# A Computational Model Of Recognizing and Revising Inappropriate Advice[1]

David Pautler and Alex Quilici
University of Hawaii at Manoa
Department of Electrical Engineering
2540 Dole Street, Holmes 483
Honolulu, HI 96822

**Abstract**

Advice-giving is improved by understanding and responding to user feedback. Previous models of this task treat user responses as misunderstandings or misconceptions and focus on generating alternate or corrective explanations. By and large, however, these models do not consider the possibility that the system's advice is inappropriate and may need revision during the course of an on-going dialog. This paper presents a model of the process of revising plan-oriented advice in response to user feedback. Our focus is on mechanisms for evaluating planning alternatives, for determining when to revise advice, and for dynamically generating explanations of these revisions.

## Introduction

Advising can be viewed as the process of providing suggested courses of action. An advice-giving dialog is one in which the advisor gives advice, the user provides feedback about that advice, and the advisor addresses that feedback.

Research into constructing computational models of advising has focused on the problem of recognizing and responding to user misunderstandings or misconceptions. Addressing misunderstandings has been modeled as a reactive planning process, where the advisor has a set of plans for getting across a particular piece of advice (e.g., trying an example or providing an analogy) and executes alternate plans in response to recognized understanding failures [1, 2, 9, 10]. Addressing misconceptions has been modelled as a process of inferring incorrect user beliefs and planning appropriate explanations for why these beliefs are incorrect [4, 5, 6, 8, 13]. In both cases, however, the mistake is assumed to lie with the user, and while the advisor's presentation changes—either by choosing different presentation plans or by extending the presentation to encompass explanations for incorrect beliefs—the advisor's suggested courses of action remain static.

Other earlier work [11, 12] extended these efforts to acknowledge that a user rejects advice not because of a misunderstanding or misconception but because of flaws in the advice itself, such as failing to take into account all of the user's goals. However, this work focused on the problem of inferring how a user response relates to previously stated advice (i.e., relating a stated user belief to the advisor's specific suggestions or explanations for those suggestions), and did not provide a model of the process of actually revising advice or of generating appropriate explanations for those revisions.

This paper extends earlier work to address these problems, concentrating on plan-oriented advice. Specifically, we provide a detailed model of the process by which an advisor can revise a plan to take into account user feedback and then generate an explanation for any revision.

[1]This work was sponsored by NSF research initiation award #IRI-9309795.

## The Problem

Our model of the advisor's behavior assumes the following scenario. The user has initially provided the advisor with an explicit request to know the best plan for a user goal $G$, and the advisor's task is to provide that plan, along with an explanation for why the plan is best. Furthermore, the advisor believes that $G$ can be achieved by any one of a set of plans, $P_1 \ldots P_n$, and that the user possesses some additional goals (and weightings of their relative importance) that influence which $P_i$ is actually the best plan for $G$.

Initially, the advisor's problem is to determine which $P_i$ to suggest to the user and to provide an explanation for why it is the best. However, the advisor's beliefs about which goals the user possesses (and their relative weightings) may well be incorrect, leading to a suggested plan that is not acceptable to the user. As a result, the user provides feedback to the advisor, and the advisor must re-evaluate candidate plans in light of this newly acquired information. In particular, the user's feedback is a description of a user goal and/or weighting, or alternatively, it is one or more beliefs from which a user goal and/or weightings can be inferred [12].

Figure 1 is an example advice-giving dialog that fits this scenario. In this dialog:

- The user asks the advisor for the best way to remove a file.
- The advisor believes that removing a file can be achieved by any one of the plans represented by the commands "rm file", "rm -i file", "mv file /tmp", "mv file /dev/floppy".
- The advisor believes that which removal plan is best depends on possible user goals such as preventing accidental file removal, allowing file recovery, minimizing removal time, and avoiding questions, as well as the relative importance the user gives to each (e.g., whether preventing accidental file removal is more important than minimizing removal time).
- The advisor believes that this particular user has the additional goal of preventing accidental file removal and allowing file recovery, with preventing accidental file removal being slightly preferred.
- The user's responses explicitly indicate the additional goals of of avoiding questions and ensuring the file no longer exists on the machine.

