A Discourse-Based Consultant for Interactive Environments

Ursula Wolz Gail E. Kaiser
Columbia University
Department of Computer Science
New York, NY 10027

ABSTRACT

Interactive computing environments provide facilities intended to support and assist the range from novice to expert users, but casual users tend to get trapped in the starter set of commands. We have developed technology for providing on-line assistance calibrated to both the task at hand and the user's past experience using the system. In this paper, we present our automated consultant and describe its application to a practical domain, the Berkeley Unix™ mail system. We take advantage of previous work on discourse-based assistants that focus on understanding the user's question and goal, and instead concentrate on the generation of suitable advice.

1. Introduction

Interactive computing environments such as mail systems provide resources and facilities intended to support and assist users. A conflict arises between creating an environment simple enough for a novice yet sophisticated enough to accommodate an expert. A solution is to expose beginners to a set of starter commands, but provide more comprehensive features they can learn later. Finin [4] points out that many beginners get trapped in the starter set, since they are not encouraged to progress to more powerful commands. This paper presents a solution to this problem through an automated consultant that answers user's questions about the environment.

We take a user's goal-centered approach in which the help given is a direct function of a user's needs within the current context. In particular, we focus on the content of the answer provided to a user. First, we provide an expert or "guru" knowledge base that includes explicit information about the relationship between the computational goals that can be accomplished in the environment, the plans used to accomplish them, and the functions that make up the plans. In a mail system, for example, a goal might be to read a set of messages and forward a subset to a colleague. The plan for executing that goal will be dependent upon the particular functions available within the mail system.

Second, we believe that classifying functions, plans and goals according to level of expertise is inappropriate and global categorization of users as "novice", "intermediate" or "expert" is inadequate. In our work, information on an individual's exposure to goals, plans and functions influences the pedagogical goals of the consultant system, that is, what specific information to present following a user's query. Expectations about what the user knows and should be told is based on the computational goals that user has attempted to satisfy in the past rather than on broad ad hoc classifications of functions and plans as "easy" or "hard".

Taking another example from a mail system, a user may have extensive experience with sending simple messages to groups of users, and almost none with modifying messages through an editor. Such a user will not fall nicely into a categorization of expertise. In order to satisfy pedagogical goals, the knowledge base also includes explicit discourse information on the relationships between computational goals.

Section 2 describes the problem in more detail and outlines our solution. Section 3 summarizes how we improve on previous work. Section 4 describes GECIE, an implementation of our technology as an extension of the Berkeley mail system. We conclude by summarizing the contributions of this paper.

2. Consulting in Interactive Environments

In order to use an environment such as a mail system effectively, a user must know the system's capabilities and how to make best use of them. This requires access to information that describes the specific features of the system -- the functions, commands or constructs (henceforth functions) available. It also requires access to methods or plans for best accomplishing goals. We claim there is a large middle ground between a novice who knows only the rudiments of a system and an expert who has gained complete mastery over it. The continuum in between is one in which learning is constantly taking place, not only about what exactly a particular function can do, but how it can be used in conjunction with other functions to accomplish useful tasks.

The user is often faced with the burden of deciding what must be learned and how to locate the appropriate informa-
tion. This is typically done by searching through manuals, asking help of others or simply experimenting with the system. Results from artificial intelligence should be able to provide mechanisms that can take some of this burden off of the user. The objective of our research is to address these issues and offer a theory of how an automated consultant can assist users in extending their expertise. The following insights are the result of informal observations of human consultants giving help in environments that support EMACS, Lisp, Unix, Pascal and Logo, and on an examination of manuals, tutorials and texts for these environments.

There is rarely a direct correspondence between a precise statement of a user's goal and a plan to satisfy it. It is more often the case that the user's goal is poorly defined, and can be accomplished in more than one way. Furthermore, a plan may be executed by a number of different sequences of commands. The problem presents itself as requiring a mapping of many user queries to many possible answers. In order to constrain the potential mappings, user queries can be categorized at least partially as relating goals to plans as summarized in figure 2-1. Although the question itself may not be stated clearly in one of these forms, informal observations indicate that the intention of most utterances falls within one of these question types.

