APPLIED THEORY

SUMMARY

- Explains the rhetorical implications of actions and states in various models of procedural discourse and in specific writer strategies
- Considers more flexible alternatives to the streamlined-step model

The Logical and Rhetorical Construction of Procedural Discourse

DAVID K. FARKAS

Procedural discourse refers to written and spoken discourse that guides people in performing a task—in other words, it is "how to" communication. User's guides for computer software, repair manuals for electronic equipment, booklets on assembling and using consumer products, online help systems, and oral instructions are all forms of procedural discourse. Often, technical communicators use the term "procedure" to designate a discrete unit of procedural discourse, the instructions for performing a single task. Not every form of procedural discourse, however, consists of distinct procedures.

In this study I systematically and thoroughly describe a set of relationships, a consistent logic, that I believe underlies all forms of procedural discourse. We see it in the standard forms of procedures that exist all around us and in more specialized forms of procedures as well. Amid all the variations of format and syntax and as much as computer technologies such as wizards change the presentation of procedures, this same logical structure is always in force.

The study also demonstrates some of the crucial rhetorical dimensions of procedural discourse, ways in which procedural discourse becomes more than logically structured information. I will use the term "model" to refer, in a general way, to procedures that seem to be most alike on the basis of such key characteristics as syntax, format, amount of detail, and use of computer technology.

While grounded in theory, this study is practical in its aims. The goal is to help writers craft effective procedures and make sophisticated decisions about which models of procedural discourse best fit their circumstances. In particular, I consider the strengths and limitations of what can be termed "streamlined-step" procedures, the model that dominates online help systems and is very widely used elsewhere.

THEORETICAL PERSPECTIVES

This study derives from two well-established theoretical perspectives: (1) human problem solving in the context of systems theory (Newell 1980) and (2) rhetoric, in particular the classical concept of source credibility and the modern concept of rhetorical roles (Ong 1975; Goodwin 1991; Coney and Chatfield 1996).

These perspectives make it possible to explain professional practice in a comprehensive, detailed, and directly useful manner, and they will remain central, I believe, to our understanding of procedural discourse even as they are extended and in some ways challenged by such emergent theoretical perspectives on procedural discourse as constructivism (Mirel 1998), feminism (Sauer 1993), and postmodernism (Cooper 1996).

Procedural discourse derives from purposeful human behavior

If human beings did not set about to accomplish tasks (in other words, to change things), we would not generate procedural discourse. Thus, procedural discourse is largely about telling someone who is in one set of circumstances how to transition to another set. In other words, at the most abstract level, procedural discourse describes system states and actions that change system states. The system states are these:

1. **Desired state:** The goal that is presented to the user. There may also be variations on this goal.
2. **Prerequisite state:** The state that is a condition for moving toward the desired state. This is often

Manuscript received 27 August 1998; revised 19 October 1998; accepted 20 October 1998.

specified at the beginning of a procedure so that the user can align his or her current state with the prerequisite state.

3. **Interim states:** States we enter as we move toward our goal. These are milestones and subgoals. We create or reach these states through our actions.
4. **Unwanted states:** States we wish to avoid. These stem from errors, system malfunctions, and conflicts with interrelated systems.

The actions are these:

1. **Human actions:** The actions we take to reach our goals.
2. **System actions:** The responses of systems to our actions (resulting in new states).
3. **External events:** Actions from outside the system (for example, a power outage) that may affect the system.

This interplay of actions and states takes place everywhere as human beings conduct their affairs. For example, if my goal, my desired state, is to climb Mt. Rainier (a complex natural system), one of many subgoals would be to reach Camp Muir by evening. Prerequisite states include decent weather, appropriate equipment, and reasonable physical conditioning. My actions will cause system responses, such as tamping down the snow on the trail or possibly dislodging a snow mass. External events, such as a storm, might force me to alter my plans. As I climb, I will be looking for milestones of my progress and even signs that I may have lost my way, an unwanted state. These milestones, plus the system responses and external events, collectively make up the interim states I will enter as I try to achieve my goal. Any guidebook on how to climb Mt. Rainier must have as its core, its logical skeleton, descriptions of these actions and states. Indeed, as I demonstrate below, descriptions of actions and system states make up the skeleton of all forms of procedural discourse down to the level of individual steps. For example, conditional steps, which are fundamental to procedural discourse, consist (almost always) of a description of a problem (unwanted state) that the user must test for followed by a description of the action(s) necessary to address the problem.

Taking a very different domain (subject-matter area) as a second example, consider an employee who wants to transfer to a new department. The employee has a clear goal state, takes actions to transition to that state, and will experience various interim states (different stages of approval). The employee might encounter and need to address impediments (unwanted states), such as a policy requiring special approvals if the employee has been with the company for less than a specified period. The policies established by the company are an administrative system. The transfer procedure the employee finds in the company's procedures and policies manual inevitably reflects that system and consists of descriptions of actions and states that will hopefully guide the employee through the system toward his or her goal.

Because procedural discourse reflects the underlying system, the nature of the domain greatly influences the actions users will need to perform and the states they will enter—and, hence, the nature of the documentation. For example, the procedures for a feature-laden system (such as a high-end software application) will tend to have many user-option steps, while procedures for an inherently trouble-prone system (such as a toxic-waste recycling plant that receives diverse and unknown materials) will tend to have more conditional steps. Procedures for tasks in which the interface is confusing—including tasks (like tying knots) for which there is no human-engineered interface—will require longer, more explanatory steps to specify actions and more frequent and careful descriptions of interim states to provide feedback. The influence of the domain is such that models of procedural discourse have evolved specifically to accommodate particular domains. For example, as we will see, flowchart procedures are optimized for domains with many conditions.

Procedural discourse is inherently rhetorical

Procedural discourse is not just descriptions of actions and states. Rather, procedures exist in a social context. We are communicating with others and guiding them (or hoping to guide them) though a task. In other words, procedural discourse, like all discourse, is always rhetorical in nature.

First and most obviously, procedures must be adapted to the users' backgrounds and information needs and to the particular circumstances in which they perform the procedures. The vocabulary must, of course, fit the audience and, likewise, the level of detail: for one audience, an unelaborated, high-level action statement might suffice; for a less knowledgeable audience, this action might need to be broken down into more specific actions. For one audience, extensive feedback will be tedious; for another, necessary. Stated differently, one aspect of the rhetorical design of procedures is figuring out how to best present the framework of actions and states to a particular audience.

