actions may be simple movements, such as, "press the RETURN key" or "Move cursor to 'File' menu." Or, they may be sequences of several simple actions that a typical user could execute as a block such as, "login to the system" for experienced UNIX users, or "Select 'Save' from 'File' menu for experienced Macintosh users. The decision as to what level of action granularity is appropriate depends primarily on the level of expertise of the expected users. One rough guideline is that the actions should be described at the same level as a successful prompt or effective

tutorial. Another guideline is that reasonable collections of similar keystrokes, such as those used to input a filename, be considered as a single action (Wharton et al. 1992).

How is the interface defined? The definition of the interface must describe the prompts preceding every action required to accomplish the tasks being analyzed, as well as the reactions of the interface to each of these actions. If the interface has been implemented, all information is available from the implementation. Earlier in the development process, the evaluation can be performed with a paper description of the interface. However, some important features of the system will be difficult to appraise, such as response time, color distinctions, timing of a speech interface, and physical interactions.

For a paper description, the level of detail in defining the interface will depend on the expertise that the anticipated users have with existing systems. For example, in preparing to analyze a Macintosh application intended for experienced Mac users, there would be no need to provide a detailed description of the appearance of the standard Mac menus; a simple listing of their contents would suffice.

Walking Through the Actions

The analysis phase of the walkthrough consists of examining each action in the solution path and attempting to tell a credible story as to why the expected users would choose that action. Credible stories are based on assumptions about the user's background knowledge and goals, and on an understanding of the problem-solving process that enables a user to guess the correct action. (Note that in earlier versions of the method, as discussed further on page 127, there was not the explicit use of credible and failure stories as are presented in this version.)

The problem-solving process is described by Polson and Lewis' CE+ theory of exploratory learning (Polson and Lewis 1990). In brief, that problem-solving process holds that users: (1) start with a rough description of the task they want to accomplish, (2) explore the interface and select actions they think will accomplish the task (or some part of it), (3) observe the interface reactions to see if their actions had the desired effect, and (4) determine what action to take next.

The theory also notes some specific heuristics that users apply when making their decisions. In particular, users often follow a "label-following" strategy, which leads them to select an action if the label for that action matches the task description (Engelbeck 1986). For instance, a user with the task of "print a document" might select an action with the label "print" or "document" (or a printer or document icon).

The critical features of the interface, then, are those that provide links between the user's task description and the correct action, and those that provide feedback indicating that the previous action advanced the user's progress. As the walkthrough proceeds, the analysts apply this theory as they tell and evaluate their story of why a user will choose the correct action at each step. In particular, the analysts ask the following four questions:

- Will the users try to achieve the right effect? (For example, maybe their task is to print a document, but the first thing they have to do is select a printer. Will they know that they should be trying to get a printer selected?)
- Will the user notice that the correct action is available? (If the action is to select from a visible menu, there is no problem. But if it's to triple-click the printer icon, the user may never think of it.)
- Will the user associate the correct action with the effect trying to be achieved? (If there's a menu item that says, "select printer," things will go smoothly; not so if the menu says "SysP.")
- If the correct action is performed, will the user see that progress is being made toward solution of the task? (If after selecting the printer a dialog box states that the "Printer is Laser in Room 105," great. The worst case is *no* feedback.)

These questions are loose guidelines, the meaning of which will be clear in the context of examples (success and failure stories) provided later in the chapter. However, it should be emphasized that the four questions are not absolute requirements. They are criteria that the analysts should consider in attempting to produce a credible story of the interaction.

Capturing Critical Information during the Evaluation

While performing the evaluation, it is important to capture information in a means that will allow the group to perform the evaluation most efficiently and effectively. A variety of recording means have been designed for and used

during this process, most of which have been used with earlier versions of the method. For example, the evaluation session can be recorded on videotape (Rowley and Rhoades 1992), electronically (Rieman et al. 1991), using group visible materials such as flip charts or overheads (Wharton et al. 1992), and paper-based forms (Wharton 1992; Wharton et al. 1992; Lewis et al. 1991a).

For group evaluations it is strongly suggested that group visible materials be used. Where convenient, capture the entire evaluation process, including evaluator comments, on videotape. The videotape can serve as a record for going back and verifying or retracing comments or decisions. The group visible materials serve to capture and summarize all decisions and key information for the group.

There are several types of information that are useful to capture using group-visible means during the evaluation process. Most useful are user knowledge requirements, assumptions about the user population, notes about side issues and design changes, and the credible success story developed during the walkthrough. We suggest the use of three displays to capture all of this information. In particular, we suggest using one display (e.g., flip chart or overhead) for recording the key points of the group story, one display for cataloging all information about each class of user, and one display for capturing notes about side issues and design changes. The information about users may also be combined with the group story, with the relevant user information appropriately marked or singled out.

When recording the group story, we suggest capturing key points like those given in the next section. For user information we suggest capturing the following information for each class of user:

- What the user must know prior to performing the task
- What the user should learn while performing the task

For the side issues and design changes that are discussed during the evaluation process, we suggest making notes about the specific issue with enough appropriate context to reconstruct and address the issue more fully at a later time. For example, suppose during the evaluation of a particular action it is found that a menu item has a misspelling. We would record the particular

menu that has the misspelling so that the problem can easily be located and corrected later. Videotapes of the evaluation can also be of use here. Examples of some information the group might record are given in the next section.

Success and Failure Stories

Recall that the analysis phase of the walkthrough consists of examining each user action and crafting a credible story as to why the expected users would choose that action based on assumptions about each user's background knowledge and goals, and on an understanding of the problem-solving process that enables a user to guess the correct action. To give a better idea of the kinds of stories the analysts derive during a walkthrough, we present several examples of credible success stories—stories that describe an interface working as it should—as well as stories that describe clear failures of interfaces.

Examples of Credible Success Stories

Success Story 1: An experienced Macintosh user begins a task by double-clicking an application's icon to start it.