1. Function specification: What does function F do?
2. Goal satisfaction:
   a. How can goal G be accomplished?
   b. Plan P accomplishes goal G in the context of situation S, but there must be a 'better' way, what is it, and why is it better?
3. Analyze or debug a plan:
   a. What does plan P (learned by rote, for example) do?
   b. Plan P ought to accomplish goal G in the context of situation S, but doesn't, why not?

Figure 2-1: Typical Types of Questions Users Ask

Magers [10] and Borenstein [2] have drawn a distinction between information that is definitional and instructional. Figure 2-2 further refines this distinction, based on the argument that definitional information is more appropriate for reminding someone about something they have previously learned, while instructional information is more appropriate for introducing new information. These types differ not only in their format and level of detail, but also in their emphasis and the degree to which related information is included. Clarifying and elucidating require a careful mixture of reminding and introducing. In this paper, we address only the first four types of answers; automatically executing plans for the users is described elsewhere [7].

Although the categorization in figure 2-1 constrains the question, while the taxonomy in figure 2-2 constrains the answer provided, the requisite knowledge and the processes needed to search that knowledge are still complex. The processes include the abilities to estimate the user's goal, to understand the user's plan, to evaluate the current situation in order to formulate an answer that does not digress from the current task, to analyze the user's plan in terms of the estimate of the goal and within the current situation, and to choose an appropriate answer and explanation depending on the user's current knowledge of the system. This requires knowledge of the functions provided by the system, the possible goals that can be accomplished with the system, the plans that may accomplish those goals, the things that typically go wrong (bugs), and what the user currently does and does not know about the functions, goals, plans and bugs.

- **Introduce**: Present functions and plans that the user has not encountered before.
- **Remind**: Briefly describe functions and plans that the user has been exposed to but may have forgotten.
- **Clarify**: Explain details and options about functions and plans to which the user has been exposed.
- **Elucidate**: Clear up misunderstandings that have developed about functions and plans to which the user has been exposed.
- **Execute**: Perform functions and plans directly for the user.

Figure 2-2: Types of Responses a Consultant Might Provide

Much of this cannot be completely known. For example, it seems unlikely that all possible goals will be known before an interactive computing environment is used extensively. It also does not seem possible to predict with certainty what the user's goal is and what the user knows. Thus the processes described above not only must operate with incomplete information, but ought to be able to do it effectively. Innovative techniques or novel applications ought to be easily and reliably incorporated into the knowledge base.

From an AI perspective, these issues can be encapsulated into two fundamental problems: (1) How can the search through a vast and complex knowledge base be restricted in order to glean the appropriate information for the immediate needs of the user? and (2) What decisions must be made in order to choose the appropriate form in which to present that information? We describe solutions to both of these problems.

2.1. A Goal-Centered Approach

We propose a goal centered approach in which the help given is a direct function of a user's needs within the current context, treated as a discourse between the user and the consultant system. In particular, we are interested in the content of the answer provided to users. The primary contributions of this research are as follows.
It is not enough for a consultant to know about the functions of an environment. Explicit knowledge of how to combine those functions into plans that accomplish computational goals is equally important. We have developed a hierarchical knowledge representation, which allows our consultant system to reason about the actions associated with functions, but also allows it to analyze whether plans can satisfy goals and which of many equally good plans is most appropriate in a given context.

In order to determine a user's needs we abandon a simple categorization of users as novice, intermediate and expert. Similarly we found it insufficient to cluster goals, plans and functions into groups such as simple or hard. Instead we exploit the structure of our knowledge base and use a goal centered representation as a user model. Decisions about how to answer a user's question are based on an analysis of the match between the knowledge base and the user model.

