The Four Components of a Procedure

—HANS VAN DER MEIJ AND MARK GELLEVIJ

***Abstract*—*As they guide people in performing a task, procedures are the heart of most manuals. It is, therefore, somewhat surprising that the theoretical and empirical knowledge of their nature has remained somewhat elusive. This paper describes a theoretical framework for procedures, summarized as the four components model, which is grounded in systems theory and rhetoric. The study addresses two research questions: (1) What are procedures made of? and (2) Which design guidelines for procedures can be abstracted from theory and research? The model distinguishes between: goals, prerequisite states, unwanted states (warnings and problem-solving information), and actions and reactions. For each component pertinent research findings are summarized and lead to the formulation of design guidelines. Occasionally these guidelines are compared with existing procedures from a sample of 104 manuals to see how well theory and practice agree. The model offers a manageable and expandable framework for creating user support that is based on scientific research. It can be used for a systematic analysis of procedures and for their (re)design.*

***Index Terms*—*Design guidelines, information typology, instructional design, screen captures.*

Information typologies are the building blocks of design. The creation of good user support hinges on their systematic use, but an information typology alone is not enough to do the trick. What is also needed is a structure or scenario in which the building blocks fall into place. In this paper, we do not discuss the question of how to create such an overall structure or scenario. Instead, we concentrate on an information typology.

The typology that is focal here concerns the building blocks of procedures. The main questions that we address are: "What are procedures made of?" and "Which design guidelines for procedures can be abstracted from theory and research?" Our discussion of design guidelines is illustrative rather than exhaustive, due to space limitations (for more information, see [1]). The guidelines generally are supported by empirical studies. We will illustrate a few discrepant cases as found in an inventory study on procedures in 52 hardware manuals and 52 software manuals.

A procedure informs a user about system states and about actions that change these states. For example, a procedure may tell users about a desired state or goal, outline the conditions for action, present intermediate states, and help the user prevent and overcome problems. In addition, the user is told which actions to take to reach the goal, how the system is likely to respond, and what else may happen. Table I displays the four system states and three action types that exist.

TABLE I
Our system states and three action types that constitute a procedure [2]

States	*Desired*	The goal presented to the user
	Prerequisite	The condition for moving toward the desired state
	Interim	The intermediate state or subgoal
	Unwanted	The to-be-avoided states (e.g. errors and malfunctions)
Actions	*Human*	The actions taken by the user
	System	The responses of the system
	External	The events or actions from outside (e.g. power shortage) that may affect the system

All kinds of combinations of these elements, together with the proper attention to rhetorics (e.g., considerations of context and audience), form procedures. These elements are incorporated in the four components of a procedure that we discuss here. The model (that is directly applicable for design) consists of the following components: goals, prerequisites, actions and reactions, and unwanted states. The components are detailed in the next sections. Because the warnings and problem-solving information component of the unwanted states each have their own unique design guidelines, they will be treated in separate sections.

Manuscript received December 4, 2002;
revised November 12, 2003.
The authors are with the Faculty of Educational Science & Technology, Twente University, Enschede 7500 AE, The Netherlands (email: meij@edte.utwente.nl; gellevij@edte.utwente.nl).
IEEE DOI 10.1109/TPC.2004.824292

GOAL(S) COMPONENT OF A PROCEDURE

A goal is a state that the user tries to realize. This state can be an end state or an intermediate state. For the designer, it is important to know for which of these goals to aim. We once came across a procedure in which the user had to complete 21 action steps to achieve a goal. There was no breakdown into subgoals and no information about the subgoals that the user was evidently completing. One of the fixes should be in presenting subgoals because, at some moment during these actions, users are likely to wonder about the goal they are pursuing.

When is it useful to break down a goal into subgoals? One guideline is that a procedure should consist of chunks of about three to five action steps. This is especially important when the user must (learn to) memorize a procedure because most people can keep only three to five items active in short-term memory [3].

A breakdown into subgoals can also be based on meaningful divisions for the domain at hand. For example, task analyses invariably reveal that some methods are used over and over again. Such methods present logical subgoals. In word processing, manipulating text blocks is one of these recurring methods.

Occasionally, designers also have to invent meaningful (sub)goals, to give tasks a meaning that they do not have by themselves. For example, users who must learn how to use a mouse are not interested in learning these movements themselves. They are a means to an end. It is useful to couch these tasks in others that are more meaningful. Thus, practicing the handling of the mouse can be incorporated in the more engaging goal of, for example, using an electronic calculator. In a sense, one has to sell the goal. This can either be done by presenting goals that have direct appeal to the user, or by adding an explanation on the value of the goal.

The description of a goal should add new information to what is already stated in the title. For example, the following one-sentence goal description *This task describes how you can save your file* is useless when given to explain the title *Saving your file*. It simply repeats the message. A better goal description would include an explanation of what happens in saving a file or indicate the conditions or reasons for the action.

