

# A Why-What-How Tool for Development and Documentation of Operating Procedures

David G. Novick  
European Institute of Cognitive Sciences and Engineering  
4 Avenue Edouard Belin  
31400 Toulouse, France  
novick@onecert.fr

## ABSTRACT\*

DSTOP, the Design Support Tool for Operating Procedures, is a relatively simple software tool for support of designers of new interfaces and their procedures for use. DSTOP is based on two complementary models: the documentation coherence maxims and the situated-act model, which distinguishes domain actions from interface actions. Use of the tool involves the writing of operating procedures using a kind of grid, where the operating procedure runs from top to bottom and is divided horizontally into why, what and how elements. The *why* element represents the operating procedure's goal; the *what* represents the operating procedure's situated act; and the *how* represents the interface action(s) required to effectuate the act. The tool has been used to develop prototype operating procedures for computer-based aircraft cockpit interfaces. DSTOP adds new functionality associated with the coherence maxims, including explicit representation of variants of terms, and explicit tracking and display of procedures (and their why-what-how components) in which any of the variants is used.

## Keywords

Authoring tool, operating procedures, rationale, acts, actions, coherence maxims

## 1. INTRODUCTION

Developers of new interfaces, particularly for safety-critical applications such as aviation, do not currently have much tool-based support for creating the content of the procedures for using these interfaces. Ideally, designers would have means to co-develop the interface and its documentation so that the each is consistent with, and informs the design of, the other. Recognizing this need, Aerospatiale Matra Airbus launched a four-year project to provide methods and tools for co-design of interfaces and documentation. At Aerospatiale, interface designers and human-factors specialist work with test pilots, line pilots and instructor pilots in a participatory design process. So as part of this project, Aerospatiale looked to create a new tool to support this process. This paper reports one of the project's central results.

There are tools available for checking compliance with standards for controlled-languages, but these do not really address the issue of what the language is supposed to represent. Tools also support consistency in documentation (e.g., Novick & Juillet, 1998) but this is again more form than substance. And the developers of tools that automatically generate manuals from specifications (e.g., Barth, 1997) do not claim that they produce

realistically usable manuals or procedures. The lack of support for the content of operating procedures is due, in large part, to the fact that writing operating procedures remains a craft based on domain knowledge and experience.

Methodologies for development of operating procedures, at least in aviation, tend to be at a higher level than the formulation of specific steps. For example, the most prominent approach, Degani and Wiener's (1997) "Four P's" model incorporates the organization's philosophy of operations, their business policies, procedures that effectuate the operations consistently with the policies, and the crews' actual practices on the flight deck. Degani and Wiener's emphasis on documentation and communication is important because procedures do not have an embodiment outside of documentation communicated to users; they exist only by virtue of knowledge and use. Within the design process itself, however, Degani and Wiener's model does not offer specific guidance on to write the actual procedures.

Similarly, an approach proposed by Drury and Rangel (1996) for reducing automation-related errors in aircraft maintenance and inspection concentrates on identifying error types and opportunities in the procedures. It starts with a set of underlying practices: describing functions neutrally with respect to whether the function will be automated, including users in the design team, and maintaining awareness of technical human factors. With these factors in place, the approach has four steps: listing system functions, listing alternative allocations for each function, listing errors for each function, and system integration. Again, while useful for design and redesign of systems, this approach lacks specific guidance for writing the procedures that make the system work.

Accordingly, this paper reports an effort to create a documentation design-support tool that

- Supports the co-design process for safety-critical interfaces and their documentation,
- Embodies a design model for operating procedures,
- Supports consistency in the documentation, and
- Encourages use and participation in adaptation.

The result is the Design Support Tool for Operating Procedures (DSTOP), a relatively simple software tool to support engineers and writers who develop and document operating procedures. The tool embodies a *why-what-how* design approach based on the situated-act model (Novick & Perez-Quiñones, 1998). The tool complements the why-what-how approach through support of key aspects of the documentation coherence maxims. And the tool is light-weight (i.e., user-extensible and non-coercive), so it reduces some of the chief obstacles to user adoption. The tool has been used to develop prototype procedures for computer-based aircraft cockpit interfaces.