A good consultant must know more than how to accomplish a particular goal, or what a particular function does. A good consultant does not simply perform a "core dump" of relevant information, but filters that information to satisfy pedagogical goals. Our knowledge representation also contains explicit discourse information to satisfy pedagogical goals.

The feasibility of our approach is explored through an automated consulting system called GECIE (Generated Explanations for Consulting in Interactive Environments -- pronounced "Jessie"). We have applied GECIE to the real world problem of the Berkeley Unix mail system [15], notorious for the great power it provides experts and the great confusion it creates for novices and even long-term non-expert users. Our goal is not to replace this mail system, as has been done by others [5, 16], but instead to augment it with consulting behavior that makes its capabilities accessible to casual users. For example, a user might ask "what does 'type' do?" If the user knows nothing about either the 'type' or 'print' functions, GECIE would provide the standard introduction to the 'type' function. But if the user already knows about 'print', then it should explain that 'type' is a synonym for 'print'. If the user has previously used 'type', but has apparently forgotten what it does, the consultant should simply remind the user of the 'type' function as briefly as possible. This is an example of a "What does function F do?" question. Examples of the other question types from figure 2-1 in the context of Berkeley Unix Mail are presented in section 4.

Since many discourse-based systems focus on determining a user's goals (e.g., Wilensky [20], Pollack [13]), we assume the output of such an understanding mechanism as input to GECIE, and instead are concerned with how to take advantage of this understanding to generate useful responses to user queries. We also assume that what the user knows about the environment can be determined by simply monitoring everything typed by the user. Therefore, the user's question and goal, and a representation of the user's knowledge are hand-coded as input to GECIE. In the above example, we assume that GECIE is given the question in symbolic form and that the user model accurately reflects whether the user already knows about 'type', 'print', or both, and concentrate our efforts on generating the best answer for the context.

3. Related Work
The development of programming environments [1] has focused on what the user can do rather than on how the user learns to do it. UC [3, 20], WIZARD [4] and ACRONYM [2] have articulated the need for comprehensive information accessing mechanisms. Evaluations of on-line help using ACRONYM indicated that the information itself is more important than the means for accessing it. UC and WIZARD both assume this, and provide information in the context of the user's goal. Both research groups acknowledge the need for pedagogical goals or "tutoring strategies", but have not studied them beyond stereotyping functions along a novice/expert spectrum.

Quilici et al. [14] have demonstrated how goal/plan knowledge can be used to answer questions, but they do not describe how the form and content of a response is affected by what the user already knows. Others [4, 6, 19, 20] identify the importance of plans, but they do not include in their knowledge bases the explicit discourse information needed to satisfy pedagogical goals. Much of the recent work on explanation [9, 17, 18] involves determining an appropriate level of detail or developing techniques for making inference chains coherent. McKeown [11] and Paris [12] go further to show how the decision of what to present from the knowledge base is dependent on the user's focus of attention and level of expertise.

Our work should be viewed as an extension of McKeown and Paris, but in an environment that is highly procedural. Our emphasis is on how to do something, rather than on what something is. A second distinction is that in an interactive computing environment there is often not only more than one way to explain something, but more than one way to do something. Therefore the analysis process that determines the most appropriate procedure affects and is affected by the generation process that produces the form and content of the answer.

4. GECIE: A Consultant for Interactive Environments
Consulting can be characterized as a three stage process of question understanding, problem analysis and answer generation. We concentrate on the latter two components: analysis, through an algorithm called the Plan Analyst; and generation, through an algorithm called the Explainer. GECIE attempts to answer a question by doing a two phase search of the knowledge base, first trying to construct a coherent relationship between the user's question and goal in an attempt to find the most appropriate information, then trying to construct a coherent textual explanation that takes
into account what the user already knows. If a perfect match cannot be found in either stage, problem solving mechanisms are employed.