Goals are often codified in the (sub)title of a procedure. Titles and subtitles should not only make sense to the user, but they should also convey the big picture of the tasks that are involved in the use of a program or apparatus. In addition to being task oriented, titles should, therefore, convey the major tasks of the user. For example, in a text-processing manual, titles such as *editing and entering text, formatting text, spell-checking text,* and *finding and changing items in a document* clearly reflect some of the major structural components for this type of software. Titles that reflect the task structure can support users by offering scenarios and by supporting the different points of view users can take in task execution. In addition, they can help users locate information easily, when the manual is consulted for reference.

Farkas indicates that a title can be framed as a noun phrase (e.g., *button duplication*), gerund (e.g., *duplicating buttons*), root (e.g., *duplicate buttons*), or infinitive (e.g., *to duplicate buttons*) [2]. Gerunds are the classic choice because they convey a sense of process and work well over a broad range of designs. Infinitives can be effective as subtitles for closely related subgoals that should be clustered.

Fig. 1 summarizes the design guidelines that can be used for presenting the goal(s) component of a procedure.

Fig. 1. Guidelines for designing goals.

On breaking down a goal into subgoals:
- It is useful to break down the desired end state (goal) into two or more intermediate states (subgoals) when the users must execute many actions
- When users must memorize a task, create subgoals that require a maximum of 3 to 5 actions
- Subgoals should be treated in the same way as goals
- When two or more subgoals belong together, their (textual and visual) presentation should show their connectivity. In addition, there may be a need to add information about prerequisites and other subgoals

On describing a goal:
- Paraphrase the title
- Sell the goal

On presenting the title of a goal:
- Be task-oriented
- Reflect the task structure
- Be in gerund-form
- Present the most general action leading to the goal state

PREREQUISITES COMPONENT OF A PROCEDURE

Prerequisites are conditions that must be satisfied so that the user can achieve a task. A general prerequisite that you will often find in software documentation is that the user must know how to use the keyboard and mouse. General prerequisites are often discussed in an introductory chapter. Prerequisites in procedures are far more specific, and they usually belong to one of the following types: system states, user skills, and user knowledge.

Users frequently find themselves in the position of having to assure that they are in the right system state. That is, they must know or discover the starting position that is needed to be able to begin a task. In

software documentation, the first screen after opening up a program is an important starting position. It can be indicated by sentences such as: *You should already have opened your application* or *Open the program before you start working on the task*.

In addition, users occasionally may need to attend to the presence of additional materials (e.g., data, sample files, additional software, and plug-ins) needed for task execution. For example, documentation sometimes revolves around a single case, or file, whose contents change as users work through the chapters. To afford an easy access to all procedures, regardless of such dependencies, the designer may prompt the user to activate a sample file before starting a procedure.

Prerequisite user skills and skill learning are especially important in tutorials. Novice users can execute some tasks only when they already know how to perform other, more fundamental ones. It is undesirable to repeat the action steps for basic tasks endlessly, however. Likewise, users should know how to achieve basic tasks at a particular moment. To handle these obstacles the designer can use a technique called fading [4]. In fading, the support for the execution of basic skills tasks is gradually decreased. From the first full instruction onward, the user gets successively less complete reminders (see Table II). In addition, fading also often includes a

TABLE II
Fading stimulates learning by systematically reducing the procedural support for tasks

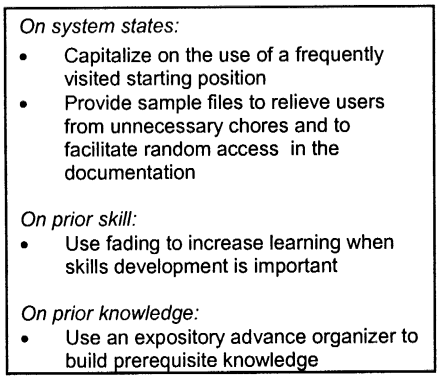
Version		
	1st	1. In the menubar click on File
		2. Click on Save
		3. Type a name for your file
		4. Click on the Save-button
	2nd	1. In the File-menu click on Save
		2. Type a name and click on the Save-button
	3rd	1. Save your file

change from predominantly procedural instructions toward conceptual information. There are no fixed rules as to how many reminders users need or how fast they know how to execute a task. Only testing can reveal what works and what does not. Empirical research from Leutner [5] indicates that fading is especially useful in the early phases of learning in which it contributes significantly to basic skills development.

Prior knowledge is the single most important human characteristic known to affect learning [6]. Prior knowledge helps users in directing and interpreting their experiences. In the absence of such knowledge, the designer may want to offer a substitute in the form of an expository advance organizer. By presenting explanatory information about how things work, the user's knowledge of a task is advanced. That new knowledge may make the task that follows more desirable and understandable. For example, a manual on Office 2000 explains that "with macro's you can execute tasks automatically. ... Macros are often linked to buttons on forms. ... you can create a button for generating a report ... and a button that allows you to return to the main menu" [7, p. 456].

Fig. 2 summarizes the design guidelines that can be used for presenting the prerequisites component of a procedure.

Fig. 2. Guidelines for designing prerequisites.