---

\* Author's current address: Department of Computer Science, University of Texas at El Paso, El Paso, TX 79968-0518, USA, Novick@cs.utep.edu

This paper will discuss the why-what-how approach and its underlying situated-act model, briefly introduce the coherence maxims, describe DSTOP, and explain a number of examples of operating procedures developed using the tool.

## 2. BACKGROUND

DSTOP derives from two principal sources. First, DSTOP builds on and extends the situated act model, which provides design guidance for authors of interfaces and their documentation by relating users' acts to their context. Second, DSTOP uses the coherence maxims that served as key concepts in the design of the prototype documentation authoring system for the COHERE project. I will briefly review the main ideas of these two sources as background for discussing the design of DSTOP.

### 2.1 Situated Acts

DSTOP embodies a design model based on the notion of situated acts (Novick & Perez-Quiñones, 1998). The model extended Suchman's (1987) pioneering work on situated action by abstracting from actions in the interface to acts in the domain context. The difference between acts and actions is analogous to the difference in speech-act theory (Austin, 1962) between illocution and locution (i.e., between the force of an expression and the formulation of the expression).

Situated acts help express what users are supposed to do by relating the behaviors prescribed in operating procedures to element of the context that give the behaviors meaning. Situated acts can be seen as more concrete than speech acts because they are expressed in terms of domain concepts (e.g., climb) rather than putatively universal meanings (e.g., request). The acts are situated in the sense that they do not have meaning outside the context of the interaction itself; they are the logical thing for the user to do under the circumstances. Indeed, the principal purpose of expressing interaction at the act level is that the salience of the situation is increased. That is, the state of the interface may be observable at the action level (e.g., a new prompt appears), but the state of the underlying system is a higher-level construct that provides a deeper level of understanding. While every action is--by definition--situated, the salience or meaning of the situation at that level may not be optimal. So the idea of situated acts is to "lift" the actions into acts that have a higher degree of situational meaning.

For DSTOP, I extended the situated-act model by considering intention. That is, above the act level, there is a rationale for the act. In speech-act terms, this is analogous to the intended perlocution of the act. So what we have, then, is a three-tier approach. At the core task level, we have a representation of situated acts that serve as communication goals, independent of specific interface style or even dialogue style. At the more concrete level, we have the interface itself, where design decisions help give shape to the interface. This level is specific to each interface design. And at the more abstract level, we have the goals that provide the rationale for the acts that compose the domain task.

### 2.2 Coherence Maxims

The second principal source influencing the design of DSTOP is the set of coherence maxims that led to the COHERE prototype authoring system for documentation (Novick & Juillet, 1998). The COHERE prototype had features that supported, mostly automatically, all three maxims and the corollary:

**Coherence of meaning:** Integrity of semantic relationships

M1 Semantic relations should not be changed unless the change is intended

**Coherence of reference:** Integrity of consistency and differentiation in referring to domain entities, actions and relations

M2 References to the same thing should appear the same.

M3 References to different things should appear different.

**Corollary:**

C1 Changes in referential expressions should be propagated

As discussed below in its description, DSTOP has strong support for M2 (similarity of reference). It also has support, but relatively less automatic, for M3 (difference of expression) and C1 (propagation of changes). One of DSTOP's design goals, though, was to be light-weight with respect to constraints on authoring, in large part because DSTOP is designed to be used at the earliest stages of the design process, when there is a premium on flexibility. As a result, DSTOP does not provide more than nominal support for M1 (coherence of semantic relations).

Accordingly, some aspects of the original COHERE prototype associated with the maxims have been omitted. Principal among these is explicit categorization in terms of actions, things and relations. If usability tests, use, or interviews with users discloses that this sort of categorization would be helpful in practice, this functionality can be added back to the lexicon item pages.

