Steps Along the On-Line Assistance Spectrum

EDWINA L. RISSLAND

Department of Computer and Information Science
University of Massachusetts
Amherst, MA 01003

Abstract

In this paper, we discuss the spectrum of on-line assistance ranging from passive, canned to active, user-customized. We discuss various aspects of on-line assistance: interactive introductory tutorials, on-line help, and on-line manuals. We then describe two steps to make on-line assistance more intelligent: (1) inclusion and customization of examples in the information provided the user; and (2) integration of various aspects of on-line assistance like tutorials and help.

1. Introduction

As any neophyte would probably attest, it is hard to get started on a new computer system. One thing contributing to this difficulty is the lack of intelligent on-line assistance in the interface. Often there is a rudimentary on-line help facility, but it is clumsy to use. It is also "dumb" in that it lacks many key ingredients of expert knowledge, like examples and heuristics; it consists of canned responses that are always the same regardless of the user's background, task, goals, context, etc.

Then too, the interaction is neither gracious, graceful nor friendly (see, for example, the help facility on VAX/VMS):

1. access is tedious (e.g., menus too long and unorganized);
2. presentation of material is often insensitive (e.g., screenfuls of text whizzing by)
3. it requires the user to speak its language to get anything useful out of the help invocation (e.g., the user might not get any useful information because he asks about "quit" when he should've asked about "logout").

Such difficulties subvert the user's expression of his intentions -- he knows what he wants to do but not how to say it -- and ignore a key source of knowledge in intelligent user interfaces [Norman 1984].

---

[1] This work supported in part by Grant IST-8212238 of the National Science Foundation.

At the other end of this spectrum are intelligent assistance providers -- interactive tutors and coaches that use A.I. techniques like user modelling (e.g., [Woolf 1984]). More intelligent forms of on-line assistance need both more knowledge and power: that is, many types of knowledge (e.g., of the user's task, domain, personal experience, etc. [Rissland 1984]), modes of interaction (e.g., natural language [Walker 1976]), and processes (e.g., inference engines that isolate the user's misconceptions [Lewis and Soloway 1984]).

Thus we see a spectrum ordered by responsiveness and intelligence. At the low or negative end of the spectrum are the dumb systems with canned responses, no interactive capability, and only minimal domain knowledge (e.g., VAX/VMS HELP). A little better are interactive "dumb" facilities, like tutorials, without even explicit models of the user or the knowledge to be explained.[2] Enriching the knowledge base and enhancing the interaction can move assistance further in the "positive" direction. Adding user-modelling and the ability to provide responses custom-tailored in content and form place the assistance facility well towards the positive end of the spectrum.

So far that end of the spectrum has not been much explored, although interesting starts have been made. Wilensky, in his UC system, allows the user to ask for assistance in natural language [Wilensky 1982a, 1982b]; his work has concentrated on request understanding. Finin, in his WIZARD system, focuses on the problem of recognizing when the user needs help; for example, when he is using inefficient means to do something, like using repeated DELETEs instead of PURGE, the system then volunteers advice [Finin 1983; Shrager and Finin 1982].

Unhappily, however, most assistance facilities available today still lie clearly towards the dumb end of the spectrum.

In the rest of this paper, we will describe some steps to move on-line assistance further along towards the intelligent end of the assistance spectrum through the use of richer domain knowledge and the integration of various aspects of assistance. We shall assume that the on-line assistance facility has already been invoked (by the user or the system) and that the facility has already 'parsed' the user's request (i.e., knows what the user requires assistance on). Our emphasis is on the generation of the response, and in particular, on the embedding of examples. Clearly, this work should eventually be tied in with work on invocation and parsing, like that of Wilensky or Finin, and an obvious extension would be the use of a program like McDonald's MUMBLE [McDonald 1982] to dynamically generate text as well as examples.

---

[2] For example, the much touted tutorial for the LOTUS 1-2-3 spreadsheet program is a step-by-step on-line tutorial; it is better than most, but its inflexibilities can be daunting to a user.

2. Aspects of Assistance: Help, Tutorials and Manuals

A recent review article [Houghton 1984], discussed several types of on-line assistance, including some pertinent here, namely, command assistance, introductory tutorials and manuals. Other types are error and prompting assistance. We will not discuss these here but many of our points are relevant to them as well. Of course, the ideal assistant is very often a human on-line consultant.

By a *tutorial*, we mean an interactive, structured program that introduces a user to a system (e.g., the EMACS tutorial [Stallman 1983]). A tutorial presents information but does not necessarily allow free rein in the interaction; some don't even allow one to jump around. What one can do next, like read further or supply parameters to a demonstration, is largely pre-determined.

By on-line *help*, we mean command assistance. The user asks for information about a particular command, like "HELP PRINT", and is then presented with information on PRINT, including relevant parameter options, but almost never including examples of standard, potentially dangerous, or clever uses.

