Weir, George (1988) *Learning from a plan-based interface.* Computers and Education, 12 (1). pp. 247-251. ISSN 0360-1315

Comput. Educ. Vol. 12, No. 1, pp. 247–251, 1988
Printed in Great Britain.

0360-1315/88 $3.00 + 0.00

LEARNING FROM A PLAN-BASED INTERFACE

GEORGE R. S. WEIR
Scottish HCI Centre, George House, North Hanover Street, Glasgow G1 2AD, U.K.

Abstract—The use of plan recognition in the user interface is presented as a basis for a learning environment within which students can assimilate the range of possible actions and objectives afforded by a target application. Support in the form of "present whereabouts" and "possible progressions" enable students to learn by doing. When combined with a facility for exploratory learning, this helps avoid the onerous air of a "teaching situation".

INTRODUCTION

While the role of the user interface is often stressed from the viewpoint of user-system performance, it is less frequently considered as a basis for user learning. Thus, in his recent survey of research in Intelligent Computer-Assisted Instruction (ICAI), Dede describes the user interface as "a service function with the ICAI system"[1]. (One exception to this neglect is the work of Slator *et al.*, who consider the learning benefits of "mnemonic feedback" in a natural language interface[2].) The presumption here is that the user interface does nothing more than direct user input and present system output. Yet, current developments in the design of intelligent interfaces suggest the possibility of an interface which acts as a learning environment.

The premium from employing an interface geared to learning is that users may acquire skills "as they work, without a large initial investment in a learning period"[2]. In keeping with this philosophy, my paper details the use of a plan-based interface design as a device for teaching. Although planning concepts have occasionally been applied to Computer Assisted Learning (e.g. Gensereth[3] and Miller[4]), these concentrate on understanding and debugging the user/student's behaviour. While this principal use should not be underplayed, plan-recognition can also serve as a technique for keeping the user appraised of his options and current situation. This, in turn, provides a strong teaching environment with learning re-inforced through constant feedback. In addition, the approach detailed below affords exploratory learning combined with an inspectable model of expertise.

THE PRIAM ENVIRONMENT

PRIAM (Plan Recogniser for Intelligent Advice and Monitoring)* is an interface design built upon a plan recognition algorithm. The design uses plan recognition in order to offer intelligent advice and monitoring in accord with plans stored in a knowledge-base. As it stands, PRIAM allows the modelling of a target domain, in terms of action sequences and the goals that these attain.

With such a plan-based description of the domain in its knowledge-base, PRIAM can then provide contextually relevant advice and supervision for students exploring the possible actions and available goals on the target system. Because PRIAM is usable independently of the real application there is no risk involved when experimenting with the available actions. This means that PRIAM can combine all the benefits of advice on appropriate user options, with no danger that naive users will make irrecoverable errors. This fact is especially reassuring to the new user. In this form PRIAM is an ideal teaching aid for command languages or any other goal-directed procedures.

Plans

The significance of plans in the PRIAM interface is that they constitute a higher level view of the goals available within the domain. Thus, possible goals are characterised in terms of the

*My development of PRIAM derives in part from work on the Alvey funded "Adaptive Intelligent Dialogues" project.

```
send mail     -    select, insert, write, send
    "         -    select, delete, write, send
    "         -    select, read, write, send
    "         -    create, insert, close, send

tidy files    -    select, read, close, purge
    "         -    select, insert, write, purge
    "         -    select, delete, write, purge
    "         -    select, read, remove, purge

run program   -    select, write, compile, execute
    "         -    select, insert, compile, execute
    "         -    select, delete, compile, execute
    "         -    create, insert, compile, execute
```

Fig. 1. Example command language plans.

component actions required for attaining these goals. A plan's components may be actions, commands, or anything which can be thought of as a constituent of the plan, such that the related goal is "accomplished" by completing an appropriate sequence of these components. From the student's perspective, such action sequences constitute the plans that may be employed toward the possible application objective. The student's initial learning aim is to establish what he can achieve on the target system. When he knows this, he must assimilate the necessary action sequences to attain his selected goals. In this respect, PRIAM assists by providing advice to students on their present position in the plan structure, and also in relating the goals that may be attained from the present action sequence. An account of this approach to interface design and a comparative example of using a PRIAM-based system against a more conventional style of interface are given in Davenport and Weir[5].

PRIAM's domain model

A description for the target domain is held in a PRIAM knowledge-base. This contains a specification of the possible plans with which the "student" should become familiar, in terms of the sequences of actions which they involve and the goals that they can accomplish. An example of a small command language plan set is given in Fig. 1.