Defense of Credibility:

- User is trying to start the application because the user knows you have to start an application to use it.
- User knows that double-clicking is possible from experience.
- User knows double-clicking is the action to use from experience.
- Changes to the display and menu bar signal start of the application.

Note that the first three parts of this story would not work for a person new to computers, and the second and third would not work for people without Mac experience.

Success Story 2: An experienced Macintosh user pulls down the GRAPH menu in preparing a graph in a presentation graphics package.

Defense of Credibility:

- User is trying to prepare a graph because that is the overall task.
- User knows to pull down this menu because the title GRAPH is clearly related to what is being attempted.

- User knows that pulling down the menu is possible, and that that is the action to take if the label looks good, by experience with the Mac.
- User knows things are going OK when a palette of graph types appears on the pull-down menu.

Success Story 3: A bank customer is using a phone-in system to transfer funds between accounts. The system says "enter your customer ID number" and the customer keys it in.

Defense of Credibility:

- User is trying to enter the number because the system said to do it.
- User uses the touch-tone buttons because they are visible and there is no other available facility for entering the number.
- User knows customer ID number because the user memorized it when the bank assigned it. (Note: this part of the story stretches credibility!)
- User thinks things are going OK when system goes on to play an audio menu of services.

Common Features of Success

With these examples in mind, we can revisit the four points that the analysts consider at each step and note some further details for each.

Users May Know What Effect To Achieve:

- Because it is part of their original task, or
- Because they have experience using a system, or
- Because the system tells them to do it

Users May Know An Action Is Available:

- By experience, or
- By seeing some device (like a button) or
- By seeing a representation of an action (like a menu entry)

Users May Know An Action Is Appropriate For The Effect They Are Trying To Achieve:

- By experience, or
- Because the interface provides a prompt or label that connects the action to what they are trying to do, or

- Because all other actions look wrong

Users May Know Things Are Going OK After An Action:

- By experience, or
- By recognizing a connection between a system response and what they were trying to do

Several of these points emphasize the importance of knowing how the user would describe the task. When the system uses the same terminology (or graphics) as the user, the user will pick the correct actions. This is the "label-following" strategy. Feedback will also be meaningful when it is expressed in the user's vocabulary. When unusual terms are used by the system, the user will find it difficult to succeed without additional knowledge.

Examples of Failure Stories—When No Credible Story Can Be Told

Success stories require success under all four of the analysts' criteria, while failure stories typically fail under a single criterion. With this in mind, we organize the examples of failure according to the criterion under which they fail. Note that each of these examples is based on user testing or field data.

Criterion: Will the User Be Trying To Achieve The Right Effect?

- *Example 1:* In an early office system it was necessary to clear a field on a menu by pressing a special key before typing into the field.
- *Failure story:* Users probably did not realize they needed to clear the field (so they never looked for the control).

Criterion: Will the User Know That the Correct Action Is Available?

- *Example 1:* In a particular graphing program, changing font and other characteristics of the graph title is achieved by double-clicking on the title to open a dialog box.
- *Failure story:* Users often do not consider double-clicking in this task context.

Command-oriented systems often have failures under this criterion. Users often know what effect they want to achieve (e.g., find the size of a file or create a new directory), but they don't know—and can't find—the name of the command.

Criterion: Will the User Know That The Correct Action Will Achieve The Desired Effect?

Example 1: In an early office system it was necessary to access key operations, like printing, from a special menu that was only accessible after a special key, labelled REQ, had been pressed.

Failure story: Users did not know what the special key REQ was used for, nor did they realize that they needed to press this key to access the requisite menu.

Example 2: In a word processor there are two menus. One menu is called FORMAT and the other is called FONT. Type styles are part of the FORMAT menu.

Failure story: Users do not know which menu to select if they want to put something in italics.

Example 3: In a telephone-based environment, an audio prompt may tell the user to "press the pound sign" on a phone keypad.

Failure story: Users often do not realize that the symbol "#" on the keypad is the pound sign.

Criterion: If The Correct Action Is Taken, Will The User See That Things Are Going OK?

Example 1: In an early office system you were required to sign off the system from a menu. After a user signed off, a sign-on menu appeared.

Failure story: Users didn't always realize that they had successfully signed off. Instead, some users would automatically fill in the sign-on menu again and get caught in a loop.

Example 2: In an early office system no feedback was presented to the user when a document was printed unless explicitly requested by the user.

Failure story: Learners didn't always realize that they had been successful. In this case, some learners repeatedly printed documents.

Some Other Problems to Watch For

- *Time outs:* Some systems, especially phone-based ones, give users only a certain amount of time to take action. Try to determine if the time allowed is adequate.
- *Physically difficult actions:* Holding down keys simultaneously, especially when they have to be pressed or released in a certain order, is hard. So is selecting small targets with a mouse, touching a screen, or the like.
- *Dropped terminator actions:* Users sometimes forget actions that signal completion of some part of a task, like pressing the pound sign after entering an ID number or a semicolon after a statement in a programming language. This may be because the completion is already apparent to the user. This may occur even though the user knows perfectly well that the action is required and even though they do it correctly most of the time.

In addition to these specific areas, the analysts need to be suspicious of supposedly credible stories that depend on users' knowledge of system terminology, or "commonly used" interaction methods, or complex plans.

5.3 Detailed Example

In this section we present a detailed example of a short walkthrough. The task is to forward calls on a campus phone system. We note that the walkthrough treats this system as a walk-up-and-use application, which may not have been the designer's intent. Nonetheless, our experience is that at least some users find themselves in the position of needing to use the system without training.

The Walkthrough: Preparatory Phase

Users: The larger class of users includes all staff and faculty on the university's campus, plus their guests and other visitors. It is expected that anyone needing to make a phone call has previously used either a touch-tone or rotary dial telephone. For the evaluation that follows we assume that our user is one of the university's professors. The professor has used the phone system several times to place outgoing and receive incoming calls. The professor further knows that you can program your phone to do assorted tasks such as forwarding your calls.

