Formalisation and implementation of motivational tactics in tutoring systems

Teresa del Soldato
and
Benedict du Boulay

The explicit teaching knowledge implemented in the current generation of Intelligent Tutoring Systems (ITSs) concerns mostly domain-based aspects of instructional processes, overlooking motivational aspects. This paper describes an instructional planner able to make decisions (about the next task to do, whether to provide hints, etc.) in order to achieve two goals: traversing the domain — domain-based planning — and maintaining the learner's optimal motivational state — motivational planning. The traditional ITS architecture is extended to include the activities of *motivational state modelling* and *motivational planning*. For example, in motivational state modelling further learners' characteristics are diagnosed, e.g. effort and confidence. Sometimes the advice offered by a motivational planner disagrees with a domain-based plan, while in other cases both plans complement each other. A method of negotiation between the motivational plan and the domain-based plan is provided in order to arrive at a decision for action by the tutor.

Introduction

The explicit teaching knowledge implemented in the current generation of Intelligent Tutoring Systems (ITSs) concerns mostly domain-based aspects of the instructional process, overlooking its motivational aspects. However, teachers often interweave motivational tactics with the domain-based decisions, aiming to build conditions that stimulate the *wish* to learn[1]. Even in systems where attention is paid to motivational issues, the theory which drives the decision making is essentially implicitly embodied in the system in contrast to the explicit representation of the domain. For instance, the coach WEST (Burton & Brown, 1982) follows pedagogical principles such as "Do not tutor on two consecutive moves, no matter what", in order to prevent excess interventions that could affect the learner's interest, independence or feeling of control[2]. However, WEST does not include in its student model an explicit model of the learner's degree of independence or feeling of control. Theories of instructional motivation elaborate the influence of issues like confidence, challenge, control and curiosity in learning processes (Keller, 1983; Malone & Lepper, 1987) and suggest instructional tactics to keep the student in an optimal learning state and provide more appealing and effective interactions. The implementation of such motivational tactics in tutoring systems requires the insertion of a *motivational state modeller* and a *motivational planner* into the system's teaching expertise (del Soldato, 1992a, 1992b).

[1] According to (Lepper, Aspinwall, Mumme, & Chabay, 1990), expert teachers include among their goals "first, to sustain and enhance their students' motivation and interest in learning, ... and second, to maintain their pupils' feelings of self-esteem and self-efficacy, even in face of difficult or impossible problems."(p. 219).

[2] The goal of such a principle is explicitly described in (Burton & Brown, 1982) as to "prevent [the coach WEST] from being oppressive" (p. 91).

The motivational planner presented here is based on the motivational tactics defined by Malone and Lepper (1987) and by Keller (1983)[3], which were formalised and implemented as production rules manipulating domain-independent teaching primitives, such as *problem*, *help*, *assessment*, *answer*, etc.

Formalisation of motivational tactics

Whereas the motivational tactics discussed in (Malone & Lepper, 1987) and (Keller, 1983) apply to generic instructional contexts, the formalisation of such tactics presented here is directed to the implementation of motivational issues in typical ITSs and therefore characteristics of current systems, such as e.g. limitations of interface devices and structures of domain representations, are taken into account. Implementing motivational techniques demands shaping the system, including domain representation and student model, in several respects. In particular, the system must:

1. detect the student's motivational state;
2. react with the purpose of motivating distracted, less confident or discontented students, or sustaining the disposition of already motivated students.

The detection of a student's motivational state is obviously very much constrained by interface limitations. However, student effort (rather than performance) is a reasonably reliable indication of intrinsic motivation (Keller, 1983). Learners who display a high level of effort (detected through their activities, suggestions, responses) deserve emphatic praise even when their performance may not be optimal. A parallel way to obtain information about the learner is through questions regularly applied during the interaction, eliciting both students' self-evaluation and their appreciation of the system's behaviour. The system should also exploit the pattern of standard reactions as, for example, when students ask for help before attempting to solve a problem (possibly indicating low confidence), or on the contrary, the total absence of help requests during the entire interaction (possibly indicating extremely high confidence).

The notion of a system's reaction — triggering particular motivational tactics — suggests that a comprehensive instructional plan should consist of a "traditional" instructional plan combined with a motivational plan. Wasson (1990) proposed the division of instructional planning into two streams: content planning ("which topic to teach next"), followed by delivery planning ("how to teach the next topic"). At first sight the motivational plan seems to be completely embedded in the delivery plan. However, motivational tactics do not always simply complete the traditional content planning: sometimes they compete with it as well. A typical example of such a conflict is the necessity for less confident students to build their confidence by accumulating experience of success, in which case the system could provide problems likely to be correctly answered — based on topics that the student *already* knows.

While the detection of a learner's motivational profile shapes the student model, the system's reaction (e.g. suggesting an easier problem, asking a puzzling question) depends mostly on the resources found in the domain representation.

[3] Many of the motivational tactics formalised and implemented in the motivational planner were also discussed in (Lepper, Woolverton, Mumme, & Gurtner, 1993).

Detection of motivational state

The detection of the learner's general motivation is simplified by supposing that each motivational state is reflected in a particular behaviour pattern (for example, that every unconfident student behaves in a largely similar way). The tutor obtains information about the students by analysing their actions. In principle there are four different sources to analyse:

1. Questionnaires applied at the beginning of the first interaction, defining the learner's evaluation of her general level of self-confidence, affinity with challenging situations and motivation to study that particular domain. Arshad (1990) used this method to model the learners confidence state. Although useful to gather relevant information, pre-interaction questionnaires are static, and the learner's motivational state is likely to change during the interaction.
2. Communication with the student during the interaction. This is a more dynamic method, and it is possible to bypass the lack of natural language interface by limiting learner input to a set of standard expressions, accessed by menus and including possible answers to the tutor's suggestions (e.g. "ok", or "too difficult", or "easy!"), requests for help (e.g. "hint please", or "give up"), etc. Particular answers are typical of either less or more confident students (for example, "no, too difficult" for a less confident learner), so they are labelled as low/high confidence answers. Although on the one hand the communication with the learner is limited, on the other hand the possible answers offer less ambiguous interpretations of confidence states.
3. Students' requests for help and perseverance to complete the task. For example, a learner who rejects any help from the tutor is probably very confident and independent, whereas students who request help even before attempting to perform a task are likely to be less confident. Similarly, students who often give up tasks do not seem to display high confidence in their skills[4] .
4. Learners' self-evaluation of their motivational state (e.g. confidence, boredom) during the interaction. The best way to obtain this information is to provide a continuous input channel that can be spontaneously updated by the student. Unlike the other three sources of information, this one depends on features of the interface (scroll-bars, icons etc.) and should be independent of prompts provided by the tutor.

The generation of the learner's motivational model, through the channels mentioned above, is described in more detail in the following sections.

Effort

Although it is not absolutely clear whether "effort" is a reliable measure of the learner's motivation, one assumes that motivated learners put more effort on the task they are performing:

> "Motivation is concerned, of course, with an individual's willingness to persist and contribute effort to the task in which he or she is engaged. (Shuell, 1992, p.32)"

[4] ...providing that the task is not excessively difficult, in which case the students would indicate, by giving up, that they are aware of their own skill limitations.

Therefore the motivational aspects of a student model should focus on effort rather than performance. On the other hand, effort (or persistence) is measured *through* performance, via the learner's actions such as attempts to solve a problem, help requests, etc. In order to establish a clear distinction between the concepts of *effort* and *performance*, this work considers performance as the *result* of the process of solving a task (e.g. right or wrong answer) whereas effort refers to *how* this result was achieved (e.g. requiring much or little work).

