Automated Consulting For Extending User Expertise in Interactive Environments: A Task Centered Approach

A Proposal For Thesis Work

Ursula Wolz*

Technical Report CUCS-393-88

Columbia University
Department of Computer Science
New York, NY 10027

Abstract

Interactive computing environments provide facilities to support and assist the range from novice to expert users, but casual and novice users tend to rely on a small starter set of commands. This proposal for thesis work addresses this problem through the implementation of GENIE (GENerated Informative Explanations), a system that answers users' questions about how to accomplish tasks in the domain of Berkeley Unix™ Mail. This work unifies three new perspectives on consulting. First, the decision on what to tell a user, including the "best" plan for the user's goal, is based on an evaluation of the user's current computational goal, and the goals the user has attempted in the past, Secondly, the decision on how to phrase the answer relies on a careful mixture of tutoring strategies. Finally, both an expert and user model are represented as declarative structures of goals with alternative plans that include explicit semantic relationships between plans.

*Supported in part by ONR grant N00014-82-K-0256 and a grant from Siemens Research and Technology Laboratories..

Table of Contents

1. Introduction

Interactive computing environments are designed to provide supportive resources for a range of users with different expertise and computational goals. Such environments may be as simple as mail systems and word processors, or encompass sophisticated data bases, design tools or programming languages. Yet all such environments contain an underlying set of functions or constructs with which users accomplish tasks. A problem arises in providing resources through which users can initially learn about the environment and then later extend their expertise. The problem that will be studied in this thesis work is *how to provide automated consulting that extends users' expertise in interactive computing environments*.

The solution that is proposed takes a user's task centered approach to consulting in which help given is a direct function of the current context, users' computational goals, and their knowledge about plans to accomplish such goals in the environment. The solution will be presented through the implementation of GENIE (GENerated Informative Explanations), an answer generating system that specifically tutors to the current needs of the user in the domain of Berkeley Unix™ Mail.

This work will unify three new perspectives on consulting, namely that:

1. The decision of what to tell a user, including the "best" plan for the user's goal, must be based on an evaluation of the user's current computational goal, and the goals the user has attempted in the past, rather than on simple spectra of user expertise and functional difficulty.
2. The decision of how to phrase the answer must rely on a careful mixture of tutoring strategies. These allow the consultant to respond directly to the question and also to present related information as enrichment. Without such strategies, the consulting is not truly individualized and may as well come from canned text, or off-line materials.
3. A declarative representation of goals with alternative plans that includes explicit semantic relationships between plans can reduce the amount of brute force problem solving required to make a choice, and also facilitates the generation of more meaningful explanations. The representation is used both as a model of expert knowledge and as a user model.

1.1. Elaboration of the Problem

Whether the environment is intended for end users of commercial products or for software development staff writing such systems, increasing one's expertise within an environment is often avoided because it tends to cut significantly into productivity. Furthermore, in some environments in which the tasks are primarily 'throw-away', users may rely on inefficient methods that are well-known rather than taking time to develop more sophisticated expertise. A primary reason for the problem is that users bear the burden of deciding what must be learned and how to locate the appropriate information. This is typically done by searching through reference material such as manuals, asking help of some one with more expertise or simply experimenting with the system.

A phenomenon in development environments such as universities and corporate research centers is that users rely on local ''gurus.'' Information about how to accomplish tasks and how to recover from failures is learned through *cultural diffusion* [Papert 80] rather than through more formal methods such as tutorials, texts and seminars. This work is an attempt to capture the advantages of such local *consultant* power. This research is not an attempt to cognitively model a human consultant; that is, it does not present a theory on the mental processes used to consult. Rather, its purpose is to automate useful consulting behavior in a computationally effective manner.

Extending users' expertise can be viewed from two perspectives. An automated consultant can *do things* for the user, or can *tell the user how to do things*. This work focuses on the latter approach. The rational is based on an issue first articulated by Waters [Waters 86] that in order to do things for a user, a system and the user must have *shared knowledge* that must first be acquired by the user. Taking this perspective, extending users' expertise can be characterized as a four-fold problem:

1. How to represent the requisite knowledge.
2. How to identify that the user needs information.
3. How to choose the most relevant information to present.
4. How to choose the form in which to present the information.

These points are based on the observation that a good consultant has extensive domain knowledge, expertise in how to analyze and use that knowledge, and expertise in how to explain

things about that knowledge. A good consultant does not simply know how to use an environment effectively, but knows what to say, when to say it, how much to say, and what approach to take depending on what he or she thinks the user knows. The problem for a consultant is how to provide the appropriate information that neither swamps the novice (or casual user) with too much complex information nor insults the expert by providing an overly pedantic tutorial.

1.2. A Task Centered Solution

Figure 1-1 shows the requisite components of an automated consultant. The components surrounded by a thick grey line represent a prototype of GENIE called GECIE (Generated Explanations for Consulting in Interactive Environments). The knowledge representations appear in boxes, the processes in boxes with rounded edges. GECIE was developed in C on an IBM PC AT in order to explore the key ideas in this thesis work. It will be described in more detail in later sections. GENIE is being developed in Common Lisp on a Sun 3/60 with crucial extensions to GECIE as will be discussed in section 6. For demonstration and testing purposes GENIE will also include simple understanding mechanisms and a surface text generator.

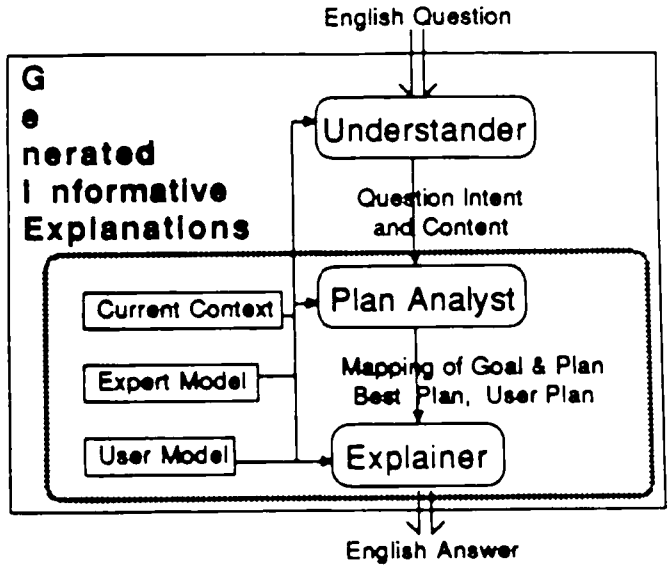

Figure 1-1: Components of GENIE

In this work the problem of representing requisite knowledge (point 1 above) is addressed through the development of a computational goal based knowledge representation used both as the Expert Model and a User Model that includes explicit semantic links between plans for those goals. The problems of choosing relevant information and the form in which to present it, (points 3 and 4) are addressed by making a separation between analytic expertise and explanatory expertise through two subsystems called the Plan Analyst and Explainer respectively. The problem of identifying the user's needs (point 2) is excluded from theoretical consideration because it has been explored extensively by others. Noticing that the user needs assistance can be done by actively watching the user's activities and inferring needs. Work by Selker [Selker 88], Quilici [Quilici *et al.* 85] and Finin [Finin 83] takes this approach. Passively waiting for the user to ask a question requires mechanisms for question understanding and is used by Wilensky [Wilensky *et al.* 84] and Pollack [Pollack 86].