Task: I want my phone calls to be forwarded to my associate's office. My associate's number is 492–1234.

Action sequence: The seven required actions for accomplishing this task and the associated system responses on the phone system are as follows:

1. Pick up the receiver.
   Phone: `dial tone`
2. Press #2 (Command to cancel forwarding).
   Phone: `bip bip bip`
3. Hang up the receiver.
4. Pick up the receiver.
   Phone: `dial tone`
5. Press *2 (Command to forward calls).
   Phone: `dial tone`
6. Press 21234.
   Phone: `bip bip bip`
7. Hang up the receiver.

Interface: The phone is a standard size, touch-tone phone located on the professor's desk. There is a template that overlays the telephone's keypad (we assume it has not been mislaid) that includes the following material:

FWD *2
CNCL #2
SEND ALL *3
CNCL #2

The Walkthrough: Step-by-Step Analysis Phase

We now use the walkthrough process to work through the interaction and appraise each step:

1. Pick up the receiver.
   Phone: `dial tone`
   Success story:
   This seems OK based on prior experience with phones. But note that there are now phones that you "program" without picking

them up!

2. Press #2 (Command to cancel forwarding).
   Phone: `bip bip bip`
   Failure story:
   - *Criterion:* Will The User Be Trying To Achieve The Right Effect?

     Big trouble is the result here. Why would the user be trying to cancel forwarding? They just have to know.
   - *Criterion:* Will The User Know That The Correct Action Will Achieve The Desired Effect?

     Even if the users know to cancel forwarding, they might not recognize CNCL on the template and they might think the required action is pressing just the number "2", not "# 2." Also they might try to press these buttons together (simultaneously) rather than in order (sequentially).
   - *Criterion:* If The Correct Action Is Taken, Will the User See That Things Are Going OK?

     Furthermore, how do users know they've succeeded? After experience they will recognize the `bips` as a confirmation, but will they at first?
3. Hang up the receiver.
   Failure story:
   - *Criterion:* Will The User Be Trying To Achieve The Right Effect?

     More big trouble results. Even if you know you have to cancel forwarding, why should you have to hang up before reestablishing it? This action has a system-oriented effect that the user will have no reason to try to achieve.
4. Pick up the receiver.
   Phone: `dial tone`
   Success story:

   This seems OK based on experience (but remember that not all phones require this now).

5. Press *2 (Command to forward calls).
   Phone: `dial tone`
   Failure story:
   - *Criterion:* Will The User Know That The Correct Action Will Achieve The Desired Effect?

     Here the issue is deciding between *2 and *3. The description on the template is of little help: Some people won't recognize FWD, and SEND ALL will look good even if they do.

     Also, there's the small aforementioned worry about whether "*" should be pressed simultaneously or as part of a sequence.
   - *Criterion:* If The Correct Action Is Taken, Will the User See That Things Are Going OK?

     Also, the feedback may be problematic: It suggests that you can dial something, but it is also unchanged from what you heard before taking this action.
6. Press 21234.
   Phone: `bip bip bip`
   Failure story:
   - *Criterion:* Will The User Be Trying To Achieve The Right Effect?

     How does the user know to enter the number now? Maybe it's not an unreasonable guess, but the dial tone doesn't constitute much guidance because it only suggests that the phone is active.
   - *Criterion:* Will The User Know That The Correct Action Will Achieve The Desired Effect?

     Also, there is a likelihood of error in not working out the *form* of the number that is needed. That is the user must understand that it is sufficient and correct to enter "21234" and that the entire number sequence of "4921234" is not needed.
   - *Criterion:* If The Correct Action Is Taken, Will the User See That Things Are Going OK?

     As mentioned previously, the bips may not mean much to someone

starting out.

7. Hang up the receiver.

   Success story:

   Seems OK based on prior experience with phones.

Fixes for Problems Discovered in the Example

The focus of the walkthrough is on spotting problems in an interface, but it also provides an explanation of those problems; that explanation can be useful in developing fixes. In the phone example, the biggest problem is the unexpected requirement to cancel forwarding. This might be fixed in one of two ways:

1. Eliminate the unexpected requirement.
2. Make it expected by providing a prompt.

Compare the two solutions. The first solution, eliminating the requirement, is obviously better. It eliminates a number of secondary problems, including the unexpected need to hang up. However, such a solution might be infeasible for some underlying technical reason. The second solution—providing a prompt so the action is expected—has some problems. Adding a spoken, audio prompt is almost certainly impossible for the underlying system in this case, which seems to have only tones as output. This leaves making an addition to the template, which is not very attractive but better than nothing. One could add material at the bottom such as "To forward calls first dial #2 and hang up. Then dial *2 and the number."

The second problem in the phone interface is the confusability of FWD and SEND ALL. In fact, these two operations are related. FWD allows you to forward calls to a number that you key in; SEND ALL allows you to forward calls to a prespecified number, not keyed in at the time of forwarding. One solution to this problem might be to add template material that attempts to clarify the role of these two functions, although brief wording of the distinction would be difficult. Other possible solutions include:

- Eliminate SEND ALL as a feature. Is its convenience worth the confusion?
- Include the feature but don't document it on the template, making it a "power user" feature that is only documented in a corresponding user guide.

- With spoken prompts, the system could offer a choice within FWD of providing a destination number or accepting a default.
- The designers could try to think of new labels for one or both operations, such as FWD or FWD TO SECY.

Each of the suggested fixes came about due to the walkthrough process and the categorization of problems according to type of failure. General guidelines about resulting fixes by failure type are described next.

5.4 *Staying on Track and General Fixes*

Staying on Track

The previous example demonstrates that the analysis always *tracks the correct actions.* That is, even if there is a major problem with the interface and digressions in the discussion occur, the analysis merely notes the problem and then proceeds to the next step, as if the correct action had been performed.