Confidence

The detection of the learner's level of confidence relies mostly on the students' beliefs on their efficacy to perform the instructional task. Schunk (1989) correlates low and high confidence to persisting with or avoiding the task:

> "People who hold a low sense of efficacy for accomplishing a task may avoid it, whereas those who believe they are more capable should participate more eagerly.
>
> ...individuals who hold a high sense of efficacy ought to work harder and persist longer than those who doubt their capabilities. (Schunk, 1989, p. 14)"

According to the quotes above, less confident learners are likely to

1. avoid tasks perceived as difficult, or
2. give up a task before attempting to perform it.

The first point can be detected by the tutor if the student is presented with options of answers that explicitly mention the difficulty of the task (e.g. "No, thanks, it is too difficult", "I prefer an easier problem"). A student's lack of persistence in solving a problem can be defined in terms of help requests and the number of steps in the problem solving process. Each time the learner reacts according to one of these patterns, the learner confidence model will be decreased.

Another point discussed by Schunk (1989) relates to variations of the degree of confidence as a function of a task's outcome: accomplishing the task raises the learner's expectancies of future successes, whereas failures affect one's sense of self-efficacy and decreases the level of confidence.

Independence

Malone and Lepper (1987) discuss the importance of having students aware of their degree of control over their success. In this sense, the student "independence model" relates to the perceived feeling of needing or not needing the tutor's help to accomplish the instructional task. For instance, when the tutor frequently intervenes in the interaction providing hints, the learner' sense of independence decreases, whereas students succeeding in a task on their own have their feelings of independence increased. Therefore the student's independence model is primarily modelled by the frequency of tutor interventions during the interaction: low independence corresponds to situations in which the tutor has intervened in excess, and the independence model increases when the students are allowed to explore the problem on their own.

Domain-based vs. motivational-based planning

Typical domain-based planners select actions according to whether the learner *knows* a topic or has mastered a skill. The methodology here is twofold: detecting the current state of the learner's knowledge and skill (student modelling) and reacting appropriately in order to increase this knowledge and skill (teaching expertise). To take account of motivational factors, the twin activities of "detecting the state" and "reacting appropriately" are extended by adding a *motivational state* and *motivational planning* to the traditional ITS architecture. Sometimes the advice offered by a motivational planner disagrees with a domain-based plan, while in other cases both plans complement each other. (In a similar way Lepper et al. (1993) consider these two cases, as well as a third situation: when the motivational and the domain-based strategies are independent of each other). Here we discuss motivational planning and compare its behaviour to the decisions taken by typical domain-based planners.

Student succeeds performing the task

Let us consider, first, a situation in which the student succeeds in solving a problem. A typical domain-based planner would acknowledge the right answer and suggest (or directly provide) a harder problem, thus making sure the student is traversing the domain in a progressive manner (see Table 1). Such behaviour is well exemplified by Peachey and McCalla's (1986) instructional planner: when the learner masters an instructional goal, the planner focuses next on goals that require the topic just mastered as pre-requisite, traversing the domain in the direction of a specific ultimate goal. Some domain-based planners elaborate the performance feedback according to the instructional context. The Meno-tutor (Woolf, 1984), for example, acknowledges the student's answer in three distinct modes: explicit, implicit and emphatic (adding details about the domain topic in question).

Table 1- Domain-based planner: tutor's actions when learner succeeds in solving problem

comment:	performance feedback (and/or praise)
next prob:	next in the pre-requisite sequence (or harder)

In this case, *knowing* or *not knowing* the topic, or exhibiting or not exhibiting the relevant skill, is the only issue in the student model that drives the selection of suitable actions, so the diagnosis methods basically aim at defining whether the student knows the topic. Such a methodology characterises more detailed domain-based instructional planners. For example, Wasson (1990) implemented a planner based on a domain network representation which links topics through a variety of relations as well as "pre-requisite", and actions like *review, focus*, and *re-achieve* are selected to be executed. Such decisions, however, are based only on the assumption of student knowing (or not) topics. In some systems (e.g. see Anderson & Reiser, 1985), the student model has been improved by expanding the knowing-or-not binary state to a more graduated mastery scale, but still it is the learner's *knowledge* which drives instructional decisions.

Motivational planning takes into account other variables in the student model and widens the tutor's space of possible reactions. Just by considering binary states of *effort* (little/large) and *confidence* (low/ok) results in four different situations, each one

requiring a suitable set of actions from the tutor[5]. In one of the situations the motivational planner generates the same action as the domain-based planner (which corresponds to *effort = large* and *confidence = ok*). Table 2 presents the four cases and the corresponding actions specified by the motivational planner.

When the student's confidence is diagnosed as being low, the major goal for the planner is to help the learner regain a reasonable level of confidence, and one of the tactics for improving confidence is to increase the student's *experience of success*. The tutor should then select a task likely to be solved successfully again (e.g. a similar task to the one the student has just accomplished). This is a clear example of disagreement between the domain-based and the motivational planner, since simply traversing the domain to the next harder topic has been deliberately avoided.

Table 2 - Motivational planner: tutor's actions when learner succeeds in solving problem

	confidence →	
effort ↓	low	ok
little	(prevent disappointment) comment: diff-level promotion next prob: harder	(stimulate challenge) comment: suggest challenge next prob: much harder)
large	(increase experience of success) comment: link effort to success next prob: similar	(ideal situation) comment: perf feedback next prob: harder

On the other hand, if providing a right answer requires little effort from the student (even an insecure one) the tutor should move to harder tasks. In this case the tutor should make the *difficulty-level promotion* very clear, both by praising the successes obtained so far and warning about the new difficulties which are likely to be encountered at the harder level. The student is prepared then to cope with new failures without feeling too de-motivated. Let us now consider the case of a task that does not require very much effort from a normally confident learner. For a typical domain-based planner such a situation would be ideal, whereas from a motivational perspective the task could be perceived as being irrelevant or "boring", or in other words, de-motivating. The tutor should then increase the degree of challenge provided by the interaction, by adjusting the difficulty level to a harder one where the student would not always (easily) perform the task, and some effort would have to be spent to achieve success.

Student fails performing the task

In the case of the student giving a wrong answer, the domain-based planner would acknowledge the error and suggest a problem of the same difficulty, or an alternative path to traverse that region of the domain (Table 3).

[5] Just as for *knowledge states*, the binary states of *effort* and *confidence* could be expanded in a more graduated scale.

Table 3 - Domain-based planner: tutor's actions when learner fails in solving problem

comment:	acknowledge (or correct) wrong answer
next prob:	same difficulty (or easier)

The domain-based planner overlooks two issues:

1. Even if the student was not able to formulate a right answer, she may have spent a good deal of effort trying to perform the task.
2. If the learner is not spending much effort on the task (therefore not succeeding) the tutor should help to make the task more interesting and appealing.

The decisions described in Table 4 show possible ways to help an unsuccessful learner to restore her confidence (if she is a less confident student) or to increase her interest in the task.

Table 4 - Motivational planner: tutor's actions when learner fails in solving problem

	confidence →	
effort ↓	low	ok
little	(facilitate success) provide: hint insist (implicitly): on same prob	(stimulate curiosity) provide: surprising result insist (implicitly): on same prob
large	(facilitate success) comment: praise effort provide: hint insist (implicitly): on same prob	(normal situation) comment: performance feedback comment: praise effort next prob: same difficulty

It has already been noted that experience of success should increase the learner's confidence, but what can the tutor do if there is no success at all? The simplest action in this case is to overlook the failure (skipping the dreadful "wrong answer" statement, or avoiding displaying the right solution the student was not able to produce), motivate the learner to keep trying and provide hints to make success easier. In fact, Lepper et al. (1990) show that this tactic is applied by expert and efficient tutors:

> "Instead of providing explicit corrective feedback, these tutors rely on a much more subtle and indirect strategy. They offer students *hints* — questions or remarks that indirectly imply the inaccuracy of their poor response, suggest the direction in which they might proceed, or highlight the section of the problem that appears to be causing them difficulty." (p. 229)

Such a tactic should not be carried out indefinitely: the tutor could consider a "maximum failure limit" and move on to another task if a particular problem is excessively effort consuming. In the case where the student has already tried hard to perform the task, the effort should be explicitly acknowledged (for both cases of low confidence and of normal confidence).