The focus then of this work is on developing a unified approach to the problems of:

- **representing functional knowledge** in a way that is useful both for analyzing the relationships among plans for a computational goal, and for explaining those relationships,
- **implementing analytic expertise** that can choose the most appropriate plan for a computational goal in a given context, or determine whether a plan will satisfy a goal,
- **implementing explanatory expertise** that can tailor responses to the current context and user's knowledge both in response to a question and to enrich the user's knowledge.

Each of these points will be discussed in turn in the next three sections, including further elaboration of the pertinent issues, the solution proposed in this thesis work and related work by other researchers. Section 6 describes extensions that will be included as part of the thesis, and addresses the limitations of the work. Section 7 discusses how this work will be validated. Section 8 presents a summary and time line for completion of the thesis.

2. Representing Functional Knowledge

Potential solutions to the problem of representing functional knowledge can be found through insights into the nature of interactive computing environments. These can be characterized as workbenches of tools with which a user accomplishes tasks. In this respect they are *procedural environments* in which learners develop skills rather than learn facts and associations between facts. Sophistication and complexity, and consequently power, are built upon interfaces such as command languages, menus with keystroke or pointing devices, or even more sophisticated technology. Good environments are often characterized as *customizable* — in which users can bend the tools to their own personal needs, and *extendible* — in which users can build new tools from those that already exist. But at the core, a set of functions must be executed by the user. The user chooses a *computational goal* to accomplish, and either chooses a *plan* to satisfy the goal which may be composed of sub-goals, or chooses a *function* that directly satisfies the goal. Such functional knowledge is important to both analysis and explanation.

Analytic expertise includes deciding which plan is most appropriate in a given context, and therefore requires knowing the relationships between alternative plans for accomplishing a goal. For example, there are at least two ways in most mail systems to send a message to a set of people. One can type each address in turn when prompted for the receiver of the message. One can also create an *alias* which is a named list of addresses that can be reused, and type the alias name at the prompt. The first method is most appropriate when the set occurs only in this instance, or is very small and easy to remember. The second method is more appropriate if over a period of time many messages will be sent to this set of people. The context also plays an important part in the content of the answer. Rather than using ''canned'' examples, the explanation of what to do can be based on actual objects in the environment. In an electronic mail environment the objects include messages, users and collections of each.

The choice of plan and the strategy used to present it are influenced not only by the current context, but by the plans the user already knows. Therefore it is necessary to be able to distinguish the plans a users knows from the potential set of plans that accomplish a goal. For example, a user who is new to sending messages may be overwhelmed by hearing about aliases, even if the message is to be sent to a group of users. Similarly, the functional knowledge of the user influences both the level of detail and the tutoring strategy of the consultant. For example,

if the user has never attempted to send mail to a group, the consultant may choose to *introduce* the plan. However, if the consultant knows the user has some inefficient plan, the consultant should *clarify* the distinctions between the user's plan and a more efficient one.

The functional knowledge of both a consultant and a user can be characterized as a *web* of interrelated goals for doing tasks, plans for accomplishing those goals, steps within plans that are either goals themselves (sub-goals), or functions that describe the actions available in the environment. Choosing the functions or plan for a goal is a matter of navigating the web, making decisions about what plans, sub-goals and ultimately functions to use. In order to generalize the functional knowledge, information about the current context must also be represented.

2.1. The Expert Model and The User Model

The web is the basis of both the Expert Model and User Model proposed in this research. Presumably the former is considerably richer than the latter. The Expert Model is traversed in order to locate relationships between goals, plans and functions. It must include information that can be used to choose between plans and explain the choice. In GECIE, the Expert Model is a declarative structure that is searched in order to locate information. Using the Expert Model to engage in dynamic planning as well as extending, updating and modifying it are discussed in section 6.

The User Model is also a web of relationships between goals, plans and functions. It may contain plans that do not exist in the Expert Model, including ones that are faulty. It is used both to decide what to present and how to present it. The User Model also affects the choice of strategy in answering the user's question. The form of an answer will depend on whether the consultant thinks the user does or does not already know something, or whether it thinks the user has a misconception. Constructing and maintaining a User Model is also discussed in section 6.

From this perspective, expertise and complexity of functions can be characterized by the richness of the web, rather than as simple spectra as described by Chin [Chin 88]. Although spectra give the illusion of quantifiable criteria, the methods used to develop and validate them are often superficial at best. In the model presented here, functions that would be classified as ''hard'' or ''advanced'' can be better characterized as requiring an understanding of complex relationships between plans in order to navigate the web. Those that are ''simple'' or ''basic''

have more straightforward paths. Similarly, classifications such as ''novice'', ''intermediate'' and ''expert'' are hard to quantify. Here they are unnecessary because users are judged by what they know about the current task, rather than how much they know about the entire environment. It is perfectly plausible for a user to have a rich web of knowledge about a portion of the environment, and almost no expertise about others. For example, a user may have extensive experience sending simple messages, and almost none with modifying messages through an editor. Such a user will not fall nicely into a categorization of expertise. A question relating to sending simple messages will require introducing very little new information, while a question about modifying messages may require an thorough introduction to editing.

2.2. Knowledge Bases in GENIE

The Expert Model in GECIE, which will be used in GENIE, is a network of the computational goals that can be satisfied in the computing environment. Figure 2-1 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 sub-goal or an ordered sequence of sub-goals that describe how it can be executed. A function is a degenerate plan that satisfies a goal directly. Encoded within a computational goal are links that describe the semantic relationships between plans required for decision making and explanation.

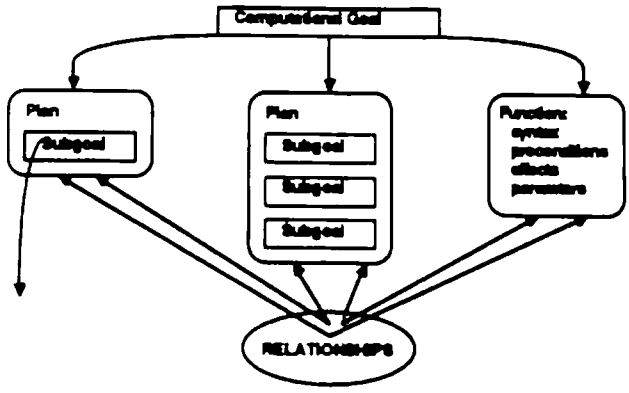

Figure 2-1: GENIE's Frames for Knowledge Representation

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. Preconditions define a state that must be true before a function can be correctly executed. They may also contain a link to a goal that could satisfy it. Effects encode the actions of functions when applied to the Current Context. The Current Context is represented as a simple add/delete list that describes possible states in the domain. Therefore, effects are encoded as directives to add or delete a state from the Current Context.

The User Model has the same representation as the Expert Model. It contains a history of what the user has done in past sessions in terms of what goals have been accomplished and what plans and functions were used to accomplish them. Its goals may contain plans that were attempted, but didn't work, or plans that do not exist in the Expert Model.

2.3. Influence of Related Research

The early stages of work on GECIE were heavily influenced by Schank's Mop theory [Schank 82] as implemented in RESEARCHER [Lebowitz 86]. Therefore, procedural knowledge was viewed from a declarative script-based perspective, rather than as condition action pairs operating on a state space [Fikes and Nilsson 71; Sacerdoti 77; Allen and Perrault 80]. The approach has strong ties to representations for program synthesis, most notably Kant's work [Kant 88] on Config, an intelligent program configuration editor. A careful analysis of the advantages of static scripted based versus dynamic planner based encoding of plans remains to be done, and will influence the degree to which the web itself can be considered a contribution to computer science.