If the problem suggests that the user would select the wrong action, then the analysis nonetheless considers how the user would react to feedback if the correct action had been taken. Further, the state of the interface at the beginning of each action is always assumed to be the correct, on-track state, never the state after an incorrect action was performed. It is as if the system and user had been reset to the proper state.

Following this principle may force the analysts to assume a "fix" that will make sense for the rest of walkthrough. For example, the analysts may assume that the user possessed the knowledge necessary to select the correct action. That same knowledge might be used to make sense of the feedback and might even be needed again to select the next action. For this reason, it is important to clearly indicate what the user knows a priori or learns during the course of the task.

General Considerations for Using Results to Fix Problems

In general, having identified successes or failures, what should the analysts and the design team do? We noted that failures are typically associated with one of the four criteria of the walkthrough analysis; the "fixes" to avoid or repair failures can be similarly organized. Consider each of the criteria:

1. *Will the user be trying to achieve the right effect?* If the interface fails on this point—that is, if the user is not trying to do the right thing—there are at least three approaches to a fix: (1) the action might be eliminated, either taken over by the system or combined with some other action; (2) a prompt might be provided to tell the user which action must be performed; or (3) some other part of the task might be changed so the user will understand the need for the action, perhaps because it is now consistent with another part of the action sequence.
2. *Will the user know that the correct action is available?* If the user has the right goals but doesn't know the action is available in the interface, the solution is to assign the action to a more obvious control. This typically requires a menu or prompt, rather than an unprompted keystroke; or it might involve assigning the action to a hidden but more easily discoverable control, such as a submenu, or a single keystroke instead of a simultaneous key combination.
3. *Will the user know that the correct action will achieve the desired effect?* To correct failures under this criterion, the designer needs to know the users and have a good idea of how they will describe their tasks. With this information, the designer can provide labels and descriptions for actions that will include words that users are likely to use in describing their tasks. It may also be necessary to reword the labels of other controls that users might select in preference to the correct one.
4. *If the correct action is taken, will the user see that things are going OK?* Clearly, in most situations any feedback is better than none—and feedback that indicates *what* happened is better than feedback that just indicates that *something* happened. Further, feedback will be most effective when it uses terms (or graphics) that relate to the user's description for the task. Note that in simple situations, the interface may forego feedback per se in favor of prompting for the logical next action.

As a general approach, dealing with problems by *eliminating actions* is likely to be more effective than trying to fix up prompts and feedback. Where the interface shows several problems that indicate a mismatch to the user's

conception of the task, the designer should look for chances to fix those problems by a *global reorganization* rather than focusing only on local improvements.

5.5 *Evolution of the Walkthrough Method*

The cognitive walkthrough is based on a theory of learning by exploration (Polson and Lewis 1990; Polson et al. 1992a) and on modern research in problem solving (Anderson 1987; Greeno and Simon 1988). The method has evolved rapidly from the original version described in Lewis et al. (1990) being shaped in part by evaluation experiments in our own and other investigators' laboratories (Jeffries et al. 1991) and Lewis and Polson's experiences in attempting to teach the method in day-long tutorials at CHI'91 and CHI'92. This section briefly describes the underlying theoretical model and summarizes our experience with and evolution of the cognitive walkthrough method. We also discuss criticisms of the method and its current status.

In summary, potential users of the cognitive walkthrough method must understand that it is focused very explicitly on one aspect of usability, ease of learning. It attempts to provide a detailed, step-by-step evaluation of the user's interaction with an interface in the process of carrying out a specific task. Both the narrow focus on a single aspect of usability and the fact that the method provides a quite detailed evaluation of ease of learning are sources of the method's strengths and weaknesses.

Underlying Theory

Modern theories of skill acquisition (Anderson 1987, 1993; Newell 1990) assume that problem-solving processes are used to discover correct actions and that the learning mechanisms store representations of correct actions with the users' current goals and task contexts. Versions of this general model have been used to account for skill acquisition in a large number of domains ranging from high school geometry (Anderson et al. 1985) to human-computer interaction (Kieras and Polson 1985; Polson and Lewis 1990). Such models predict that teachers and designers can facilitate the skill acquisition process by facilitating the problem-solving processes (Anderson et al.

1984; Anderson et al. 1989; Anderson et al. 1990; Anderson et al. 1992). The cognitive walkthrough attempts to provide guidance to developers to enable them to design interfaces that facilitate problem-solving processes that users employ to discover an action sequence necessary to perform a task.

The model of problem-solving underlying the cognitive walkthrough is based on laboratory research on problem-solving done in the 1960s through the early 1980s demonstrating that people with limited experience in a domain employ variates of means-ends analysis (Newell and Simon 1972) in a very large number of situations (Greeno and Simon 1988; Polson and Lewis 1990). Means-ends analysis is a problem-solving heuristic that selects the next action by choosing the action that will reduce the most important difference between the current state and the goal.

The walkthrough process hand-simulates the user's problem-solving processes: formulating a current goal, selecting a next action, and modifying the goal based upon the consequences of the action. The key idea is that correct actions are chosen based on their perceived similarity or relevance to the user's current goal. The sophistication and robustness of such a problem-solving process is dependent upon the users' knowledge of both the task and the interface based on their training and experience. The next action is selected based upon the users' representations of their current goal and available actions, also determined by training and experience.

Users with very limited backgrounds are almost completely dependent upon the label-following strategy (Polson and Lewis 1990). Such individuals will have simple and very concrete representations of a task and its goal structure. They have little or no knowledge of the consequences of actions, so they have a strong tendency to select menu items or other actions based on how well the labels of those actions match one or more components of their current goal.

