Tutoring That Responds to User Questions and Provides Enrichment

Ursula Wolz*

Technical Report CUCS-410-88

Columbia University
Department of Computer Science
New York, NY 10027

Abstract

An aspect of human tutoring that epitomizes individualized instruction, is the ability to take advantage of a situation to teach something new. This type of tutoring can be characterized as *enrichment* as opposed to tutoring that is in *response* to a particular need. At Columbia, we have developed a model for consulting in interactive environments that permits answers to queries to be both in response to a user's questions and as enrichment. Our demonstration system GENIE (Generated Explanations for Consulting in Interactive Environments) produces answers in the domain of Berkeley Unix™ Mail. Our emphasis is on skill rather than factual acquisition. Consequently, GENIE is able to answer questions about alternative plans for goals, and whether plans satisfy goals. We abandon stereotypes of both functionality and expertise, and choose the best plan for a goal based on the current context and what the user has previously done. Four tutoring strategies are used both in response and as enrichment. They are reminding, introducing, clarifying distinctions and elucidating misconceptions.

*This Research is supported in part by ONR grant N00014-82-K-0256, by NSF grant IST-84-51438, a grant from DARPA, and a grant from Siemens Research and Technology Laboratories.

1. Introduction

An aspect of human tutoring that epitomizes individualized instruction, is the ability to take advantage of a situation to teach something new. This type of tutoring can be characterized as *enrichment* as opposed to tutoring that is in *response* to a particular need. At Columbia, we have developed a model for consulting in interactive environments that permits answers to queries to be both in response to a user's questions and as enrichment. Our demonstration system GENIE (Generated Explanations for Consulting in Interactive Environments) produces answers in the domain of Berkeley Unix™ Mail. We are guided by the principle that computing technology can have the greatest impact on education by providing exploratory environments in which the user, not the machine initiates an interaction. Therefore our focus is on building support systems through which users can learn to manipulate the functions of the environment.

Although a mail system may not appear to be a pedagogically rich environment, we chose it for several distinctive features. First, it offers a simple straightforward domain that increasing numbers of people learn to use only at a minimal level. Secondly, the basic functionality of the environment, such as reading, sending and filing mail are intrinsically useful, so that user's are motivated to learn about them. Third, the activities in a mail system do not require the sophisticated problem solving skills necessary for programming languages such as C, Pascal or Lisp. Therefore, we can separate issues of teaching the mechanics of use from those of teaching problem solving. This paper focuses entirely on the former.

GENIE is an attempt to model the human "guru hacker" usually found in academic and research computing environments. GENIE is an attempt to behaviorally rather than cognitively model such consultants. It is a hybrid of a tutoring system and help system. It tries to extend user expertise as a tutor would, but does not contain a curriculum. It passively answers users queries as a help system would, but makes more sophisticated choices about what to say when.

2. Choosing What To Say

Interactive environments are procedural rather than factual, consequently a consultant must be able to explain *how to do something* rather than *what something is*. Question answering therefore requires knowing how to satisfy computational goals through plans that map to sub-goals, or that map directly to the functions defined in the environment. A computational goal may be satisfied by more than one plan, and the "best" plan for a goal will depend on what the

user knows and the current context, such as the state of the environment and the existence of specific objects during a session. GENIE's Expert and User Model explicitly represent alternative plans for goals, along with the semantic relationships between plans so that the best plan can be calculated within a given context. Functions are represented as degenerate plans and include information on syntax, parameters, preconditions and effects. In the discussion that follows, plans will often refer to both functions and plans.

We abandon stereotypes of functionality and expertise such as those used by Chin [Chin 88]. We neither classify the functions of the environment along a spectrum of difficulty, nor users along a spectrum of novice to expert. GENIE chooses what to say based on the question intent, the user's task (the computational goal), the current context, which is represented as an add/delete list, and the User Model, which contains a goal-centered history of the user's past actions. We classify the potential questions that can be asked in procedural environments into these question intents:

1. How can I satisfy goal G?
2. What does plan P (function F) do?
3. Does plan P satisfy goal G?
4. Plan P doesn't seem to satisfy goal G, why not?
5. Is there a better plan than P to satisfy goal G?
6. What is the difference/similarity between plans P_1 and P_2?