The initial work on GECIE focused on explanation rather than planning. Consequently work on Intelligent Tutoring Systems and User Modeling have had a strong effect on the choice of knowledge representation. Constructing and maintaining a User Model has proven to be a difficult task in building Intelligent Tutoring Systems and more recently in Natural Language Question Answering systems. One problem is reliably determining what the user knows. A second is choosing how to represent the user's knowledge. A third is updating the User Model dynamically. Within the thesis itself it is hoped an argument can be made that the web paradigm of goals, plans and functions offers insight into solutions to the first two problems. The third, like updating the Expert Model, falls within the domain of knowledge acquisition and learning.

Early research by Brown and Burton [Brown & Burton 78] and by Sleeman and Smith [Sleeman & Smith 81] illustrate the difficulty in determining what the student knows in

unguided settings. The difficulty lies in diagnosing what the student doesn't know based on unexpected behavior. Since within GENIE, no behavior is expected in the first place, this problem is avoided by waiting for the user to notice that something is wrong. The user model is a reference point for choosing the form and content of the consulting dialogue, but is not the primary source for choosing what to say. There is only one instance — elucidating misconceptions as enrichment — where GENIE expects to find a "buggy plan" in the User Model. This case is a by-product of the representation rather than its focus and therefore does not play a significant role in how GENIE works.

The second problem is addressed by restricting the representation to knowledge of *what the user knows how to do* rather than the larger domain of *what the user knows*. For example no attempt is made to draw analogies to knowledge the user might have of things outside the environment. The second problem is also addressed by the nature of the domain. Since the user is engaged in using an environment, it should be possible to build a system to monitor his or her actions. For example, the Marvel software development environment [Kaiser *et al.* 88] monitors all commands entered by the programmer. Inferences could therefore be drawn on whether particular goals have been attempted and successfully accomplished and through what plans or functions. Selker [Selker 88] and Quilici [Quilici *et al.* 85] among others have demonstrated how such plan recognition is possible and useful for developing a User Model. Their work, however, concentrates on the user's activities, rather than the form and content in which to present information.

More recent research on User Modeling falls into two categories: either stereotypes are used to represent large categories of users and answers are geared to a category, or explicit beliefs and goals of the user are represented and reasoned about to determine an answer. Related work on beliefs and goals will be presented in the context of analytic expertise in the next section.

Chin's work [Chin 88] on the UC system [Wilensky *et al.* 84] best exemplifies the stereotype approach and is most relevant to GENIE. Chin's system provides help within the Unix environment based on a dual set of stereotypes. Users are classified as novice, intermediate, or expert and functions are classified as easy, intermediate[1], and hard. The system uses rules stating how much detail to provide about the different classes of functions depending on which

[1]Chin actually uses two levels of intermediate.

class the user falls into. The approach in this work is in direct opposition to Chin's: it is based on the assumption that knowledge cannot be quantified in discrete chunks and users are unlikely to learn and progress from one neat division to another. Rather, they are likely to learn functions based on tasks they have performed and this will vary substantially depending on the tasks.

The representation of the Expert and User Models is based on work by Goldstein [Goldstein 82], Genesereth [Genesereth 82] and Clancey [Clancey 82]. Goldstein introduced the notion of a network, a *genetic graph* of knowledge that represents stages of development of a student's understanding of a domain. Rather than overwhelm the beginner with an optimal method, the genetic graph provides a basis for choosing what to say. Its structure however, is based on degrees of student development which are very hard to quantify and evaluate. Furthermore, it asumes that all users follow certain stages of development. Since the web is task connected as well as knowledge connected it should be more tolerant of eclectic user knowledge. Genesereth's *dependency graph* of goals and plans suggests that this is the case. Genesereth focused primarily on diagnostics. He was therefore not concerned with encoding semantic information about the relationships between plans, which are seen here as crucial to explanation. Clancey first articulated the need for the separation of domain expertise from tutoring knowledge. Although he implicitly captures goal/plan knowledge in his rules, unlike GENIE his tutors can not reason abstractly about relations between goals and plans. Finally, none of these researchers seriously addressed the problem of generating a coherent response.

3. Analytic Expertise

Expertise in interactive computing environments is not only a matter of knowing what functions allow you to perform simple tasks. One must be able to analyze how functions and plans interrelate or interfere with each other when attempting to achieve more complex goals. A particular plan is directly related to a particular goal, but the components of a plan, the steps or sub-goals, may occur in many different plans. Furthermore, in most computing environments there is often more than one plan to achieve a goal.

Another insight into such analytic expertise comes from the observation that the intent of a user's question influences the consultant's behavior. This is often termed the *discourse goal*, since it is the goal of constructing some expression to elicit some information. For example, a user may ask a question in order to satisfy the discourse goal of "getting help". It is

distinguished from the computational goal of actually accomplishing some task in the environment. In order to decrease confusion between the two, the computational goal about which help is sought will be referred to as the ''goal'', and the discourse goal will be referred to as the *question intent.*

3.1. Influence of the Question Intent on Analysis

Within a procedural environment it is possible to reason about, and consequently ask questions about, the relationship between goals and plans, goals and functions, plans and functions, goals and other goals, plans and other plans, functions and other functions. The range of question intentions reduces to those in Figure 3-1. The utterance identifies a goal or plan (or function) or both, and implies an assumption about their validity. Its form also implies an expected answer. Therefore, in order to reply informatively, a consultant system must provide the information users expect, namely a goal, plan, function or relationship. It must also confirm or deny their assumptions about the validity of the goal, plan or function mentioned in the question.

3.2. Analyzing Goals, Plans and Functions in GENIE

The design of the analytic component of GECIE was based on informal observations of human consultants, and on analysis of written texts. In particular, we identified a need for reasoning about goals and plans, and being able to choose a ''best'' plan within a context.

Five basic parameters are required for choosing what to say and how to say it. These are:

- **A question intent - QI**, that provides an expected discourse focus of a goal, plan and/or function.
- **A computational goal - G**, which is either identified by the user in the question, or is inferred by the consultant from the plan or function specified in the question.
- **A stated plan - S**, which only exists if the user identifies it in the question. If it does exist, it may not be the same as a plan in the User Model — the user may have just learned it from someone else.
- **A User Model plan - U**, for the computational goal, which is the plan that the consultant thinks the user has used in the past for accomplishing the goal.
- **A best plan - B**, for the computational goal which is inferred by the consultant from the Expert Model, given the current context and the goals, plans and functions in the User Model. It may or may not be the same as S or U.

Question Intent	Question Identifies	Question Assumes	Question Expects in Reply	Intuitively
What plan/function is required to satisfy the goal?	Goal	Goal is possible	Plan or Function	How do I do it?
What goal is satisfied by this function or plan?	Function or Plan	Function or Plan satisfies some Goal	A goal	What does it do?
Does this plan satisfy this goal?	Function or Plan and Goal	Plan or Function satisfies Goal	Confirmation of assumption or explanation	Does it do it?
This plan doesn't work, for this goal, what's wrong?	Function or Plan and Goal	Plan or Function should not satisfy Goal	Confirmation of assumption or explanation	What's wrong with it?
Is there a better way to accomplish this goal?	Function or Plan and Goal	Plan or Function may not be best way to satisfy goal	Confirmation of assumption or explanation	What's a better way?
How are these Goals, Plans or Functions similar or different?	Pair of Functions, Plans or Goals	Plans, Goals or Functions exist	Relationship between the pair	What are the similarities or differences between them?