To participate in this dialog, the advisor must able to determine the best plan for the user, accept user feedback (in the form of additional goals and changed weightings), revise this plan to take the feedback into account, and provide a suitable explanation.

## The Advisor Architecture

Our advisor model consists of a stereotypical user model, a dialog model, a plan library, a plan evaluator, a response generator, and a feedback processor.

*User*: What's the best way to remove a file?

*Advisor*: Use "rm -i file". It asks you to confirm before removing the file.

*User*: I don't want to answer questions when I remove a file.

*Advisor*: An alternative is to use "mv /tmp file". That way you can copy back any file you accidentally remove.

*User*: I want the file off the machine.

*Advisor*: "mv file /dev/floppy" will also allow you to recover your file. It removes your file onto a floppy disk.

Figure 1: An Example Dialog

- The stereotypical user model contains a "standard" set of UNIX user beliefs [3]. This includes a standard set of goals and their relative importance to the user. For example, with file removal, the standard user goals are to prevent accidental file removal and to allow for file recovery, with the former preferred slightly over the latter. These goals are used in determining the advisor's initial suggestion of a plan.
- The dialog model contains the set of stated user and advisor beliefs (including user goals), along with their inferred connections to the dialog [12]. For example, one inferred connection is that a user not wanting to answer questions is addressing an unstated effect of the advisor's suggested plan of using "rm -i". These beliefs are used in evaluating plans (overriding the advisor's defaults in the stereotypical user model) and in generating appropriate explanations (allowing the advisor to generate explanations that are customized to what's been previously discussed in the dialog).
- The plan library contains descriptions of available plans, along with a list of each plan's enablements (conditions that must be true before the plan can be executed), effects (conditions that are true after the plan is executed), and execution properties (conditions that are true about the plan's execution). For example, the plan of "mv file /tmp" has the effect that the file no longer exists in the original directory and that the file exists in "/tmp", it has an enablement that "/tmp" is writeable, and it has a property that it takes a time on the order of seconds to execute the command. The effects can be both direct and indirect, with the indirect effects linked to direct effects through causal chains (e.g., "rm -i" has the direct effect of asking a question, which leads to the user answering a question, which can prevent accidental file removal). This information is used both in evaluating plans (supporting plan comparisons) and in generating appropriate explanations (supporting causal rationales for plan choices)
- The plan evaluator determines the best plan for the user, taking into account the set of possible plans that achieve the user's stated goal, the user beliefs in the dialog model, and the information about these plans found in the plan library. This determination is done by a weighted, numeric computation, rather than symbolically, and results in a ranking of plans by scores and an enumeration of the relative influence of the various factors that determined this ranking.
- The response generator forms a symbolic, causal explanation for why a given plan is most appropriate. This explanation takes advantage of the prior dialog context (from the dialog model), the advisor's causal knowledge of the relevant plan alternatives (from the plan library), and the relative influence of the different factors that determined the final plan ranking.
- The feedback recognizer determines the connection between a stated user goal and previously-given advice, infers the other causally-related user beliefs and goals, and determines which factors in the advisor's earlier computation of the best plan need to be revised. This component has been described in detail elsewhere [12].

The focus of this paper is on plan evaluation and explanation generation.

## Plan Evaluation

Our model evaluates a plan in the context of a set of effect goals and a set of property goals. An effect goal is a desired condition of the world after executing a plan. One example is the goal of allowing file recovery, a condition users want to hold true after executing a file removal command. A property goal is a desired condition while executing a plan. One example is the goal of fast execution of the remove command, as it is a constraint on the execution of the plan.

Our model assumes the user has a single explicitly provided user effect goal, $G$ (e.g., remove a file), and a set of unstated effect and property goals. Given $G$, the advisor locates an initial set of plans, $P_1$ through $P_n$, by finding all planning operators that have the user's stated effect goal as one of their effects, and eliminates any from the set of possible alternatives that have currently unachievable enablements.