## 3 HOW DSTOP WORKS

DSTOP, like the original COHERE conceptual prototype, was developed in HyperCard, which is a direct-manipulation, hyper-linking and scripting graphical user interface construction environment. DSTOP is a fully functional prototype that is being transitioned to use by industry. From the user's perspective, DSTOP's interface consists of three windows:

- A main window for writing, editing and analyzing procedures using the situated acts model's "Why-What-How" model,
- An index of the procedures in the main window, and
- A lexicon and index of terms used in the procedures.

### 3.1 Main Window

The main window contains fields for the why, what and how components of the operating procedure. The "what" components are the main representation of the operating procedure, and are supposed to be at the act level of the situated acts model. The "what" field actually comprises two adjacent sub-fields, one for the assignment of procedure responsibilities within the members of the crew (i.e., between the pilot flying (PF) and the pilot not flying (PNF)) and one for the acts as such. This division into of sub-fields eases the user's burden of formatting and avoids confusion in searching the operating procedure for terms in the lexicon. The "why" components are the goals associated with each of these acts. And the "how" components are the interface actions (and possibly responses) that comprise the performance of the acts, and are supposed to be at the action level of the situated acts model. The titles associated with these fields are the reminders for the engineer to structure the operating procedure they are writing according to the situated act model. DSTOP, in itself, does not compel the engineer to adhere to the model, although the fields and the titles help the engineer to do so. In fact, the relationship among why, what and how components of a procedure step is logical one, represented spatially in the interface. That is, it is the user who determines

how and why the three components of a step are related, and who shows the existence of these relationships by formatting the operating procedure in the why, what and how fields so that the components of a single step appear on associated lines of their respective fields. A set of controls at the top right of the window enables the user to display or hide each of the what, why and how fields. The main window also contains a set of header fields for the operating procedure's name, the name of the author, the date, the sequence number, and the version number. Figure 1 presents a view of the main window, with a draft operating procedure "Respond to request to report."

The main window also embodies functions analogous to those built for the COHERE prototype, namely the ability to enter text in a window and then use a point-and-click technique to see if a term in the new text is consistent with existing terms in the tool's lexicon. Accordingly, the main window contains controls that set the window's "mode." In edit mode, the user can freely edit any of the what, why or how fields. In "find" mode, clicking on any term in a visible what, why or how field will find the term in the lexicon and make the appropriate lexicon page active. If there is a lexicon page for the term but the use of the term for this particular operating procedure is not already noted, the tool will automatically update the use. If the term does not appear in the lexicon, the tool will present a dialogue that enables the user to create a new lexicon entry or to note that this term is a variant of an existing lexicon entry.

### 3.2 Index Window

The operating procedure index window contains a list of the procedures in the main window. Clicking on one of the operating procedure names in the index will make that procedure active in the main window. The operating procedure index window also contains a button that enables the user to refresh the index; this would be done if, for example, the user added or deleted a procedure.

#### Lexicon Window

The lexicon window can display an index of all the terms in the lexicon, a page for adding an unknown term, or a page for any term in the lexicon. These pages, including example pages for lexicon terms, are depicted in Figures 2 to 4. From the lexicon index page, the user can click on any entry and cause the page for that entry to be displayed.

Each page for an individual lexicon item (see Figure 2) consists of fields for

- The name of the term (i.e., one or more word that represents the core appearance of the term, such as "send").
- The possible variants of the term, normally including the term's name (i.e., variations in spelling or form due to number or verb tense, such as "send" and "sends").

Procedure:	<b>Respond to a request to report</b>			<input checked="" type="checkbox"/> Why <input checked="" type="checkbox"/> What <input checked="" type="checkbox"/> How
Sequence:	<b>000</b>	Author:	<b>David G. Novick</b>	
Version:	<b>10</b>	Date:	<b>05 Nov 1998</b>	

Goals	Acts	Actions
Delay response due to transient cockpit workload.	PNF • If you <b>need a short-term delay</b> to respond to the clearance, respond STBY.	<b>Select</b> DCDU LSK for STBY.
Ensure mutual awareness.	PNF • Announce: "Responding standby."	Read aloud from DCDU.
Inform ATC that the aircraft will comply with the request to report.	PNF • Respond ROGER.	<b>Select</b> DCDU LSK for ROGER.
Ensure mutual awareness.	PNF • Announce: "Responding roger to request to report."	Read aloud from DCDU.