Figure 3-1: Information Imbedded in the Question Intention

In GECIE, the question intent, the goal, stated plan or function were given as input where appropriate. For example if the question intent was "How do I do it?", only a goal was given. In GENIE, the question intent must be derived from the form of the question through a question understanding component which will be described in section 6. Depending on the question, the goal and stated plan may or may not be derived from the question. If the latter cannot be derived from the question, then it is not used in the analysis process. If the goal cannot be derived from the question, it must be derived by the Plan Analyst. Similarly the Plan Analyst must derive a User Model plan and a best plan.

Given a plan or function and a goal, the Plan Analyst uses the Expert Model to find a "match" between them. A match is defined as confirmation that a plan or function satisfies a goal. Given a function and goal the Plan Analyst confirms or denies that they match. Given a

plan and a goal, it tries to find a match. If it is unsuccessful it uses the cases in Figure 3-2 to identify the mismatch. These are based on work by Joshi, Webber, and Weischedel [Joshi, Webber & Weischedel 84]. Given a plan, the Plan Analyst attempts to locate the goal it satisfies. If no actual match emerges from a list of candidates, then the goal with the least complex mismatch is chosen, but is marked as a mismatch along with its causes.

- A step in the plan has missing preconditions.
- A step in the plan is missing.
- A step in the plan is extraneous.
- A step in the plan that has missing preconditions is related to a step that is missing.
- A step that is missing is related to a step that is extraneous.
- A missing precondition is related to an extraneous step.

Figure 3-2: Types of Invalidities Found in Plans

Given a goal, the Plan Analyst searches the Expert Model for the "best" plan for the goal using two sets of heuristics. First it tries to choose steps in plans using "world knowledge" such as efficiency and temporality based on the current context. For example, users normally prefer plans that accomplish goals now rather than later, or that require fewer rather than more steps. For example, in a mail environment the choice may be affected by whether a message, address or alias already exists. The second set of heuristics compares the candidate plans to knowledge in the User Model, choosing the plan whose sub-steps occur most frequently in the User Model. The Plan Analyst uses one of three search strategies in order to complete the parameter list. They are:

- Given G, but not S, attempt to find B and U.
- Given S, but not G, attempt to find G, B and U.
- Given S and G, find a relationship between them, and find B and U.

The Plan Analyst also returns other information, such as whether the best plan, the stated plan and the User Model plan are the same. If there is no user plan, or if it does not match the best plan, then the Plan Analyst also attempts to locate components of the stated plan and the

best plan in the User Model.

3.3. Research Related to Analytic Expertise

The development of programming environments [Goldberg 87; Habermann & Notkin 86; Stallman 81; Kaiser *et al.* 87; Walker *et al.* 87; Reiss 87] has focused on what the user can do rather than on how the user learns to do it. These systems have attempted to reduce the amount of detail, and in the process, the amount of complexity to which the user must attend. Although users may be able to operate at a higher level of abstraction, they must still master the surface functionality of the system to use it. Furthermore, there are times when users do want to attend to detail. Even systems that attempt to protect them, should include sufficient explanatory power to justify their actions. Within Programming Environment research the focus has been on helping the user do tasks, rather than on how to explain or justify how tasks are done. Therefore there has been less reason to include mechanisms for evaluating the conceptual trade-offs between plans.

A body of work by Allen and Perrault [Allen and Perrault 80], Pollack [Pollack 86] and Appelt [Appelt 85], uses detailed information about user beliefs and plans in combination with a formal reasoning system to determine what to include in an answer. Because they have relied on such detailed formal models in combination with a theorem prover, they have tended to operate in limited, well constrained domains, producing shorter responses than those at which this research aims. Pollack is an exception, although she has focused more on the representation required to produce helpful responses for plan invalidities and less on the generation of the responses themselves. Pollack also encodes beliefs, plans and goals in logical predicates which are resolved with a theorem prover. In GENIE, these are separated and therefore made more accessible. Knowledge of goals and plans is put in the Expert Model, beliefs about the user in the User Model, and analytic expertise in the Plan Analyst.

The knowledge conveyed in a consulting session is determined by the current task or goal of the user and the user's questions about that task. Therefore the analytic process provides a valuable test bed for theories about personalized tutoring because of the unique relationship between the consultant and the user. By contrast, in most tutoring, both on and off the computer, the teaching agenda is predetermined by an implied curriculum chosen by the tutor. This is not appropriate for a consultant since users have a multitude of different needs, backgrounds, and

deficiencies in what they know. In task centered settings, consulting is based on tasks initiated by the user. In an exploratory learning centered setting, a tutoring agent could initiate the task, but the student may still require consultation in order to accomplish it.

In classic settings the tutor assumes some external force motivates the student to learn, or the tutor must find incentives to motivate the student. In a consultation session, motivation for learning is a matter of being able to do the task independently of the consultant. Brown and VanLehn [Brown & VanLehn 80] introduced the notion of *felicity conditions* under which new learning is most likely to take place. Within a procedural setting, two situations are extremely motivating. The user attempted to do something and it didn't work, or the user finds a task tedious and suspects there is a more efficient way to do it.

4. Explanatory Expertise

A good consultant does not simply perform a 'core dump' of relevant information, but filters that information to satisfy a pedagogical agenda. Further insight into implementing explanatory expertise comes from an analysis of on-line and off-line documentation. Although such materials and systems help the user learn about the functions themselves, such documentation can be inadequate at providing specific goal oriented help for the task at hand. Furthermore, since most tutorials only scratch the surface of the capabilities of an environment, users tend to rely on the few commands they learn initially and never develop broad expertise.

An analysis of on and off-line help, including help for Lisp, UNIX, Pascal, BASIC, Logo, and a number of word processing programs, reveals that reference material tends to fall into three categories:

- **Reference manuals** that provide details and definitions of the environment. The material is either alphabetically ordered or grouped according to the function of the constructs.
- **Support manuals** that provide more explanation about how to use the functions. These are organized according to the function of the constructs.
- **Tutorials and textbooks** that introduce the concepts behind the functions. These tend to be much more explanatory and less definitional than reference or support material. Although they may be organized according to the function of the constructs, there is a greater emphasis on how constructs are combined.

From these, four necessary tutoring strategies in computing environments have been

identified. They specify the kind of information that is typically included. Tutorials and to some degree support manuals are intended to *introduce* new material, while the concise definitions in reference manuals can efficiently *remind* users of how functions work. All three kinds of material may help users *clarify* distinctions or *elucidate* misconceptions of functions, but the user must possess strategies for locating the relevant information. Clarifying or elucidating goals 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. Although the information necessary for learning a better way may exist, the user is responsible for finding that information and must often piece it together from various points in the book.

4.1. Explanatory Expertise in GENIE

The Explainer uses the tutoring strategies to choose the form in which to present an answer. Specifically it chooses to:

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

The choice of strategy is based on the relationship between QI — the question intent, B — the best plan for the goal under discussion, and if they exist, U — the plan for the goal in the User Model, and S — the plan for the goal stated by the user. Figure 4-1 summarizes the rules for choosing strategies.