The process of determining the best $P_i$ then involves analyzing each $P_i$ against user goals, either stated or inferred, in terms of three factors:

1. *The relative desirability of $P_i$'s side effects.* All else being equal, a plan that not only achieves the user's main goal but also an additional user goal, is preferred to a plan that only achieves the user's main goal.
2. *The relative undesirability of $P_i$ failing to have all desired user properties.* All else being equal, a plan whose execution properties are similar to what the user desires is preferred to one whose properties are far from what the user desires.
3. *The relative undesirability of having to achieve $P_i$'s unachieved enablements.* All else being equal, a plan whose enablements are all satisfied is better than a plan with an unsatisfied enablement.

The key problem is: How do we create a computational model of plan evaluation that encompasses these factors? Our model is that the process requires computing a desirability (or undesirability) measure for each of these factors and then combining them.

### Computing Effect Desirability

*Effect desirability* captures the relationship between the plan and the user's effect goals (other than the main effect goal, which triggered this particular set of plans). It is defined as $(EGD \times EA)$, where $EGD$ represents *effect goal desirability*, $EA$ represents *goal achievement*.

$EGD$ is a vector that captures the relative desirabilities (weightings) of the possible user effect goals. This vector contains one entry per possible user goal, which represents the goal's strength. The entry corresponds to an integer between 10 and $-10$, where 10 represents the strongest goal, 0 represents apathy, and $-10$ represents the opposite being a goal. These weightings are initially formed from the stereotypical user model and augmented, as the dialog progresses, by stated and inferred user goals.

$EA$ is a matrix that captures the relationship of each candidate plan to the set of possible user effect goals, where there is one column per plan, one row per goal, and the entries are values that capture the relationship between the plan and the

goal. These values also correspond to an integer between 10 and −10, where 10 represents achieving the goal, 0 represents not being relevant to the goal, −10 represents thwarting the goal, A value between 0 and 10 means the plan only partially or sometimes achieves the goal (e.g., "rm -i" only prevents accidental file removal if the user answers "n" for files that actually shouldn't be removed). This matrix captures only the final relationship between plans and goals, not the causal relationship of how these plans achieve or thwart the goals.

The result of multiplying $EGD \times EA$ is a vector that captures the relative ability of these plans to satisfy the user's effect-related goals. The higher the score for an individual plan, the closer it comes to satisfying the users goals.

As an example, the stereotypical starting values of $EGD$ for our example dialog is [10, 8, 0, 0], with the entries representing the desirability of preventing accidental file removal, allowing file recovery, avoiding questions, and ensuring the file is actually removed from the host. That is, users have preventing accidental removal as a strong goal, with allowing file recovery a close second, and users are apathetic about avoiding questions or ensuring the file is really removed). $EA$'s initial values are determined based on causal relationships and are fixed across all users, as shown below:

| | use rm | use rm -i | use mv /tmp | use mv floppy |
|---|---|---|---|---|
| prevent accidental file removal | −10 | 9 | 0 | 0 |
| allow recovery of an accidentally removed file | −10 | −1 | 9 | 10 |
| avoid questions after command | 10 | −10 | 10 | 10 |
| ensure file not on host after command | 10 | 10 | −10 | 10 |

The result, for our example file removal plans, is [−180, 82, 72, 80] ("rm -i" is ranked the highest, "mv floppy" is ranked a close second, "mv /tmp" is ranked third, and "rm" is ranked a distant fourth). This low ranking for "rm" is a direct result of its thwarting both goals the user is assumed to have.

This computation of effect desirability ranks plans in terms of effect goals, without considering property goals or enablements. An advisor using only this information to evaluate plans could not address user responses dealing with properties or enablements, such as "But I want a faster file-removal command" or "But I don't want to have to find a floppy disk".

## Computing Property Undesirability

Property desirability captures how close the properties of each plan's execution come to meeting the user's desired constraints on that execution. It is computed and defined in a way that's similar to effect desirability. In particular, it is defined as $PGD \times PD$, where $PGD$ represents *property goal desirability* and $PD$ represents *property distance*.