If the planner is not concentrating its actions on restoring the student's confidence, other actions may be selected. Often a wrong answer provides a good clue about an inconsistency in the learner's comprehension of the topic. Provoking an incongruous or

paradoxical event is one of the tactics to stimulate cognitive curiosity (Keller, 1983; Malone & Lepper, 1987). Depending on the nature of the answer and the learner's mistake, the tutor may be able to use the wrong answer to generate a "clash" between what the student believes and what the domain model states [6].

Student gives up performing the task

Producing right or wrong answers are not the only ways of having a task done. The student may sometimes give up and request a new task, abandoning an incomplete problem.

Table 5 - Motivational planner: tutor's actions when learner gives up, after good general performance

effort ↓ \ confidence →	low	ok
little	(remind success) comment: previous successes provide: hint insist (implicitly): on same prob	(encourage effort) comment: on lack of effort suggest: help insist (explicitly): on same prob
large	(facilitate success) comment: praise effort provide: hint insist (implicitly): on same prob	(normal situation) comment: praise effort next prob: same difficulty

For the less confident learners (see Table 5), this situation is very similar to failing when performing a task, and again the tutor could ignore the failure (for a certain number of times) and provide hints to help perform the task correctly. If the student has presented a generally good performance, previous successes should be mentioned, making the learner aware of her capabilities.

From more confident students a bit more of effort could be required (if the effort spent on the task was low), especially if later the tutor praises the resulting effort linking it to good performance. Note that in this case rather than providing hints at once, the tutor negotiates the help delivery with the student. When interacting with reasonably confident learners the tutor should not be intrusive but should share decisions with the student. Another situation concerning the tutor's "intrusion" in the interaction is discussed in the following example.

Student requests help

The situations cited so far concerned the effect of confidence and effort in motivational planning, referring to tactics for stimulating challenge, curiosity and confidence. Another important issue in motivation is the degree of control the student is able (or allowed) to exert in the interaction.

[6] Such an "entrapment" tactic is used by systems which perform Socratic dialogues (see e.g. Clancey, 1982).

Tutors usually provide hints and clues when the student requests help. Lepper and Chabay raise the question of whether help should be *always* available to the student:

> "Should the tutor *always* intervene when the student requests help, or should some evidence of effort and independent work be demanded first?" (Lepper & Chabay, 1988, p. 248)

The approach adopted in this work is that independence should be encouraged, specially if the tutor has already intervened too much, and therefore decreased the student's feeling of control and independence over the interaction. Avoiding further interventions, at least for a while, is the most basic action to take in order to restore the learner's sense of independence. Help can be skipped in two situations:

1. if the student is requesting help in excess, or
2. if the student is lost and help should be delivered, but at the same time the tutor assumes that it has already intervened in excess[7].

However, if the *confidence* model is low, help should be provided in order to facilitate the learner succeeding on the task. The priority of *confidence* over *independence* assumed here is due to the fact a less confident student is eager to be helped, and less likely to feel annoyed by excessive interventions from the tutor. Examples of a motivational tutor's behaviour when the learner requests help are presented in Table 6.

Table 6 - Motivational planner: tutor's actions when learner requests help

	confidence →	
independence ↓	low	ok
low	(facilitate success) provide: specific help	(increase independence) comment: encourage independence skip: providing help
ok	(facilitate success) provide: specific help	(normal situation) provide: generic help

One can note the distinction between providing *specific* help (to less confident students) and providing *generic* help. Specific hints present more details about the problem and help the student in a more direct way, whereas generic help is "less intrusive". Delivering help of different degrees of generality is a tactic also considered by Lepper at al. (1993)[8]: "Increase or decrease the specificity of hints provided to the student as a function of the student's difficulty at a particular point" (p. 83).

The discrepancies between domain-based planning and motivational planning revealed here suggest that the inclusion of motivational tactics in a tutor's instructional planning mechanisms alter in a significant way the behaviour of the tutor.

---

[7] The second situation was included in the pedagogical principles of the coach WEST (see Burton & Brown, 1982), as discussed previously in this paper.

[8] Lepper at al. relate such a tactic to the goal of increasing (or decreasing) the challenge of a task, whereas here the specificity of hints concern the degree of confidence presented by the student. Nevertheless, strategies for enhancing challenge and confidence are closed related in (Lepper et al., 1993).

Implementation of motivational tactics

The motivational tactics described above were implemented through the application of production rules to a database consisting of information about the state of the interaction, the student's progress in mastering the domain and the motivational state of the student. The set of production rules detects the student's motivational state and reacts in order to maintain the student's motivation, producing a "MOtivational REactive plan". The system is named MORE.

Constructing the database required the definition of a set of instructional primitives to represent objects and actions in a teaching interaction. Such instructional primitives are dynamically manipulated by the instructional planner and student modeller.

This section describes:

1. the set of instructional primitives adopted;
2. the rules which represent the student modellers, concerning both progress across the domain and motivational state;
3. the instructional planner, which is split into three different modules: the domain-based planner, the motivational planner and the negotiation rules between the planners.

The instructional primitives, the student modellers and the instructional planners are domain-independent, although it is assumed that the domain can be organised in a particular problem-solving pattern. Examples of the system performance when applied to a concrete domain (Prolog-debugging) are provided in this paper.

Instructional primitives

A language describing instructional primitives was defined in order to build a database containing information about the interaction and the student. This language is not intended to cover all aspects of instructional interactions: its goal is to establish the necessary primitives to represent the learner's motivational state and the motivational tactics. The set of instructional primitives described here is versatile enough to be included into more complex sets of instructional objects.

Objects (e.g. problem, help, answer) often require properties, or attributes, such as type or content. Besides, many objects change "state" during the interaction: *help*, for instance, may be *requested* or *not-needed*. The instructional objects used in this work, along with their respective properties (possible types, contents and states) are described here.

Problem

Problems are tasks that the student should perform (learn, master, solve), and basically correspond to domain topics. As in a typical network representation of domain, *problems* are linked through relations such as pre-requisite or similarity. Several motivational tactics place special emphasis on the *next* task to be performed by the student, such as e.g. suggesting a similar task to increase the learner's experience of success, therefore one of the properties of a *problem* is its relation to the previous task performed by the student. Basically, tasks have to be ordered by difficulty level, so that a problem can be *harder* or *easier* than the previous problem, or present the same degree

of difficulty (*same-diff*). *Similar problems* should also present the same degree of difficulty, as well as require similar reasoning to be successfully performed.

Problems usually require a certain number of *steps* or attempts to be successfully solved (see next section). While the learner is dealing with steps towards a final answer or solution, the *problem* state is set as *solving*. When the student produces a final answer, whether the task is considered successfully performed or not generates the states *succeeded* or *failed*. This is a rather simplistic classification, since complex domains include problems with many different degrees in which a solution may be considered "correct". However, the emphasis of this work does not rely on the aspects of instruction related to the subject domain, although potentially it can include more detailed domain-centred approaches. The learner may also give up working on the task, or reject a task suggested by the tutor. All possible states and types for the object *problem* are presented in Table 7.

Table 7 - Problem properties

Problem	
states:	*suggesting, solving, rejected, succeeded, failed, given-up*
types:	*harder, easier, same-diff, similar*

Step

The *steps* required to perform a task or solve a problem may depend on:

1. characteristics of the domain,
2. features of that kind of task,
3. characteristics of that particular task.

Therefore some steps in the problem solving process can be defined in advance, but the complete list of possible steps to solve a particular task is set only at the moment the student agrees to perform the task. MORE keeps a record of all the steps performed by the learner, detecting whether a particular step is redundantly repeated.

