
# Intelligent Help Facilities: Generating Natural Language Descriptions with Examples

**Vibhu Mittal and Cecile Paris**
**USC Information Sciences Institute**
**4676 Admiralty Way**
**Marina del Rey, California 90292**

**April 1993**
**ISI/RR-93-334**



*To appear in the Proceedings of HCI International 1993*

# REPORT DOCUMENTATION PAGE

*FORM APPROVED*
*OMB NO. 0704-0188*



| 1. AGENCY USE ONLY *(Leave blank)* | 2. REPORT DATE | 3. REPORT TYPE AND DATES COVERED |
|---|---|---|
| | April 1993 | Research Report |

**4. TITLE AND SUBTITLE**

Intelligent Help Facilities: Generating Natural Language Descriptions with Examples

**5. FUNDING NUMBERS**

NCC2-250
DABT63-91-C-0025
NSF ISI-9003087

**6. AUTHOR(S)**

Vibhu O. Mittal and Cecile Paris

**7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)**

USC INFORMATION SCIENCES INSTITUTE
4676 ADMIRALTY WAY
MARINA DEL REY, CA 90292-6695

**8. PERFORMING ORGANIZATON REPORT NUMBER**

RR-334

**9. SPONSORING/MONITORING AGENCY NAMES(S) AND ADDRESS(ES)**

| NASA AMES | ARPA | NSF |
|---|---|---|
| Moffett Field, CA 94035 | 3701 Fairfax Drive Arlington, VA 22203 | 1800 'G' Street NW Washington, DC 20550 |

**10. SPONSORING/MONITORING AGENCY REPORT NUMBER**

**11. SUPPLEMENTARY NOTES**

To appear in the Proceedings of HCI International 1993

**12A. DISTRIBUTION/AVAILABILITY STATEMENT**

UNCLASSIFIED/UNLIMITED

**12B. DISTRIBUTION CODE**

**13. ABSTRACT** ***(Maximum 200 words)***

On-line help facilities are essential in any complex system, especially for introductory or naive users. Previous studies have highlighted the need for appropriate examples along with the description. This paper describes a help/documentation facility built within an explanation framework that plans the presentation of text and examples using techniques in natural language generation. The paper shows how text and examples can influence each other and enumerates some of the other issues that arise in planning such presentations.

**14. SUBJECT TERMS**

help facilities, Natural Language Generation, examples

**15. NUMBER OF PAGES**

**16. PRICE CODE**

| 17. SECURITY CLASSIFICTION OF REPORT | 18. SECURITY CLASSIFICATION OF THIS PAGE | 19. SECURITY CLASSIFICATION OF ABSTRACT | 20. LIMITATION OF ABSTRACT |
|---|---|---|---|
| UNCLASSIFIED | UNCLASSIFIED | UNCLASSIFIED | UNLIMITED |

# GENERAL INSTRUCTIONS FOR COMPLETING SF 298

**The Report Documentation Page (RDP) is used in announcing and cataloging reoprts. It is important that this information be consistent with the rest of the report, particularly the cover and title page. Instructions for filling in each block of the form follow. It is important to stay within the lines to meet optical scanning requirements.**

**Block 1. Agency Use Only (Leave blank).**

**Block 2. Report Date.** Full publication date including day, month,a nd year, if available (e.g. 1 jan 88). Must cite at least the year.

**Block 3. Type of Report and Dates Covered.** State whether report is interim, final, etc. If applicable, enter inclusive report dates (e.g. 10 Jun 87 - 30 Jun 88).

**Block 4. Title and Subtitle.** A title is taken from the part of the report that provides the most meaningful and complete information. When a report is prepared in more than one volume, repeat the primary title, add volume number, and include subtitle for the specific volume. On classified documents enter the title classification in parentheses.

**Block 5. Funding Numbers.** To include contract and grant numbers; may include program element numbers(s), project number(s), task number(s), and work unit number(s). Use the following labels:

| | |
|---|---|
| C - Contract | PR - Project |
| G - Grant | TA - Task |
| PE - Program Element | WU - Work Unit Accession No. |

**Block 6. Author(s).** Name(s) of person(s) responsible for writing the report, performing the research, or credited with the content of the report. If editor or compiler, this should follow the name(s).