Another rhetorical aspect of procedures is that to succeed, they must "sell" themselves, establish their own credibility. The user must be convinced that the procedures come from a fully knowledgeable and trustworthy source, people who respect the user's investment of time and energy. If not convinced, the user may well plunge ahead without following the procedures (or will seek out other procedures). Many procedural documents, notably trade books on using software applications, aggressively establish their (initial) credibility by means of promotional

writing on the front and back covers. Other procedural documents, such as a software vendor's own help system or manual, establish credibility simply through their professional appearance and the reputation of the people who produced them. Of course, the credibility that is generated through these means is readily dispelled if the documentation turns out to be a disappointment.

A third rhetorical aspect is "selling" the domain itself. If the user believes that the effort is too great for the reward, he or she may well give up on the whole enterprise. So, for example, Durack (1997) points out that vendors of sewing patterns emphasized in their instructions that home sewing was something a homemaker could easily succeed in. Similarly, a company's documentation will often "sell" the product it documents as the best means to work in the domain.

As explained by Ong, Goodwin, and Coney and Chatfield, procedures inherently dramatize (if only very faintly) an implied author (a persona) and an implied (or "mock") reader, a dramatized image of who the actual readers are presumed to be. Through the implied author and reader, procedures may set out to achieve a broad range of rhetorical functions including selling themselves and entertaining users.

For example, in striking contrast to online help systems and vendor-issued manuals, many commercial computer books—notably the "Dummies" series—establish and maintain throughout a vivid and elaborate implied author (something like the intrusive narrator in many Victorian novels) who attempts to engage, impress, and encourage a mock reader who is depicted as technically unsophisticated and adverse to learning about computers—a fictional entity that real readers may or may not accept ("I am very comfortable with technology. This book is not for me."). As we will see, many rhetorical strategies, such as trying to engage or persuade a user, will require a more verbose kind of writing that includes more than bare statements about actions and states.

HELP SYSTEM PROCEDURES AND "STREAMLINED-STEP" PROCEDURES

This study focuses on help system procedures. This is because help systems are extremely prevalent, because they have been an arena of innovation in procedure writing, and because their highly structured and terse construction makes them relatively easy to examine and explain. A standard help procedure is shown in Figure 1.

There are, of course, a great many procedures similar to that shown in Figure 1 that do not come from help systems or the software industry. Consider, for example, the print procedure from the instruction booklet for a telephone and the brief procedure stenciled on an industrial vacuum cleaner, presented in Figures 2 and 3.

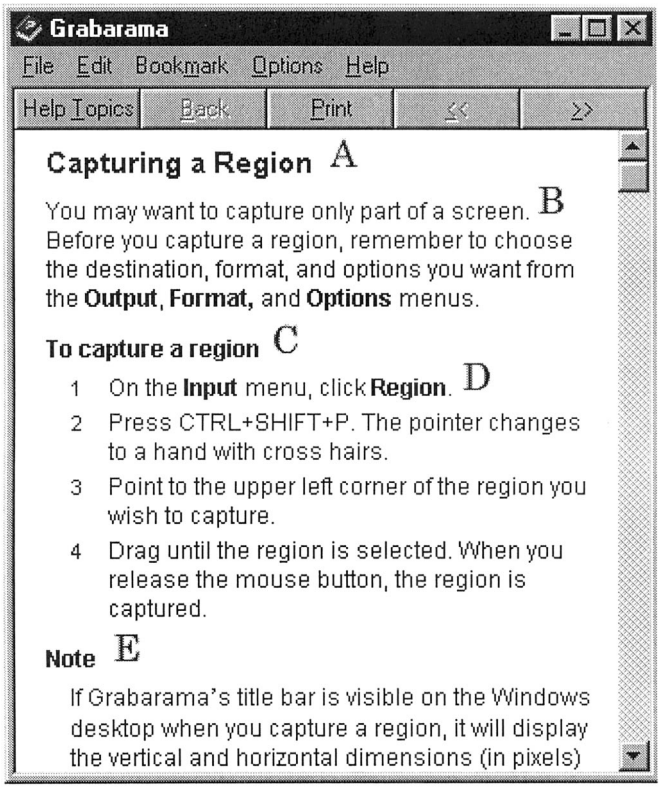

Figure 1. A typical help system procedure. The components (discussed in a later section) are A, title; B, conceptual element; C, infinitive subheading; D, steps; and E, notes. NOTE: The examples used in this study have been created or adapted for illustrative purposes and no longer represent the original documentation.

On-Hook Dialing

On-Hook Dialing offers the convenience of dialing without lifting the handset. All the telephone functions are available except actual conversation.

1. **Press the DIAL (*) button.**
2. **Dial the number (manually or automatically).**
3. **Monitor call progress through the speaker.**
4. **When the call is answered, lift the handset and begin talking. The speaker will cut out when the handset is lifted.**

Figure 2. A procedure from a telephone instruction booklet.

In contrast, in many domains, such as consumer product instructions, procedures consist of longer, more loosely constructed steps. Such a step, from a first-aid procedure, appears as Figure 4.

Clean Filter Daily

- Stop Motor.
- Rap on front filter housing with fist to shake dust from filter to rear cover.
- Remove rear cover, empty, and replace.

Figure 3. A procedure stenciled on a vacuum cleaner.

I call procedures such as the first-aid procedure "rich-step" procedures. This contrasts to the brevity and simple formatting of the "streamlined step" procedures we find in help systems and other domains. Because streamlined-step procedures are so familiar to us and because we will later examine very different procedures, it is worth taking a moment to note their defining characteristics:

1. Steps are brief. Each step consists of a brief action statement (or perhaps two short, closely related action statements). Some steps contain a supplementary sentence (or possibly two) that explains how the system responds to the action (feedback). Brief action statements are well suited to the domain of computer software and graphical user interfaces. In fact, to explain how to tie a knot or slide a knife against a whetstone (natural systems), longer action statements would be necessary. Similarly, brief feedback statements are hard to write in domains such as first aid in which the (human) system's responses to the actions taken are uncertain and hard to describe. Streamlined-step procedures may also include a brief example, explanation, or comment but never become "mini-essays," as we saw in the first-aid procedure.