$PGD$ is a vector that captures the relative desirabilities (weightings) of the possible user property goals. Like $EGD$, it contains one entry per possible user goal that represents the strength of the goal, with a value between 10 and −10, and these weights are initially formed from the stereotypical user model.

$PD$ is a matrix that captures the relationship of each candidate plan to the set of possible property goals, where there is one column per plan and one row per property. With properties, however, we are concerned with measuring distance, so the entries in the matrix represent how close the plan's execution comes to having the desired property. That is, because there is no notion of thwarting a property goal, the values in this matrix are non-positive, between 0 and −10, with 0 representing a plan that exactly has the particular property and the negative numbers representing how far the plan is from having that property.

As an example, for file-removal there are two common user-desired properties: fast execution and command simplicity. Our stereotypical starting values of $EPD$ for these goals is [4, 6] (command simplicity is slightly preferred to fast execution). $PA$'s initial values are determined based on causal relationships and are fixed across all users, as shown below:

| | use rm | use rm -i | use mv /tmp | use mv floppy |
|---|---|---|---|---|
| executes quickly | 0 | -4 | -4 | -8 |
| command simplicity | 0 | -2 | -4 | -6 |

That is, "rm" has both of these properties, but the others tend to execute more slowly and are not as simple (since they all require command-line arguments).

The overall result for property desirability for our example (computing $EPD \times PD$) is [0, −28, −40, −68] ("rm" is ranked the highest, "rm -i" is ranked a distant second, "mv /tmp" is ranked third, and "mv file /dev/floppy" is ranked the lowest). Here, "rm" has a high ranking is a direct result of its having exactly the properties the user desires in a removal-command.

## Computing Enablement Undesirability

Enablement undesirability captures the negative consequences to the user of having to achieve enabling conditions. Our model computes enablement undesirability as $EU \times RE$, where $EU$ is a *enablement undesirability* vector. and $RE$ is a *required enablements* matrix.

$EU$ is a vector that captures the relative undesirability of each of the known enablements for the set of plans under consideration. It is intended to model the user's perceived attitude toward having to achieve the enablement. Since any enablement is somewhat undesirable, the values range between −10 and 0, with a −10 indicating a highly undesirable enablement condition and a 0 indicating apathy toward the enablement.

$RE$ captures which plans are associated with which enablements. Its values range from 0 to 10, with a 10 indicating an unsatisfied enablement, a 0 indicating that it's not an enablement or that it's a currently satisfied enablement, and an in-between value indicating a partially unsatisfied enablement. The idea is that even if a plan has an enablement that's a lot of work to achieve, that work is irrelevant if it's already achieved.

As an example, for the file-removal commands, the possible plan-specific enablements are space in "/tmp" and locating a floppy disk (directory write permission is an enablement of *any* plan for removing a file, so it can be ignored). A stereotypical $EU$ is [−3, −6], which indicates the user minds trying to clean up "/tmp", but minds much more having to locate a floppy.

| | use rm | use rm -i | use mv /tmp | use mv floppy |
|---|---|---|---|---|
| space in /tmp | 0 | 0 | 10 | 0 |
| available floppy | 0 | 0 | 0 | 10 |

By multiplying $EU$ and $RE$, we come up with an undesirability cost vector for these plans of [0, 0, −30, −60].

## Computing An Overall Ranking

Essentially, effect desirability captures the progress the user can make toward achieving various user goals, while property undesirability and enablement undesirability capture the price the user has to pay—just how far the plan is from ideal. Our model combines these measurements to achieve an overall desirability for the plan by normalizing each of these measurements and then computing a weighted sum,

$$score_i = w_1 \times n_{ed} + w_2 \times n_{pu} + w_3 \times n_{eu}$$

where $score_i$ is $P_i$'s score, $w_i$ represents a weight, $n_{ed}$ is the normalized effect desirability, $n_{pu}$ is the normalized plan undesirability, and $n_{eu}$ is the normalized enablement undesirability.