Help

Help refers to hints and clues offered and provided by the tutor. Hints can be either requested by the student (state *requested*), suggested by the tutor when the system suspects the student needs help (state *suggesting*), or actually provided when the system assumes the student surely needs support (state *providing*). Besides, the offer of help may be rejected by the student (state *rejected*), or the tutor may decide that even if help is needed, it would be more appropriate to avoid intervening in the interaction (state *skipping*). Otherwise, the state *not-needed* is applied. Hints vary in their degree of generality, here referred to as "detail", and content. More detailed and helpful hints are constructed from combinations of simpler hints.

The content of the hint provided by the tutor refers to the present step (when the student requests help to complete a step), to the next step (in cases such as when the learner is

lost, performing the same step instead of progressing towards the solution) or to a "surprise result", which aims to present a contradiction to stimulate the student's curiosity. Previous steps or problems may also be re-presented to the student, highlighting similarities between solutions already achieved and the present step.

The states of an object (in this case, the states of the object *help*), refer to the situation of the object at every moment of the interaction. However, sometimes it is necessary to establish a decision about an eventual state further in the interaction. When the student rejects a help offer by the tutor, a second offer immediately following the first hint suggestion should be avoided even if the conditions of the interaction indicate that help is needed, in order to respect the learner's decision and independence. For this reason the property *skip-next* was created, which holds information about whether the following help offer should be avoided.

Possible states, contents and degrees of detail are presented in Table 8.

Table 8 - Help properties

Help	
states:	*suggesting, providing, rejected, skipping, requested*
content:	*present-step, next-step, previous-step, surprise-result*
detail:	*general, specific*
skip-next:	*yes, no*

Answer

Although every action performed by the learner does not directly relate to a question proposed by the tutor, MORE refers to any action expected from the student as an *answer*. Positive and negative answers refer to the learner's reactions to suggestions posed by the tutor. The student may agree or disagree with a suggestion, reflecting different degrees of confidence as stated in the section on Confidence. Statements made by the student are checked by the system and classified as *right* or *wrong* answers. Answers which are not reactions to suggestions or statements are considered *steps* of the problem solving process. MORE checks the *step* and classifies it according to the set of all possible steps for that particular task.

Table 9 - Answer properties

Answer	
states:	*get, check, checked*
content:	*positive, negative, right, wrong, step, give-up, help-request*
type:	*low-confidence, high-confidence, neutral*

Answers, or actions performed by the student, are expected at certain moments during the interaction. On such occasions, the answer state is set as *get* and the system does nothing until the learner reacts (*help* may be requested at any time, though). As soon as the student reacts, the *answer* is analysed by the student modeller (state *check)*. The

state *checked* is set when the answer has been analysed and the system is planning its next action. *Answers* consist of states, contents and types presented in Table 9.

Assessment

Assessment is feedback provided by the tutor on whether the student's answer is right or wrong. In many systems, e.g. SCHOLAR (Carbonell, 1970) and BUGGY (Brown & Burton, 1978), positive assessment delivery may include or be replaced by a praising element such as "Very good". In this work assessment and praise are explicitly distinct. Possible states and contents of *assessment* are listed in Table 10.

Table 10 - Assessment properties

Assessment	
states:	*providing, not-needed*
content:	*right, wrong*

Comment

Several instructional (motivational) actions such as praising, encouraging and challenging, are performed through *comments* provided by the tutor. The set of possible *comment* states is very narrow: either a *comment* is provided (state *providing*) or not needed. It is the scope of possible *comment* contents that bears the rich variety of this object, as shown in Table 11.

According to Schunk (1989) praising the learners' effort as opposed to their performance produces different reactions in the students' motivational state and should be delivered on particular occasions. For example, praising effort should be avoided if there is a reason to praise performance. Therefore the content of praising *comments* eventually delivered by MORE consists of either effort (content *praise-effort*) or performance (content *praise-perf*).

Table 11 - Comment properties

Comment	
states:	*providing, not-needed*
content:	*praise-perf, praise-effort, level-promotion, challenge, trying-harder, previous-successes, encourage-indep*

Content *trying-harder* refers to *comments* that encourage the learner to continue working when the task is abandoned, suggesting that more effort is necessary to complete the task. Other comments provided by the tutor may consist of reminding the student that success in a similar task has been previously achieved (content *previous-successes*), or encourage the learners to work on their own when help has been excessively requested and delivered (content *encourage-indep*).

Previously in this paper we mentioned the necessity of having the less confident students aware of their level promotions (i.e. when the task gets harder), preventing eventual de-motivation caused by new difficulties immediately after a certain degree of success had

been achieved. In this sense, comments of content *level-promotion* inform the student that the next tasks will get more difficult because the current topic or skill being studied has already been mastered. On the other hand, increasing the level of difficulty of the task may be necessary in order to challenge more confident students. Remarks provided by the tutor to make the challenge more explicit are labelled as comments of content *challenge*.

Student modelling

In typical ITSs, the student's performance is analysed in order to build a model of what the student *knows*. In MORE such a task is twofold: not only is the learner's *knowledge* important, but also the learner's *motivation* is relevant. Therefore two sets of rules are necessary to generate a model for the student's performance and a model for the student's motivational state. The following describes the generation of both models, and it is important to notice that since they are independent modules the generation of the learner's performance model could be replaced by another (more detailed) modelling method.

Performance modelling

Since performance modelling is not the major focus of this work, the student's competence in mastering a skill is basically classified as good or bad according to the rate between tasks tried and tasks completed successfully (see Table 12, rules P1 and P2). In this work the threshold between good and bad performance has been set to the rate (successfully completed tasks)/(total tasks) at 0.5, but such a limit can be adjusted to different values. The performance modelling mechanism can be easily upgraded without affecting the basic architecture of MORE.

A second feature in the performance model of the student refers to the path traversed by the learner to solve the problem. In the event of the student repeating the same step for a pre-determined number of times (which is set as the step-repetition-limit), the solution path is modelled as *lost* (rule P3), therefore the student needs help (but whether help will be suggested or provided is decided by the instructional planner). Rule P4 restores the value *ok* for *path*, when the student's focus moves from the repeated step to a to a different step. The step-repetition-limit is set for every task according to its degree of difficulty.

Table 12 - Performance modelling

rule	rate right/total tasks	performance model
P1	above *perf-threshold*	good
P2	below *perf-threshold*	bad

rule	last-step-repetitions	path model
P3	above *step-repetition-limit*	lost
P4	below *step-repetition-limit*	ok

A similar but more elaborate mechanism to detect deviations from an optimal learning path is adopted in the Meno-tutor (Woolf, 1984). There the learner's state of "confusion" is measured "as a function of the number of questions asked, the number of

incorrect responses given, and the extent to which the student's frontier of knowledge has been explored" (p.80). The Meno-tutor implementation also includes a wrong-answer-threshold (similar to the step-repetition-limit), defined as "the number of permitted wrong answers" (p. 67).

Confidence modelling

Confidence is represented as a value (*conf-value)* in a linear scale, and the limits for the lowest and the highest possible confidence values are set before the interaction with the student takes place. The confidence value is incremented and decremented in large or small (normal) steps. The values for these steps (named *conf-inc, conf-dec, large-conf-inc, large-conf-dec*), are previously set like the confidence limits. As a trial value, the confidence limits were set as 10 and 0, the *conf-inc* as 1 and the *large-conf-inc* as 2 (and the values for *conf-dec* and *large-conf-dec* were set as -1 and -2 respectively), so that the student's confidence model at any moment during the interaction corresponds to any integer value within the range 0-10. A threshold value (*conf-threshold)* is defined to distinguish between low and high confidence. For instance, if the *conf-threshold* value is set to the value 4 then *conf-value* 5 corresponds to a normal degree of confidence, and *conf-value* 3 is considered low confidence. The limits for the confidence scale and the low confidence threshold value may be altered if more precision is required.

The student's confidence model (the numerical value associated to *conf-value*) is dynamically adjusted during the interaction according to the rules described in Table 13.