This example has three possible goals: send mail, tidy files and run program. Each of these can be accomplished in four different ways, i.e. there are four unique sequences of actions which would result in the respective goal being achieved.

In using the PRIAM environment, the student can imagine that he is attempting to achieve any particular goal by entering the required actions in an appropriate sequence. For the command language example shown above this would be similar to entering commands on the target system save that PRIAM does not emulate the behaviour of that system. Rather, it provides guidance on the propriety and viability of the student's chosen input in accord with the domain specification in its knowledge-base.

Since it operates on plans and component actions, PRIAM is well suited to command language applications, but its scope is not limited to this field. Any domain which lends itself to description in terms of goals, plans and action sequences can be used with PRIAM, as is shown from the example in Fig. 2.

Unlike our command language example, each possible goal in this domain is attainable by only one unique plan, i.e. a single sequence of actions. Although this simplifies the learning task, the main difficulty for the potential student is that the same component actions may appear in several different plans. With so much similarity in the composition of viable plans, the student needs assistance in assimilating the connections between the possible goals and the requisite action sequences. This is the central issue which PRIAM addresses.

Monitoring and advice

In operation, PRIAM presents an environment to the student which "knows about" the domain of the application modelled in the PRIAM knowledge-base, whether it be a computer command language or a domain such as cookery. Using PRIAM's knowledge-rich environment the student

pancake batter	-	**sift flour, sift salt, add egg, add half milk, add melted butter, beat mixture till creamy, stir in other milk.**
sweet fritter batter	-	**sift flour, sift salt, add water, add butter, mix to thick smooth batter, whisk egg white, fold egg white into mixture.**
sponge pudding	-	**sift flour, sift salt, add butter, add sugar, beat until light and fluffy, add egg, add third of flour, beat mixture, add egg, add third of flour, beat mixture, add third of flour, add milk, fold in flour.**
crumble topping	-	**sift flour, rub in butter, add sugar, mix ingredients.**

Fig. 2. Example cooking domain plans.

can become familiar with the target application via the PRIAM model. In effect, the student can rehearse his actions on the PRIAM system, reflecting the way that he would attempt to accomplish the target objectives in reality. As he rehearses his plans to perfection, PRIAM not only adjudicates on his choice of action but provides support and advice to facilitate learning.

Each student entry is monitored by PRIAM which considers whether this action initiates any of the plans in the knowledge-base. If so, the student is given details of the related goals and relevant next action for each initiated plan.

For subsequent actions, PRIAM checks first against plans which have already been initiated to determine whether any of these is continued by the student's action. If so, details of the relevant goal and next command are reported to the student, otherwise the input is checked against plans which have been started but suspended (with a view to re-activation). Should there be no match in either case, PRIAM indicates that the entered action neither starts any plan nor furthers any that are presently active. It follows this by indicating the possible next actions for presently active plans and makes an offer of help on the basis of a mismatch between what is appropriate in the domain context and what the student has tried to do. The user can of course choose to ignore PRIAM's help. In either case the 'stray' command may be recorded (with the context of its occurrence) as an aspect of the student's history.

When completion of an action sequence achieves its corresponding goal, PRIAM reports the goal achieved and the plan employed. Completed plans are filed by the plan recogniser for later reference, and are retained beyond the end of the session. Clearly, such profiles might support heuristics to supplement PRIAM's present technique for identifying user's objectives. They may also provide a basis for adapting PRIAM's response to individuals, depending upon their past history and current position in the domain model.

By monitoring students as they execute action sequences, PRIAM is able to provide relevant advice and information at crucial points where they might stray from the predefined target goals. Advice and orientation may also be offered when users change direction in mid-task. Using its knowledge-base PRIAM is able to meet the following queries:

—What goals are possible?
—How do I achieve that goal?
—Where am I at present?
—What is the appropriate next action?
—What plans are suspended?
—What have I accomplished already?

Students can seek answers to such queries—in most cases, by explicitly requesting the information—relative to their current context in the domain model. PRIAM's response also accounts for the plans accomplished thus far. With such information available at all times students need never lose track of their present whereabouts, what they were doing, or what progressions

are possible in the current domain. A user may ask about the possible objectives that PRIAM recognises in relation to the application; he can specify a particular goal and have PRIAM detail the required actions it recognises as suitable for achieving that objective. He can also get information at any time on his current context as understood by PRIAM. This is provided as a list of the actions he has performed in the currently active plan(s). Also on request, the student can invoke a list of the possible next actions from his current context. Finally, details of any suspended plans and all plans accomplished to date can be viewed when desired.