Find  
 Mode  Edit  
 Add

Update

Figure 1. DSTOP Main Window

<b>Item</b>	LSK
<b>Variants</b>	LSK LSKs
<b>Meaning</b>	line select key
<b>Uses</b>	Delete a hold, {How} Enter a hold, {How} Respond to a message {How}

Enable Change     Enable Delete

Figure 2. Lexicon Object Page

- The meaning of the term, which the user should provide.
- The uses of the term in operating procedures in the main window. For each operating procedure in which any of the variants of the term is used, this field contains the name of the operating procedure followed by indicators of which of the what-why-how fields are involved.

The lexicon pages contain controls that permit knowledgeable users to change or delete entries, variants and uses. The lexicon pages also contain a button that enables knowledgeable users to update the “uses” field for one or more contiguous operating procedures in the main window. This feature is a great time saver, because it means that uses of terms can be recorded automatically rather than by having to search through all the operating procedures and click on every use. If the “Enable Change” or “Enable Delete” buttons are selected, the corresponding controls are displayed and the response to clicking in the “Uses” field is limited to selection; that is, clicking a use will not cause the tool to activate the corresponding operating procedure in the main window. The buttons are shown as enabled in Figure 2.

The Lexicon Window can display a list of all terms contained in the lexicon. Figure 3 depicts this list. Clicking the “Refresh” button rebuilds the list from the pages currently in the lexicon. Clicking the “List of things to ignore” button displays a similar list of terms, each of which does not have a lexicon page but rather is explicitly noted to be ignored during search. Some terms, such as “the” are so common that the user may not want a lexicon page for the term and will thus want DSTOP to skip the term when examining an operating procedure for terms not in the lexicon. The user can add or delete terms from this list.

When DSTOP does encounter a term that is not in the lexicon and not in the list of things to ignore, it presents the user with a dialogue for handling the unknown term. Figure 4 depicts this dialogue box. The dialogue presents the unknown item (in this case, the word “cockpit”). The user can define the unknown item as a variant of an item already present in the lexicon (for example, “LSKs” would be a variant of “LSK”), as a new entry in the lexicon, or as a thing to ignore.

- |             |   |
|-------------|---|
| <b>Find</b> | Read<br>respond<br>response<br>retain<br>roger<br>Scratchpad<br>select<br>SEND<br>shared<br>standby<br>status<br>STORE<br>stored<br>system<br>the |
|-------------|---|

Figure 3. List of Terms in Lexicon

#### 4. EXAMPLES

To demonstrate the why-what-how approach to the development of operating procedures, I developed six example operating procedures dealing with a new, text-based "DataLink" interface to supplement normal voice-based air-traffic control communications, and three example procedures dealing with the Airbus A340's flight management and guidance functions for holding at a waypoint.

To illustrate the main ideas, here is one step from a DataLink procedure, showing the goal, the domain-level act, and the interface-level action:

**Why:** Establish datalink contact with ATC.

**What:** SEND LOGON message.

**How:** Select MCDU LSK for LOGON.

The “why” corresponds to the users’ top-level goal for the operating procedure. In this way, DSTOP’s what-why-how approach an extension of the situated act model, as the “why” field represents an additional level of abstraction above the act level. The “what” represents the symbolic action to be taken by the users, i.e., sending a message. The words “SEND” and “LOGON” are in uppercase because they correspond exactly to uppercase words in the interface. The “How” represents the

**Unknown Item:** cockpit

a Activate aircraft alert already and Announce appropriate as ATC awareness
---

  

Figure 4. Unknown Item Dialogue Box

users' action in the interface itself, in this case pushing a key corresponding to the "LOGON" label.