We normalize effect desirability and property undesirability by dividing the respective scores by the number of effect and property goals, respectively. For effect goals, this results in a score between $-100$ and 100 (where 100 is a plan that achieves all the goals; a score of 0 is a plan that is essentially side-effect neutral, and $-100$ is a plan that thwarts all the goals). For property goals, we do the same, but this results in a score between $-100$ and 0, where 0 is a plan that is ideal in terms of its properties, and $-100$ is a plan that's as far as possible from ideal. We normalize enabling conditions by dividing by the number of enablement conditions, which leads to a score between $-100$ and 0, where 0 is a plan with no negative attitude toward achieving its enablements and $-100$ is a plan with maximal negative attitude.

The weights, $w_i$, represent the user's overall weightings of each of these types of factors. If each $w_i$ is 1, the user weights each factor evenly (which, in turn, implies that the same desirability scale has been used for effect goals, property goals, and attitude toward enablements). By providing these weightings, the model allows each of these factors to be placed on their own relative, rather than absolute, scale. In addition, the weightings can be used to capture general attitudes of the user, such as a user for whom it is much more important to achieve as many effect goals as possible than to have a plan that's a lot of work to execute.

We have used stereotypical starting weightings of 1, .5, and .5 for our plan evaluations, giving even weight to effect desirability on one hand and the combined undesirability of plan properties and plan enablements on the other hand. The result for our example is approximately $[-45, 14, 3, -11]$, which suggests for the standard user "rm -i" is the best, "mv /tmp" is a distant second, and both "rm" and "mv floppy" leave the user worse off when their side effects are considered. However, exactly what the various stereotypical weightings should be is an open question that needs to be experimentally addressed.

## Revising Goal Weightings

Our model of plan evaluation is heavily dependent on the relative desirabilities of effect goals, property goals, and plan enablements. A key question is how to the advisor can modify these weightings during the course of an on-going dialog.

Our model currently revises weights differently depending on several different classes of goal-based user responses: those that provide unqualified goals (e.g., "I want X" or "I do not want X"), those that provide qualified goals (e.g., "I really want X" or "I very much do not want X"), and those that provide goal preferences (e.g., "I prefer X to Y"). The assumption is that the user's underlying purpose in making these statements is to inform the advisor that the current plan is insufficient because it doesn't appropriately take the newly stated goal into account.

For unqualified goals, the advisor simply changes the goal weighting to the maximum value ($-10$ or 10, depending on whether it's a positive or negative goal), rather than trying to infer an appropriate value. This "maximal change" approach is attractive because in re-evaluating a plan, there are only two outcomes: the plan either remains the best plan or another plan becomes the best. If maximally revising the weighting does not result in a new "best" alternative, it suggests that the current alternative cannot be improved on in terms of this goal. On the other hand, if it does result in a new alternative, that alternative is provided as the best plan, and the user is given the opportunity to reduce this weighting in subsequent responses (through qualified goals or goal preferences).

In our example, when the user specifies that he doesn't want to answer questions, the goal of avoiding questions is automatically given a 10. This results in an effect goal desirability vector of $[10, 8, 10, 0]$, resulting in a new ranking for file removal plans in terms of goal effects, $[-80, -12, 172, 180]$. Having this new information makes "rm" a better alternative than before, and it makes "mv /dev/floppy" the best plan when considered solely in terms of goal effects. However, when the advisor considers property undesirability and enablement undesirability, "mv /tmp" becomes the best plan.

For qualified goals, the advisor changes the goal weighting by a either a fixed amount or a "delta", depending on the linguistic term used as a qualifier (e.g., "really" causes the goal weighting to be set to a high value, "prefer not" causes the goal weighting to be a medium value, and so on). The assumption here is that the user is aware the advisor considered the goal, but believes the advisor put an incorrect weighting on it.

Finally, for goal preferences, the advisor changes the goal weightings to reflect the stated preference. In a user statement such as "I prefer X to Y" causes X's weighting to be increased to a delta over Y (with the precise delta depending on any qualifiers).

As with our initial weightings and our relative weightings of effect goals, property goals, and enablements, exactly what the deltas should be for various linguistic terms and preferences is an open question.