2. Concomitant with the brevity of steps, the formatting is simple. While graphics and tables are often used, most steps are nothing more than simply formatted paragraphs. In contrast, some procedure models are formatted more elaborately (as can be seen in Figure 11 later in this article).

3. Action statements are built around an imperative verb. While there may be an introductory phrase or clause, the main action of the step is conveyed by an imperative verb that begins the main clause. Such syntactic variations as beginning a step with a descriptive statement—as we saw in the first-aid example—violates the streamlined-step model.

4. There is some, but relatively little, information preceding the steps. Often there is simply a title or else a title and a brief introductory paragraph (conceptual element). Significant amounts of supplementary information, however, may appear after the steps in the form of notes.

5. When streamlined-step procedures are presented online—as in the case of online help—hypertext jumps

5. Frozen tissues are painless and appear waxy and yellow. They will become swollen and painful and prone to infection when thawed. Do not re-warm rapidly. Thawing may require 15-60 minutes. For white people, thawing should continue until the pale blue tint of the skin turns pink or red. For black people, assess frostbite by the swelling and blistering of the skin. Reduction of swelling indicates alleviation of frostbite. Morphine or tranquilizers may be required to control the pain during thawing and should be administered under professional medical supervision.

Figure 4. A lengthy, loosely constructed step from a first-aid procedure.

are often used to "layer" the procedural information. In other words, conceptual overviews, definitions, and other information do not always appear in the procedures but can be displayed if the user clicks a button or hot text (Farkas 1998).

The streamlined-step model offers much to the user. The brevity and simple formatting contribute to legibility on the computer screen as well as efficient cognitive processing. The consistency of the design (a function of simplicity) enables users to form expectations about the format, which are fulfilled when readers return repeatedly to the help system. A further benefit is that keeping one's place in the procedure is especially easy—in contrast, say, to procedures that are formatted in lengthy paragraphs. Finally, this model embodies, in the best-case scenario, a highly functional style of work: a distinct "decision-action sequence." That is, the model ideally enables users to quickly decide (while reading the initial components) whether the particular procedure meets their needs, and if the procedure is appropriate, to move crisply from decision-making to carrying out actions.

Is the streamlined-step model a form of minimalism? No. Minimalism is a complex and sophisticated set of strategies for inducing users to engage as much as possible in inferencing and problem-solving (Carroll 1990). One important minimalist strategy is providing sparse information. Streamlined-step procedures, however, may be either complete and explicit or sparse and inferential.

Because of its brevity and simple format, the streamlined-step model offers major efficiencies in production, particularly so for large procedure sets such as help systems. These procedures are relatively easy to write, format, and localize. It is relatively easy to orient newly hired employees and contract writers to help systems that follow this model, and it is relatively easy for a large team of writers to follow a style guide and achieve consistency in their work.

ANALYZING STREAMLINED-STEP PROCEDURES

Expanding on Boggan, Farkas, and Welinske (1993 and 1996), I now offer a detailed analysis of the construction and function of streamlined-step procedures, with a focus on help systems. I approach the streamlined-step model as a set of mandatory, near mandatory, and optional components that function together to produce efficient procedures. Any given design following this model will make use of all or some of these components. Following the theoretical claim made above, we will find that these components are built around descriptions of actions and states. The components are these:

1. **Title** (nearly mandatory). Introduces and briefly explains the purpose of the procedure.
2. **Conceptual element** (optional). Elaborates on the title and provides other kinds of information that will enable the user to decide whether to perform the procedure.
3. **Infinitive subheading** (optional). Primarily used when multiple procedures appear under the same title. The infinitive subheadings make clear the purpose of each procedure.
4. **Steps** (mandatory). Consist of action statements, descriptions of the system's response, and related information that enables the user to execute the procedure.
5. **Notes** (optional). Present information that lies outside the main flow of the procedure. Most often, notes consist of less important information and are placed at the bottom of the procedure. Warnings, Cautions, and other important notes are placed more prominently.

In the world of help, the term *topic* (or sometimes *module*) refers to individual units of content that are physically distinct and separate: they appear in separate windows (scrolling or nonscrolling) or other display areas. Thus, we talk of "procedure topics," which contain one or more than one individual procedures. But there are also other kinds of help topics—overviews, definitions, and reference topics, for example—that supplement procedure topics and are linked to them via hypertext jumps. The idea of distinct *topics* applies less well to print documentation because print procedures often span page boundaries (that is, the print page is a less distinct unit than a window), but the notion of procedure topics is still useful in understanding not only help but procedural discourse in general.

The title

Most streamlined-step procedures and virtually all help procedures begin with a title. A title introduces a procedure or procedure topic, and in procedure sets, such as help systems (as opposed to a single stand-alone procedure), the title enables the user to distinguish a particular procedure from the others. In fact, in help systems, titles appear not just at the top of the individual procedures but as part of various navigation devices—tables of contents, online indexes, lists of jumps to related topics, and so forth—that users scan to find the information they want. From the perspective of actions and states, titles are (1) the most general action (the sum of all the rest) and (2) the action that represents the purpose of the procedure, the goal state that the procedure will result in.

There are four main ways to phrase titles:

Noun phrase: Macro automation
Gerund: Automating your work with macros
Root: Automate your work with macros
Infinitive: To automate your work with macros

Noun phrases are the least informative and are less often seen than the other phrasings, especially in the software industry. Gerunds convey a sense of process and work well over a broad range of designs. They are the "classic" choice. Roots are concise, but have the drawback of sounding like directives. This may be disconcerting to users when they scan a list of titles, for their job is to pick one, not perform each of the listed actions. (The root form does work well in the vacuum cleaner procedure shown in Figure 3 because the title is meant as a directive.) Infinitive phrases are effective, but because of the strong sense of causality conveyed by the preposition "to," infinitive phrases preclude the option of adding a conceptual element and require the writer to proceed immediately to steps (the "title-to-step" design). When writers are required to follow the title-to-step design throughout a procedure set, they may well encounter instances in which they cannot convey enough information in the title for the user to readily decide whether to carry out the procedure. Another design implication of phrasing titles as infinitives is that writers lose the option of clustering several procedures on one topic (explained below).