**Block 7. Performing Organization Name(s) and Address(es).** Self-explanatory.

**Block 8. Performing Organization Report Number.** Enter the unique alphanumeric report number(s) assigned by the organization performing the repor.

**Block 9. Sponsoring/Monitoring Agency Names(s) and Address(es).** Self-explanatory

**Block 10. Sponsoring/Monitoring Agency Report Number.** (If known)

**Block 11. Supplementary Notes.** Enter information not included elsewhere such as: Prepared in cooperation with...; Trans. of ...; To be published in... When a report is revised, include a statement whether the new report supersedes or supplements the older report.

**Block 12a. Distribution/Availability Statement.** Denotes public availability or limitations. Cite any availability to the public. Enter additional limitations or special markings in all capitals (e.g. NOFORN, REL, ITAR).

| | |
|---|---|
| DOD | - See DoDD 5230.24, "Distribution Statements on Technical Documents." |
| DOE | - See authorities. |
| NASA | - See Handbook NHB 2200.2. |
| NTIS | - Leave blank. |

**Block 12b. Distribution Code.**

| | |
|---|---|
| DOD | - Leave blank. |
| DOE | - Enter DOE distribution categories from the Standard Distribution for Unclassified Scientific and Technical Reports. |
| NASA | - Leave blank. |
| NTIS | - Leave blank. |

**Block 13. Abstract.** Include a brief (Maximum 200 words) factual summary of the most significant information contained in the report.

**Block 14. Subject Terms.** Keywords or phrases identifying major subjects in the report.

**Block 15. Number of Pages.** Enter the total number of pages.

**Block 16. Price Code.** Enter appropriate price code (NTIS only).

**Blocks 17.-19. Security Classifications.** Self-explanatory. Enter U.S. Security Classification in accordance with U.S. Security Regulations (i.e., UNCLASSIFIED). If form contins classified information, stamp classification on the top and bottom of the page.

**Block 20. Limitation of Abstract.** This block must be completed to assign a limitation to the abstract. Enter either UL (unlimited) or SAR (same as report). An entry in this block is necessary if the abstract is to be limited. If blank, the abstract is assumed to be unlimited.

# Intelligent Help Facilities: Generating Natural Language Descriptions with Examples

Vibhu O. Mittal and Cecile L. Paris

USC/Information Sciences Institute, 4676 Admiralty Way, Marina del Rey, CA 90292, U.S.A.

Department of Computer Science, University of Southern California, Los Angeles, CA 90089, U.S.A.

## Abstract

On-line help facilities are essential in any complex system, especially for introductory or naive users. Previous studies have highlighted the need for appropriate examples along with the description. This paper describes a help/documentation facility built within an explanation framework that plans the presentation of text and examples using techniques in natural language generation. The paper shows how text and examples can influence each other and enumerates some of the other issues that arise in planning such presentations.

## 1. INTRODUCTION

Good help facilities are a crucial component of any complex system (e.g., [1,2]), and there have been many studies on generating effective online help (e.g [3--6]). Most attempts at providing intelligent help facilities have focused on either structured, hierarchical menu style help, as in the VMS online help system [7], or on hypertext style browsing capabilities (e.g [8, 9]). However, static, online help, in the form of canned text is not always helpful: in one study, Houghton found that in most current systems, while online help enhanced the performance of experienced users, it was either not helpful, or sometimes even detrimental to inexperienced users [10]. It is therefore clear that online help systems that can tailor their descriptions to the user (particularly the inexperienced, introductory user), would be very helpful.

Tailoring descriptions to the user often involves more than just a change in the terminology, or the level of detail presented to the user [11]. The type of information presented is often different as well: for instance, syntactic information vs structural

information *vs* functional information. One of the differences between introductory material and non-introductory material is the importance of examples: studies by Beard and Calamars [12], and Tuck *et. al* [13] found that the component help most often cited as necessary, and the one most necessary for introductory users, was the presence of examples. It is thus clear that a help facility designed for introductory users must be able to include examples in its explanations.

In this paper, we shall describe a framework for an explanation/help facility that can generate natural language descriptions (of objects in the knowledge base) that incorporate examples along with descriptive text. We shall briefly describe some of the issues that arise, and using a simple example from our system on help in LISP, we shall describe how the system goes about planning its presentation.