Sophisticated users have knowledge of how to decompose a task into a collection of subtasks in order to organize the task in the fashion that permits effective use of a relevant application. They will also have extensive knowledge of possible actions and the actual consequences of those actions. However, the same basic principles apply. These users select actions that are relevant or similar to a current goal. However, the evaluation of similarity is driven by much more sophisticated knowledge of the task and the interface.

A cognitive walkthrough is a set of reasonable speculations about a user's background knowledge and state of mind while carrying out a task. This is one of its major strengths. The designer is forced to consider in some detail the kind of background knowledge a user must have and what sort of mental gyrations a user must go through to complete a task successfully. We feel that the method encourages a designer to directly confront the assumptions that are implicitly or explicitly incorporated into an interface about the user's task representation and background knowledge.

History—Current View

The history of our work with the cognitive walkthrough is the story of our attempts to achieve a balance between two conflicting goals: On the one hand, the procedure should be concise and simple so it can be used efficiently. On the other, it should provide guidance to analysts with no background in cognitive psychology and little experience in interface evaluation, leading them to examine an interface in detail and identify subtle problems that they would otherwise miss.

Our first version of the walkthrough used a single-page form, containing a series of brief questions which the authors of the method expected to be sufficiently instructive and complete (Lewis et al. 1990). But the single-page form failed when we tested its use with untrained analysts, including students in a user-interface design class and designers in industry. The most critical shortcoming of the form was its terminology, which implicitly assumed some background in cognitive science. Analysts without this background had trouble distinguishing "goals" from "actions," a distinction that is critical to the method's success. They also failed to identify some of the problems that we thought were obvious, often because questions that should have pointed to those problems were subsumed within broader general questions.

Our response to these difficulties was to develop the second version of the method, which was much more formal and far more complex. We expanded the form, breaking some of the questions into subparts, and supplied detailed instructions for each question (Lewis et al. 1991a; Polson et al. 1992a; Wharton 1992). This version required designers to perform an extremely detailed analysis of the problem-solving process, including providing an explicit description of a user's current goal structure, a detailed analysis of how the user would select an action based on this goal structure, and a

description of how feedback and a user's interpretation of that feedback would modify the goal structure appropriately facilitating selection of the next correct action.

This was the version Lewis and Polson taught at a CHI tutorial in New Orleans in 1991. Generation and manipulation of detailed goal structures turns out to be an art that is difficult even for someone with an extensive background in cognitive science. In other words, there were serious usability problems with this version of the method. To address some of these problems, particularly the requisite cognitive science background, one of us (CW) also experimented with a richer structure for training, question presentation, and recording of results (Jeffries et al. 1991). This approach included the use of a trained facilitator or "champion" to guide a group of designers in performing the walkthrough. It also recorded the results of the process as several separate documents, most importantly a knowledge analysis and a series of problem reports, which were designed to be immediately usable in the next phase of the design process (Wharton 1992).

A side-effect of all these changes was to make the walkthrough process more tedious and time-consuming, which many analysts reported as a major shortcoming (Jeffries et al. 1991; Wharton et al. 1992). One attempt to rectify this problem was the "automated" walkthrough, an Apple HyperCard stack that prompted for each question of the walkthrough and provided space for recording the analysts' evaluations, which could then be printed in summary form (Rieman et al. 1991). The automated system included a "help" screen for each question; it automatically disabled certain questions when they weren't relevant; and it allowed other questions to be manually disabled for an entire session (e.g., questions about time-outs in a system where none will occur).

Most of the published evaluation studies have examined variants of this second, more detailed, version of the method (Jeffries et al. 1991; Wharton et al. 1992; Desurvire et al. 1992; Cuomo and Bowen 1992). Discussion of some of these evaluation studies is presented later in this section.

Even with the aid of the automated system, however, many analysts found this version of the walkthrough to be inordinately time-consuming (Desurvire et al. 1992). In recognition of that difficulty, a third version of the walkthrough method was developed. This is the version that is presented in this chapter.

This current version represents a revised emphasis on our original goals for the procedure: This version de-emphasizes explicit consideration of the user's goal structure. The designer is now asked to try to motivate a user's choice of a correct action in the sense of motivation that one would expect to underlie the actions of characters in a novel or murder mystery. The designer is asked to tell a "credible story" that motivates the selection of the next correct action. We conjecture that designers will be able to tap the large amount of tacit knowledge that human beings have about individual's goals and the motivation of action.

In part, our conjecture is based on Lewis and Polson's experience with this new version of the method at their tutorial at CHI'92. The response received was dramatically different from the CHI'91 experience and quite favorable. Students using the simplified version of the method successfully completed several problems provided by Lewis and Polson during the practical portion of the tutorial. Working in groups of four to five, students rapidly and efficiently worked through the training problems, uncovering the difficulties that we had incorporated into the problems as well as having additional insights.

Another one of us (JR) has also used this version of the method with other students in a course on user interfaces. Results have been mixed in that students do not always make effective use of the success and failure criteria, but certainly they are much more favorable when compared to experiences with past versions. We note, however, that these acclamations are not the product of a formal evaluation study, but rather anecdotal reports of user experiences. We do not yet have any empirical data validating the new version of the method.

While we still believe that the process can now be used quite effectively by designers, we recognize that this can only happen if the designers are also given more training in cognitive theory than they would implicitly receive from the simple forms and brief instructions of our original attempts. Such

training may come through a short class, such as the tutorials we have given at the CHI conference or it may be provided through the "champion" approach, where a member of the analysis group is familiar with the method and with basic cognitive psychology. A good starting point is the overview of psychological theory (presented by Wharton and Lewis in Chapter 13), which points out some of the theoretical aspects that are relevant to software design. A critical part of the training is to present examples of interfaces that fail or succeed, analyzed in terms of the underlying theory. We have included one such set of examples in this chapter.

Since we now recommend that analysts understand the theory that underlies the walkthrough, in this latest version we have shifted our attention away from the details of the questions and the forms on which the walkthrough's results are recorded. The questions should focus on the basic cycle of goal formation, action selection, and response evaluation, but the details of what to examine within each phase may vary from interface to interface. Equally important, what is recorded as the analysis proceeds should also be tailored to the needs of the design process in which the walkthrough is embedded.