Conceptual element

The conceptual element, shown in Figure 5, consists of a paragraph or two of explanatory text that is provided when at least some users will need more information than is provided in the title to decide whether to execute the procedure. Often the job of the conceptual element is simply to elaborate on the

Viewing serials by category

Often the Holdings window will display a very long list of serials belonging to many disciplines. If you view serials by category (for example, economics), you can control the length of the Holdings window and view only the serials you are interested in. You can also view serials that belong to two or more categories.

Figure 5. A conceptual element that explains the purpose of the procedure.

title so as to clearly explain, in the user's own terms, the goal state the procedure will result in.

Note that the conceptual element in this example also states a variant goal: "You can also view serials that belong to two or more categories." There are tasks for which the procedure can do no more than describe in general terms the goal state and how to achieve it. For example, a help procedure explaining setting margins cannot address every possible margin setting the user might have in mind.

In addition to making clear the goal state—the purpose of the procedure—the conceptual element should indicate whether there are any prerequisites (prerequisite states) for carrying out the procedure (conditions that must be met before meaningful actions can be taken). For example, the conceptual element for the procedure "Connecting to an online information service" should make clear, before the user tries to do anything, that there must be a modem and communications software installed on the user's computer. The conceptual element must also point out any less-than-obvious but potentially undesirable implications of carrying out a procedure (unwanted states). For example, the conceptual element for a procedure on saving a file in ASCII format should make clear that most formatting will be lost. In some designs, noncritical conceptual information does not appear on the procedure topic but is available to users via a hypertext jump to an overview topic.

Infinitive subheading

The infinitive subheading provides a means to cluster two (or more) closely related procedures under a single title and (if it is needed) a single conceptual element. That is, multiple procedures can appear in one topic, as shown in Figure 6.

In Figure 1 we saw an infinitive subheading in a single-procedure topic. This infinitive subheading does provide a cue to the user that steps will follow directly, but this function probably doesn't justify including a component that essentially restates the title in infinitive form. So if there is a compelling role for the infinitive subheading, it is to make possible closely related procedures in a single topic.

Even though multiple-procedure topics are longer and more complex than single-procedure topics, they offer certain benefits. Often, the procedures help explain each other. That is, the user is apt to better understand the meaning of one procedure in the context of the other(s). Second, clustering closely related procedures reduces the total number of topics in the help system (or other procedure set). Fewer topics make navigation easier (fewer entries to scan in the table of contents, the found topics list of the index, and related topics lists) and make for easier maintenance of the help system.

Collectively, the title, conceptual element, and infinitive subheading can be regarded as the "decision-oriented" components. Now we turn to steps, which guide users through actions.

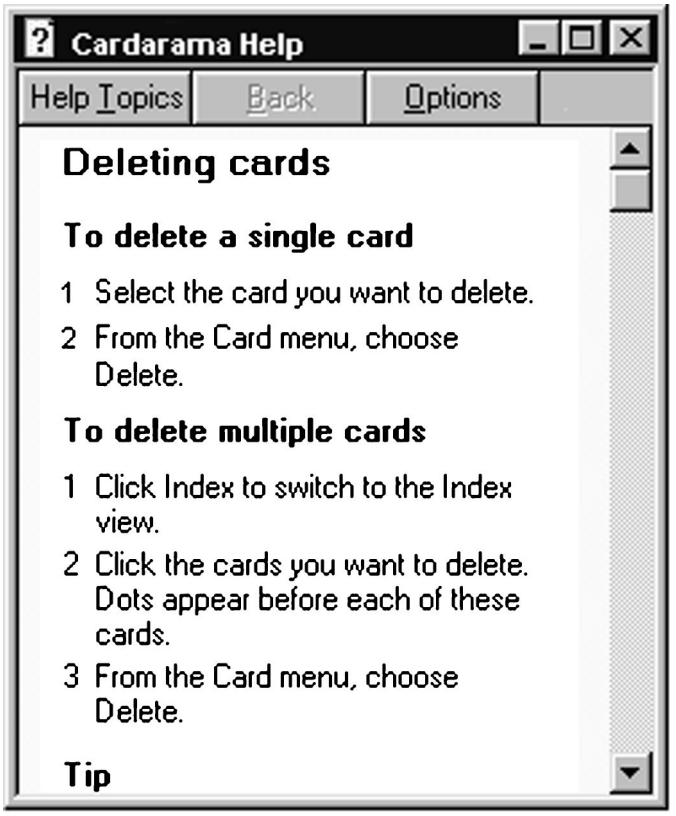

Figure 6. A procedure topic with two procedures, each introduced by an infinitive subheading.

Steps

In the streamlined-step model, the main element of a step is the action statement. Procedure writers, however, can use action statements in various kinds of steps.

Basic action steps Some steps are a direct, unelaborated statement of what the user is supposed to do:

> Click Format Painter.

> Select the file or files that you wish to mark.

Steps with a facilitating modifier Very often, users will benefit from more guidance than what is provided in the basic action step. The most common form of elaboration is to provide a modifier that assists the user in carrying out the action. I use the phrase "facilitating modifier" to distinguish these modifiers from those that introduce an option or condition (describe system states) or explain the purpose of the step. Most often facilitating modifiers are "locator phrases" that indicate the location of the object to be acted on.

On the File menu, click New.

But facilitating modifiers can express other relationships such as time or the manner in which an action should be taken, as shown below, respectively:

When the call is answered, lift the handset and begin talking.

Drag the handles so that they extend beyond the borders.

User option steps User option steps combine a description of a state—a variation on the procedure's main goal with a statement of an action—the means of achieving that variant goal:

To open a document so that it can be read but not changed in any way, click Read Only.

User option steps are akin to the "You can also . . ." sentences found in conceptual elements in that both point out a variant goal.

It is possible to create a procedure set without a single user option: each variant goal becomes a separate procedure very similar to the procedure it varies from. This multiplication of procedures, however, is very undesirable; and so, writers should use user option steps whenever feasible.

It is worth noting the progression from general to specific that we see in the streamlined-step model. This progression parallels the general-to-specific organization that characterizes many kinds of documents. The title is the most general component (broadest goal state), the conceptual element exists on the same level of generality as the title. The infinitive subheading is more specific, though it is still relatively broad because it conveys the purpose of a complete procedure. The variant goals expressed in user option steps are normally one more level down in specificity. Basic action steps and action steps with facilitating modifiers are the most specific; they don't describe any sort of goal, but consist of command syntax for achieving goals.