## 2. ISSUES IN THE USE OF EXAMPLES

The use of examples can contribute a great deal to the effectiveness of the response. Indeed, empirical studies have found that examples can greatly increase user comprehension (e.g., [14, 15]). However, studies also show that badly integrated text and examples can actually be detrimental compared to using either text or examples alone (e.g. [16]). It is thus clear that in order to provide useful documentation automatically, a system must be capable of providing well-integrated examples to illustrate its points.

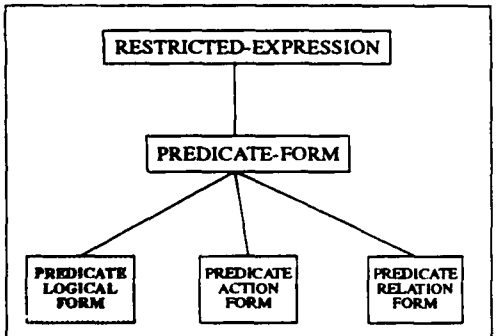

Figure 1: Part of the grammar non-terminal hierarchy.

There are many issues that arise in generating complex descriptions which include examples. Although some tutorial systems used examples in explanations, they were not considered as an integral part of the complete explanation and were inserted in the explanations without any representation of how the examples related to and complemented the accompanying explanation. Consider, for instance, a help description generated for the term 'predicate-relation-form' in our system. A predicate-relation-form is a non-terminal in our system's grammar, and its parent and child relationships are shown in Fig. 1. A part of the English description is shown in Fig. 2.

Consider the description in Fig. 2: the system first describes the predicate-relation-form by describing it as a specialization of a predicate-form (which returns a boolean value). It then describes the syntax of a predicate-relation-form. However, the description does not include the facts that:

- the syntactic detail that there is a left parenthesis before the relation-name and a right parenthesis after the last argument
- the different types of arguments that can appear after the relation

This is because these facts are communicated through the examples which follow. The parentheses are noted in the system as 'fixed features': features that are constant and will appear in the same location in every example. From this, the user can infer that those features are necessary. The different types of parameters that can appear as arguments in a predicate-relation-form are (in our system: numbers, symbols and instances) also illustrated through the use of different parameters in the examples. A natural language generator should be able to take properties of both the descriptive text and the examples into account and generate a coherent, comprehensive (yet non-redundant) explanation with examples. Such descriptions are not possible to generate if the text generator and the example generator do not interact closely in planning their presentation.

From the preceding discussion, it is clear that the inclusion of examples into explanations can cause certain portions of text to be elided. However, the incorporation of examples into descriptions can also cause additional text to be included -- information that would not originally have been communicated. This is illustrated by the last example of a function-form that is presented in the description. The system attempts to find a negative example (an example that is *not* a predicate-relation-form) that is as similar as possible to one of the positive examples that have already been presented. In this case, the system attempts to generate a negative example by changing the number of arguments from two to one. This results in the third example changing from a predicate-relation-form to a function-form. The system attempts to point this out, resulting in additional explanation.

A PREDICATE-RELATION-FORM is a predicate-form. It consists of a relation followed by some parameters, the number of which is equivalent to the arity of the relation. Examples of predicate-relation-form are:

```
(VERSION  LOAD-SOFTWARE  5.1)
(STATUS   LED-1          'ON)
(CONNECTED COMPUTER-A  PRINTER-B)
```

However, the following is not a predicate-relation-form, because the number of arguments is not equal to the arity of the relation. It is an example of a FUNCTION-FORM:

```
(CONNECTED COMPUTER-A)
```

The difference between a FUNCTION-FORM and a predicate-relation-form is that the number of arguments in a function-form are one less than the arity of the relation, and ...

Figure 2: Part of the English description of predicate-relation-form.

Thus, as even this brief description has shown, there is strong interaction between the descriptive text and the accompanying examples. There are a number of other issues that must be addressed in a practical generation system, such as the number of examples to be presented, the order of presentation, whether the examples should be presented *before*, *after* or *within* the description, etc. Due to lack of space, we shall not discuss these issues here.[1] In the following section,

[1]See [17] for further details.

we shall describe the planning framework within which such descriptions are generated.