By an on-line *manual*, we mean a version of the hard copy manual (whatever its content or organization) which is available to the user on-line together, hopefully with some sort of access interface, the case of a standard text-editor (in read-only mode) being the bare minimum.

Help and tutorials are clearly more interactive than manuals, although one can easily imagine interactive manuals as well. Tutorials can provide a "guided tour" of a system and an opportunity for the user to try things out in a sheltered, or hypothetical, environment.

3. Embedding Examples in On-line Assistance

One important component of knowledge that is missing in most on-line (and off-line) assistance is examples. Examples, by which we mean specific cases and instantiations, are one of the most important ingredients of expert knowledge. They offer concrete illustrations of what is being explained and memorable hooks into more general information. They are especially important for the beginner.

Examples can provide easily understood and remembered usages. For instance,

PRINT VITA.MEM

is clearly more perspicuous than

"**PRINT** [[*d:*][*filename*[.*ext*]][/T][/C][/P]...]" (from [IBM 1983])

A novice can use simple cases: to figure out how to instantiate the general syntactic description, to use as "recipes" for standard tasks, as a basis for generalizing, and as a basis for a "retrieval+modification" approach to generate another instance. For the more expert, examples can serve as a reminder of syntax and things previously done, much like an icon; this is especially useful with commands used only infrequently.

[Rissland 1978] presented a taxonomy of examples: "start-up" (easy, perspicuous cases); "reference" (standard, textbook cases); "model" (paradigmatic, template-like cases); "counter-examples" (limiting, illegal cases); "anomalous" (ill-understood, strange cases).

Here we use such a taxonomy to select and order the presentation of examples. For instance, we provide the neophyte user with "start-up" examples and the more experienced user with "references". Where a sequence of examples is called for, references are presented before models which are presented before counter-examples and anomalies. This taxonomy can enable the user to ask specifically for examples in a certain class (e.g., "easy" or "dangerous").

Another aspect of examples which we have previously studied is their generation [Rissland 1981]. Two modes of generation are "retrieval+modification" and instantiation. We use these techniques in on-line assistance by linking the assistance program with an example generator, which has an "examples-space" of already existing examples, and procedures for modification and instantiation. The examples, which have been harvested and organized by an expert, are represented as frames; they contain slots for information such as a graphics demonstration, difficulty rating, and pointers to more and less complicated examples. Modification operators include procedures to personalize examples (e.g., if an example needs a file-name, use one of the user's). Instantiation procedures include ways to generate a range of cases, including those that satisfy and violate legal parameter values.

This ability to dynamically generate examples allows the assistance facility to provide examples tailored to the user, his tasks, goals, context, domain, etc.; it depends, of course, on having some sort of user-modelling capabilities. The idea is to work examples into the assistance given the user, and better still to make the examples meaningful in the sense of relating to the user.

4. Integrating Aspects of On-line Assistance

One difficulty in learning or checking out a feature of a system is that the various aspects of on-line assistance do not share a common language or set of examples, and thus it is hard to integrate one's knowledge or to apply information from one source to another. This violates the pedagogical strategy of using information seen before by the learner, like examples which have become "old friends".

Our approach to this consistency/integration problem is to have all the aspects of on-line assistance share common source material -- examples and text -- which is represented in a way usable by each individual aspect. Each aspect then puts its presentations together by retrieving the text and examples it needs from the common source.

In our work, we use a script-like control structure of a text and examples template, a "TEXPLATE". A TEXPLATE typically contains pointers to chunks of text, calls for examples, and control information. It can also contain "literal" material (like text used nowhere else) which is presented "as is". Calls to examples are either requests for explicitly named examples in the Examples Knowledge Base (EKB) or constraints by which the the example generator can generate a new example. For instance, an example call could be for a named counter-example or for an example generated to fulfill prescribed constraints (like one using the name of the user's most recently created file). Control information includes options to present to the user and the appropriate assistance-module response actions: for instance, MORE to cause the tutorial to go on, EXAMPLE for an example, QUIT, etc. Control information also contains directions for which sequence of examples the system should present if the user repeatedly selects the EXAMPLE option.

5. Two On-going Assistance Studies: IA-LADYBUG & VMS

In our on-going work, we are working within two systems. The first is IA-LADYBUG, a system designed specifically for novice programming students. It introduces them to notions useful in the Pascal programming language (like subprocedures) by having them work with a graphics icon, the LADYBUG, which can be commanded by LOGO-like commands like CRAWL, RIGHT-TURN, etc. [Levine and Woolf 1984]. The second is a subset of VAX/VMS command language [DEC 1978] dealing with directory commands like PURGE, DELETE, and SET PROTECTION.