Conditional steps Conditional steps begin by directing the user to test for a condition. This condition is almost always a state (or symptom of a state) that is troublesome or threatening. Conditions range from some awkwardness in the interface, to a deep-level problem in the system, to a problem caused by an external system (such as a power outage). Conditional steps join a description of this special state with the action (or actions) the user will need to address this state if the user ascertains that it exists.

If your view of the tape is not expanded, double click the icon to display the tape volume.

If activity in the Primary unit reaches the yellow zone and the backup units are unavailable, click Emergency Shutdown.

When appropriate, the step may explicitly state the purpose of the action (how it addresses the condition):

If activity in the Primary unit is approaching the yellow zone, and the backup units are unavailable, click Phased Shutdown to initiate a rapid but "soft" shutdown sequence.

Many special states, in particular threatening ones, can be monitored by the system itself and communicated to the user via an error message (or some other system-initiated message). But there are still many instances in which the user must test for conditions and take action. Relatively unimportant conditions (those that will affect few users and will not affect any users unduly) are often relegated to supplementary, end-of-procedure notes. Particularly important conditions are given special emphasis as Warning or Caution notes.

Steps are not without rhetorical implications. User options suggest the power and flexibility of the product. But numerous user option steps (even when they are presented in table format) may push the user too far into decision-making mode, violate the decision-action sequence, and suggest design problems in the product and documentation. An occasional conditional step subtly dramatizes the implied author as a careful guide on the lookout for potential trouble. But numerous and complex conditional steps (although they may well be necessary) are taxing and disruptive. They force the user into problem-identification and problem-solving mode, violate the decision-action sequence, and suggest the vulnerability of the product and the limitations of the documentation—especially because documentation often offers only general or probable solutions to complex conditions. Finally, in some cultures, imperative verbs may strike users as overly authoritative.

Purpose explanations in steps The title, conceptual element and infinitive subheading all exist to make clear the overall purpose of a procedure. Even so, there are times when a more specific statement of purpose and other explanatory information are included within steps (and in notes). These are "local" purpose statements because they pertain to a single step or a few related steps:

Rap on front filter housing with fist to shake dust from filter to rear cover.

Click Bottom to complete the box.

In the first example (from the vacuum cleaner procedure shown in Figure 3), the "to" phrase makes clear why the action should be performed. People are often more willing to execute a step and often can perform it better when they understand its purpose. In the second example, the "to" phrase sums up the result of four clicks used to create a box with a drawing tool and provides closure for this portion of a lengthy procedure. (In the "phased shutdown" example in the previous section, we saw a local purpose statement used to explain the means of addressing a condition.)

It is important not to confuse local purpose explanations with user option steps. Syntactically, the difference is that user option steps begin with a "to" phrase, whereas the "to" phrase of a local purpose explanation follows the imperative verb. Semantically, the distinction is that the local purpose explanation does not open up the prospect of a variant goal state, whereas a user option does.

Feedback statements in steps Action statements are occasionally followed by a feedback statement. This is a brief description of the system's response to the user's action and the new state the system has entered. The most basic role of feedback is to provide verification (make clear that the user did the right thing and the system has responded properly) and to draw the user's attention to the result of the action. This kind of feedback is only needed occasionally in documentation for interactive systems, especially graphical user interface (GUI) computer systems, because these systems are themselves designed to provide clear feedback. But even in GUI systems, feedback is useful when the system's response is non-routine or hard to notice:

Click Display Sequence. Each display item highlights during the interval it is visible to the subjects.

Click Delayed Posting. New options appear below.

The following feedback statement (a rephrasing of a local purpose statement) serves an additional role; it functions as a milestone and provides closure after four closely related steps in a lengthy procedure:

Click Bottom. This completes the box.

At times, feedback statements explain only a system's change of state; no perceptible response is described:

Click Copy. The selected text and graphics will be copied to the Clipboard.

If the change of state has important and especially negative implications, it should be written as a note, probably a Caution or Warning. Even though writers may not use feedback statements often, their deft use helps guide the user smoothly and confidently through the procedure.

Notes

Notes make up the final component of the streamlined-step model. Their overall function is to identify and convey content that should appear outside the main flow of the procedure. Caution and Warning notes emphasize information. They appear just before the step they most closely pertain to or else in the conceptual element. They point out any highly undesirable implication (an unwanted state) of a condition or action. Supplementary, end-of-topic notes usually employ the headings "Note" and "Tip." They are most often user options and conditions that contain noncritical information and that will interest relatively few users.

While supplementary notes keep steps brief, they do add length and clutter to the procedure as a whole. Users are intimidated when they find a long list of notes at the bottom of the procedure. They wonder: Which of these are important enough to read? Which one of these may have the answer to my unresolved question? Thus, writers should consider whether the information they include in supplementary notes could perhaps be dispensed with altogether. Unfortunately, notes sometimes become a dumping ground for information that should be incorporated into the conceptual element or steps.

ANALYZING COMPLEX STEPS AND NOTES

Because of the enormous variety possible in human discourse, we inevitably find hybrids and borderline cases among these components. But even these exceptional instances can be understood and analyzed using the framework I have set forth. For example, it is relatively easy to analyze this somewhat unusual step:

On the File menu, click Open. The Open dialog box will generally appear. If the Drawing Modification alert box appears, click Save Changes or Discard Changes.

The first sentence is an action statement with a facilitating modifier (a locator phase). The feedback statement is unusual in that it presents a condition (If the Drawing Modification alert box appears, action must be taken).

This next step is also unusual in that it begins with the conditional presentation of a user option:

If the Beta_4 folder is available on the project server, you are authorized to install it. Drag the folder to your own hard drive, open it, and double click Install.exe.

The following complex note begins with a variant goal (resizing a datasheet column). The writer assumes the user knows what action to perform but explains an implication of this goal and does so in terms of what it does not do:

Note: Resizing the datasheet column doesn't change the defined size of the field in the table.

This kind of analysis can be very helpful when writers encounter unusual steps, either in their own draft procedures or in the drafts of other writers. An unusual step may be poorly written or it may be an appropriate response to an unusual situation, but in either case a writer should understand why the step is constructed as it is.