The strategies are extended by using them differently to satisfy distinct tutoring needs: the need for tutoring that is in *direct response* to the question and tutoring that is intended as *enrichment*. The former prevails in order to satisfy principles of informativeness: answer the question that was asked. But it is also possible to present new skills to the user opportunistically. For example if the user asks whether a particular plan will accomplish a particular goal, the consultant must respond informatively that it does or does not. However, if the consultant knows of a better way to accomplish the goal, the opportunity should be taken to mention it. Each strategy can be used both responsively and as enrichment. In the subsections that follow, steps

```
Introduce responsively when:
  (AND (QI <>
         get a better plan B than S)
       (OR (U does not exist)
           (AND (U exists)
                (OR (U is invalid)
                    (U <> B) ]

Introduce as enrichment when:
    (AND (U exists)
         (U <> B)
         (S exists)
         (S is not valid))

Remind responsively when:
  (AND  (QI <> get a better plan B than S)
        (OR (AND (S does not exist)
                 (U exists)
                 (U = B))
            (AND (S exists)
                 (U exists)
                 (U = S))))

Clarify responsively when:
  (AND (QI =
         get a better plan B than S)
       (S <> B))

Clarify as enrichment when:
  (AND (QI <>
         get a better plan B than S)
       (S exits)
       (OR (U <> B)
       (S <> B)))

Elucidate responsively when:
  (AND (S exists) (S is not valid))

Elucidate as enrichment when:
     (U is not valid)
```

Figure 4-1: Summary of Conditions that Determine the Strategy

within a strategy that occur in response but not as enrichment are marked with '*'.

4.1.1. Introducing

Introducing is used to provide generic details about a function or plan assuming no previous background about it. It occurs most frequently in tutorials. An informal description of the process of introducing is shown in Figure 4-2. 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 stated by the user does work and none of the plans the consultant thinks the user knows is the best plan. For example if the user asks "To send mail to a group of users, do I type `alias` at the `TO:` prompt?", and the consultant believes this is the only plan the user knows (it is also the plan in U), then the consultant will first elucidate why the plan does not work, and will then introduce as enrichment the best plan, which

```
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.  *
```

Figure 4-2: Description of Rules for Introducing

may not require the alias command at all.

4.1.2. Reminding

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. Manuals most often use this strategy. An informal description of the process of reminding is shown in Figure 4-3. 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. For example, if the consultant has seen the user reply to messages in the past and the user asks, "How do I answer a message?", (no S is stated), then the user just needs to be reminded about the command; a long introduction is not necessary. A justification for reminding as enrichment has not been found, in fact, it seems rather pedantic.

4.1.3. Clarifying Distinctions

Clarifying is used to compare two functions or two plans for a goal. This is done occasionally in tutorials and textbooks, but is essential for face to face consulting and questions, when the user often queries about the difference between two plans or specifically asks for a

1. Stating the goal.
2. If the goal maps to a function, reminding about the function, otherwise:
3. Summarizing the sub-goals for the plan for the goal.
4. If this is the top level goal, reviewing the steps in the plan through an example.

Reminding about 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. If not in the context of introducing a goal, giving an example.
5. If not in the context of introducing a goal, relating the function to other pertinent information.

Figure 4-3: Description of Rules for Reminding

better way to achieve a goal. An informal description of the process of clarifying is shown in Figure 4-4.

1. Stating that a best way (B) exists for the goal.
2. Summarizing an alternative plan P. *
3. Summarizing B.
4. Describing the relationship between B and P.
5. For steps in B that differ from P, introducing or reminding those sub-goals based on whether those goals exist in the user model, and describing the relationship between this step and the corresponding step in P.
6. Summarizing the steps of B through an example.

Figure 4-4: Description of Rules for Clarifying

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. For example the user asks "Normally, to send mail to a group of users, I just type all the addresses at the `TO:` prompt, is there a better way?" The consultant may decide that it is time to introduce *aliasing*, and will clarify the difference between using a new plan that includes the `alias` function and the user's stated 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. For example, the user asks "Can I type more than one address at the `TO:` prompt" and the context suggests that the best way would be to create an alias. First, since the plan is the focus of the question, the consultant must responsively introduce or remind about the plan depending on what it thinks the user knows. Only then would it say, "by the way..." and clarify aliasing as enrichment.

4.1.4. Elucidating Misconceptions

Elucidating is used to clear up misconceptions and will be used most often when dealing with an individual's problems. Because most texts do not specifically address an individual reader, elucidating is found infrequently, to forewarn the user of a possible misconception that can occur. An informal description of the process of elucidating is shown in Figure 4-5.

---

1. Stating that the plan does not work for the goal.
2. Summarizing the plan, identifying the problem.
3. If the problem is missing preconditions, either state that no plan exists to satisfy them, or introduce or remind about a plan to satisfy them depending on whether the plan exists in the user model.
4. If the problem is a missing step, introduce or remind about it depending on whether it exists in the user model.
5. If there is an extraneous step, identify it, and describe why it is extraneous - what effects does it have that are redundant with some other step.
6. If missing preconditions are related to a missing step, identify the relationship between the two, and introduce or remind about the missing step depending on whether it exists in the user model.
7. If a missing step is related to an extraneous one, clarify the difference between them.
8. If an extra step is related to a step with missing preconditions, clarify the difference between them.

Figure 4-5: Description of Rules for Elucidating

In GENIE, a plan is elucidated responsively when the user states a plan for a goal that the consultant does not think is valid based on those in Figure 3-2 drawn from [Joshi, Webber & Weischedel 84]. For example if the user asks "To send mail to a group of users, do I type `alias` at the `TO:` prompt?", the consultant notices that a precondition to using the `alias` command is that one be at the `Mail>` prompt. The consultant provides a solution: return to the `Mail>` prompt, create the alias, then begin to compose the message.

Elucidating for enrichment occurs when the consultant thinks the user has a plan for the goal that is not valid. Since the plan in the User Model is never the focus of discourse (unless it is equal to the stated plan, in which case the stated plan is still the focus of the discourse) it can never be elucidated in response. However under some circumstances it may seem opportune to address what the consultant thinks the user knows. This must proceed in a delicate manner since it is based on knowledge the consultant believes rather than knows.

4.2. Research Related to Explanatory Expertise

Research on UC [Wilensky *et al.* 84; Chin 86], WIZARD [Finin 83] and ACRONYM [Borenstein 85] has 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 tutoring strategies, but have not studied them beyond stereotyping functions along a novice/expert spectrum.

Evidence for two of the strategies, *introduce* and *remind*, is also provided by Magers [Magers 83] and Borenstein [Borenstein 85] who have drawn a distinction between information that is *definitional* and *instructional*. 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. We therefore choose to *remind* or *introduce* depending on the user's knowledge and goals. We further refine the distinction of Magers and Borenstein by including the possibility of elucidating or clarifying.

Quilici *et al.* [Quilici *et al.* 85] have demonstrated how goal/plan knowledge can be used to answer questions in computing environments, but they do not describe how the form and content of a response is affected by what the user already knows. Others [Wilensky *et al.* 84; Johnson 86; Waters 86; Finin 83; Pollack 86] 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 [Kukich 85; Swartout 83] involves determining an appropriate level of detail or developing techniques for making inference chains coherent, but

similarly is not concerned with the form of the answer.

The use of tutoring strategies is quite similar to the use of schemata in natural language generation work by McKeown on TEXT [McKeown 85], Paris on Tailor [Paris 87] and McCoy on Romper [McCoy 86]. This work differs in the complexity of schemata and in the domain of discourse. The TEXT system generated text on factual information about objects and their components. Similarly, Tailor mixed factual information about objects' components and purposes. Neither system however attempted to take into account users' goals. GENIE explores new ground by answering questions about skill acquisition. The use of schemata has also grown gradually more sophisticated. TEXT chose one of four schemata to produce text. Tailor was able to gracefully intermix two schemata. GENIE currently uses a mixture of four schemata in a limited way. Section 6 describes plans to generalize these for more flexible mixing.