Question answering is as a process of understanding the user's question, analyzing the best course of action, and explaining that course to the user. GENIE does this through three components: an *Understander*, a *Plan Analyst* and an *Explainer*. The Understander parses an English question, and maps the utterance to a question intent, and if provided in the utterance, a computational goal and a stated plan or function. Given a Goal, the Plan Analyst[1] locates the *best plan* for the goal. Given a plan, it finds the most appropriate goal, then checks whether the given plan is the best plan. Given both, it confirms or denies that the plan satisfies the goal, if it does, then it again checks for the best plan. The Plan Analyst also searches the User Model to determine if it contains a plan for the goal. If the User Model contains the best plan, then the Plan Analyst also produces a relevant related plan, so that the Explainer can opportunistically

[1]A more detailed discussion can be found in [Wolz & Kaiser 88; Wolz 88].

introduce it.

2.1. Choosing How to Say it: Responding and Enriching

The Understander and Plan Analyst provide the Explainer with a computational goal (G), a question intent (QI), a best plan (B), a user model plan (U). If provided in the question itself, they also supply a stated plan (S), and if U = B, a related plan R. The Plan Analyst also provides information on the relationships between the plans and the goal. Specifically, whether each of the follow propositions is true:

```
G exists, U exists, B exists, S exists, R exists

S satisfies G, U satisfies G S = U, S = B, U = B
```

Note that the user may ask about a plan that is not in the user model, and by definition if B exists, then B satisfies G.

The Explainer chooses to respond and possibly enrich through four tutoring strategies:

- **Introduce:** Presenting plans that the user has not encountered before.
- **Remind:** Briefly describing plans that the user has been exposed to but may have forgotten.
- **Clarify:** Explaining distinctions and options about plans.
- **Elucidate:** Clearing up misunderstandings that have developed about plans.

These strategies have been identified through an analysis of reference manuals, support manuals, textbooks, tutorials and on-line help [Wolz *et al.* 88]. The strategies are similar in form, but not content to the schemata used by McKeown [McKeown 85], McCoy [McCoy 86] and Paris [Paris 87]. They specify the kind of information that is typically included. For example, Figure 2-1 describes the process of introducing. Tutorials, textbooks and to some degree support manuals are intended to *introduce* new material, while the concise definitions in reference manuals and on-line help such as the Unix man function, can efficiently *remind* users of how functions work. All of these resources may help users *clarify* distinctions or *elucidate* misconceptions of functions, but the user must possess strategies for locating the relevant information. Clarifying or elucidating only occurs in a limited way in tutorials and textbooks. The user's goal may not occur in a written text. Even if it does, texts tend to introduce the simplest techniques for accomplishing a goal.

Figure 2-2 summarizes the rules for selecting strategies. These were developed as a result of

```
1. Stating the goal. *
2. If the skill maps to a function, introducing the function, otherwise:
3. Summarizing the sub-goals for the plan for the goal.  For each sub-goal either
   introducing or reminding the sub-goal depending on whether the user model does
   or does not contain the sub-goal.
4. If it is a top level goal, reviewing the steps in the plan through an example.
5. Relating each step in the example to a sub-goal. *

 Introducing a function consists of:

1. If not in the context of introducing a goal, stating the goal.
2. Presenting the syntax.
3. Describing the parameters.
4. Describing any preconditions that must exist for it to work.
5. Describing the effects (which is not the same as stating the goal.)
6. If not in the context of introducing a goal, giving an example.  *
                '*' indicates that a step occurs only in response
```

Figure 2-1: Description of Rules for Introducing

systematic analysis of the combinatoric possibilities between the QI, G, S, U and B. In any circumstance one responsive strategy and possibly one enrichment strategy will be invoked.

Introducing is used to provide generic details about a function or plan assuming no previous background about it. In GENIE, a plan is introduced responsively when the consultant thinks the user has no previous knowledge of it, doesn't know the best plan for the goal, or has a faulty plan for the goal. An exception is when the question intent specifically asks for a better way, in which case the clarify strategy is used. Introducing as enrichment occurs when the Plan Analylst produces a related plan R.

Reminding is used to present the bare minimum of information about a function or plan under the assumption that the user has some knowledge about it from previous experience, but may have forgotten it, or seeks confirmation about its relationship to a goal. In GENIE, a plan is reminded responsively when the user does not state a plan and the consultant thinks the user already knows the best plan. Reminding is also used when the stated plan is a plan the consultant thinks the user already knows. A justification for reminding as enrichment has not been found, in fact, it seems rather pedantic.