BEYOND THE STREAMLINED-STEP MODEL

The streamlined-step model is straightforward and efficient. Users benefit from its simplicity, economy, and consistency—characteristics that also contribute to efficient writing and production. This model is extremely prevalent, but it should not be taken as a standard or universal solution, but simply as the best strategy for particular circumstances. Now we will consider other circumstances and the models that have arisen to address them. First, we will look at models quite different from the streamlined-step model (though they share many of the same components). These are playscript procedures, flowchart procedures, wizards, and interface annotations. (I exclude Information Mapping™ from this discussion because the considerable strengths of this document format and development methodology pertain to information design generally rather than to procedural discourse in particular. See Robert E. Horn, *Mapping hypertext*, Lexington, MA: Lexington Institute, 1989.)

We will see that the function and benefits of these models are readily revealed by the same framework of actions and states we applied to streamlined-step procedures. Then, we will consider two further models of procedural discourse: rich-step procedures and paragraph-style procedures. We will see that the freedom provided by these models allows for a greater variety of cognitive and rhetorical strategies.

The playscript procedure

The playscript procedure, shown in Figure 7, arose to document tasks in which more than one person takes part (Matthies 1977; Barnett 1993). Very often these are administrative tasks such as processing a bank loan or placing an employee on disability leave.

The distinctive feature of this model is that the procedures are formatted like the script of a play, so that each person's actions in the overall task are separately listed,

Request for New Facilities	
Requester	1. Completes form 347, "Facilities Request," in three copies.
	2. Sends 2 copies to Facilities Engineering.
Facilities Engineering	3. Assigns job number to both copies of form 347, indicating when preliminary action will be taken.
	4. Returns copy of form 347 to Requester.
Requester	5. In case of a delayed response or any other inquiry, refers to job number of request
Facilities Engineering	6. Reviews and prioritizes all requests, allocates available funds, and issues a work order for approved projects.

Figure 7. A playscript procedure.

somewhat as a script shows each actor and the actor's lines. Not only is each individual shown what to do, but each can see his or her role in the context of the complete procedure. The framework of actions and states is easy to see in this model: for example, most of the steps in this example are basic (unmodified) steps. Step 5 is a conditional step. There are no user options in this example, but a writer could create one easily enough by beginning a step with "If desired."

Flowchart procedures

As I have noted above, certain domains (such as a toxic-waste disposal facility) inherently present a great many conditions, a situation which complicates documentation. The streamlined-step model (and other kinds of step-based procedures) will not work well here: there can be no distinct transition from decision-making mode to action mode, for the user is threading his or her way through branching steps from beginning to end. In fact, there is little point to a list of steps, for the user will need to jump around a great deal within the list. As in the case of the playscript procedure, these circumstances prompted the emergence of a more appropriate procedure model.

Flowchart procedures are specially adapted to represent complex branching. Their special format graphically shows the decision points and the pathways that users must follow through the branching logic of the procedure. A flowchart procedure for a diagnosis and repair task is shown in Figure 8.

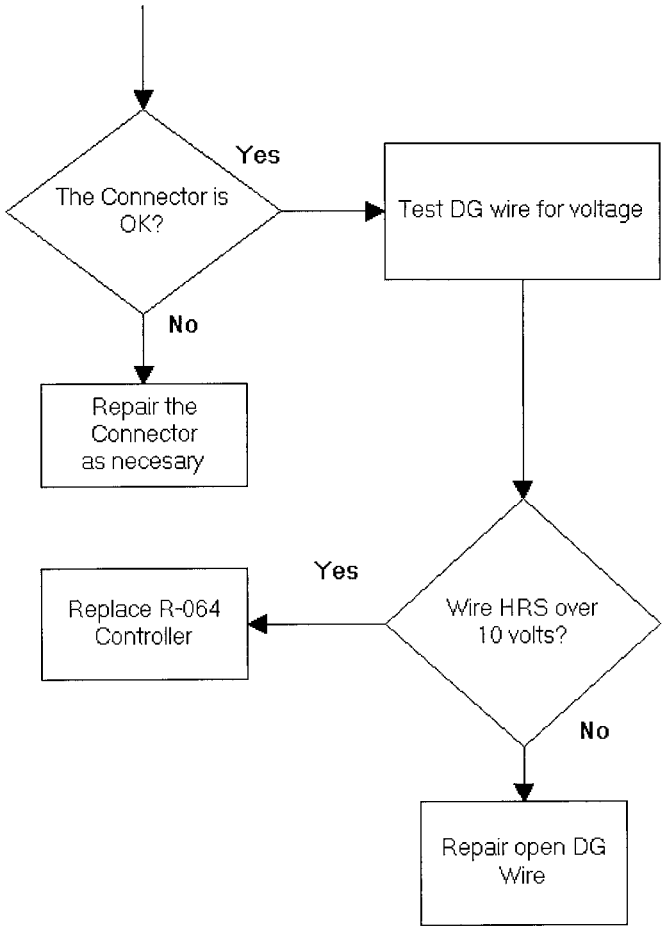

Figure 8. A flowchart procedure.

Wizards

Wizards exploit the power of the computer to act on behalf of the user and to simplify and standardize both the presentation of information and the means of performing actions. In a typical software wizard, each wizard panel asks the user to click an option button to indicate a choice among multiple options. (The user has already chosen a general goal by entering the wizard.) By executing the action and displaying the next panel, the wizard shields the user from the complexities of the software product's regular interface. Non-software wizards—which range from automatic teller machines to hardware-store kiosks that guide users through designing a backyard deck—work in a similar manner.

In addition to carrying out the action portion of user option steps, wizards often simplify conditions—either by silently resolving issues or by ascertaining for the user that the condition is present and proposing a course of action. So, for example, a wizard might tell the user that there is an older version of a particular file on the user's hard drive and propose overwriting it. Finally, wizards simplify branching. If the user chooses an option or responds to a conditional statement in a manner that makes later options or conditions irrelevant, the user never even sees the irrelevant options and conditions. For tasks that can be "wizardized," wizards are a compelling alternative to streamlined-step procedures, flowchart procedures, and some other non-computerized models of procedural discourse (Horton 1993).

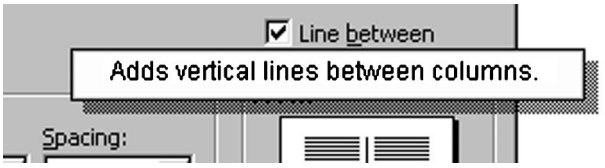

Figure 9. Balloon-type (What's this?) help.