GENIE extends work by McKeown and Paris further. McKeown suggested, and Paris showed, how a User Model affects the choice of content in the schemata. Work on GENIE shows how the use of the current context is equally important. McKeown identified the need for selecting a *knowledge pool* of potentially relevant information, however the mechanisms used in TEXT and later in Tailor were fairly simple. Work on GENIE suggests that, at least in procedural environments, extensive analysis within a separate component enhances the selection of the knowledge pool.

5. An Example

One scenario will be presented here in order to give an example of how GECIE produces text. It shows how the User Model affects the choice of strategy. Other examples can be found in [Wolz & Kaiser 88].

In this example the best plan (B) chosen by the Plan Analyst is described by the Explainer through three different tutoring strategies depending on what the user knows. 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 that included the user. Figure 5-1 is a graphic representation of the Expert Model required to answer this question. 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. Three different tutoring

strategies are invoked depending on what is in the User Model. In all three cases we assume that the user has a plan for *compose.message*.

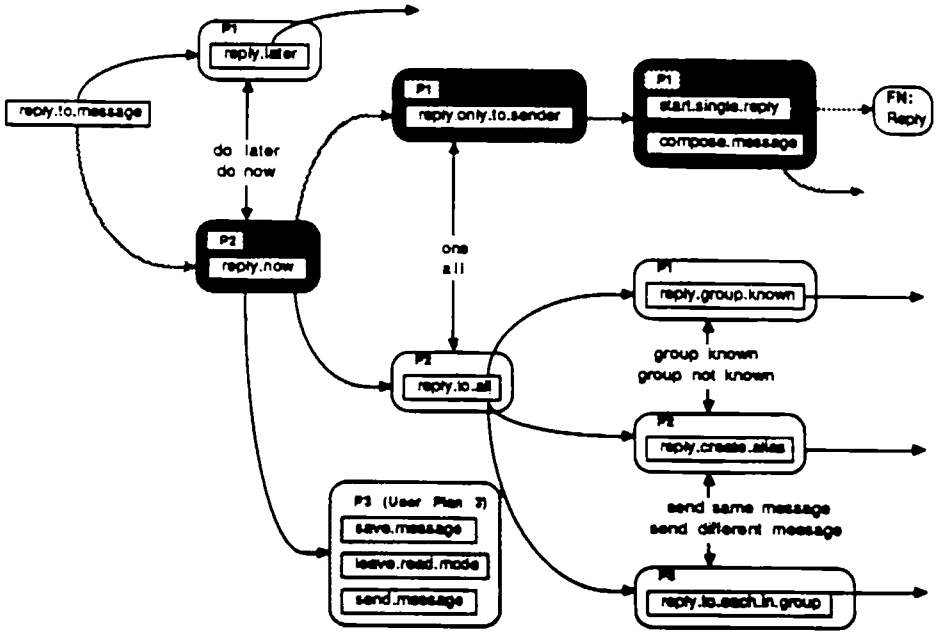

Figure 5-1: GENIE's Expert Model Knowledge of Replying to a Message

In the first case the User Model contains no plan for the goal, that is the Plan Analyst was unable to find a plan like B. The Explainer chooses to *Introduce Responsively* because no plan U exists in the User Model for the goal. Figure 5-2 shows both a process trace of the directives that are assembled for producing text and the template based text that is produced.

In the second case the User Model contains a plan U that is identical to B, that is, GENIE believes the user has replied to messages correctly in the past. The Explainer chooses to *Remind Responsively* because U exists and is equal to B. Figure 5-3 presents a portion of the process trace.

In the third case, 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". The Explainer chooses to *Clarify Responsively* because the user has a valid plan for the goal, but

```
Introducing:  reply.to.message
  STATEGOAL:  reply.to.message
    SUMMARIZE:  reply.to.message.P2
      Introducing:  reply.now
        STATEGOAL:  reply.now
        SUMMARIZE:  reply.now.P1
          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 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, you must indicate you wish to reply
and compose a message.  You can indicate you wish to reply by using
the command 'Reply'.

The syntax is: Reply

To use this command you must be in read mode. The
command removes you from read mode, makes the addressee of your
message the current sender and temporarily puts you in write mode.

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

Figure 5-2: *Reply to a message* When User Model Contains No Plan for the Goal

```
Reminding: reply.to.message
  STATEGOAL:  reply.to.message
  SUMMARIZE: P2
  EXAMPLE: (FN: Reply, Compose.message)
```

Figure 5-3: *Reply to a message* When User Model Contains a Plan for the Goal

that plan isn't the best. Figure 5-4 presents a portion of the process trace.