Notation: P_E = P exists $P_{\sim E}$ = P does not exist
P_V = P is valid $P_{\sim V}$ = P is not valid
P_f = P after invalidity is fixed
Q_5 refers to question intent "find a better plan than S"
& = AND, | = OR

Introduce Responsively:

Intro(B): $S_{\sim E}$ & (U_E | $U_{\sim V}$)
Intro(S): S_E & (U_E | $U_{\sim V}$ |[U_V<> B)])

Introduce As Enrichment: Intro(R): S_E & (U_V = B)

Remind Responsively:
Remind(B): S_E & (U_V = B)
Remind(U): S_E & (U_V <> B)
Remind(S): (S_V = U_V) & ([S_V = B] | [~Q_5 & (S_V<> B)])

Clarify Responsively: Clarify(S): Q_5 & (U_V = B)

Clarify As Enrichment:
Clarify(U): $S_{\sim E}$ & (U_V = B)
Clarify(S): (Q_5 & [S_V <> B]) | ($S_{\sim V}$ [(S_f <> B])

Elucidate Responsively: Elucid(S): $S_{\sim V}$ & ~Q_5
Elucidate As Enrichment: Elucid(U): U_V

Figure 2-2: Summary of Conditions that Determine the Strategy

Clarifying is used to compare two functions or two plans for a goal. In GENIE a plan is clarified responsively when the intent of the question is specifically for a better plan for the goal. A plan must be stated, and clarification occurs if it is not the best plan. Clarifying as enrichment occurs whenever the question intent is not specifically asking for a better way, but the plan stated is not the best plan, or when there is no stated plan and the User Model plan is not best.

Elucidating is used to clear up misconceptions and will be used most often when dealing with an individual's problems. In GENIE, a plan is elucidated responsively when the user states a plan for a goal that the consultant does not think is valid. Elucidating for enrichment occurs when the consultant thinks the user has a plan for the goal that is not valid.

3. Examples of Enrichment

Two examples are presented that show how GENIE provides enrichment. Others can be found in [Wolz & Kaiser 88]. Consider the question "How can I answer a message?". The question identifies a goal (*reply to a message*) and has a question intent (QI) to receive a plan in response. Assume that the Current Context contains a message that the user is currently reading that was sent only to the user, not to some group. Figure 3-1 is a graphic representation of the Expert Model required to answer this question. The Plan Analyst uses two sets of heuristics. First it chooses steps in plans based on "world knowledge" such as efficiency and temporality. For example, user's normally prefer to do things now rather than later, and do things in the current context. The Current Context dictates that the Plan Analyst choose as the best plan B the path indicated in the figure by the darkly shaded boxes. If it can't make a choice based on world knowledge the Plan Analyst uses a second set of heuristics that compare candidate plans to knowledge in the User Model.