## Providing An Explanation

Our model assumes that the advisor will respond to the user by either providing a new alternative (and an appropriate explanation), explaining that the previous alternative is the best (even taking into account the user's latest feedback), or providing an explanation that corrects a mistaken user belief [12]. Due to space considerations, we tackle only the first: explaining why a new alternative is the best.

Our model for providing an explanation relies on four advice-giving principles, derived from Grice's conversational maxims [7]. These are:

1. Don't waste the user's time by stating anything the user already knows.
2. Don't confuse the user by changing the dialog focus once it's on the new alternative.
3. Don't confuse the user by providing information that forms a case against this alternative.
4. Don't waste the user's time by telling the user something that can be inferred by an assumption of advisor cooperativeness.

We illustrate these principles by showing how the model applies them to determine the content for the response explaining why "mv /tmp" is now the best plan for the goal.

Goal Effects:
1a) rm -i does involve answering questions
1b) mv /tmp does not involve answering questions
2a) rm -i does prevent accidental file removal
2b) mv /tmp does not prevent accidental file removal
3a) rm -i does not allow recovery
3b) mv /tmp allows recovery

Enablement costs:
4a) rm -i does not require space in /tmp
4b) mv /tmp requires space in /tmp

Property Effects:
5) rm -i executes faster than mv /tmp
6) rm -i is a simpler than mv /tmp

Figure 2: The factors where "mv tmp" and "rm -i" differ.

From examining dialogs, it appears that advisors consistently use comparative explanations to explain why a new plan is a better choice than an existing alternative. Thus, the advisor's first task is to determine to which alternative the new plan should be compared. One approach is to compare the recommended plan to each of its alternatives; however, this can lead to a complex and lengthy response even in our simple example. Another approach is to compare the plan to the next best alternative. However, if the next best alternative has not yet been mentioned in the dialog, this suddenly changes the focus of the dialog, and has the potential to confuse the user. As a result, our model applies the principle of maintaining focus, and the advisor compares the new alternative with the most recently suggested alternative. For our example, this means comparing "mv /tmp" to "rm -i", the plan currently under discussion.

The advisor's next task is generate an explanation for why "mv /tmp" is better than "rm -i". This explanation involves a comparison of costs and benefits between the two plans, which have been computed as part of the advisor's calculation of the plan's desirability, For our example, Figure 2 lists the various costs and benefits in general order of most influential difference to least (items with the same number reflect the same underlying goal, such as answering questions). Because of the large number of influences, it is cumbersome to discuss all of these influences in a single response. As a result, the advisor must determine an appropriate subset of these influences in a given response.

One approach to determining which costs and benefits to discuss is to mention only the most salient differences between the plans, with salience defined in terms of the largest influence on the plan's overall score. However, this heuristic often fails, as many different factors may contribute relatively equally, or the most salient difference may be one the user is already aware of. As a result, our approach is to apply the advice-giving principles above to reduce the set of candidate costs and benefits.

In particular, the advisor first avoids confusing the user by eliminating from consideration any factor that argues against the advisor's new recommendation. For our example, the advisor ignores any factor that negatively influences whether "mv /tmp" is the best plan. This eliminates (2a) and (2b), which indicate a failure of "mv /tmp" to achieve a goal, (4a) and (4b), which indicate that it requires an additional enablement, and (5) and (6), which indicate that "rm -i" has some additional, desirable properties.

Next, the advisor avoids telling the user what the user already knows by eliminating any belief found in the dialog model. This eliminates (1a), which the user's previous response explicitly addressed. At this point, the remaining choices are (1b), which points out that mv "/tmp" does not involving answering questions, and (3a) and (3b), which point out that mv "/tmp" allows recovery and "rm -i" does not.

After narrowing down the set of factors to those that will not confuse the user and that the user doesn't already know, the advisor has an acceptable response.

*Advisor*: Use mv "/tmp". It does not require that you answer questions. It allows you to recover your file and "rm -i" does not.