ACTIONS AND REACTIONS COMPONENT OF A PROCEDURE

The actions & reactions component is the focal point for all others. Users must act on the system. Only in this way can they be expected to develop their skill. Rather than discussing the actions of the user and the reactions of the system as two separate entities, we prefer to discuss the two in tandem because of their intricate relationship. The discussion of the component action and reaction concentrates on three critical design issues: (1) the balance between *let go* and *support*; (2) the use of screen captures to support vital user actions; and (3) the description of (a series of) action steps.

Balancing *Let Go* and *Support* The designer often needs to find the proper balance between direct instructions to act and invitations to explore. By balancing one against the other, the designer adapts to the user's tendency to explore. Another compelling reason for varying between *support* and *let go* is that such a mixture affords the development of strategic knowledge. Self-directed learning can then lead to deep-seated knowledge as the user learns not just *how to do it* but also *how it works*. The user acquires more strategic knowledge with a properly balanced approach [8]–[10].

An example is shown in Fig. 3. The "on your own" section in this figure does not merely ask students to explore. It cues them to consider what other goals they might want to pursue in relation to searching a text. The suggested goals for exploration are related conceptually and procedurally to the operations practiced. The user has just been thinking about forward search and, thus, is prepared to discover backward search. The exploration is also quite tractable. In this case, searching backward requires only two slightly different actions (i.e., positioning the cursor at another place, and selecting another option from the search menu).

Fig. 3. When users are invited to explore on their own they should have adequate prior knowledge and skills (slightly adapted version) [44].

Searching a text
You can position the cursor quickly to a word or part of a sentence by searching for this text.

1. Position the cursor at the beginning of the file
2. Go to the menubar and press twice on the → key
3. Choose the command FORWARD and press the ENTER key

WordPerfect asks what you want to search for. Check to see if the prompt **Search:** is on your screen.

4. Type any word(s) from the text
5. Press the F2 key

On your own: Searching text
The commands NEXT and PREVIOUS enable you to find out if the word you have been searching can also be found elsewhere in the text. You can find these commands under the SEARCH option. Try them and see.

Using Screen Captures to Support Vital User Actions In their taxonomy, van der Meij and Gellevij suggest that screen captures can support mental model development, switching attention, verifying screen states, and identifying and locating window elements and objects [11]. These ideas accord with Mayer's SOI-model [12], in which pictures, along with text, can help users select, organize, and integrate information in working memory and build a mental model or schema.

Screen captures can support mental model development when they convey the look and feel of a program and when they elucidate its underlying structure. Thus, they should depict system topology and component behavior [13]. A screen capture conveys system topology when it displays the important elements of a screen, such as a menu or an icon, and shows these in a wider context, such as a toolbar or the home page of a program. Component behavior requires that the main components of a picture must be labeled and that changes in the system state must be depicted or described. Research indicates that screen captures designed in this fashion can improve mental model development in a statistically significant way [14].

By their very nature, screen captures invite users to switch attention. The pictures clearly tell the user when it is important to look up from the manual to the screen. Research suggests that switching occurs regularly for users with moderate computer experience, even in the absence of screen captures. For these users, screen captures support switching acts only for complex tasks that benefit from repeated consults of the same picture [15].

Screen captures can also help users in verifying screen states. When users consult screen captures and discover that they are still on the right track, the pictures serve as positive feedback, which reinforces motivation. Especially for the novice user, this may be important to allay initial anxiety. Apart from checking progress, users can also use screen captures to verify whether the program has processed their input correctly. The special advantage of pictures compared to text is that the user needs merely to compare the computer screen with the depicted screen. The comparison is direct; there is no need for a mental transformation of a description. Research shows that offering a legible screen capture, in which the information to be verified is clearly cued, decreases time needed to verify that information and increases the ability to correct errors [16].

User interfaces are complex, especially when seen through the eyes of a novice. Among the multitude of menu options, windows, icons, and symbols, the user has no easy task in picking the element or object that is needed. Making the right choice hinges on two factors: (1) knowing which element to look for and (2) knowing its placement. Screen captures showing system topology and component behavior can help with this identification process as well as with locating a screen object [14].

Action Steps In its simplest form, an action step consists of a combination of a verb and noun. The verb tells the user what to do. The noun indicates the object involved. The basic action step thus tells the user how to act upon an object. The user should: *Press F7, Click the Enter key, Select Install, Lift the handset, Remove the Out tray, Switch off the computer,* etc. Considering their vital role, designers may want to signal this core part to make it stand out from additional information they present in the action steps.

If necessary, an action step should also inform the user of the whereabouts of the object. Examples of action steps with locators are: *Choose Download Fonts from the file menu, In the File Manager, select Up,* and *Unplug your computer from the ac outlet.* The use-order principle of Dixon [17], [18] suggests

that the preferred order is a presentation in which the locator phrase precedes the action information because it better accords with the user's action sequence. That is, the user must first go to the right location, which may actually be also an action that needs to be executed, before performing the action.

When a procedure involves more than one action step, and these steps are to be executed in consecutive order, then the designer must ascertain how to signal the relationship between these separate steps. Oddly enough this issue is not debated very often in the literature. However, the authors that do discuss it unanimously agree on the desirability of numbering each action step [19]–[21]. There is also some experimental support for this stance [22]–[24].