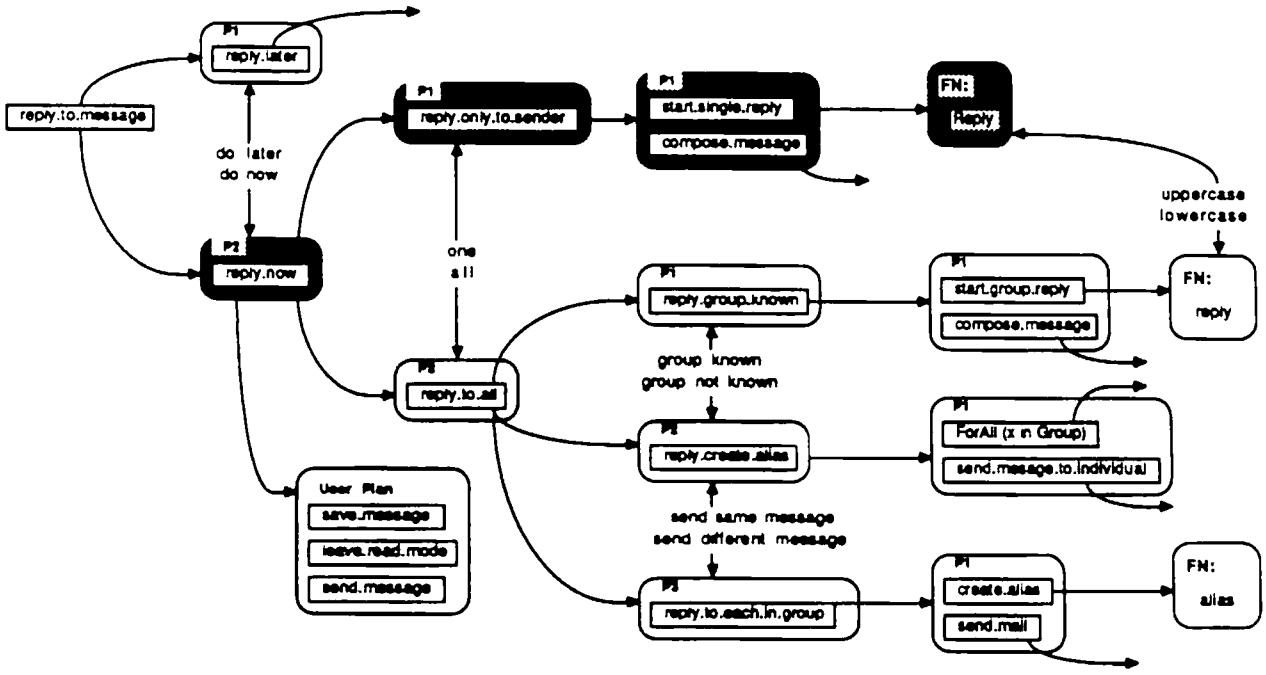

Figure 3-1: GENIE's Expert Model Knowledge of Replying to a Message

Assume the User Model contains a plan that does not exist in the Expert Model, but that is

valid. This plan is indicated in the figure by the lightly shaded box labeled ''User Plan 3''. Also assume that the User Model has a plan for *compose.message*. The Explainer chooses to *Remind Responsively* because the user has a valid plan for the goal, and then *Clarify as Enrichment* because the user's plan is not the best plan. Figure 3-2 presents both a process trace of the directives that are assembled for producing text and the template based text that is produced.

A second example illustrates how a related plan is introduced. Consider the question "What does `Reply` do?" The question identifies the function *Reply* (a degenerate plan), and has a question intent (QI) to receive a goal in response. Assume the User Model contains the goal *start.single.reply* that is satisfied by the function `Reply`. The Plan Analyst locates the goal for `Reply` which is *start.single.reply*, and notices that the goal and plan exist in the User Model. It therefore looks for any relevant related plans and discovers that the function `Reply` has a semantic link to the function `reply`[2] which allows a user to reply to every one to whom the original message was sent.

The Explainer chooses to *Remind Responsively* because `Reply` is both the best plan (B), and the User Model plan (U). Since the Plan Analyst found a related plan (R) in the form of function `reply`, the Explainer also chooses to *Introduce as Enrichment.* Figure 3-3 presents a process trace.

4. Summary

In this paper we have shown how a component of tutoring can be implemented as a consultant system. We have presented rules for choosing between answers that are in response and those that are as enrichment through four tutoring strategies. The focus of this paper has been on deep structure generation. Therefore we have made no claims as to how a question is actually understood. Furthermore we have made no claims as to how the User Model, and for that matter, the Expert Model are created and updated automatically. However, our emphasis on goals with alternative plans, and our abandonment of stereotypes provides a more direct focus that should shed light on both problems.

In our current approach the tutoring strategies are implemented as separate entities that map to templates. We have found that this approach make the content of the answer is rather rigid,