However, the advisor then tries to make the response as brief as possible by repeatedly applying the cooperativeness principle, while ensuring that the focus principle is not violated. In particular, the advisor assumes that if the user mentions a plan flaw, $F$, the user expects that any new suggested plan will not have this $F$ (or that if it does, the advisor will mention it). It can therefore eliminate from consideration any response that suggests the new plan will not have a previously mentioned flaw. In this case, this eliminates (1b), since if the advisor simply states "use mv /tmp" in response to the user's desire not to answer questions, the user will infer that this plan does not involve answering questions—so there's no need for the advisor to explicitly mention it.

After this first application of the cooperativeness principle, the advisor is left with (3a) and (3b), which leads to a briefer response.

*Advisor*: Use mv "/tmp". It allows you to recover your file and "rm -i" does not.

However, by again applying the cooperativeness assumption, the advisor can narrow this response down to simply (3b). In particular, the advisor assumes that if it mentions that a new plan has an effect $E$, the user will infer that the old plan does not result in effect $E$. As a result, stating (3b) alone will imply (3a). While the advisor can also use the converse of this rule to reduce the response to (3a), which implies (3b), doing so violates the focus principle. That is, mentioning that "rm -i" doesn't support file recovery allows the user to infer that "mv /tmp" does support recovery; however, it shifts the focus from the new alternative to an old one.

## Future Work

There are several key areas of future work: verifying the behavioral predictions made by the model, experimenting with alternate weightings, extending and revising the model, and integrating the model with existing work in dialog response planning.

### Verifying The Model's Predictions

Our model makes several predictions about advisory dialogs that need to be verified experimentally. One prediction is that advisor explanations for why one plan is better than another will not necessarily involve the factor that is "largest" difference between the two candidate plans. A related prediction is that explanations that follow our model's advice-giving principles will be rated to be more cooperative than explanations based on the heuristic of choosing the "largest" difference between plan candidates. While our model is based on analysis of a variety of UNIX-related advice-giving dialogs, and it appears to model the behaviors in many of these dialogs, these experiments are necessary to evaluate the model's overall quality.

### Experimenting With Alternate Weightings

One drawback to our current approach lies in how we have formed our stereotypical weightings. For the UNIX domain,

the stereotypical values for users have been assigned by applying general principles to specific UNIX commands. We have treated file removal, for example, as an instance of the general notion of destroying something, and we applied the principle that users want to ensure they don't destroy anything accidentally, and if they do, they want a way to get it back—but preferring not to destroy something in the first place. We have then tried to approximate this notion with our weight values. For a full-blown UNIX advisory system, however, it is necessary to have a more accurate model of user goals. One way to address this problem is by providing different UNIX users with questionnaires that have them rank various goals in different situations. This empirical approach can also be applied to determining the weightings of property goals and of enablement undesirability.

A feature of our current approach is that we can readily experiment with different weightings and study the resulting advisor behavior. In particular, it appears that we can use our model to begin to explore the effects of user personality on advice-giving. For example, it may well be possible to represent laziness by giving a high undesirability to enabling conditions (i.e., that the best solution is the one requiring the least effort, regardless of whether it achieves other goals) and to explore how the dialogs generated by our system compare to real-world advisory dialogs involving "lazy" users. While others have represented personality traits as weighted combinations of goals [14], there has been little work on studying how personality affects user-advisor interaction in advice-giving dialogs.

### Extending And Revising The Model

Our model currently uses a straightforward algorithm for revising its weightings: it simply substitutes a maximal positive or negative weighting. There are a variety of alternative, more subtle approaches that may lead to a more accurate model of the advice-giving process. One is to use a "delta" in response to user goals, rather than a fixed target. That is, the model can slightly increment or decrement its weighting each time the user mentions a goal. A more subtle version of this approach is to have the amount of change dependent on the value of the weighting: the less initial information about a goal, the more it changes (e.g., an initial weighting of 0, indicating an apathetic user, would result a large change; a weighting of $-7$ indicating a strongly negative goal would change little in response to a user statement). It's clearly necessary to compare the behavior of these approaches with our own.