How Are Actions and Reactions Currently Presented? There was no proper balance between *let go* and *support* in 102 of the 104 procedures that we analyzed. Only direct support was given. In addition, the two cases with *let go* did not heed important conditions for engaging in explorations. That is, the context for exploration did not guarantee safe progress and, also, it was not assured that the user would have the necessary prior knowledge and skill.

The most striking finding on the use of screen captures in practice is that most of these pictures lack any form of signaling as called for by theory and empirical research. For example, 24 out of the 30 procedures with screen captures from the sampled software manuals contained no labeling or cueing and also did not convey changes in the system state. The designs of these screen captures were simply not optimized or well-integrated into the design of the action and reaction component in the procedure.

Theory and practice converge on many but not all design issues discussed for the action steps. That is, most manuals signal the verb-noun core. They do so predominantly by marking the key object that the user must manipulate. Numbered lists of action steps are also quite common with an 83% score for software manuals and a 75% score for hardware manuals. The presence of locators was frequently mentioned in the action steps, especially in software manuals. However, in most cases (72%), action descriptions preceded rather than followed locator information.

UNWANTED STATES COMPONENT OF A PROCEDURE, PART 1: WARNINGS

Warnings are given to prevent certain actions of the user or to alert the user to the presence of a risk. Industry has always given considerable attention to the design of warnings. International standardization organizations describe the directions for presenting warnings in some detail [25]–[28]. In addition, there is a sizeable body of research on the design of warnings.

Fig. 4 summarizes the design guidelines that can be used for presenting the actiona nd reaction component of a procedure.

Fig. 4. Guidelines for designing actions and reactions.

On balancing between let go and support:
- Balance direct instructions and invitations to explore to address the user's propensity to explore and to build strategic knowledge
- Prepare the user well before inviting explorations

On using screen captures to support vital user actions:
- Screen captures should explain the layout of the screen, describe its elements and show successive screen states to support mental model development
- Screen captures intended to support switching attention are most useful for complex tasks
- Screen captures must be legible to support verification processes
- Screen captures should cue screen elements and present these in a useful context (i.e. display a full screen) to support identification and location of screen objects

On describing (a series of) action steps:
- An action step should always include a combination of action – object (verb-noun)
- If there is additional information, signal the action – object (verb-noun) part
- When a series of action steps must be completed in consecutive order, each step must be numbered in sequence

The design guidelines for warnings in the literature generally are derived from models of communication and models of human information processing [29]. Information processing theories decompose the receiver's processing of a warning into stages of attention, comprehension, motivation, and behavior. We will use the "see-think-use" model that fits this perspective to ground the design guidelines for warnings. According to this model users must first perceive—**see**—the message, then they must understand it—**think**—, and finally they must act accordingly—**use**.

Stage 1: Seeing a Warning To help the user in perceiving a warning, it must be made conspicuous. It must stand out from other stimuli. The key here is salience. To attract the user's attention, designers can use contrast, highlighting, size, signal words (e.g., caution or danger), pictures, and location, among others. Some redundancy can be helpful here, especially a combination of text, signaling, and picture.

In technical documentation, the location of a warning is special in view of the difference between theory and practice. Some research suggests that warnings are

more effective when given "just-in-time" (i.e., within procedures) for tasks that the user is not familiar with [30]–[34]. In contrast, company policies may dictate the presentation of warnings in a separate section up front in the book. Occasionally, the dilemma of placement is solved by giving a warning twice (using a slightly different presentation format), once as part of a series of general warnings about usage and once as a specific warning within a procedure.

Stage 2: Thinking About a Warning Warnings must inform the user of various issues of hazard control. That is, a fully informed user must comprehend the hazard, know how to avoid it, and know the potential consequences of unsafe behavior. In addition to filling a knowledge gap, users may also need warnings as a reminder or cue to prevent them from forgetting to act safely at the critical moment. In short, warnings must be comprehensible and memorable. There are problems associated with using only textual information or only pictures for these functions. We concentrate the discussion below on the comprehensibility of warnings.

To indicate the nature of a hazard, industry has proposed the usage of standardized signal words. For example, the words *caution*, *warning*, and *danger* connote low to high levels of hazard [25]. This works well for the experienced user who knows the meaning of these cues. However, uninitiated users may be unaware of the distinction. For them, these words mainly give off a general alert, producing an overall impression of hazard [35].

Pictures are not as easily understood as is sometimes assumed. Standardization institutes have different criteria on the level of comprehensibility that they find acceptable. For example, the American National Standards Institute requires an 85% success criterion for pictures with no more than 5% critical confusion [25], whereas the European Organization of International Standards requires a 67% success criterion for just the picture [36].

A combination of text and pictures is preferable because the redundancy increases the saliency of a warning. In addition, text can compensate for some of the vagueness of the picture and vice versa. For example, the abstract nature of a text can be counterbalanced by the presence of an icon depicting the type of risk that the user may be running.

Stage 3: Using a Warning The ultimate goal of a warning is that the user acts as safely as possible under the given circumstances. The user should avoid certain actions or take precautions to minimize the risk. To achieve this, a warning must tell the user what to do or what to avoid. The guidelines for describing these actions are, by and large, the same as for regular instructions.