The relationship between the "why" and "what" components can be assessed by looking at whether the "what" field would be a good answer to a question posed by prefacing the why field with the word "to." In this case, it makes sense that "To Establish datalink contact with ATC" one should "SEND LOGON message." Analogously, the relationship between the "what" and "how" components can be assessed by looking at whether the "how" field would be a good answer to a question posed by prefacing the "what" field with the word "to." In this case, it makes sense that "To SEND LOGON message" one should "Select MCDU LSK for LOGON."

#### 4.1 Representing Levels

This relationship raises the issue of whether it would be possible to abstract up from the "why" field or down from the "how" field. In fact, it should be possible to abstract up from the rationale, although possibly not on a line-by-line basis. That is, as the rationale becomes more abstract, individual steps may no longer be differentiable. Indeed, in some operating procedures I created, a single "why" entry covered more than one "what" entry. It should also be possible to abstract down from the interface actions, although at some point the exercise loses interest. There could certainly be available how-to information on what it means, in the example above, to "select" an LSK; but it is doubtful beyond this that even lower levels of explication would be useful. It would be both difficult and excessive to try to break down the steps in pushing a button, for instance.

Consequently, I believe that the why-what-how fields presented in DSTOP are actually distinctively useful levels. One reason for this is that the "what" level contains differentiation by crew role; that is, the step is something that is accomplished by the crew as a whole. In contrast, interface actions—at least in the cockpit—are normally executed by an individual person. This suggests that the levels are anchored at the level of situated acts, which is central field in the tool's main window.

Another reason to believe that three levels are a sufficient representation arises out of the analogy to speech acts. If the act and action levels represent respectively the illocutionary and locutionary acts in speech-act theory, then the goal level represents the intended perlocution. This analogy has its limits, though, and these limits reveal an unresolved issue in the why-what-how model: does the "why" level represent an *abstraction* of the "what" level or does it represent an *intention* that motivates the "what" level? This issue may have to be resolved through use.

#### 4.2 Defining Contents of Fields

The natural orientation toward anchoring the what, why and how levels does not mean that it will automatically be obvious to an author how to define the contents of the three fields for a given procedure or step. For example, in adapting the current A340 FMGS operating procedure for entering a hold to the tool's representation, precisely formulating the level components can pose issues that may be difficult to resolve. Here is an example step from the "Enter a Hold" operating procedure:

**Why:** Display LAT REV page for point of hold.

**What:** Choose present position for hold.

**How:** On MCDU F-PLN page, select LSK for PPOS or applicable WPT.

Among the difficulties encountered was the fact that the rationale I developed refers to the interface rather than to the domain task. The "what" field was written in terms of domain acts in order to conform with the situated act model, but while this act really does effectively choose the point of the hold it does not capture very well the impression given in the MCDU that the user is causing the LAT REV page to be displayed.

Aerospatiale Matra Airbus conducted an extended set of simulator trials of the DataLink interface. In the documentation used in the simulator experiments, only the "what" field was included. The rationale for this design decision was that the "what" field represents the act level, as it addresses the act to be taken by crew in terms of the domain rather than in terms of specific interface actions. So from the why, what and how fields for a step from the DataLink procedure "Respond to a message":

**Why:** Clear message from DCDU and retain in MCDU's ATC message logbook.

**What:** STORE the message, unless the system has already stored the message.

**How:** Select DCDU LSK for STORE.

the documentation used just the "what" field. In this case, the "what" field discusses the act of storing a message. The "why" and "how" fields thus represent aids for the author of the procedure in terms of (1) making explicit and thus understanding the design rationale for including a procedure step, (2) explicating how the step is to be carried out, typically through actions in the interface, and (3) assuring that act and action levels can be cleanly distinguished. In the example, the "why" field explains the rationale for taking the step, namely moving messages from the DCDU to the MCDU's logbook. The "how" field explains how to carry out the step in terms of interface actions, in this case selecting a line-select key in the DCDU associated with the label "STORE."

It was not clear how to represent all three fields in written documentation, other than presenting the full table of fields. For future documentation, such as electronic on-board documentation, the presence of the "why" and "how" fields at operating-procedure-design time should make it easier to include information on rationale and technique.