Table 13 - Confidence modelling

rule	answer type	answer content	confidence model
C1	*low-conf*	*pos/neg*	decrement by *conf-dec*
C2	*high-conf*	*pos/neg*	increment by *conf-inc*

rule	steps	answer content	confidence model
C3	none	*help request*	decrement by *conf-dec*

rule	problem state	with / without help	confidence model
C4	*succeeded*	without *help*	increment by *large-conf-inc*
C5	*succeeded*	with *help*	increment by *conf-inc*
C6	*failed*	without *help*	decrement by *conf-dec*
C7	*failed*	with *help*	decrement by *large-conf-dec*

Rules 1 and 2 refer to the answer expression, as explained in the description of *answer types* (see previous section). Rule 3 reflects the case of a student asking for help from the tutor before even trying to perform the task. The four last rules concern the result of the task. If the task is accomplished, the student's confidence in future successes rises, whereas if the student failed in performing the task, the expectancy of a following success decreases. Refining this model, successes obtained completely independent of help from the tutor are likely to increase the learner's confidence in a more dramatic way than successes obtained after being helped. On the other hand, a failure despite the hints provided by the tutor saps the learner's confidence more than if the student fails but success was not facilitated in any sense.

Effort modelling

Table 14 presents a model for classifying students' effort as a function of their persistence to solve the problem and requests for help to perform the task. It is assumed that persistence to solve the problem can be measured through the number of attempts to get a solution, or *steps* performed, so that *many* steps reflects a greater degree of effort from the learner. The quantification of few/many attempts is defined by the domain expert, according to each problem's level of difficulty. A value is set as a threshold between few and many steps (*few-steps-lim)*, so any quantity of attempts higher than that limit is considered *many steps*, otherwise the student has only performed *few steps*. Besides the number of steps performed, a student who requests hints from the tutor or accepts help offered by the tutor spends less effort than learners who try to perform the task on their own. The result of the task performance is another relevant factor, and giving up the task obviously denotes less persistence than working until the problem is solved[9] .

Table 14 - Effort modelling

problem state	steps	with/without help	effort model
given-up	none	—	none
given-up	*few*	without *help*	little
given-up	*few*	with *help*	↓
succeeded	*few*	with *help*	↓
succeeded	*few*	without *help*	medium
given-up	*many*	with *help*	↓
given-up	*many*	without *help*	large
succeeded	*many*	with *help*	↓
succeeded	*many*	without *help*	maximum

Independence modelling

The independence model (see Table 15) is similar to the confidence model in many respects. Independence is represented as a numerical value (*indep-value*), and limits are set for the highest and lowest *indep-value*, as well as incremental and decremental steps (*indep-inc/dec, large-indep-inc/dec*) and an independence threshold (*indep-threshold*). The aim of the independence model is to evaluate whether the tutor is intervening to excess. In this sense, each time the tutor interrupts the interaction, e.g. offering help or providing it directly, the *indep-value* is decremented (rules I1, I2 and I3). One assumes that not only the "quantity" of interventions affects the learner feelings of independence, but also the "quality" of such interventions is relevant. Offering a detailed hint, which directs the learner to the task's solution, implies that the learner *needs* help to succeed, whereas a vague hint about the learner having solved a similar problem in a previous task may affect the student's feeling of independence in a less evident way. To accommodate such distinction in the model, *specific* hints decrement the *indep-value* by a larger

[9] If the student fails solving the problem, the effort model is evaluated as for the *given-up* case.

amount (rule I3). On the other hand, when the student rejects help, the *indep-value* is obviously incremented (rule I4). The *indep-value* is also incremented when the tutor skips offering help, even if the student is not following an optimal solution path (rule I5). The student is not aware of the tutor refraining from intervening, otherwise the learner's independence would be affected anyway (there is no point in the tutor advertising that it *should* intervene but will not in fact interrupt the interaction, since that would be an interruption anyway). So in this case, the *indep-value* is incremented on account of the balance between the tutor's interventions and the student's freedom to explore the solution path, in favour of the latter.

Table 15 - Independence modelling

rule	help state	help detail	independence model
I1	*suggesting*	—	decrement by *indep-dec*
I2	*providing*	*general*	decrement by *indep-dec*
I3	*providing*	*specific*	decrement by *large-indep-dec*
I4	*rejected*	—	increment by *large-indep-inc*

Instructional planning

The necessity to divide the student model into two independent parts (the performance and the motivational aspects of the student's state) was discussed earlier in this paper. In an analogous way, the instructional planner comprises two modules, one referring to the progress across the domain (domain-based planner) and the other referring to increasing or maintaining the student's motivation to learn. Whereas domain-based planning only concerns the performance model, the motivational planner is driven by both the motivational state and the performance state, especially when it concerns the refinement of motivational top-level tactics. For instance, if the learner is not confident, the motivational planner sets the goal *increase confidence*. Nevertheless, the tactics appropriate to increase the student's confidence after a task failure are different to those required in the case where the learner succeeded the task.

Domain-based planner

The ultimate goal of a domain-based planner is to have the student master a particular set of topics, or skills, in the domain. Usually topics are largely ordered through pre-requisite links, and the planner reasons about sequences of topics to be learned, navigating towards a goal topic. MORE includes a simple domain-based planner which aims to "advance" across the domain every time a topic or skill is mastered by the learner (see Table 16).

Rule D1 shows the case when the student succeeded and therefore the tutor suggests a *harder* topic (or next in a pre-requisite progression). When the student does not perform the task, one of rules D2, D3 and D4 provides an alternative path towards the topic goal, by suggesting a problem of the same level of difficulty (*same-diff*). If the student requests help, the tutor provides a hint about the topic (rule D5) and if the student is lost the tutor intervenes with a hint about the next step (rule D7). Although the domain-based planner does not "offer" help, rather providing it directly, the motivational planner

described in the next section may *suggest* help. Therefore rule D6 deals with the case of a *rejected* offer of help, in which case the tutor does nothing.

Table 16 - Domain-based planner

rule	STUDENT MODEL / HISTORY	ACTION
D1	*problem-state = succeeded*	provide *assessment type right* suggest *problem type harder*
D2	*problem-state = failed*	provide *assessment type wrong* suggest *problem type same-diff*
D3	*problem-state = given-up*	suggest *problem type same-diff*
D4	*problem-state = rejected*	suggest *problem type same-diff*
D5	*help-state = requested*	provide *help content present-step*
D6	*help-state = rejected*	*(help not-needed)*
D7	*path-state = lost*	provide *help content next-step*
Note: in order to make this table more clear, the actions were described in a simplified format. For instance, in rule D1 the action "provide *assessment type wrong*" actually means that the *assessment-state* is set to state *providing* and the *assessment-type* is set as *wrong* .		

Motivational planner

The motivational planner embedded in MORE determines tactics to increase or maintain the student's motivation to work. The decision about which tactics to apply depends both on the state of the interaction, such as problem state, confidence value, etc., and on the top-level tactics already present (or necessarily absent) in the motivational plan. For example, the tactic *increase experience of success* is a specific tactic for the top-level tactic *increase confidence*. If the planner sets the tactic *increase confidence* to be executed and the student performs a task successfully, then the tactic *increase experience of success* can be included in the plan, specifying the way the tutor should try to increase the learner's confidence. The set of rules that generate motivational tactics is presented in Table 17.

Rules M1, M2 and M6 are straightforward: when the student model values for confidence, effort or independence (control) decrease below the respective thresholds, the tutor should apply tactics aiming to increase such motivational aspects. Increasing the learner's confidence and independence *at the same time* is a contradictory strategy because less confident students need to succeed in order to raise their confidence, and this may require a great deal of tutor intervention delivering hints that could facilitate the student's success. Excess of intervention, on the other hand, should be avoided when learners need reassurance of their feelings of independence. One should note that in this motivational planner, raising the student's confidence (*tactic increase confidence*) was given priority over raising student's independence (*tactic increase control*, in rule M6), assuming that a less confident student is not really annoyed by excessive help and attention from the tutor. Once the student's confidence is restored, the tutor is then able to apply tactics to increase the student's independence.