Recent studies challenge the conventional order of presenting the risk description before the instruction. For example the participants in the study of Maes, Maas, Van der Meulen, and Verbunt were asked to rate risk perception, tendency to comply, and naturalness of warnings with instructions before risk descriptions (e.g., *never pull out jammed papers from the printer by hand, the printer may get damaged*) and warnings with risk descriptions before instructions (*the printer may get damaged, never pull out jammed papers from the printer by hand*) [37]. The results clearly favored a presentation in which the instruction preceded the description of the risk. The authors suggest that an instruction—risk order of presentation is more effective because it emphasizes the instructive nature of warnings (also see [17], [18]).

Many people who have noticed and understood a warning still fail to comply. For example, Friedman's study reports a decline from 88% subjects seeing the warning, 46% reading it, and only 27% subjects who complied [31]. A warning should induce compliance. It should be persuasive so that, once the user knows how to act safely, the user also holds the correct attitudes and beliefs to do so. The factors in play here concern the user's assessment of the effort needed to act safely and the likelihood and severity of the hazard.

How Are Warnings Currently Presented? Table III presents findings from our inventory on what are probably the minimum acceptable standards for presenting warnings [38]. The first thing to note about the data is that the presence of warnings in the sample seems low. Only 20 of the 104 manuals included a warning.

TABLE III
How well do manuals satisfy the minimum acceptable standards for presenting warnings?

	Software manuals (n=8)	Hardware manuals (n=12)
The warning stands out on the page	62.5%	66.6%
There is a signal word	50%	66.6%
The hazard is described	25%	16.6%
The consequences of non-compliance are described	62.5%	50%
The instruction of what to do or to avoid	100%	83.3%

n = number of warnings found in the analyzed procedures

About sixty-five percent of the warnings are easy to perceive thanks to their presentation format (e.g., in italics, another font), a picture, or a signal word such as *Caution* or *Warning*. The percentage of unsignaled warnings in hardware manuals is high considering the risk that may be involved. Occasionally we saw that critical information that was hidden in the regular text (e.g., *Confirm the line voltage designated on the rear panel of the monitor* and *Discharge any static electricity from your body by touching any metal surface*).

The majority (60%) of the warnings was placed between the action steps rather than before or after these steps. This "just-in-time" placement is more common in hardware manuals than in software manuals (69.2% and 37.5% respectively). The different percentages for the factors risk description and specification of the consequence may have to do with our coding. We defined the risk factor as a state and coded the consequences as a type of (system) action. Even so, these two features of a warning were sometimes hard to distinguish from each other. Nearly all warnings stipulated what users must do or not do to avoid running a risk.

Fig. 5 summarizes the design guidelines that can be used for presenting the warnings component of a procedure.

Fig. 5. Guidelines for designing warnings.

On supporting seeing a warning:
- Make warnings salient
- Use a combination of text, signaling, and pictures to draw attention
- If applicable, follow company or international standards

On supporting thinking about (understanding) a warning:
- Beware of the fact that standardized signal words may not be understood as such by inexperienced users
- Use a combination of text and picture to improve comprehensibility

On supporting using a warning:
- Instruct the user on how to avoid or minimize the risk
- Place the instruction to act safely before the risk description
- Persuade the user to act safely

UNWANTED STATES COMPONENT OF A PROCEDURE, PART 2: PROBLEM-SOLVING INFORMATION

Mistakes are inevitable. No matter how hard designers try, the user is likely to be confronted with problems that can have a considerable impact on user motivation and acceptance of a program. In addition, users generally spend a considerable amount of time on problem-solving [39]–[42]. In short, it is well worth the effort to support the user with problem-solving information.

Minimalism has been among the very few design approaches that specifically addresses this issue. Within this tradition, a model for problem-solving information has been suggested that closely resembles the information processing model for warnings. Just like the see-think-use model this "detect-diagnose-correct" model consists of three main phases. In dealing with a problem the user must first experience—**detect**—the problem, then define—**diagnose**—it, and finally solve—**correct**—it (see also [43]).

Stage 1: Detecting a Problem To experience a problem the user must both see and assess it. In seeing, the user becomes aware of the presence of a problem. This awareness may be prompted by a system action such as an error message on the screen, an external act (e.g., power failure), or by the presence of problem-solving information in the documentation. It is important to catch a problem early to prevent it from getting worse and complicate correction. Just as in warnings the location of problem-solving information can be critical in this respect. Whenever possible, it should be given just-in-time [44].

Users do not deal with all problems they encounter. Some problems are ignored because they are mere inconveniences. For example, the user may experience problems working with styles, decide that they are not worth an exerted effort, and ignore these by and large. Whether or not the user decides to further pursue a problem thus partly depends on problem assessment.

Problem-solving information should be given when actions are error-prone or correction is difficult [44]. This guideline indicates that the factors likelihood and severity suggest where problem-solving information may be needed. Likelihood is closely tied to the user's prior knowledge. Problems often occur when the user's prior knowledge is insufficient, or the user's real-world experience conflicts with the way in which the program works [45]–[47]. For example, users frequently forget to create a text block before underlining existing text. They do so because they do not have to perform these actions in a noncomputer environment. The presence of problem-solving information is also desirable for complex problems. Complexity may be high when the consequences of certain actions are hard to understand or when the problem really needs to be fixed before the user can move on [41].