Gm's "guru" knowledge base is a hierarchy of the computational goals that can be satisfied in the environment. Figure 4-1 shows the structure of this frame-based knowledge representation. Computational goals contain links to alternative plans for satisfying the goal. A plan can be linked to a subgoal or an ordered sequence of subgoals that describe how it can be executed indirectly, or to a function that executes it directly. Encoded within a computational goal are links that describe the relationship between plans. Functions describe the operators of the environment. Their representation includes information about the correct syntax of the function, any preconditions and effects, and the actions associated with parameters.

Figure 4-1: Gm's Frames for Knowledge Representation

Preconditions and effects of functions describe requisite and resulting world states, respectively. The current world model is represented as a simple add/delete list. Preconditions define a state that must be true. They may also contain a pointer to a goal that could satisfy it, which is extremely useful discourse information. Effects encode the actions of functions when applied to the world model: namely to add or delete a state.

Besides the guru knowledge base, Gm requires as input symbolic representations of the user's question and the current world model. It also requires a reasonable estimate of the user's goal and the user's knowledge of the environment. Finally, in some instances such as diagnosing a faulty plan, it also requires a symbolic representation of the user's attempted plan.

The user model uses a representation similar to the guru knowledge base. It contains a history of what the user has done in terms of what goals have been accomplished and what plans and functions were used to accomplish them. Selecting pedagogical goals to form a response is then simply a matter of matching what the user knows and has done with the knowledge base.

Since most of Gm's responses are rather stereotypical, the final text generated is the result of a pass through a very simple functional unification grammar [8]. For the purposes of this paper, this pass can be viewed as executing directives for filling textual templates. A few templates are presented in figure 4-2. Operations appear in capital letters; variables are surrounded by braces ("{" ").

We present five example queries that demonstrate Gm's capabilities. The first two are described in detail, while the processes for answering the other three are examined only briefly.

4.1 Example 1

The first question is: What does 'type' do? This is an instantiation of the "What does F do?" category of figure 2-1. Gm requires four inputs: GKB -- the guru knowledge base; UM -- the user model; WM -- the world model; and F -- the function about which information is sought. Figure 4-3 shows the por-
tion of GKB required to answer this question. GECIE's Plan Analyst phase determines whether the user already knows about type, and in this case, since there is a relational link to print, whether the user knows about print. The outcome of this analysis is passed to the Explainer.

Four possible responses are illustrated in figure 4-4. If the user knows nothing about either 'type' or 'print', GECIE generates the standard introductory template for type, and does not overwhelm the user with the fact that print is a synonym. If the user knows about print, GECIE states the fact that type is a synonym, reminds the user about print, and then introduces type. If the user knows about type but not print, GECIE reminds the user about type and makes an aside that there is a synonym for type called print. Finally, if the user knows about both, GECIE just reminds him about type.

The second question is: How can I forward a message? This question is an instance of the "What does plan P do?" category of figure 2-1. GECIE again requires four inputs: GKB, UM, WM and G. GECIE constructs a trace through the goal hierarchy and passes it to the Plan Analyst. The Plan Analyst searches for P in GKB. If P is found, the Explainer uses GOAL-REMINDS to justify why the user's plan is best. If no plans are found, the Explainer uses GOAL-INTRODUCE-COMPLEX to explain the plan, then DESCRIBE_LINK to justify why the new plan is better.

Three possible responses are illustrated in figure 4-6. If the user does not know anything about how to forward a message, GECIE selects the default plan. This information is encoded in the relational links. If the user has forwarded messages in the past, then GECIE simply reminds the user about the command Reply. But, if the user seems to know how to forward messages, but does it awkwardly, then the user is told that GECIE knows the user's plan, and introduces a better way, explaining why it is better by providing the relational links between goals.

Figure 4-4: GECIE's response to Question 1

4.2. Example 2a
The second question is: How can I forward a message? This is an instantiation of the "How can I satisfy G?" category of figure 2-1. GECIE again requires four inputs: GKB, UM, WM and G -- the goal about which information is sought. Figure 4-5 is a graphic representation of a portion of GKB required to answer this question. In this case the Plan Analyst constructs a trace through the goal hierarchy and passes it to the Explainer.