To summarize, we originally conceived of the walkthrough as a simple question-and-answer process that designers could use effectively with little understanding of cognitive theory, much the way a person can fill in an income tax form with no deep understanding of tax law and policy. We now believe that a basic understanding of the cognitive theory is essential, and that given that understanding, designers can structure the walkthrough process to best fit the needs of their individual situations.

Criticisms Addressed and Current Appraisal

A variety of criticisms have been levied against the cognitive walkthrough. Most of these criticisms have been based on the second version of the method; we believe we have addressed nearly all of them in our third version of the method. The criticisms have fallen into two general classes. The first is the type of usability problem found. The second is how the method fits to the development process. We will address concerns in each of these categories by discussing the method's strengths and weaknesses.

Types of Usability Problems Identified

A number of people have identified concerns regarding the types of usability problems that the walkthrough identifies (Jeffries et al. 1991; Cuomo and Bowen 1992; Wharton et al. 1992). Issues that have been raised include usability problem severity, content, scope, and total numbers. We address each of these.

Severity

By problem severity two things are meant: How much the problem impedes user progress and how much the problem is in need of repair. Using such definitions, the Jeffries et al. study (1991) compared four usability evaluation methods: a variant of heuristic evaluation, standard usability testing, guideline application, and the cognitive walkthrough. In absolute severity ratings, for the first type of severity mentioned above, the walkthrough had the lowest mean at 3.44 on a 9-point scale (where 1 is trivial and 9 is critical). Usability testing had the highest with a mean of 4.15. But when compared to the other methods the difference was significantly less: Heuristic evaluation had a mean value of 3.59 and guidelines had a mean value of 3.61, reducing the mean difference between the methods to only 0.17. We (like Jeffries et al. 1991) believe that these results, particularly for usability testing, in part reflect the biased phrasing used to identify the problems: Biased phrases such as "users had trouble. . ." occurred with the usability testing problems whereas the others used personal references or more neutral language.

Additionally, when comparing the ratios of all usability problems found, of most severe to least severe, they found that guidelines and the walkthrough were quite similar, identifying roughly an equal number of each. The heuristic evaluation performed more poorly, identifying nearly twice as many problems of lesser severity than it did of those that were more severe, and usability testing found almost exclusively problems that were considered to be most severe. Such results are not surprising to us, reflecting what we believe to be the underlying approach of each method. (cf. the results obtained by Desurvire et al. 1992.)

Each method promotes a different style of analysis. The guidelines, usability testing, and walkthrough methods each suggest specific issues to be examined: The "path" of analysis required (i.e., examine *only* a suite of user tasks or apply specific guidelines) is expectedly quite narrow and deviations from the path are not rewarded because you still need to complete other

required analyses. Conversely, heuristic evaluation promotes the notion of straying off a rigid path of analysis: The evaluator can look at any given part of an interface in any order and manner chosen. The reward for the heuristic evaluator is based on the breadth of an analysis, not a narrow one. For the walkthrough, this narrowness is in part due to the focus on user actions. Usability testing may be even more restricted.

Content

Both Jeffries et al. (1991) and Cuomo and Bowen (1992) have raised concerns about the content of the usability problems. Jeffries et al. (1991) performed two related types of content analyses: consistency and recurrence. Cuomo and Bowen (1992) examined problem content in light of the seven stages of user activity in both action execution and evaluation as defined by Norman (1986) and according to functional areas as defined by Smith and Mosier (1986). We will address each of these.

The walkthrough did not rate as well on recurring and general problems as the other methods. Consistency is a measure of how well problems indicate that one part of the interface is in conflict with another. Usability testing was found to be the method which addressed consistency problems the least, while all others were comparable. These results again mirror the narrow versus broad path of analysis used by each method.

A recurring problem is one that continually interferes with the interaction, as opposed to only interfering the first time. The walkthrough and heuristic evaluation were lower (by 20 percent) than the guidelines and usability testing methods. We believe this result mirrors the underlying learning principle of both methods. The walkthrough method assumes that users learn as they go along and also that the evaluation will "stay on track." Thus, the evaluation results will reflect these assumptions. Presumably this happens, too, in heuristic evaluations, but not in the other methods.

Norman's (1986) seven stages of user activities are: (1) establish the goal, (2) form the intention, (3) specify the action sequence, (4) execute the action, (5) perceive the system state, (6) interpret the state, and (7) evaluate the state with respect to goals and intentions. In a study by Cuomo and Bowen (1992) comparing three evaluation techniques—cognitive walkthroughs, guidelines for designing user interface software, and heuristic evaluation—problems found by each method were categorized into one of the last six stages of user

activity in an attempt to learn the types of problems each evaluation technique addresses. As would be expected, the walkthrough predominantly addressed problems in stage 3 with which the method (particularly the second version) is most concerned. It also did comparable to the other methods in stage 2, although all methods were considered weak in this stage. The latter stages of user activities (4-7) we believe are now addressed better by this latest version of the walkthrough method. More generally, those problems identified across all stages by the walkthrough rarely overlapped with any of the other methods of guidelines and heuristic evaluation, finding unique types of problems.

Examination of the results that categorized problems by functional area indicates similar results to the stages of user activity. Of the four functional areas of (1) data entry, (2) data display, (3) sequence control, and (4) user guidance, the walkthrough performed better on those areas more closely aligned with the earlier listed execution stages of user activities. That is, there were more problems in the data entry and sequence control areas, than in the data display and user guidance areas. Again, we believe that the second version of the method masked the role that feedback plays in a user's interaction because of its focus on goals. Our new version has now brought both of these concerns to a more level playing field.