Stage 2: Diagnosing a Problem In this stage the user must articulate the problem and elect a course of action. Many people find it difficult to diagnose the problem. One of the ways to come to an understanding of the nature of the problem involves examining the system state. When the problem is cued by an error

message the user can study this message to find out what is wrong. Another possibility is that the user reflects about the actions that preceded the problem or considers the context in which it appeared.

Just-in-time problem-solving information facilitates diagnosis. A placement in context can help the user in understanding the problem and may give the designer a chance to present a specific rather than a general solution [48]. An additional benefit is that problem-solving information can be used for exploratory purposes [44].

Diagnostic information is optional. It is not a necessary ingredient for problem solving because the user can get away with studying only the components of detection and correction. It should be included in the help because it presents the designer with a chance to present conceptual information about the program that is likely to interest the user. That is, when people are experiencing a problem, they are often eager to learn more about it. By helping the user to understand the specific problem, the diagnostic information can contribute to mental model development (see Fig. 6).

Fig. 6. Use of the sections detection (problem), diagnose (cause), and correction (solution) to present problem-solving information.

Problem	***The keyboard does not work***
Cause	The keyboard is locked because the Ctrl-key has been pressed
Solution	Press the Ctrl-key again to unlock the keyboard

When a user cannot solve the problem without support, the question arises where to look for help. The user can consult the online help or look for references or pointers in the table of contents and index that are promising for the problem at hand. The user can also browse the main body text of a document to find relevant information. To facilitate this effort the designer can use text, pictures, and signals to draw the user's attention to the presence of problem-solving information.

Stage 3: Correcting a Problem Solving or correcting a problem can be a daunting task when the user receives a jargon-loaded error messages. Because error correction may require the user to delve deep into a system the use of some jargon sometimes cannot be avoid. On the whole, however, corrective information should be simple to understand and enable even the inexperienced user to solve the problem.

The nature of the problem can play an important role here. For well-defined problems information it is easier to use the help that is given than for ill-defined ones [49]. One study even found that about 50% of the college students participating in the experiment failed to extract the proper information for ill-defined problems when the relevant graphs and illustrations were presented to them [50]. This research thus indicates that the help may need to be attuned to the type of problem that the user is facing. More difficult problems may require more elaborate instructions on how to solve these.

How Is Problem-Solving Information Currently Presented? A striking finding from the inventory is the scarcity of problem-solving information. On a total of 104 procedures randomly drawn from 104 manuals, only 11 assisted the user in dealing with problems that they might encounter. In this respect very little has changed for the better when compared to an inventory reported about seven years ago [43]. The main conclusion from that study is still valid: the user is not given enough adequate support for handling problems.

Fig. 7 summarizes the design guidelines that can be used for presenting the problem-solving information component of a procedure.

Fig. 7. Guidelines for designing problem-solving information.

On supporting detecting a problem:
- Present problem-solving information as a distinct information type
- Present problem-solving information when actions are error-prone or when correction is difficult

On supporting diagnosing a problem:
- Decompose problem-solving information into the sections problem (detection), cause (diagnose), and solution (correction)
- Present problem-solving information right before, in, or immediately after the action steps

On supporting correcting a problem:
- Avoid as much as possible the use of jargon in problem-solving information when correction is difficult
- Give more problem-solving information when problems are more complex

CONCLUSION

The four components model of a procedure attempts to support practice by offering an easy-to-use framework that is firmly based on theory and research. The same is true for the design guidelines that accompany each component. For this purpose we occasionally presented existing models and findings in a slightly different way than in the research literature. This is an unavoidable consequence of our choices. Theory and practice are not always easily comparable or compatible. We welcome discussions about these choices, as they can help bring the science and art of designing procedures closer to each other.

REFERENCES

[1] H. van der Meij, P. J. Blijleven, and L. M. Jansen, "Codeboek voor het analyzeren van procedures in software— en hardware handleidingen," Twente Univ. Press, Enschede, The Netherlands, Internal Rep., 2001. [Codebook for the analysis of procedures in software and hardware manuals.].

[2] D. K. Farkas, "The logical and rhetorical construction of procedural discourse," *Tech. Commun.*, vol. 46, no. 1, pp. 42–54, 1999.

[3] J. L. Doumont, "Magical numbers: The seven-plus-or-minus-two myth," *IEEE Trans. Profess. Commun.*, vol. 45, pp. 123–127, June 2002.

[4] J. M. Carroll and H. van der Meij, "Ten misconceptions about minimalism," in *Minimalism Beyond the Nurnberg Funnel*, J. M. Carroll, Ed. Cambridge, MA: MIT Press, 1998, pp. 55–90.

[5] D. Leutner, "Double-fading support—A training approach to complex software systems," *J. Comput.-Assist. Learn.*, vol. 16, pp. 347–357, 2000.

[6] P. L. Smith and T. J. Ragan, *Instructional Design*, 2nd ed. New York: Wiley, 1999.