Table 17 - Motivational planner

rule	student model / history	top-level tactics	tactic
M1	*conf-value* < *conf-threshold*	—	*increase confidence*
M2	*effort-value* < *medium*	—	*increase effort*
M3	*effort-value* > *medium*	—	*maintain effort*
M4	*help-state* = *rejected*	—	*respect control*
M5	*problem-state* = *given-up* above giv-up-lim	—	*respect control*
M6	*indep-value* < *indep-threshold*	not *increase confidence*	*increase control*
M7	*problem-state* = *succeeded*	*increase confidence*	*inc. experience success*
M8	*problem-state* = *failed*	*increase confidence*	*facilitate success*
M9	*problem-state* = *given-up*	*increase effort* not *increase confidence* not *respect control*	*encourage effort*
M10	*problem-state* = *given-up*	*increase confidence* not *respect control*	*facilitate success*
M11	*problem-state* = *succeeded*	*increase effort*	*stimulate challenge*
M12	—	*stimulate challenge* *increase confidence*	*emphasise promotion*
M13	*problem-state* = *failed*	*increase effort* not *increase confidence*	*stimulate curiosity*
M14	*perf-value* = *good*	*facilitate success* *increase effort*	*remind successes*
M15	*path-state* = *lost*	*increase control*	*avoid intervention*
M16	*help-state* = *requested*	*increase control*	*encourage indep*
M17	—	*encourage indep*	*avoid intervention*
M18	*help-state* = *rejected* *help-skip-next* = *no*	*respect control*	*avoid next intervention*
M19	*help-state* ≠ *rejected* *help-skip-next* = *yes*	—	*avoid intervention*
M20	*help-state* ≠ *requested*	not *increase confidence* not *stimulate curiosity*	*share control*

Assuming that there is a direct correspondence between high effort and general motivation, rule M3 aims to *maintain* the learner's state of motivation when a great deal of effort has been spent on the instructional task. The four rules discussed so far rely exclusively on the motivational student model. Rules M4 and M5, on the contrary, concern solely aspects of the interaction history. The point of both rules is that once the learner has explicitly refused to be helped by the tutor, or insisted on abandoning the task, the tutor should respect such decisions. The *giv-up-lim* parameter, or "giving up

limit", defines a value for the number of times the tutor can insist on helping the students when they explicitly abandon the task. Analogous to all the other parameters in the system, the value for *giv-up-lim* is set for every interaction, and the trial value suggested here is 2: if the learner gives up performing the task for the second time the tutor respects the learner's decision. Therefore Rule M5 still bears a certain degree of malleability, since *giv-up-lim* may be set to a high value.

Rules M7, M11 and M12 formalise the situations described in Table 2, generating tactics to challenge confident learners or encourage successful but less confident students. The actual translation of the motivational tactics into actions (such as, for instance, *stimulate challenge* resulting in suggesting a much harder problem) will be performed by the negotiation planner. In the same way, rules M8 and M13 reflect the situations described in Table 4, whereas rules M9, M10 and M14 correspond to Table 5. Since giving up denotes a degree of decision from the student (whereas a failure is obviously involuntary), even when the student's confidence is low the "giving up limit" should be observed, which explains the necessary absence of the tactic *respect control* in order to activate rules M9 and M10.

The remaining rules concern raising or maintaining the learner's feelings of independence (see Table 6). Rule M15 forces the tutor to skip interrupting the student even when help is needed, avoiding excessive intervention. Rules M16 and M17, on the other hand, reflect the need of encouraging (M16) the tutor-dependent student to work without help (M17) even if the learner is asking for clues. One should remember that skipping help interventions raises the student independence model, so eventually the top-level tactic *increase control* will be removed and the promised help delivered, if still needed or requested. Rule M18, in conjunction with rule M4, avoids a possible help offer or delivery immediately following the student's rejection of hints, irrespective of the independence model value or solving path state. If the possibility of a following help offer actually arises, rule M19 applies the tactic *avoid intervention,* at the same time dealing with the contradictory situation of a student requesting help immediately after having rejected it, in which case the hint is delivered. Apparently rule M19 is in contradiction with rules M16-M17, since the former prevents avoiding interventions in case the student requests help, while the latter avoids interventions even when requested. The contradiction is resolved by the fact that rules M16-17 only apply when the student's independence model is low, in which case the tutor does not intervene to offer help. Rule M19, on the other hand, implicitly depends on a previous help offer rejected by the student generating the tactic *avoid future intervention* (rule M18) which in turn generates through the negotiation planner the condition *help-skip-next* = *yes*, as it will be discussed later in this paper (see section on Negotiation planner).

Rule M20 does not radically restrain the tutor from intervening, but concerns the sharing of responsibility over the help delivery. In other words, when the tactic *share control* is applied the tutor should not intervene directly, but only *offer* to help instead. This rule presents three negative conditions. First, the offer does not apply when there is need to increase the learner's confidence, for the same priority reasons discussed for rule M6. Second, stimulating curiosity benefits from a degree of surprise, which could be affected if a previous indication of intervention is given. Finally, it would be redundant to offer to help soon after the student has explicitly requested help. One should notice that the domain-based planner included in this work embeds an implicit instance of the tactic *share control* concerning the delivery of tasks to be performed by the student, since

problems are always suggested rather than imposed. If a less negotiating domain-based planner is adapted to MORE, it would be necessary to include in the motivational planner a rule similar to rule M20, in order to share the control over the tasks to be performed and widen the learner's scope for relevant choices and responsibility.

Negotiation planner

MORE includes two independent planners in its instructional planning process, one generating domain-based actions and the other generating motivational tactics. Sometimes the actions and tactics are complementary, as in cases such as the action *provide help* and tactic *facilitate success*. However, it may happen that the two planners disagree, and a mechanism to negotiate between traversing the domain or increasing the student's motivation has to be applied. A third set of rules has been designed to amalgamate the tactics suggested by the motivational planner with the actions suggested by the domain-based planner in order to produce a combined action. Since MORE is designed to investigate motivational states, the decisions taken by the motivational planner overrule the domain-based planner. However, because the planners are independent, the system can be set so that the motivational planner is by-passed and the decisions are wholly taken by the domain-based planner only. The negotiation rules are listed in Table 18. Examples of instructional plans generated by the three planners described in these sections are presented later in this paper.

As mentioned in the previous section, the negotiation planner is responsible for translating the motivational tactics into instructional actions to be performed by the tutor, overriding (deleting), altering or complementing the actions already provided by the domain-based planner. There are cases in which the "disagreement" between the actions and tactics is not extreme, so the deleted action is actually replaced by a similar action. For instance, if the domain-based action determines that general help should be provided to the student, and the tactic *increase confidence* is generated by the motivational planner, the negotiation planner combines both decisions resulting in *specific* help being delivered. Therefore the domain-based action would be only partially altered. If the motivational planner generates the tactic *avoid intervention* instead, the help delivery is totally neglected by the negotiation planner and the tutor skips intervening, which is a case of complete disagreement. In the case where the tactics in the motivational plan do not interfere with the help delivery, and the negotiation planner decides that a comment should be delivered together with the hint, then the domain-based plan and the motivational plan complement each other.

The first three rules in Table 18 were discussed in the previous section (see Table 2). Both rules N1 and N2 alter the degree of difficulty of the next task, still maintaining the act of suggesting a new problem. The first rule decreases the difficulty of the problem in order to provide a similar task, likely to be successfully performed and so increasing the student's confidence. The second rule, on the contrary, amalgamates the need to challenge the student with the suggestion of a more demanding task, and adds a comment about the challenge to make sure the increased level of difficulty is perceived by the student. Rule N3 complements the suggestion of a harder problem with a comment warning the student about the increasing level of difficulty. Completing the cases described in Table 2, rule N6 provides the deserved praise in case the student has spent a good deal of effort on the task.