The interface annotation model (including balloon-type help)

The interface annotation model encompasses a broad variety of fairly disparate designs. The model includes the balloon-type (or What's This?) pop-up messages found in many online help systems. The model's defining feature is that procedural information is located close to the controls of a human-engineered interface. So, for example, as shown in Figure 9, if a user wishes to create a multicolumn document and is pondering the somewhat cryptic check box label "Line between" in the Columns dialog box, the user need only right-click the check box to receive a brief annotation explaining its purpose:

If that particular control turns out not to match the user's purpose, the user can quickly display annotations on the other controls. Notice that this particular annotation assumes that users will know how to perform the correct action (to click on this control). It is also possible for the writer to explain both the purpose and the action:

> To add vertical lines between columns, click this box.

What you will not find is any sort of locator phrase (a form of facilitating modifier). Why? Because both the strength and limitation of this kind of documentation is that the user must find the relevant portion of the interface. Interface annotations, therefore, are not necessarily ideal for novices; however, users who (1) are able to locate relevant dialog boxes on a GUI interface and (2) do not need detailed explanations can work much more quickly with balloon-type help than by finding the correct procedure topic in a table of contents or index and then reading steps.

For a non-software implementation of this model, consider the descriptive labels that are often stenciled directly

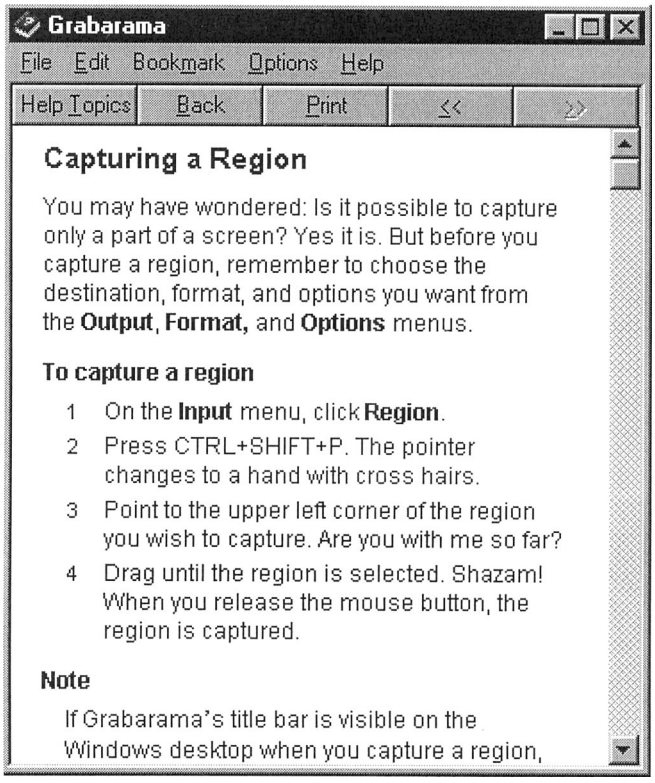

Figure 10. A streamlined-step procedure with a vivid implied author.

on such electromechanical controls as switches, knobs, valves, and levers. For an experienced, knowledgeable operator, these labels are more convenient than looking through a manual for the appropriate procedure. Manuals in which steps appear as numbered callouts on diagrams of the interface are a hybrid of the streamlined-step model and the interface annotation model.

Like the streamlined-step model, these models have their own rhetorical implications and possibilities. For example, the playscript model, potentially at least, can convey a sense of efficient organizational activity and cooperation among coworkers. Likewise, wizards are sufficiently powerful to evoke a sense of wizardry in at least some users, especially if the designer chooses to exploit the wizard metaphor with, say, a graphic of a wizard with conical hat and wand.

FURTHER RHETORICAL STRATEGIES AND OTHER MODELS OF PROCEDURAL DISCOURSE

There is no limit to the kinds of rhetorical strategies achievable in procedural discourse. Although streamlined-step procedures generally project nothing more than subdued professionalism, it is not impossible, even within this terse and rigid model, to create a vivid implied author. We see this in Figure 10, an exaggerated variation on Figure 1.

The rhetorical effects shown here, though gimmicky, might work in some circumstances. In particular, the sentence "Are you with me so far?" might function well if used sparingly in a procedure set with some lengthy procedures. Effects like these often appear in the very popular "Dummies" series of computer books.

Various goals, both rhetorical and cognitive in nature, may, however, require more freedom than the streamlined step model allows. Therefore, we will consider two procedure models that are alternatives to the streamlined-step model. These are the "rich-step" model and the "paragraph-format" model. (One can imagine corresponding variations on some of the other models described above, though in practice they are rarely found.) We will now consider the rich-step and paragraph-style alternatives to the streamlined-step model.

The rich-step model is simply a less restrictive variation on the streamlined-step model. Such procedures do not follow a highly specific set of conventions, and the steps are typically longer and syntactically more varied, as we saw in Figure 4 (the excerpt from the first-aid procedure).

Paragraph-format procedures give up entirely on separately formatted steps. Although some designs consist of a succession of imperative-verb sentences, this model usually employs a still wider range of syntactic options than rich-step procedures.

In both models, we often see simply sloppy, undisciplined procedures (often for consumer-product instructions) in which writers simply clump procedural information into wordy, disorganized steps. But there are also many fruitful uses of the extra freedom that these models allow.

Tutorial documentation is one. Tutorials, which often combine the cognitive goal of retention and the rhetorical goal of generating confidence in timid users, often adhere to one of these models because the writer includes frequent explanations to promote concept-building as well as previews, reviews, very extensive feedback, questions, encouraging comments, and other special elements. (They may also follow a hybrid model that intersperses streamlined steps and explanatory paragraphs.)

In the case of lengthy procedures, the rich-step model provides an opportunity to chunk several related steps into one longer step, a strategy that benefits users cognitively because each of these longer steps usually corresponds to a distinct part of the overall task. Often, the first sentence of a rich-step procedure is a broadly stated action that (like the topic sentence of a paragraph) encompasses the entire (chunked) step. We see this in the first sentence of the step shown in Figure 11. In this example, from a printed user's

3) Select the Palette

Paintarama offers two types of 256 color palettes. You select a palette by clicking on it in the Palette Group box.