[7] B. Bruck, *The Essential Office 2000 Book*. New York: Prima, 1999.

[8] S. K. Bhavnani, "Beyond command knowledge: Identifying and teaching strategic knowledge for using complex computer applications," in *Proc. ACM CHI'01 Conf. Human Factors Comput. Syst.*, 2001, pp. 229–236.

[9] R. C. Thomas and M. R. K. Foster, "A pilot study of teaching the strategic use of common computer applications," in *Proc. Austral. User Interface Conf.*, vol. 23, no. 5, Australia, Jan. 29–Feb. 1 2001, pp. 85–92.

[10] S. Wiedenbeck, P. L. Zila, and D. S. McConnell, "End-user training: An empirical study comparing on-line practice methods," in *Proc. ACM CHI'95 Conf. Human Factors Comput. Syst.*, 1995, pp. 74–81.

[11] H. van der Meij and M. Gellevij, "Screen captures in software documentation," *Tech. Commun.*, vol. 45, no. 4, pp. 529–543, 1998.

[12] R. E. Mayer, "Designing instruction for constructivist learning," in *Instructional-Design Theories and Models Volume II: A New Paradigm of Instructional Theory*, C. M. Reigeluth, Ed. Mahwah, NJ: Lawrence Erlbaum, 1999.

[13] R. E. Mayer and J. K. Gallini, "When is an illustration worth ten thousand words?," *J. Educ. Psychol.*, vol. 82, pp. 715–726, 1990.

[14] M. Gellevij, H. van der Meij, T. De Jong, and J. M. Pieters, "Multimodal versus unimodal instruction in a complex learning context," *J. Exp. Educ.*, vol. 70, no. 3, pp. 215–239, 2002.

[15] M. R. M. Gellevij and H. van der Meij, "Screen captures to support switching attention," *IEEE Trans. Profess. Commun.*, vol. 45, pp. 115–122, June 2002.

[16] M. R. M. Gellevij, "Visuals in Instruction: Functions of Screen Captures in Software Manuals," Ph.D. dissertation, Twente Univ. Press, Enschede, The Netherlands, 2002.

[17] P. Dixon, "Plans and written directions for complex tasks," *J. Verbal Learn. Verbal Behav.*, vol. 21, pp. 70–84, 1982.

[18] ——, "The structure of mental plans for following directions," *J. Exp. Psychol.: Learn., Memory, Cognit.*, vol. 13, no. 1, pp. 18–26, 1987.

[19] S. Feinberg, *Components of Technical Writing*. New York: Holt, Rinehart and Winston, 1989.

[20] R. Krull, "Documentation for a physical world," *J. Tech. Writing Commun.*, vol. 24, no. 2, pp. 181–195, 1994.

[21] J. Price and H. Korman, *How to Communicate Technical Information. A Handbook of Software and Hardware Documentation*. Redwood City, CA: Benjamin Cummings, 1993.

[22] M. J. Floreak, "Designing for the real world: Using research to turn a target audience into real people," *Tech. Commun.*, vol. 36, no. 4, pp. 373–381, 1989.

[23] L. T. Frase, "Writing, text, and the reader," in *Writing: The Nature, Development, and Teaching of Written Communication: Vol 2: Writing: Process, Development and Communication*, C. H. Fredericksen and J. F. Dominic, Eds. Mahwah, NJ: Lawrence Erlbaum, 1981.

[24] R. F. Lorch and A. H. Chen, "Effects of number signals on reading and recall," *J. Educ. Psychol.*, vol. 8, no. 4, pp. 263–270, 1986.

[25] Amer. Nat. Stand. Inst., "Z535.1-5," in *Accredited Standards Committee on Safety Signs and Colors*. ANSI Z535.1–5, Washington, DC: Nat. Elect. Manuf. Assoc., 1998.

[26] Deutsches Institut für Normung, *Benutzerinformation. Hinweise für die Erstellung*. DIN 8418, Berlin, Germany: Deutsches Institut für Normung, Beuth Verlag, 1988. [User guides. Guidelines for their construction].

[27] Int. Org. Standard., *Instructions for use of products of consumer interest*. ISO 37, Geneva, Switzerland: Int. Org. Standard., 1993.

[28] Int. Org. Standard., *Safety Colors and Saftey Signs*. ISO 3864, Geneva, Switzerland: Int. Org. Standard., 1984.

[29] M. S. Wogalter, D. M. DeJoy, and K. R. Laughery, "Organizing theoretical framework: A consolidated communication-human information processing (C-HIP) model," in *Warnings and Risk Communication*, M. S. Wogalter, D. M. DeJoy, and K. R. Laughery, Eds. Philadelphia, PA: Taylor & Francis, 1999, pp. 15–23.

[30] J. P. Frantz, "Effect of location and procedural explicitness on user processing of and compliance with product warnings," *Human Factors*, vol. 36, pp. 532–546, 1994.

[31] K. Friedmann, "The effect of adding symbols to written warning labels on user behavior and recall," *Human Factors*, vol. 30, no. 4, pp. 507–515, 1988.