The tutor's behaviour envisioned in table 4 is generated through rules N4, N5 and N7. Rules N4 and N5 generate actions which completely disagree with the domain-based plan, encouraging the student to keep solving the problem instead of "accepting" the learner's failure. Rules N6 and N7 refer directly to the research results obtained by Schunk (1989): facing the choice of praising both the student's performance and effort, the tutor favours the former (rule N6). Nevertheless, when large effort was spent although success was not achieved, the tutor acknowledges the student's persistence (rule N7).

When the student gives up accomplishing the task (see Table 5), one of rules N8 or N9 is activated. Whereas the domain-based plan moves to an alternative problem, the motivational plan determines that either the student's success should be facilitated (for less confident learners, rule N8) or more effort should be encouraged (for confident but not persistent learners, rule N9). In both cases, though, the negotiation planner determines that the student should be encouraged to persist in solving the problem. If the tutor is trying to facilitate the learner's success, hints are directly provided, and if the student has been successful in previous tasks, those results are flagged in order to encourage the learner's persistence (rule N10). For confident students, on the other hand, the tutor comments on the lack of effort (the tutor does get a bit demanding sometimes) and offers help. Obviously the learner may insist on abandoning the task anyway by rejecting the tutor's help, in which case the tutor moves to a new problem as the domain-based planner determines, because the motivational tactic *respect control* will be generated and prevents the tactic *encourage effort* being included in the motivational plan again.

The last six rules concern whether to intervene to help the student succeeding with the task or to skip interruptions at all, as stated in Table 6. Rule N11 simply disregards the help delivery present in the domain-based plan in order to avoid interventions as decided by the motivational planner. Therefore help is not delivered in that moment of the interaction. If the motivational plan includes the tactic *encourage independence* as well (rule N12), the tutor encourages the learner to work without support or suggests that help will be delivered later (after the learner's *indep-value* model rises above the *indep-threshold*). Table 6 also states that if the learner's confidence is low then help should be delivered and it should be detailed (*help-detail specific*), which is accomplished by rule N13. On the other hand, if help is needed and the student's feelings of independence are reassured, the tutor suggests help rather than directly intervening (rule N14). One should notice that rule N14 is also activated in the case presented in the previous paragraph, when the tutor *offers* help after complaining about the student's lack of effort. Finally, rule N15 states that the immediately next intervention should be avoided, respecting the learner's control. The action *skip-next-help* actually means that *help-skip-next* value is set to *yes*.

Table 18 - Negotiation planner

rule	DOMAIN-BASED PLAN action	MOTIVATIONAL PLAN tactic	NEGOTIATION PLANNER delete action	 add action
N1	suggest *problem type harder*	*increase experience success* not *stimulate challenge*	suggest *problem type harder*	suggest *problem type similar*
N2	suggest *problem type harder*	*stimulate challenge* not *increase confidence*	suggest *problem type harder*	suggest *problem type much-harder*
N3	suggest *problem type harder*	*emphasise promotion*	—	provide *comment level-promotion*
N4	provide *assessment type wrong* suggest *problem type same-diff*	*facilitate success*	provide *assessment type wrong* suggest *problem type same-diff*	provide *help content next-step*
N5	provide *assessment type wrong* suggest *problem type same-diff*	*stimulate curiosity*	provide *assessment type wrong* suggest *problem type same-diff*	provide *help content surprise-result*
N6	provide *assessment type right*	*maintain effort*	—	provide *comment praise-perf*
N7	provide *assessment type wrong*	*maintain effort*	—	provide *comment praise-effort*
N8	suggest *problem* not provide *assessment*	*facilitate success* not *respect control*	suggest *problem*	provide *help content next-step*
N9	suggest *problem* not provide *assessment*	*encourage effort*	suggest *problem*	provide *comment trying-harder* suggest *help content next-step*
N10	—	*remind successes*	—	provide *comment previous-successes*
N11	provide *help*	*avoid intervention*	provide *help*	skip *help*
N12	—	*encourage indep*	—	provide *comment encourage-indep*
N13	provide *help detail general*	*increase confidence*	provide *help detail general*	provide *help detail specific*
N14	provide *help*	*share control*	—	suggest *help*
N15	(*help not-needed*)	*avoid future intervention*	—	skip *next-help*

Application to a concrete domain

The formalisation and implementation of motivational tactics described in this paper made use of domain independent elements (generic *problem, help, answer*, etc.). However, evaluating the motivational planner requires its application to a concrete domain. A simple tutor for teaching Prolog debugging was designed and implemented with the purpose of being a "vehicle" for MORE[10]. In this sense, the tutor described here is simply a illustrative example of how MORE interferes in the behaviour of a tutoring system, providing the means to evaluate the motivational planner potentialities. This prototype is not meant to "compete" effectually (in domain terms) with purpose-built Prolog debugging tutors such as TADP (Brna et al., 1993).

The *problems* in the Prolog-debugging tutor consist of Prolog programs with bugs and the task for the student is to find and correct the bugs[11]. In this implementation the set of programs is limited to very simple programs, and each problem contains only one bug. The solution for a problem in the domain space is the correct version of the respective program.

Examples of bugs are a variable starting with a lower-case letter, a mistyped functor, or a wrong argument in a clause. Each of these bugs presents many distinct possible instances, even when one considers the application of the bug to one single program.

The level of difficulty of the *problem* depends on the complexity of the program combined with the degree of difficulty of the bug. In this work, the complexity of a Prolog program was defined according to Gegg-Harrison's schemata (Gegg-Harrison, 1989). The degree of difficulty of bugs, on the other hand, is not as well determined as the complexity of programs. For the purposes of the limited domain representation in this tutor, we assume that a bug of a syntactic nature, such as lower-case variable, is "easier" to detect than a semantic bug, such as a wrong argument in a clause. This assumption originates from the idea that syntactic bugs may be noticed without the need of running the program. Besides degree of difficulty, the other property for *problems* is similarity. *Similar problems* in the Prolog debugging domain consist, for example, of buggy programs generated by the application of the same bug to different programs of the same degree of difficulty.

Preliminary formative evaluation studies were performed with subjects who volunteered to interact with the Prolog-debugging tutor. The subjects were asked to report their motivational state during the interactions (e.g. level of confidence) and the interactions were recorded on video tape (both sound and screen image) through a scan converter. A more detailed discussion about the data gathered in this experiment is provided in (del Soldato, 1994). Nevertheless, this consisted of a simple study to help designing forthcoming (and more complex) evaluation studies.

Here we present several examples of typical interactions with the Prolog-debugging tutor, and discuss some questions raised from the preliminary evaluation study.

---

[10] The tutor was implemented in Pop11 and Prolog, within the Poplog system. For a more detailed description of the tutor discussed here, see (del Soldato, 1994).

[11] A more elaborate tutor would embody a theory of debugging and teach that theory through the student's experience of debugging programs.

Student succeeds, with little effort but low confidence:

This case was discussed in Table 2: the problem was "easy" to solve, so the focus of the interaction can shift towards the next level of difficulty, but because the student is not confident the system comments on the level promotion.

Had the student been feeling more confident, the system would have highlighted the increasing difficulty of the next task in a more challenging way (e.g. "The next problem will be much harder!"). One subject, who was continuously challenged by the tutor, reported being particularly stimulated by such comments ("It makes me feel more *interested* about the next problem ... and I don't need to check myself if the problem is too easy, the tutor tells me").

Table 19 - Level promotion

dialogue	student model	instructional plan
S — *(promptly corrects the program)*	conf = 4 (low) effort = little	
S — There is no bug in the program.	conf = 4 effort = little	provide *assessment right* comment *level-promotion* suggest *prob harder*
T — Right answer. This looks easy for you now, it's time to move to harder problems. How about this program? *...suggests harder problem*	conf = 5 (ok) effort = little	

In the example provided in Table 19 (here "S" stands for "student" and "T" for "tutor") one can notice how the confidence model is increased by the right answer acknowledgement. During the evaluation study, one subject provided a clear example of confidence raised by successes. When the next problem was presented, the subject spotted the bug at once (it was a syntactic bug). Nevertheless, she decided to go through with the exercise, exclaiming: "I think it is easy, but I want to do it. It makes me feel good!".