- An optimized color palette contains the 256 best colors for recreating the active image
- The standard palette covers the full color spectrum and includes the sixteen standard Windows colors.

If you want to recreate multiple images with the same palette, select the standard palette. Otherwise, select the optimized palette because it will yield better image quality.

4) Select the Reproduction Method . . .

Figure 11. A multilevel, rich-step procedure.

Friends, there has been an ongoing problem with the finishing stapler in our copying machine. You may be part of the problem without even knowing it. When the "talking display" asks you to add staples, the machine is not in fact entirely out of staples. There is still a little stub of staples about 1/4 to 1/2 inch long. Before you add any more staples, it is necessary to remove this stub with a little tool expressly for this purpose. You will find this tool hanging from a hook inside the copier's front cabinet at the far left. After you have removed and disposed of this stub, you can then add a new row of staples, which must be in ONE PIECE, or the stapler will jam. Thanks.

Figure 12. A paragraph-format procedure that works hard at persuasion.

guide, there is an additional strategy at work. This step entails only one easy-to-execute action—clicking an option button to choose one or the other palette. The challenge for the user is to make the right choice. The writer, therefore, treats the action as an incidental issue and focuses and formats the step around the concepts underlying the decision. This strategy is better than the streamlined-step alternative, a conceptual paragraph (without any special formatting) followed by a single step. Also, using a multilevel format exploits one of the strengths of the print medium.

Writers may also be wise to diverge from the streamlined-step model when they document complex natural systems (such as the first-aid procedure) in which descriptions of actions and states require several sentences.

The streamlined-step model may be inadequate for those intent on sophisticated persuasive strategies. Below, in Figure 12, is a paragraph-format procedure written casually, but skillfully, by an office assistant with an MFA in creative writing.

Here, the rhetorical objective of motivating the audience to pay heed and cooperate is as important as explaining what must be done. The writer, therefore, dramatizes a warm, appreciative implied author who is on very good terms with her helpful coworkers. Part of this rhetorical strategy is to create a subtle drama built around humorous criticism of the copier: this advanced machine with its sophisticated "talking display" produces misinformation that causes problems for humans. Humans, then, must work together to prevail over the subversive machine. Goodwin characterizes this strategy as "emplotting the reader."

The syntactic variety and the lack of visually distinct steps reduce the efficiency of the copier procedure, but this is a reasonable trade-off, especially since the actual task is simple enough that most users will not need to consult the procedure while performing the task. All this rhetorical subtlety would be lost if the writer had chosen to create an admittedly briefer and more efficient streamlined-step procedure. Along the same lines, much of the humor and liveliness found in the "Dummies" series of computer books requires rich-step and paragraph-format procedures.

Because streamlined-step procedures are so familiar, they often seem to be a default choice for writers, an unexamined preference over rich-step procedures, paragraph-format procedures, and other potentially effective models. While not forgetting the efficiencies of this model both for users and producers, technical communicators should not become so conditioned to it that they fail to adequately consider the full range of alternative models or the possibility of their own innovations.

CONCLUSION

We see, then, that the framework of actions and states underlies not only the very prevalent streamlined-step model (where the framework is most easily seen) but all forms of procedural discourse. The framework clarifies the operation of procedural discourse and helps us write effective procedures. In addition, this framework helps us choose among models. For example, the presence of numerous conditions suggests the use of a flowchart procedure or a wizard whereas domains in which actions and feedback are especially difficult to describe might necessitate rich-step procedures.

At the same time, all procedures are rhetorical in nature, and the range of rhetorical effects and strategies is broad indeed. Various models of procedural discourse have inherent rhetorical implications and make possible certain rhetorical strategies. For example, the relatively unconfining nature of the paragraph-format model makes possible such strategies as emplotting the reader. Finally,

we can see that procedural discourse is complex, much more complex than it often appears at first glance. Thus, a wide-ranging, highly synthetic understanding of procedural discourse is necessary if we will achieve the best possible professional practice and advances in research and theory. TC

REFERENCES

Barnett, Robert. 1993. *Practical playscript*. Canberra, Australia: Robert Barnett and Associates.

Boggan, Scott, David Farkas, and Joe Welinske. 1993. *Developing online help for Windows.* Carmel, IN: Sams Publishing.

Boggan, Scott, David Farkas, and Joe Welinske. 1996. *Developing online help for Windows 95*. Boston, MA: International Thomson Computer Press.

Carroll, John M. 1990. *The Nurnberg funnel: Designing minimalist instruction for practical computer skills*. Cambridge, MA: MIT Press.

Coney, Mary B., and Carl S. Chatfield. 1996. "Rethinking the author-reader relationship in computer manuals." *The journal of computer documentation* 20, no. 2:23–29.

Cooper, Marilyn M. 1996. "The postmodern space of operator's manuals." *Technical communication quarterly* 5, no 4:385–410.

Durack, Katherine. 1997. "Patterns for success: A lesson in usable design from U.S. patent records." *Technical communication* 44, no. 1:37–51.

Farkas, David K. 1998. "Layering as a 'safety net' for minimalist documentation." In *Minimalism beyond the Nurnberg funnel*. J. M. Carroll, ed., pp. 247–274. Cambridge, MA: MIT Press.

Goodwin, David. 1991. "Emplotting the reader." *Journal of technical writing and communication* 21, no. 2:99–115.

Horton, William K. 1993. "Let's do away with manuals before they do away with us." *Technical communication* 40, no. 1:26–34.

Matthies, Leslie H. 1977. *The new playscript procedure.* Stamford, CO: Office Publications.

Mirel, Barbara. 1998. "Applied constructivism for user documentation: Alternatives to conventional task orientation." *Journal of business and technical communication* 12, no. 1:7–49.

Newell, Allen. 1980. "Reasoning, problem solving, and decision processes: The problem space as a fundamental category." In *Attention and performance,* vol. 8. R. S. Nickerson, ed., pp. 693–718. Hillsdale, NJ: Lawrence Earlbaum Associates.

Ong, Walter. 1975. "The writer's voice is always a fiction." *PMLA* 90:9–21.

Sauer, Beverly A. 1993. "Sense and sensibility in technical documentation: How feminist interpretation strategies can save lives in the nation's mines." *Journal of business and technical communication* 7, no. 1:63–83.