[32] J. A. Strawbridge, "The influence of position, highlighting and imbedding on warning effectiveness," in *Proc. Human Factors Soc. 30th Annu. Meeting*, Santa Monica, CA, 1986, pp. 712–715.

[33] A. Venema, "Produktinformatie ter preventie van ongevallen in de privésfeer. Gevaars-en veilighiedsinformatie op verbruiksprodukten," Institute for Consumer Research, SWOKA, Leiden, The Netherlands, Res. Rep. 69, 1989. [Product information for the prevention of accidents in the home and during leisure activities: Hazard and safety information on nondurable products.].

[34] M. S. Wogalter and S. D. Leonard, "Attention capture and maintenance," in *Warnings and Risk Communication*, M. S. Wogalter, D. M. DeJoy, and K. R. Laughery, Eds. Philadelphia, PA: Taylor & Francis, 1999, pp. 123–148.
[35] S. D. Leonard, H. Otani, and M. S. Wogalter, "Comprehension and memory," in *Warnings and Risk Communication*, M. S. Wogalter, D. M. DeJoy, and K. R. Laughery, Eds. Philadelphia, PA: Taylor & Francis, 1999, pp. 149–188.
[36] Int. Org. Standard., *General Principles for the Creation of Graphical Symbols*. ISO 3461-1, Geneva, Switzerland: Int. Org. Standard., 1988.
[37] A. Maes, A. Maas, I. Van der Meulen, and F. Verbunt, "Wanneer waarschuwen waarschuwingen?," *[When do Warnings Warn?] Taalbeheersing*, vol. 20, no. 2, pp. 126–140, 1998.
[38] M. S. Wogalter, "Factors influencing the effectiveness of warnings," in *Proc. Public Graphics*, Lunteren, The Netherlands, 1994, pp. 5.1–5.21.
[39] S. K. Card, Th. P. Moran, and A. Newell, *The Psychology of Human-Computer Interaction*. Mahwah, NJ: Laurence Erlbaum, 1983.
[40] J. M. Carroll and C. Carrithers, "Blocking learner error states in a training-wheels system," *Human Factors*, vol. 26, pp. 377–389, 1984.
[41] A. W. Lazonder, "Minimalist computer documentation. A study on constructive and corrective skills development," Ph.D. dissertation, Dept. Instruct. Technol., Twente Univ., Enschede, The Netherlands, 1994.
[42] A. W. Lazonder and H. Van der Meij, "Error-information in tutorial documentation: Supporting users' errors to facilitate initial skill learning," *Int. J. Human Comput. Studies*, vol. 42, pp. 185–205, 1995.
[43] H. Van der Meij, "Does the manual help? An examination of the problem-solving support offered by manuals," *IEEE Trans. Profess. Commun.*, vol. 39, no. 3, pp. 146–156, 1996.
[44] H. Van der Meij and J. M. Carroll, "Principles and heuristics for designing minimalist instruction," *Minimalism Beyond the Nurnberg Funnel*, pp. 19–53, 1998.
[45] *Handbook of Human-Computer Interaction*, M. Helander, Ed., Elsevier, Amsterdam, The Netherlands, 1998, pp. 67–85. J. M. Carroll, R. L. Mack, W. A. Kellog, "Interface metaphors and user interface design,".
[46] S. A. Douglas and Th. P. Moran, "Learning text editor semantics by analogy," in *Proc. ACM CHI'83 Conf. Human Factors Comput. Syst.*, 1984, pp. 207–211.
[47] S. J. Payne, "A descriptive study of mental models," *Behav. Inform. Technol.*, vol. 10, no. 1, pp. 3–21, 1991.
[48] P. Hunt and K. Vassiliadis, "No easy answers: Investigating computer error messages," in *Effective Documentation, What We Have Learned From Research*, S. Doheny-Farina, Ed. Cambridge, MA: MIT Press, 1988, pp. 127–142.
[49] P. Wright, "Quality or usability? Quality writing provokes quality reading," in *Quality of Technical Documentation*, M. F. Steehouder, C. Jansen, P. van der Poort, and R. Verheijen, Eds. Amsterdam, The Netherlands: Rodopi, 1994, pp. 7–38.
[50] J. T. Guthrie, S. Weber, and N. Kimmerly, "Searching documents: Cognitive processes and deficits in understanding graphs, tables and illustrations," *Contemp. Educ. Psychol.*, vol. 18, no. 2, pp. 186–221, 1993.

Hans van der Meij studies information-seeking, instructional design, minimalism, and usability testing. He is author or coauthor of more than 100 publications in scientific books and articles. His awards include STC's Frank R. Smith Outstanding Article Award in 1996 and 1998, and IEEE's Professional Communication Society's Best Transactions Article Award in 1997.

Mark Gellevij studies the use of visuals in software documentation and the cognitive processing of text-picture combinations. His Ph.D. thesis is a combination of articles on these topics, published in *The Journal of Experimental Education*, *STC's Technical Communication*, and the IEEE TRANSACTIONS ON PROFESSIONAL COMMUNICATION. The book is available at: www.tup.utwente.nl. Apart from his research, he teaches several courses on instructional theory and (motivational) design of instruction.