Figure 4-5: GECIE's Guru Knowledge for Question 2a

4.3. Example 2b
A refinement of the second question is: To forward a message to a group of users I reply to each individually - is there a better way? This is an instance of the "Given P is there a better P for G?" category of figure 2-1. In addition to GKB, UM, C and G, GECIE now requires a fifth input -- the user's stated plan, which is represented as an ordered list of goals and functions. This question type is analyzed in the same way as those in example 2, but a greater emphasis is put on finding alternative plans, that is, the Plan Analyst searches more carefully to construct an alternative plan. If none are found, the Explainer uses GOAL-REMINDS and DESCRIBE_LINKS to justify why the user's plan is best. If a better plan is found, the Explainer generates a response that uses GOAL-INTRODUCE-COMPLEX to explain the plan, then DESCRIBE_LINK to justify why the new plan is better.

4.4. Example 3a
A third question is:

This question is an instance of "What does plan P do?" GECIE's inputs are GKB, UM, WM and P. The Plan Analyst searches for P in GKB. A detailed discussion of the process employed by the Plan Analyst is presented in [21]. If P is
found, the Plan Analyst searches UM for knowledge of any goals directly satisfied by the plan. If some are not found, that is, if the user's knowledge is not that specific, the search continues to higher level goals until a match is found. The Plan Analyst reports the plan and the associated trace to the Explainer. The Explainer, in turn, uses the highest level goal it receives to summarize what the plan does, then for each step in the plan chooses REMIND or INTRODUCE templates depending on the user's knowledge represented in the trace. When the plan as input to Gm is found to be faulty, the Explainer uses DESCRIBE_FAULT to articulate the problem, and then proceeds as if the question were of type 3b as described in the next example.

4.5. Example 3b
The last question we consider is a modification of question 3a, but the user actually identifies the plan as bad: I'm trying to save some messages as I read them using the 'copy' command but I keep getting the same old new mail. Why? This is an instance of "Plan P ought to accomplish goal G in the context of situation S, but doesn't, why not?". The input is GKB, UM, WM, G and P. Since both G and P are given, the Plan Analyst uses means-ends analysis to construct the relationship between them. It matches this trace with UM so that the explainer can choose the most appropriate templates.

5. Implementation Status
Gm is implemented in C on an IBM PC-AT running MSDOS. We are currently integrating Gm with the mail system available with Unix 4.3 BSD on Digital Equipment Corporation VAX™ 750s. Our next step is to augment the functional grammar to produce graceful text, and to expand both the Plan Analyst and Explainer algorithms to handle combinations of the simple question types discussed here.

We plan to substitute Gm for the Berkeley Unix mail system for obliging human guinea pigs, to enable a fieldstudy. Users will be required to construct questions from an interactive menu, and the user model will be updated by hand, based on an analysis of the history of recent mail sessions.

6. Conclusions
This paper describes our extension of previous research on question answering. We focus on answer generation within the context of extending user expertise in interactive computing environments. Rather than categorizing users along a spectrum of expertise, and functions along a spectrum of level of difficulty, we present a goal centered approach where an answer to a question about the environment is based on knowledge of what the user has done in the past. Finally, we have applied this technology to a practical domain, the Berkeley Unix mail system.

ACKNOWLEDGMENTS
Ursula Wolz is supported by ONR grant N00014-82-K-0256. Gail Kaiser is supported in part by grants from AT&T Foundation, IBM, Siemens Research and Technology Laboratories, and New York State Center of Advanced Technology — Computer and Information Systems, and in part by a Digital Equipment Corporation Faculty Award. We thank Michael Lefkovitz and Kathy McKeown for their helpful insights and ongoing support of this project.

REFERENCES