Follow-up work done by Cuomo (personal communication, October 1992) indicates that the walkthrough, when compared to guidelines and heuristic evaluation, actually is a better predictor of problems when considering the above stages of user activity. In detail, Cuomo and Bowen performed a usability test and the results were then compared to the previously identified problem types. For each of the predicted problem types, evidence that the problem affected user performance was sought. Across all stages of user activity, of the 51 problem types identified by guidelines, only 11 problem types (22 percent) actually appeared to cause at least one noticeable user difficulty. Of the 35 problem types predicted by heuristic evaluation, 16 (46 percent) caused users difficulty and of the 24 problem types predicted by the walkthrough, 14 (58 percent) problems affected users. The cognitive walkthrough performed the best in terms of predicting problems which actually affected users' performance.

Scope

Two concerns have been brought up regarding the scope of the those problems identified by the walkthrough method. In particular, Jeffries et al. (1991) found that walkthroughs identified more specific problems than general problems. This is related to the point made by Wharton et al. (1992) regarding no high-level treatment of user tasks. Because the walkthrough is task-based and consequently follows a narrow path of analysis, there is no high-level treatment of user tasks nor are corresponding global interface problems identified. By a high-level task treatment it is meant that there is no way to determine whether a given task evaluates well as a whole (since only the actions are examined) or whether the interface as a whole matches well to a user's conceptual needs. A general flaw is one that affects several parts of the interface and not simply a single part. We believe this to be an expected trade-off. If more tasks are evaluated or more partial solution paths can be evaluated, the number of general problems would be expected to increase.

Total Number of Problems

As for the sheer number of problems identified by a particular method, this is still an open question. Jeffries et al. (1991) indicates that heuristic evaluation will yield more problems, with all other methods yielding fewer problems but equivalent numbers. However, Cuomo and Bowen (1992) found that guidelines may yield the most problems with the walkthrough yielding the second highest number, due to the method's narrowness. Heuristic evaluation then follows third. These studies have conflicting results because of the differing applications of such techniques. In these studies the number of evaluators, the differing man-hours or training and availability, and experience with the overall task and environment, were quite varied. For example, in the Cuomo and Bowen study, Smith and Mosier's (1986) guidelines differ not only in number—944 compared to 62—for those used by Jeffries et al. (1991), but also in content. Consequently, no general conclusions can be drawn at this time. However, we note a few points about the walkthrough method and the larger results generally.

Although we only have a couple of studies to draw on, there seems to be a "magic" number of roughly 28 to 43 identified problems (let's say 35± 7) that applies to both studies for each method group. For these two studies, we list the mean results in order of Jeffries et al. (1991) first, followed by Cuomo and Bowen (1992) second. A single heuristic evaluator (which could be considered as a single group) yielded 30.25 and 28.5 problems per evaluator.

The walkthrough yielded 35 and 43 problems. Guidelines yielded 35 and 113 problems. And usability testing yielded 32 problems in the first study. Exclusive of the results for guidelines (which cannot be compared directly because different guidelines were used), the walkthrough appears to yield roughly the same number of problems as the other methods, and in some cases slightly more. Thus, the method competes well with all other methods using this criterion.

Finally, it is important to note that when discussing comparative studies such as these, many other factors will influence the results. For instance, these include the number of inherent problems in the interface, the number of tasks used for the evaluation, the reality of the databases, the skill of the evaluators, evaluator familiarity with the tasks, and the style of interface (e.g., there are fewer guidelines for direct manipulation style interfaces).

Fit to the Development Process

Concerns have also been raised about how well the walkthrough fits into the development process (Jeffries et al. 1991; Wharton et al. 1992). With one exception, we believe that we have remedied all of these process concerns. Many of these concerns have already been addressed in some detail throughout this chapter, thus we only highlight them here, as necessary.

Bookkeeping

In addition to recording information about the walkthrough proper, we also recommend capturing information regarding any side issues for later discussion and about the success or failure story being told, particularly as it applies to users. We further suggest using group-visible materials and videotaping where convenient. (For details see page 112.)

Cumulative Evaluation Time

The amount of time it takes to perform a walkthrough in general will be longer than other methods because more detailed analyses are undertaken. The second version of the method required much more time due to the extensive bookkeeping requirements. Looser applications of the second version by one of us (CW) and our experiences with this current version have reduced the amount of time substantially, by a factor of between 2 to 4.

Group Process Concerns

Wharton et al. (1992) made several suggestions for ensuring a good group walkthrough. Further experimentation with the method in group settings by one of us (CW) and refinements made to our latest version have verified that by following many of the Wharton et al. (1992) suggestions, the evaluation will be much more successful and be rated more favorably. One issue not noted, however, is the need for a good facilitator. We cannot emphasize enough that the facilitator needs to be aware of and to manage expected group dynamics and discussions so that time is not wasted during the process.

Task Selection

Task selection is a critical component of the process. For an evaluation to be most informative, tasks should be selected by adhering to those guidelines given in the section starting on page 109. Working closely with marketing folks and others concerned about the product to derive well thought out benchmark tests serves the evaluation well. Task selection more generally is an area of research for the HCI community.

Requisite Knowledge for the Evaluators

As has been noted by various critics of the method, there still appears to be a need for at least one evaluator to have some basic understanding of cognitive science for a most effective application of the method. Currently there appears to be no easy way around this. However, the walkthrough method is not alone in this requirement. Like the walkthrough, other methods have been applied successfully by the typical software developer, but better results are yielded when someone has knowledge of cognitive science (Nielsen 1992a; Desurvire et al. 1992).

Current Appraisal

The adjustments and refinements that we have made to the method since its inception in 1990 have been many. We have responded to the number of criticisms made about earlier versions of the method. The result has been the new version of the method presented in this chapter. We believe that this new version of the method remedies as many of the concerns raised about best fitting the method to the development process, as is possible. As our comments indicate, the walkthrough is quite effective at the job it is designed to do. It is not a panacea, nor the ultimate method. There are trade-offs when

one method is selected in place of another, as others' results have shown. Again, we suggest the use of multiple inspection methods for a thorough evaluation.