## 3. THE SYSTEM

Our current framework implements the generation of examples *within* a text-generation system that explicitly plans text to achieve a communicative or discourse) goal. Given a top level communicative goal -- such as (DESCRIBE OBJECT), the system finds plans capable of achieving this goal. Plans typically post further sub-goals to be satisfied, and planning continues until primitive speech acts -- i.e., directly realizable in English -- are achieved. The result of the planning process is a discourse tree, where the nodes represent goals at various levels of abstraction, with the root being the initial goal, and the leaves representing primitive realization statements, such as (INFORM ...) statements. In the discourse tree, the discourse goals are related through coherence relations. This tree is then passed to a grammar interface which converts it into a set of inputs suitable for input to a natural language generation system called PENMAN [18]).

A fragment of the text plan generated by the system for the description in Fig. 2 is shown in Fig. 3. The skeleton shows the structure of the plan, along with the discourse goals posted by the system. The two top-level goals posted by the initial goal are (DESCRIBE (SYNTAX PRED-REL-FORM)) and (EXEMPLIFY-FEATURES (SYNTAX PRED-REL-FORM)). The first goal, to describe the features ultimately results in the first part of the description, which states that a predicate-relation-form consistes of a relation-name followed by some arguments. The second goal, to exemplify the features of a predicate-relation-form results in the generation of the actual examples, along with the associated background text ("Examples of predicate-relation-form are ... ").

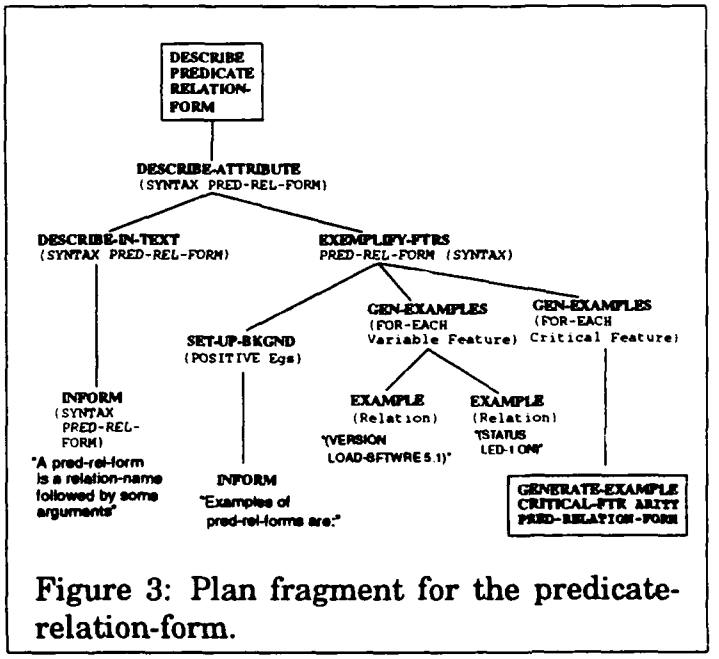


Figure 3: Plan fragment for the predicate-relation-form.

Thus, examples are generated by *explicitly posting a goal within the text planning system*: i.e., some of the plan operators used in the system include the generation of examples as one of their steps, when applicable. This ensures that the examples embody specific information that either illustrates or complements the information in the accompanying textual description. Additional sources of knowledge such as the user model, text type, dialogue context, etc, can be added

to the system by incorporating additional constraints in the plan operators.

Using this framework, we have generated descriptions of constructs in the programming language LISP [19] and documentation for the grammar in our own plan language [20] for the EES expert system framework [21]. The generation of these descriptions highlight some of the issues that must be addressed by *any* interface that must integrate natural language and examples together. Some of these issues are also applicable to the generation of multi-media explanations, where the constraints imposed by the non-textual component upon the text planner must be taken into account.

## 4. CONCLUSIONS

Examples are essential in explanations, especially for naive users. It is therefore important that on-line help systems designed for non-expert users be able to present effective and appropriate examples to the user. As systems evolve over time, it is essential that the documentation/help facility be updated to reflect the changes. A system that generates documentation from the underlying code would help alleviate the 'maintenance of documentation' problem. However, such a system would then need to be able to present examples and text in a coherent and integrated manner. In this paper, we have presented some of the issues that arise in the planning of such presentations. The issues we have described are not specific to a particular framework. Our implementation demonstrates that it is not just desirable, but also feasible to build such on-line help systems by making use of advances in natural language generation and knowledge based systems. This work is important in other related application areas such as intelligent tutoring systems, expert system explanation and user interfaces.