For IA-Ladybug (over whose environment we have total control), the student manual, on-line introductory tutorial and interactive on-line HELP share material. The TEXPLATES are indexed by command and topic and reference the manual's

text file for textual material and a separate EKB for examples. Often, the tutorial and interactive HELP present dynamic examples that are merely summarized in the manual, for instance drawing a ball bouncing or a seven color sunburst. The tutorial and HELP present examples that are too complicated or whose effect (like color) would be lost in the manual.

The simpler "start-up" and "reference" examples presented in the manual are the first examples presented in the tutorial and HELP. HELP, especially, goes on to present more complex or difficult examples, like counter-examples to show the limits of commands (e.g., RIGHT 362 exceeds the parameter range for degrees of turning). At this time, HELP also does some very simple tuning of its examples to the user, for instance by using information about the user's directory and the user's own answer to whether or not he is an expert.

6. Summary

In this paper we have discussed two steps to making on-line assistance more intelligent: (1) inclusion and customization of examples in the information provided the user; and (2) integration of various aspects of on-line assistance like tutorials and command help. We have used knowledge about the structure, types, and generation of examples to implement (1) and a control structure of text and examples, called a "TEXPLATE", to achieve (2).

Currently we are experimenting with our prototype on-line assistance modules. One thing we have learned is that subjects do not read very well and that examples are a quick way to impart a lot of information. We have also found that using texplates to separate the control from the substance makes it easy to re-write the assistance scripts. Another observation is that for examples that are graphics demos, it would be nice for the user to be able to do "instant replays in slow motion" and to be able to take a deeper look at the code behind the example.

7. Acknowledgements

The author acknowledges the work of the members of the "help team", specifically E. Valcarce who implemented the on-line command assistance, L. Gordon who developed the IA-Ladybug tutorial, and R. Filoramo who wrote the IA-Ladybug Manual. Also thanks to B. Woolf and L. Levine for their critical discussions and to O. G. Selfridge for sharing his ideas.

8. References

DEC, *VAX/VMS Command Language User's Guide*. Digital Equipment Corporation. Order No. AA-D023B-TE, 1978.

Finin, T. W., "Providing Help and Advice in Task Oriented Systems". In *Proceedings IJCAI-83*. Karlsruhe, W. Germnay, 1983.

Houghton, R. C., "Online Help Systems: A Conspectus". *CACM*, Vol. 27, No. 2, February 1984.

IBM, *Disk Operating System by Microsoft, Inc.*. IBM Personal Computer Language Series, IBM Corp, 1983.

Johnson, L., and Soloway, E. M., "PROUST: Knowledge-Based Program Debugging". In *Proceedings Eighth Int'l Software Engineering Conference*, Orlando, FLA, March 1984.

Levine, L., and Woolf, B., "Do I Press Return?" In *Proceedings ACM-SIGCSE Symposium on Computer Science and Education*, Philadelphia, February 1984.

McDonald, D. D., "Natural Language Generation as a Computational Problem: An Introduction". In Brady (Ed.) *Computational Theories of Discourse*, MIT Press, 1982.

Norman, D. L., "Stages and Levels in Human-Computer Interaction". To appear in *International Journal of Man-Machine Studies*, summer 1984.

Rissland, E. L., *Constrained Example Generation*. COINS TR 81-24, Department of Computer and Information Science, University of Massachusetts, Amherst, 1981.

Rissland, E. L., "Ingredients of Intelligent User Interfaces". To appear in *International Journal of Man-Machine Studies*, summer 1984.

Rissland, E. L., "Understanding Understanding Mathematics" *Cognitive Science*, Vol. 2, No. 4, 1978.

Shrager, J., and Finin, T. W., "An Expert System that Volunteers Advice". In *Proceedings AAAI-82*, Pittsburgh, PA, August 1982.

Stallman, R. M., "EMACS: The Extensible, Customizable, Self-Documenting Display Editor". In Barstow, Shrobe and Sandewall (Eds.) *Interactive Programming Environments*, McGraw-Hill 1984. Also available as MIT AI Lab Memo 519a, 1981.

Walker, D. E. (Ed.), *Speech Understanding Research: Final Report*. Stanford Research Institute, Menlo Park, CA, 1976.

Wilensky, R., "Talking to UNIX in English: An Overview of UC". In *Proceedings*

AAAI-82, Pittsburgh, PA, August 1982a.

Wilensky, R., *Talking to UNIX in English: An Overview of an On-line Consultant*. Report No. UCB/CSD82/104, Computer Science Division, University of California, Berkeley, September 1982b.

Woolf, B. P., *Context Sensitive Text Planning in Tutorial Discourse Generation*. Ph. D. Dissertation, Dept. of Computer and Information Science, University of Massachusetts, Amherst, May 1984.