```
 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)
      Introduce:  Reply
```

Figure 5-4: *Reply to a message* When User Model Does Not Contain Best Plan

6. Proposed Extensions to the Prototype and Limitations of the Work

The demonstration system GECIE is by no means as sensitive as a skilled human consultant. At the present time, it cannot handle certain aspects of context, one cannot ask questions in any natural way, the knowledge bases must be updated by hand, and the answer that is generated could stand stylistic improvement. Work has begun on GENIE, in order to build a more robust system. This section will discuss issues that pertain to this thesis work first, and then those that fall outside it, and could form PhD thesis topics in themselves.

6.1. Extensions to be Included in the Thesis

Figure 6-1 shows the framework of question answering as it is viewed in this work. To answer a question, GENIE is given an utterance in English text. An Understander parses the text using an Augmented Transition Network (ATN) [Woods 73], takes the resulting lexical form and identifies a goal, plan or function in the utterance. It also maps lexical forms to the user's assumption and expectations in asking the question, and identifies the question intent. This is passed to the Plan Analyst that produces a goal, a best plan, a User Model plan, or when a User Model plan cannot be produced, any knowledge in the User Model pertaining to the best and stated plans. This information in turn is passed to the Explainer which constructs a set of directives for the text based on the tutoring strategies. The directives are given over to a Functional Unification Grammar (FUG) [Kay 79; McKeown & Paris 87] which produces English text.

The focus thus far has been on the Plan Analyst and Explainer components since these extend theories of deep level explanation generation. No attempt will be made to augment

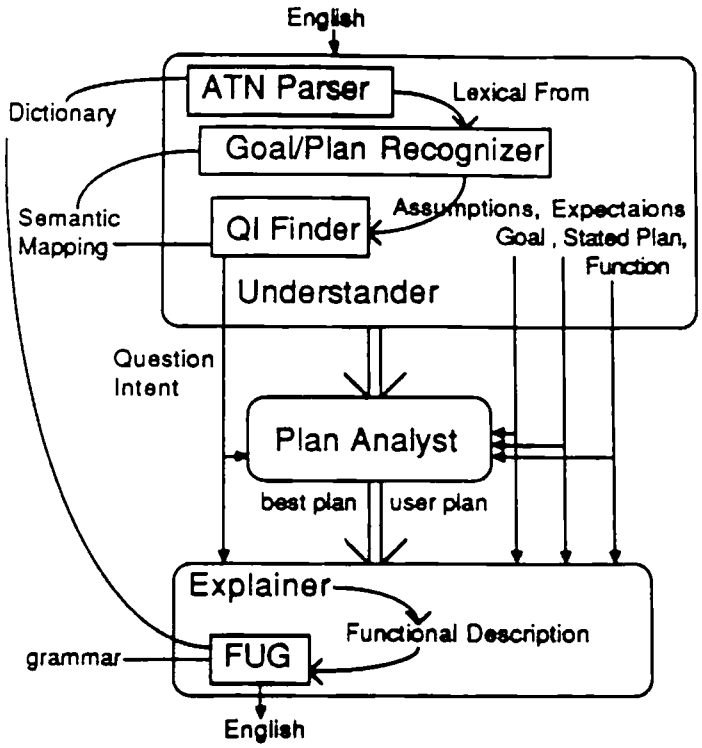

Figure 6-1: Framework For Question Answering

theory on the use of either ATN parsers or FUGs. Although work has begun on an Understander for demonstration purposes, its design is not expected to shed much light on the complex problem of natural language understanding. At the present time, it is expected that a small sample set of questions will be able to be parsed and mapped to goals, plans and functions in the Expert Model. Implementation has begun on using a simple ATN for parsing the sentence types in Figure 3-1 into lexical forms. Preliminary analysis of the grammar indicates that mapping lexical forms to the categories of Figure 3-1 will be straightforward given our restricted question set. Implementation has also begun on the Question Intent Finder, which seems to reduce to a simple set of cases. The remaining problem is to develop efficient mechanisms for mapping semantic categories to representations in the Expert Model. For example the phrase "answer a message" must be mapped to the goal *reply.to.message*.

At the other end of question answering, GECIE relies on textual templates to produce actual text. We have discovered that the prose generated is stylistically stilted, referentially awkward, often redundant and therefore inadequate. We are currently exploring the use of a functional unification grammar that should allow us to produce more graceful text.

The major emphasis for the thesis will focus on two intertwined issues: making the Plan Analyst responsive to a wider context and more complex goal/plan representation, and mixing strategies in the Explainer in a less rigid manner. Both of these problems require a better understanding of the representations of state in the Current Context, and the representations of semantic relation links in the Expert Model. At the present time, both representations are rather ad hoc and yet wield considerable influence. It seems likely that a systematic analysis (which has been started) will reveal some sort of taxonomy in both which could be exploited for more efficient search in the Plan Analyst and more flexible choice of strategies in the Explainer. Furthermore, work is about to begin on how to represent a query history in order to take into account not only what the user has done in the past, but what the user has asked. For example, if the user asks the same question twice in a row, it is likely the first answer was inappropriate.

At the present time the simplifying assumption is made that the user either knows or does not know about things in the domain. By definition, if something exists in the User Model, even if it is wrong, then it is known. If it does not exist, then it is not known. In practice this assumption is inadequate first because knowledge is not binary, and secondly because tutoring can occur across a broader continuum. Rather than fall for the seduction of simple categories of levels of mastery, we would like to evaluate and record knowledge in a less quantified manner. This in turn will affect how GENIE's heuristics and strategies must be changed.

While an improved representation may aide the Plan Analyst, there are also plans to make web traversal more efficient and robust. Issues under consideration include: using information from the Current Context and User Model to help guide search, expanding the definition of a match between goals and plans based on the Current Context, supporting non-sequential plans, and introducing general problem solving when a plan is not explicitly encoded in the knowledge base. Creating more flexibility in the Explainer is also of interest at the present time. In particular the following issues will be addressed: How do strategies combine, conflict and interleave; what aspects of context (what preconditions) affect the firing of steps within a strategy; what options are possible. For example, the current *Remind* strategy uses both syntax

and an example. Under what conditions might just one of these be appropriate?

6.2. Limitations of the Current Theory

Even with the enhancements just described, GENIE, like most intelligent systems falls far short of human ability. The problems to be described here deserve mentioning, but solving them does not contribute to the theory developed in this thesis.

One major goal of this work is to answer questions within the current context. At the present time the context is provided in symbolic form as input to GENIE. The assumption is that it can be derived by a subsystem that observes users' actions. In order to answer "what if" questions where the user presents a hypothetical context, a different subsystem would be required that allows users to construct a context in some friendly way such as natural language. This is an issue of understanding rather than generation. Similarly, in human discourse a question is rarely answered fully through a single utterance. Often a user's initial question spawns other questions that may be combinations of question types and require a mixture of answering strategies. Here too, a different subsystem, that monitors a discourse history, would contribute to the construction of the current context. Although there are plans to construct the requisite *representation* for the history, there are no plans to build the mechanisms to construct it. Such mechanisms belong more appropriately in systems such as Marvel [Kaiser *et al.* 88].

The final limitation concerns failure mechanisms. For example, the system may not possess vocabulary, may not associate vocabulary with goals, or may not have particular plans attached to goals. Each of these failures requires a different sort of discourse with the user to clarify the problem, and update the appropriate knowledge representation. Furthermore, all possible goals and plans for those goals cannot be known when a computing environment is set loose on users. If the environment is extendible, even the relationships between functions cannot be fixed. The problem of extending, updating and modifying the knowledge bases are viewed as problems in knowledge acquisition and learning and therefore not within the immediate scope of this research. Work on machine learning by Lebowitz [Lebowitz 86] has influenced the design of the Expert and User Model, and we are confident that both can be updated automatically in the future.

7. Validation

Established methodology from Computer Science is not completely appropriate for validating a thesis involving man/machine interfaces. Similarly, no claim can be made that people will learn "more" or "better" with GENIE. Proving such a claim may indeed be impossible given the number of variables introduced that are not the focus of this research. For example, the interface for forming questions may completely thwart the answering process. Instead, this thesis must be able to justify why design choices were made, that is, what principles were followed. It must also be able to show that a fully robust system is feasible in theory if not in practice in the immediate future. Three approaches to validation will be discussed here. They are analysis of current materials, a formal study of human preferences, and a complexity analysis of a theoretically robust system.

Section 4 described how the strategies can be found in typical textual support materials. A systematic analysis of texts will be included in the dissertation, both to show how GENIE models text, but also where GENIE surpasses print media.