## References


[1] G. Kearsley, *Online Help Systems: Design and Implementation*. Ablex Publishing Co., 1988.

[2] W. K. Horton, *Designing and Writing Online Documentation : Help files to hypertext*. New York: John Wiley and Sons, Inc., 1990.

[3] A. H. Duin, "Computer Documentation -- Centering on the Learner," *Journal of Computer-Based Instruction*, vol. 17, pp. 73--78, Spring 1990.

[4] E. Reiter, C. Mellish, and J. Levine, "Automatic Generation of on-line documentation in the IDAS project," in *Proceedings of the Third Conference on Applied Natural Language Processing (Trento, Italy)*, pp. 64--71, April 1992.

[5] W. K. Horton, *Illustrating Computer Documentation: The Art of Presenting Information Graphically and Online*. New York: John Wiley and Sons, Inc., 1991.

[6] B. Wasson and S. Akselsen, "An overview of on-line assistance: from on-line documentation to intelligent help and training," *The Knowledge Engineering Review*, vol. 7, pp. 289--322, December 1992.

[7] DEC, *Command Language User's Guide*. Digital Equipment Corporation, 1978.

[8] G. Fischer, A. C. Lemke, and T. Mastaglio, "JANUS: Integrating hypertext with a knowledge-based design environment," in *Proceedings of Hypertext '89*, pp. 105--117, 1989.

[9] R. Rada and J. Barlow, "Expert systems and hypertext," *The Knowledge Engineering Review*, pp. 285--301, 1988.

[10] R. C. Houghton, "Online help systems: a conspectus," *Communications of the ACM*, vol. 27, pp. 126--133, 1984.

[11] C. L. Paris, *The Use of Explicit User Models in Text Generation*. London, England: Frances Pinter, 1993.

[12] R. E. Beard and P. V. Calamars, "A Method for Designing Computer Support Documentation," Master's thesis, Department of Communication, AFIT/LSH, WPAFB, Ohio 45433, September 1983.

[13] R. Tuck and D. R. Olsen, "Help by guided tasks: utilizing UIMS knowledge," in *Proceedings of the CHI'90 Conference*, pp. 71--78, ACM, 1990.

[14] J.-A. LeFevre and P. Dixon, "Do Written Instructions Need Examples?," *Cognition and Instruction*, vol. 3, no. 1, pp. 1--30, 1986.

[15] D. H. Charney, L. M. Reder, and G. W. Wells, "Studies of Elaboration in Instructional Texts," in *Effective Documentation: What we have learned from Research* (S. Doheny-Farina, ed.), ch. 3, pp. 48--72, Cambridge, MA.: The MIT Press, 1988.

[16] M. Ward and J. Sweller, "Structuring Effective Worked Examples," *Cognition and Instruction*, vol. 7, no. 1, pp. 1 -- 39, 1990.

[17] V. O. Mittal, *Generating descriptions with integrated text and examples*. PhD thesis, University of Southern California, Los Angeles, CA, 1993 (forthcoming).

[18] W. C. Mann, "An Overview of the Penman Text Generation System," in *Proceedings of the Second National Conference on Artificial Intelligence*, (Washington, D.C.), pp. 261--265, 1983.

[19] V. O. Mittal and C. L. Paris, "Generating Object Descriptions which integrate both Text and Examples," in *Proceedings of the Ninth Canadian Artificial Intelligence Conference (AI/GI/VI 92)*, pp. 1--8, Canadian Society for the Computational Studies of Intelligence (CSCSI), Morgan Kaufmann Publishers, 1992. (Vancouver, Canada).

[20] V. O. Mittal and C. L. Paris, "Automatic Documentation Generation: The Interaction between Text and Examples." To appear in the *Proceedings of the 13th International Joint Conference on Artificial Intelligence (IJCAI-93)*, 1993.

[21] W. R. Swartout, C. L. Paris, and J. D. Moore, "Design for explainable expert systems," *IEEE Expert*, vol. 6, no. 3, pp. 58--64, 1992.