Exploration

PRIAM provides two features whereby users may change their plans (context), thereby permitting them maximum flexibility in exploring the full range of available options. There is effectively no restriction on excursions. Active plans may either be abandoned or "held". In either case the option is invoked by the student starting an alternative plan. So, the signal to abandon or hold a plan is starting something new. This is more efficient than a hold command followed by a new initiation. Any number of plans may be held in this fashion, and can be re-activated by restarting the desired plan.

By means of a backstep command the user can also retreat to any previous choice point and take a different route. Thus the student may progressively undo his actions all the way to the initiating entry. This combination of backstepping and plan suspension offers the user greater flexibility than would a simple "abort plan" option.

LEARNING WITH PRIAM

From the above description of the facilities afforded by the PRIAM interface design it is apparent that it holds considerable potential for learning-support. In effect, the free exploration of the available plan structures, with the constantly available orientation advice and on-request goal information, combine to produce an environment which enables new users to familiarise themselves with a domain specification in planning terms, by simulating the tasks available on the target domain. Importantly, this format allows access to the expertise modelled in the PRIAM knowledge-base.

In the role already described, PRIAM acts as a stand-alone advisor able to guide students through the actions and objectives available on the modelled system. This allows users to gain experience of the application with no risk involved in experimenting with the available actions, and with all the benefit of PRIAM's context sensitive advice. Furthermore, the PRIAM engine can be used in the same form with any number of domain descriptions allowing a uniform mode of interaction and support for a variety of knowledge-bases.

PRIAM experiments

Using the simple command language structure shown in Fig. 1, a small experiment was devised to compare the performance of a plan-based interface against a more conventional command language interface. Subjects were asked to complete a set of six tasks, chosen at random, on each of the two interfaces. Notes were kept of the time taken for each task and of any incorrect goals.

Based on the data from 16 subjects—8 subjects for each interface—a comparison was made of the accumulated time for each condition. Results indicated a significant improvement in performance between the first and second interface, suggesting a considerable learning of the command language between the two sets of tasks, regardless of which interface was first encountered. Additionally, there was a marked interaction effect between order of presentation and the type of interface. When presented first, the conventional interface resulted in a significantly slower task set completion (see[5] for greater detail). Although these experiments are taken only to indicate a greater ease of use in favour of the PRIAM interface style, further experimental work is underway to measure the learning benefits which accrue from this approach. For the present, informal use of the system encourages the belief that users find the PRIAM interface style to be a supportive learning environment.

Designer support

In order to facilitate use of the PRIAM system, a support program (DBE) has been written which allows easy specification of domain goals. This also allows checks for name clashes between command and goal names as well as conflict checks on PRIAM commands. The actual commands employed by the PRIAM interface, i.e. to access additional information on the knowledge domain, can also be specified by the designer using DBE. (A full account of DBE and its use with PRIAM is given in Weir[6].)

CONCLUDING REMARKS

Although to date PRIAM has only been employed on command language domains, this should not be taken as a limitation on its range of possible application. In principle, its strategy may be employed to provide a learning-rich environment for any domain which lends itself to appropriate structuring. In this vein, a proposal has recently been made for PRIAM's use in a courseware authoring system for Computer Assisted Learning. (cf. Van Schaik and De Diana[7]). In such a context, PRIAM may fill a dual role. Firstly, it may allow structuring of the requirements on the courseware author by allowing authoring strategies to be pre-defined. In this case, PRIAM's context-sensitive environment provides a learning/support facility for the author. In the second case, PRIAM acts simply as a learning and support environment for students, in the manner outlined in the present paper.

REFERENCES

1. Dede C., A review and synthesis of recent research in intelligent computer-assisted instruction. *Int. J. Man–Machine Stud.* **24**, 329–353 (1986).
2. Slator B. M., Anderson M. P. and Conley W., Pygmalion at the interface. *Commun. ACM* 7(29), 599–604 (1986).
3. Genesereth M. R., The role of plans in intelligent teaching systems. In *Intelligent Tutoring Systems* (Edited by Sleeman D. and Brown J. S.), pp. 137–155. Academic Press, London.
4. Miller M. L., A structured planning and debugging environment for elementary programming. In *Intelligent Tutoring Systems* (Edited by Sleeman D. and Brown J. S.), pp. 119–135. Academic Press, London.
5. Davenport C. and Weir G., PRIAM: Plan Recognition for Intelligent Advice and Monitoring. In *People and Computers: Designing for Usability* (Edited by Harrison M. D. and Monk A. F.), pp. 296–315. CUP (1986).
6. Weir G., *PRIAM: A User's Guide*. Scottish HCI Centre, Report No. AMU8604/01S (1986).
7. Van Schaik P. and De Diana I., A methodology for object oriented courseware development. Presented at *CAL '87* (1987).