Offshoots and Other Uses

The cognitive walkthrough is aimed at the analysis of highly prompted interfaces. The questions posed by the method direct the analyst to examine the interplay between the user's intentions and the cues and feedback provided by the interface. But the underlying logic of the cognitive walkthrough can be applied in situations in which there is little or no prompting, and users must reason their way through a problem without help from an interface. The essence of the cognitive walkthrough is the description and evaluation of a hypothetical process— a conjecture about the steps the user will take in determining the correct actions in a problem situation. In the cognitive walkthrough we assume that these steps are largely guided by the interface. But nothing prevents us from imagining what steps might be taken in the absence of such guidance. We have seen and continue to see opportunities to apply and adapt the method in other situations.

Programming Walkthroughs

The programming walkthrough (Bell et al. 1994) applies this same idea to evaluating the design of programming languages. A typical programming environment provides no prompting and no immediate feedback for most actions: The programmer writes whatever code desired, and gets only delayed, collective feedback on the effect of all this code taken together. (So-called structure editors provide some immediate feedback as code is written, but only on the syntactic legality of the code.) The cognitive walkthrough as such cannot be applied in this situation. But it still makes sense to imagine the decisions faced by a programmer in writing a program in a language and to critique the language if some of these decisions appear to be difficult.

The programming walkthrough shares with the cognitive walkthrough a dependence on specific tasks. The analyst must have one or more specific problems for which programs might be written and examines the process of solving these. The whole process is subject to comment, including any needed reframing of the problem to make it amenable to solution, preliminary decisions about how to break the problem up into parts, and any other problem-solving steps, even those that may not depend on the details of the

language. As with the cognitive walkthrough the analysis aims to point out places where the process is likely to break down; that is, where the programmer is likely to make a wrong move.

Because of the lack of cues from the environment, programming requires a good deal of knowledge to choose the right steps. The programming walkthrough produces an inventory of the knowledge required for the steps in the tasks that it examines. For example, if successful use of a language requires the programmer to define certain basic data structures before writing the code for operations on these structures, then the programmer has to know this. If the design is to succeed, this piece of knowledge and other necessary knowledge, must be conveyed to the programmer in some way. If this is not feasible or desirable, then the language design must be changed to eliminate the need for this knowledge. This inventory of necessary knowledge (called "guiding knowledge") is a useful by-product of the programming walkthrough method.

The programming walkthrough has been used in a number of language design projects (Bell et al. 1994). But the idea is not limited in its application to programming languages as such. Any tool can be examined in this way, since the key idea of the analysis is simply to describe and critique a plausible sequence of steps leading from a problem to a solution using the tool. The idea has been applied to a geographic information system, which combines highly prompted interfaces for some functions with programming-language-like unprompted support for other functions. We have applied the method informally to the analysis of algorithm animations, by asking how observations users might make in viewing an animation would or would not aid them in answering specific test questions later (Stasko, Badre, and Lewis 1993).

Designers' Self-Criticism and Design Rationale

In our earlier discussion we focused on the use of the cognitive walkthrough by a group of people, all serving as analysts for a particular design. Another approach is to have a separate group of analysts critique the design of another group. This division of labor between design and evaluation is traditional, but we think it can have undesirable effects. Having a separate evaluation group is costly in staff and in communication. Further, and sometimes more important, it can lead to an "*us* versus *them*" attitude, on both sides, in which the designers view the evaluators as kibitzers who do not contribute real work

but slow down those who do, and the evaluators see the designers as not sensitive to the demands of usability. If designers know they must satisfy usability evaluators, rather than their own values, they may avoid taking any responsibility for usability beyond what is forced on them, instead of seeing usability as just another aspect of design for which they are professionally accountable and in which they should take pride.

In principle, at least, we think the cognitive walkthrough could ease this situation. Designers themselves might use the method and thereby gain some ability to assess the usability of their own designs. The logistical simplicity of the method is an advantage here; anyone can do a walkthrough by themselves, any time. It is not necessary to always perform group walkthroughs.

We have successfully used the programming walkthrough in this way in our own design work on various versions of the ChemTrains graphical programming language (Bell et al. 1994). Besides being a general aid to design, we found the walkthroughs also useful in producing design rationale; that is, the reasons for design decisions, as described in Lewis et al. (1991b). We found that choosing and analyzing specific problems, as demanded by cognitive or programming walkthroughs, is a good way to see and keep track of the strengths and weaknesses of alternative designs.

5.6 *Value of the Walkthrough in Design*

The cognitive walkthrough method promises to be a valuable addition to the designer's suite of tools. The new version of the method is flexible enough to fit into any given software development process. The method identifies problems with a design early in the process and, by describing the reasons for those problems, it suggests design changes early on. Testing the interface with actual users, by comparison, can only be performed after a prototype of the system is available; it generally yields less obvious insight into the reasons for the problems it discovers. The walkthrough can also be used to identify questions that can be answered by specific, focused user testing, such as questions about user terminology or task description.

Experience with an earlier version of the walkthrough has shown that it can be applied effectively by software engineers, although best results are obtained where one of the analysts has some background in cognitive psychology (Jeffries et al. 1991; Wharton et al. 1992). Our experiences with the newest version of the method are similarly positive. However, like other methods, the walkthrough is not designed to discover every problem with an interface. The method's developers suggest that it is best used along with other methods that designers may find effective in answering different questions about the interface.

Acknowledgments

We thank all participants in the CHI'92 Usability Inspection Methods Workshop for their comments and helpful discussions. We also thank the following people for their helpful discussions and detailed comments on earlier drafts of this paper: Randolph Bias, Louis Blatt, Donna Cuomo, Heather Desurvire, Barbara Diekmann, George Engelbeck, Bob Mack, and Dennis Wixon.