---

[2]We concede that this distinction is "user hostile" but the example is too good to resist.

```
Reminding: reply.to.message
  STATEGOAL:  reply.to.message
  SUMMARIZE: U
  EXAMPLE: U
Clarifying: reply.to.message
  SUMMARIZE: B
  COMPARE_STEPS(send.message, reply.only.to.sender)
    SHOW-RELATIONSHIP: (initiate addressee) (sender is addressee)
    Introducing:  reply.only.to.sender
      STATEGOAL:  reply.only.to.sender
      SUMMARIZE:  reply.only.to.sender.P1
        Introducing:  start.single.reply
              STATEGOAL:  start.single.reply
          IntroducingFunction:  Reply
               PRESENT-SYNTAX: [Reply]
                  DESCRIBE-PRECONDITIONS:
                     (mode = reading.messages)
            DESCRIBE-EFFECTS:
                     Delete (mode = reading.messages)
               Add (addressee = current.sender)
                        Add (interrupt.mode = compose.message)
          Reminding:  compose.message
                    ......
        EXAMPLE: (FN: Reply, Compose.message)

In order to reply to a message you save the message, then stop
reading mail, then send mail.  To do this, use the save command,
then the quit command, then the send command.  For example:
type: save then type: quit then type: mail [user-name]

There is a better way:
In order to reply to a message it is assumed you want to reply
right away.  In order to reply right away it is assumed you want
to reply only to the sender.  To do this, use the Reply command,
then type the message.  The difference is when you use the send
command you must specify the addressee, when you use the Reply
command the sender is the addressee.

The syntax is: Reply
To use this command you must be reading mail.  You temporarily
stop reading mail; you start sending mail; the addressee of your
message is the current sender.

For example, type the command: Reply
Then just type your message and end with <esc>.
```

Figure 3-2: An Example of Reminding then Clarifying As Enrichment

and the text itself awkward. We are currently attempting to mix the strategies in a more flexible way, and have plans to use a Functional Unification Grammar [Kay 79; McKeown & Paris 87] rather than templates to produce more graceful text.

```
Reminding: start.single.reply
   STATEGOAL: start.single.reply
     RemindingFunction: Reply
       PRESENT-SYNTAX: [Reply]
       EXAMPLE: (FN: Reply)

Introducing: start.group.reply
   STATEGOAL: start.group.reply
     IntroducingFunction: reply
       PRESENT-SYNTAX: [reply]
       DESCRIBE-PRECONDITIONS:
                          (mode = reading.messages)
       DESCRIBE-EFFECTS:
                          Delete (mode = reading.messages)
                 Add (addressees =
                        {current.sender current.receivers)
                          Add (interrupt.mode = compose.message)
   EXAMPLE: (FN: reply)
```

Figure 3-3: GENIE's An Example of Reminding then Introducing as Enrichment

We have however, shown how a consultant system can operate in an interactive environment. GENIE is able to answer questions about the task at hand, and within the current context. More importantly, it is able to take advantage of the situation in order to enrich the user's expertise, providing information that is not simply in response to a question.

References

[Chin 88] Chin, D. N.
Intelligent Agents as a Basis for Natural Language Interfaces.
PhD thesis, University of California, Berkeley, 1988.

[Kay 79] Kay, M.
Functional Grammar.
In *Proceedings of the 5th meeting of the Berkeley Linguistics Society.* Berkeley Linguistics Society, 1979.

[McCoy 86] McCoy, K. F.
The ROMPER System: Responding to Object-Related Misconceptions Using Perspective.
In *Proceedings of the 24th Annual Meeting of the ACL.* Association of Computational Linguistics, New York City, New York, June, 1986.

[McKeown 85] McKeown, K.R.
Text Generation: Using Discourse Strategies and Focus Constraints to Generate Natural Language Text.
Cambridge University Press, Cambridge, England, 1985.

[McKeown & Paris 87]
McKeown, K. R. and C. L. Paris.
Functional Unification Grammar Revisited.
In *Proceedings of the 25th Annual Meeting of the Association for Computational Linguistics.* Association for Computational Linguistics, Stanford, Ca., July, 1987.

[Paris 87] Paris, C. L.
The Use of Explicit User Models in Text Generation: Tailoring to a User's Level of Expertise.
PhD thesis, Columbia University, 1987.

[Wolz 88] Wolz, U.
Automated consulting for extending user expertise in interactive environments: A task centered approach.
Technical Report CUCS-393-88, Department of Computer Science, Columbia University, New York, NY, 1988.

[Wolz & Kaiser 88]
Wolz, U. and G.E. Kaiser.
A Discourse-Based Consultant for Interactive Environments.
In *Proceedings of the Fourth IEEE Conference on Artificial Intelligence Applications*, pages 28 - 33. 1988.

[Wolz *et al.* 88] Wolz, U., K.R. McKeown and G. E. Kaiser.
Automated tutoring in interactive environments: A task centered approach.
Technical Report CUCS-392-88, Department of Computer Science, Columbia University, New York, NY, 1988.