The focus of this work is on the content and quality of the text produced. Consequently, the evaluation of the system should not be based on the complete question answering cycle, but on whether the text generated is preferred to currently available material. Therefore, we plan to conduct an experiment with groups of users from different backgrounds, for example, Columbia Computer Science Undergraduates, Computer Science Department "Wizards", Long Island University Education Majors, and Academic Administrative Staff. A number of scenarios will be described to subjects who will be asked to rate the relevancy of a variety of texts drawn from manuals and output from GENIE.

The final method of evaluation is concerned with whether a system like GENIE can run in real time. A theoretical analysis will be done to determine how increasing the size of the knowledge bases will affect the Plan Analyst and the Explainer. It is often assumed that most search algorithms have exponential time complexity. If this proves to be the case for GENIE, then at a minimum, the thesis must include mechanisms for bailing out in real time, that is, punting gracefully if search takes too long.

8. Summary and Agenda for Completion

It is impossible to put temporal objectives on the development of new ideas. It is possible however, to create a time line for implementation. Therefore, what follows is a schedule merely for implementation. It is hoped that the theoretical issues described in section 6.1 will fall naturally out of work on the implementation.

- Understander complete by 12/15/88 (for demonstration purposes) — Lourdes Andre, David Robinowitz
- Extensions to knowledge bases by 12/15/88 (to develop theory on representation of current context, relational links and discourse history, user mastery of plans) — Michael Tanenblatt, and UW
- Extensions to Plan Analyst 12/15/88 (to develop theory on how to exploit knowledge bases better to make goal/plan choices) — DR & UW
- Conversion to FUG 1/31/89 (for demonstration purposes) — UW
- Strategy Enhancement 2/15/89 (to make answers more flexible) — UW
- Really make it work 3/15/89 (to get ready for testing) — UW and ??
- Systematically evaluate texts 1/15/89 — UW
- Test and evaluate use with people 3/15/88 - 4/15/89 — UW
- Do time complexity analysis 5/15/89 — UW
- Write, write, write 9/89
- Defend (!?)

To summarize, this thesis work proposes a unified approach to consulting as question answering in interactive programming environments. It posits three integrated components, namely a Functional Representation, Analytic Expertise, and Explanatory Expertise.

A functional representation is needed in order to represent the requisite knowledge. A declarative structure of goals, with alternative plans and semantic relationships between plans is proposed for both the Expert and User Models. The intent is to facilitate generation and reduce the amount of brute force problem solving required.

A separate component that models analytic expertise is needed in order to choose the most relevant information. Mechanisms that use computational goals, the current context, and abandon stereotypes of functionality and user expertise are proposed. The intent here is to allow choices among plans to more precisely reflect users' needs.

Finally, a separate component that models explanatory expertise is needed in order to choose

the form in which to present the information. Use of an explicit mixture of tutoring strategies is proposed that both respond to the query and enrich. The intent here is to present information succinctly, while exploiting opportunities to present new information.

Acknowledgments

Thanks are in order to Kathy McKeown, Gail Kaiser, Steve Feiner and Elaine Kant for reading and discussing earlier versions of this paper. The work presented in this paper also benefited from many valuable discussions with Mike Lebowitz and Cecile Paris.

References

[Allen and Perrault 80]
Allen, J. F. and Perrault, C. R.
Analyzing Intention in Utterances.
Artificial Intelligence 15(1): pages 143-178, 1980.

[Appelt 85] Appelt, D. E.
Planning Natural Language Utterances.
Cambridge University Press, Cambridge, England, 1985.

[Borenstein 85] Borenstein, N.S.
The design and evaluation of on-line help systems.
PhD thesis, Carnegie Mellon University, April, 1985.

[Brown & Burton 78]
Brown, J.S. and R.R. Burton.
Diagnostic Models for Procedural Bugs in Mathematics.
Cognitive Science 2:155-192, June, 1978.

[Brown & VanLehn 80]
Brown, J.S. and K. VanLehn.
Repair theory: A Generative theory of bugs in procedural skills.
Cognitive Science 4:379-415, 1980.

[Chin 86] Chin, D.N.
User modeling in UC, the UNIX Consultant.
In *Proceedings of the CHI'86 Conference*, pages 13-17 . Boston, MA, April, 1986.

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

[Clancey 82] Clancey, W.J.
Tutoring rules for guiding a case method dialogue.
Intelligent Tutoring Systems.
Academic Press, London, 1982, pages 201-225.

[Fikes and Nilsson 71]
Fikes, R. E. and Nilsson, H. J.
STRIPS: A New Approach to the Application of Theorem Proving to Problem Solving.
Artificial Intelligence 2: pages 189-205, 1971.

[Finin 83] Finin, T.
Providing help and advice in task oriented system.
In *Proceedings of the Eighth International Joint Conference on Artificial Intelligence*, pages 176-178. Karlsruhe, West Germany, 1983.

[Genesereth 82] Genesereth, M.R.
The role of plans.
Intelligent Tutoring Systems.
Academic Press, London, 1982, pages 137-155.

[Goldberg 87] Goldberg, A.
Programmer as reader.
IEEE Software 9:62-70, September, 1987.

[Goldstein 82] Goldstein, I.R.
A genetic graph model for tutoring.
Intelligent Tutoring Systems.
Academic Press, London, 1982, pages .

[Habermann & Notkin 86]
Habermann, A.N. and D. Notkin.
Gandalf: Software development environments.
IEEE Transactions on Software Engineering SE-12(12):1117-1127, December, 1986.

[Johnson 86] Johnson, W.L.
Intention-based diagnosis of novice programming errors.
Morgan Kaufmann Inc., Los Altos, CA, 1986.

[Joshi, Webber & Weischedel 84]
Joshi A., B. Webber and R. Weischedel.
Living up to expectations: Computing expert responses.
In *Proceedings of AAAI-84*, pages 169 - 175. American Association of Artificial Intelligence, 1984.

[Kaiser *et al.* 87] Kaiser G. E., S. M. Kaplan and J. Micallef.
Multiuser, istributed language-lased environments.
IEEE Software 11:58-67, November, 1987.

[Kaiser *et al.* 88] Kaiser G. E., P. H. Feiler and S. S. Popovich.
Intelligent Assistance for Software Development and Maintenance.
IEEE Software :40-49, May, 1988.

[Kant 88] Kant, E.
Interactive problem solving using task configuration and control.
IEEE Expert 3(4), Winter, 1988.

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

[Kukich 85] Kukich, K.
Explanation structures in XSEL.
In *Proceedings of the 23rd Annual Meeting of the Association for Computational Linguistics.* Chicago, IL, 1985.

[Lebowitz 86] Lebowitz, M.
An experiment in intelligent information systems: RESEARCHER.
Intelligent Library and Information Systems.
Ellis Horwood, London, 1986.

[Magers 83] Magers, C. S.
An experiemntal evaluation of On-line HELP for non-programmers.
In *CHI'83 Proceedings*, pages 277-281. 1983.

[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.

[Papert 80] Papert, S.
Mindstorms: Children, computers and powerful ideas.
Basic Books, 1980.

[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.

[Pollack 86] Pollack, M.
Inferring domain plans in question-answering.
PhD thesis, Moore School, University of Pennsylvania, May, 1986.

[Quilici *et al.* 85] Quilici, A.E., G. Dyer and M. Flowers.
Understanding and advice giving in AQUA.
Technical Report, UCLA Artificial Intelligence Laboratory, Los Angeles, CA, 1985.

[Reiss 87] Reiss.
Working in the garden environment for conceptual programming.
IEEE Software 11:16-26, November, 1987.

[Sacerdoti 77] Sacerdoti, E.
A Structure for Plans and Behavior.
American Elsevier North-Holland, New York, 1977.

[Schank 82] Schank, R. C.
Dynamic Memory: a Theory of Reminding and Learning in Computers and People.
Cambridge University Press, New York, 1982.

[Selker 88] Selker. T.
Cognitive Adaptive Computer Help - A Research Overview.
Technical Report, T.J. Watson Research Center, IBM, T.J. Watson Research Center, Yorktown Heights, N.Y., 1988.

[Sleeman & Smith 81]
Sleeman, D.H. and M.J. Smith.
Modelling student's problem solving.
AI Journal :171-187, 1981.

[Stallman 81] Stallman, R.M.
Emacs The extensible, customizable, self-documenting display editor.
In *SIGPLAN SIGOA Symposium on Text Manipulation*, pages 147-156. June, 1981.

[Swartout 83] Swartout, W. R.
xplain: a system for creating and explaining expert consulting systems.
Artificial Intelligence :285-325, September 1983, 1983.

[Walker *et al.* 87] Walker, J. H., D. A. Moon, D. L. Weinreb & M. McMahon.
The symbolics genera programming environment.
IEEE Software 20:36-87, November, 1987.

[Waters 86] Waters, R.C.
KBEmacs: Where's the AI?
The AI Magazine 7(1):47-56, Spring, 1986.

[Wilensky *et al.* 84]
Wilensky, R., Y. Arens, and D. Chin.
Talking to Unix in english:An overview of UC.
Communications of the ACM 27(6):574-593, June, 1984.

[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.

[Woods 73] Woods, W.
An Experimental Parsing System for Transition Network Grammars.
In Rustin, R. (editor), *Natural Language Processing*. Algorithmics Press, New York, 1973.