### 5. CONCLUSION

The DSTOP tool is based on two complementary models: the documentation coherence maxims and the situated act model, which distinguishes domain actions from interface actions. The tool involves the writing of operating procedures using a kind of grid, where the procedure runs from top to bottom and is divided horizontally into why, what and how elements. The "why" element represents the procedure's goal; the "what" represents the procedure's situated act; and the "how" represents the interface action(s) required to effectuate the act. Not all of these components will necessarily appear in the final version of the documentation. The "why" element can be used to verify the correctness of the design, to maintain the operational perspective through the design process and to assure consistency between the task model, the interface, and the operating procedures. Nevertheless, the tool's structure supports eventual implementation of the documentation in a hypertext environment, as is currently contemplated for on-board documentation in commercial aircraft.

The DSTOP prototype embodies much of the functionality of the original COHERE prototype aimed at respecting the

coherence maxims. Tool functions that address the maxims include the lexicon, the lexicon index, and the abilities both to add and to search text in the body of procedures. In this respect, DSTOP helps developers of interfaces and their related procedures of use make sure that each term has a single meaning and, to a lesser extent helps to make sure that each meaning has a single associated term to represent it. DSTOP also adds new functionality associated with the coherence maxims, including (1) explicit representation of variants of terms, and (2) explicit tracking and display of procedures (and their why-what-how components) in which any of the variants is used. Thus if all uses are recorded in the lexicon pages, then terms with unique or limited uses can be easily discerned. And authors can track whether or not a term is used uniquely within a single why-what-how category.

## 6. ACKNOWLEDGMENTS

This research was supported by a contract from Aerospatiale Matra Airbus. Thanks to Florence Buratto, Florence Beaujard, Florence Reuzeau, Meriem Chater, and Saïd Tazi for their contributions to the project.

## 8. REFERENCES

Austin, J. (1962). *How to do things with words*. Cambridge, MA: Harvard University Press.

Barth, H. (1997). DiDoLog—A system for developing and automatically generating guaranteed consistent and complete user manuals for interactive systems, RVS-Demo-01, <http://www.rvs.uni-bielefeld.de/~ihbarth/DiDoLog.html>, 14 April, 1997

Budde, R., Kautz, K., Kuhlenkamp, K., and Züllighoven, H. (1992). *Prototyping: An approach to evolutionary system development*. Berlin: Springer-Verlag.

Carroll, J. (1997). Reconstructing minimalism. *Proceedings of the 15th Annual Conference on Computer Documentation (SIGDOC 97)*, Salt Lake City, October, 1997, 27-34.

Degani, A., and Wiener, E. Procedures in complex systems: The airline cockpit. *IEEE Transactions on Systems, Man, and Cybernetics*, 27, 3 (1997), 302-312.

Drury, C., and Rangel, J. Reducing automation-related errors in maintenance and inspection, in Federal Aviation Administration, *Phase VI Technical Report on Human Factors in Aviation Maintenance and Inspection*, Chapter 11. Available at <http://hfskyway.faa.gov/document.htm>, June, 1996.

Novick, D., and Perez-Quiñones, M. (1998). Cooperating with computers: Abstracting from action to situated acts, *Proceedings of the Ninth European Conference on Cognitive Ergonomics (ECCE-9)*, Limerick, IR, August, 1998, 49-54.

Novick, D., and Juillet, J. (1998). Documentation integrity for safety-critical applications: The COHERE project, *Proceedings of SIGDOC 98*, Quebec, September, 1998, 51-57.

Paris, C., and Hartley, A. (1996). Multilingual document production from support for translating to support for authoring. Information Technology Research Institute Technical Report No. ITRI-96-17.

Suchman, L. (1987). *Plans and situated actions*. Cambridge: Cambridge University Press.

David G. Novick earned his Ph.D. in Computer and Information Science at the University of Oregon in 1988 and his J.D. at Harvard University in 1977. He is professor and chair of Computer Science at the University of Texas at El Paso. His research focuses on interactive systems, especially development methods for interfaces and their documentation.