### Integrating With Dialog Planning

Our plan evaluator and response generator together determine the high-level content of an explanatory justification for why a given plan is the best alternative. This content, however, may not be suitable as the entire response. Our model, for example, generates an explanation that "mv /tmp" allows the user to recover a file, but our illustrative dialog shows that the advisor has embellished that explanation with an additional causal justification of exactly how the file can be recovered: namely, that the file exists in "/tmp" and can be recovered by copying it back. While we can get reasonable advisor behavior by tweaking our model to automatically provide a causal explanation for any goal, a better approach is to think of our response generator as generating a goal for a dialog planning process (e.g., inform the user that "mv /tmp" allows the user to recover a file), where the dialog planner is responsible for determining whether it it is necessary to provide the causal justification.

## Conclusions

This paper has presented a model of the process by which an advisor can revise plan-oriented advice in response to user feedback. In particular, the model provides a general mechanism for evaluating candidate plans in response to different stated user goals, and it demonstrates how applying a small set of advice-giving principles can determine the content of an appropriate explanation for any newly recommended plan.

These advice-giving dialogs can be viewed as a collaborative planning process [4, 5, 6, 15], where the advisor and user cooperate to locate the best plan for the user's goals. Our work is complimentary to earlier efforts in this area. It differs in that it provides a formalization of the process of evaluating a plan that takes into account all factors (effects, properties, and enablements), as opposed to only a subset of them. It also provides a mechanism based on general conversational principles for determining the factors to discuss in a response, as opposed to the heuristic of simply discussing the "most important" factor.

## References


[1] Cawsey, A. *Explanation and Interaction: The Computer Generation of Explanatory Dialogs*, MIT Press, Cambridge, MA., 1993.

[2] Cawsey, A. Planning Interactive Explanations. *International Journal of Man-Machine Studies*, 38, 169–199, 1993.

[3] Chin, D. Acquiring User Models, *Artificial Intelligence Review*, 7(3-4), 1994.

[4] Chu- Carroll, J. and S. Carberry. Response Generation in Collaborative Negotiation. In *Proceedings of the 33rd Annual Meeting of the Association for Computational Linguistics*, pp. 136-143, 1995.

[5] Chu-Carroll, J. and Carberry, S. Generating Information-Sharing Subdialogues in Expert-User Consultation. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence*, Montreal, CA, pp. 1243-1250, 1995.

[6] Chu-Carroll, J. and Carberry, S. A Plan-Based Model for Response Generation in Collaborative Task-Oriented Dialogues. In *Proceedings of the Twelfth National Conference on Artificial Intelligence*, pp. 799-805, 1994.

[7] Grice, H. *Logic and Conversation.* In *Syntax and Semantics 3: Speech Acts*, Cole and Morgan, eds., New York, NY: Academic Press, 1975.

[8] McCoy. K.F. Generating Context-Sensitive Responses to Object-Related Misconceptions, *Artificial Intelligence*, 41(2), 1989.

[9] Moore, J. D. *Participating in explanatory dialogues: Interpreting and responding to questions in context.* Cambridge, MA: MIT Press, 1994.

[10] Moore, J. D. and Paris, C. L. Exploiting user feedback to compensate for the unreliability of user models. *User Modeling and User-Adapted Interaction*, 2(4), 287-330, 1992.

[11] Quilici, A. Using Justification Patterns To Advise Novice UNIX Users . *Artificial Intelligence Review*, 1998 (to appear).

[12] Quilici, A. Forming User Models by Understanding User Feedback. *User Modeling and User Adapted Interaction*, 3(4):321-358, 1994.

[13] Quilici, A., M. Dyer, and M. Flowers. Recognizing and Responding to Plan-Oriented Misconceptions, *Computational Linguistics*, 14(3):38–51, 1988.

[14] Reilly, W. and J. Bates. *Building Emotional Agents*, Technical Report 143, Computer Science Department, Carnegie Mellon University, Pittsburg, PA, 1992.

[15] Sidner, C. An Artificial Discourse Language for Collaborative Negotiation. In *Proceedings of the 12th AAAI*, pp. 814–819, 1994.