Another interesting question about confidence modelling was raised by reports from some subjects about their confidence on what *they were doing at that moment*, as opposed to the idea of *global* confidence on their abilities. For example, subjects expressed comments such as, e.g., "I am not sure why I am doing this" while performing a particular step, showing low confidence on a *local* situation, even if their global confidence about the eventual success in performing the whole task was not (apparently) affected. In other words, there were situations where a distinction raised between *global confidence* ("I don't know if I can solve this problem") and *local confidence* ("I don't know if this query is a good one"). This suggests that further research on confidence modelling could explore the distinction and correlation between local and global confidence, and how this affects instructional planning (e.g. should a "high global-confidence" student be offered help in a case of "low local-confidence"?).

Student fails, effort large:

This case was discussed in Table 4. The tutor acknowledges the learner's effort, even if the performance's result was not satisfactory (Table 20). One should note that in the case where the confidence model is a value below the *conf-threshold* (which here is set as 4), the tutor does not provide the performance assessment and offers help instead, insisting on the same problem (Table 21).

Table 20 - Praising effort

dialogue	student model	instructional plan
S — *(modifies and tests the program)*	conf = 5 (ok) effort = large	
S — There is no bug in the program. *(wrong answer)*	conf = 5 effort = large	provide *assessment wrong* comment *praise-effort* suggest *prob same-diff*
T — Wrong... But it was a good effort! How about *...suggests problem same-diff*	conf = 4 (low) effort = large	(wrong answer assessment: *conf value* is decremented)

Table 21 - Insisting on same problem

dialogue	student model	instructional plan
S — *(modifies and tests the program)*	conf = 4 (low) effort = large	
S — There is no bug in the program. *(wrong answer)*	conf = 4 effort = large	provide *help specific* comment *praise-effort*
T — Have a look at clause 1. You have tried hard. *(insists on same problem)*	conf = 4 effort = large	

Student gives up, effort little:

If the student abandons the task after little effort, the tutor also insists on the same problem, and offers (or directly provides) help, as discussed in Table 5. Here we show two examples of such cases. The interaction reproduced in Table 22 presents the situation where a less confident student gives up the task, although the overall performance has been good (many tasks satisfactorily accomplished). The tutor can then evoke the previous successes to encourage the learner's persistence in solving the task.

Table 22 - Reminding successes

dialogue		student model	instructional plan
S —	*(works only a little)*	conf = 4 (low)	provide *help specific*
	I give up...	effort = little	comment *prev-successes*
		perf = good	
T —	Have a look at clause 1.	conf = 4	
	You are doing fine,	effort = little	
	keep trying.	perf = good	

In interactions with more confident students, as presented in Table 23, the system "demands" further effort in a more direct way, without the need for extra encouragement. However, subjects were not really pleased with such comments, suggesting that the expressions included in this implementation for the *trying-harder* type comments could be replaced by more supportive expressions.

Table 23 - Encouraging effort

dialogue		student model	instructional plan
S —	(works only a little)	conf = 6 (ok)	comment *trying harder*
	I give up...	effort = little	suggest *help next-step*
T —	Maybe you can try a bit harder.	conf = 4	
	May I help you?	effort = little	

A similar situation occurred in the preliminary evaluation study, when a confident but bored[12] subject abandoned finding the bug after little effort. The tutor "ignored" the request to abandon the task, providing a hint instead, which annoyed the subject who kept working, mumbling that the tutor was not "letting him leave". However, another subject in the same situation was quite pleased to be offered a hint after having given up performing the task. The disparity between these two reactions shows that students can abandon a task for different reasons: the first subject gave up performing the task because he was annoyed with the task, whereas the second subject wanted to complete the task but was not able to progress further. Maybe a better option for the second subject would have been to request help, so the tutor coped quite well with the situation by offering a hint. However, it is clear that motivational planning should also consider the case of "bored students" giving up.

Student fails, confidence ok, effort little

If the student fails after little effort, the tutor also insists on the same problem and tries to stimulate the learner's curiosity, as discussed in Table 4. The actual behaviour of the tutor in such a situation is showed in Table 24. In the case reproduced here the buggy program being investigated by the learner was:

[12] At the very start of the interaction he exclaimed in his first language the equivalent to the English expression "How boring...".

member(X, [X | Tail]).

member(X, [Head | Tail]) :- member(X, tail).

(Checking the head member of a list does not reveal the bug, whereas checking any element in the tail of the list results in a buggy query).

Table 24 - Stimulating the learner's curiosity

dialogue		student model	instructional plan
S —	?- member(1, [1, 2, 3]).	conf = 5 (ok) effort = none	
T —	Solution for this query: yes	conf = 5 effort = little	
S —	There is no bug in the program. (wrong answer)	conf = 5 effort = little	provide *help surprise-result* (insists on same problem)
T —	The solution for ?- member(b, [a, b, c]) may surprise you...	conf = 5 effort = little	

Students request help, confidence low/ok:

Comparing the tutor's reaction to help requests when the *conf-value* is above or below the *conf-threshold*, one can note the different level of detail on the hints provided to the learner. The less confident student is given a more direct hint (Table 25) whereas the confident learner is given a more general hint (

Table 26). In the former case the problem is the buggy problem

last (Item , [Item]).

last (Item , [First | Rest]) :- last (First , Rest).

(where the bug is in the second clause) and in the confident learner case the buggy program is

last (Item , [Rest]).

last (Item , [First | Rest]) :- last (Item , Rest).

The independence model is not considered in the "low confidence" case, as priority is given to restoring the learner's confidence. For the confident student, the requested hint is provided since the *indep-value* is above the *indep-threshold*. (In the next paragraph we discuss the situation where the *indep-value* is below the threshold). Generic hints were particularly appreciated by one of the subjects ("...it gives you some direction, but still lets you work").

Table 25 - Example of specific hint

dialogue	student model	instructional plan
S — Help, please	conf = 4 (low)	provide *help specific*
T — Hint: Have a look at clause 2	conf = 4	

Table 26 - Example of general hint

dialogue	student model	instructional plan
S — Help, please	conf = 6 (ok) indep = 5 (ok)	provide *help general*
T — Look at the original problem: there is a wrong argument somewhere	conf = 6 indep = 4 (low)	(help is provided: *indep value* is decremented)

Student requests help, independence low:

If the learner has already requested hints too many times, or the tutor has excessively intervened on its own (if the *conf-value* was low), resulting in a low value for the independence model, the student's help request will not be satisfied at this moment and postponed to a later help request or situation in which the student is lost (Table 27). This situation was discussed in Table 6.

Table 27 - Skipping help

dialogue	student model	instructional plan
S — Help, please	conf = 6 (ok) indep = 4 (low)	skip *help*
T — I will help you later.	conf = 6 indep = 6 (ok)	(*indep value* is incremented)

It is interesting to note that subjects were in general quite annoyed when the tutor would "refuse" to provide help. Although the tactic of skipping help concerns the students' independence, what seemed to disturb the subjects was the machine's independence. One subject made a particular remark: "I want to feel I am in control of the machine, and if I ask for help I want the machine to give me help". When reminded that human teachers happen to encourage students' independence the subject answered: "But a human teacher knows when to skip help. I interact with the human teacher but I want to be in control of the machine"[13]. Further evaluation should investigate if individuals with minimal experience with computers feel less prompted to control the tutor, as this was not possible to determine in this experiment using the strongly computer-related domain of Prolog-debugging. Because the relationship between a student and a human teacher is

[13] It is important to mention here that the subjects in the evaluation study, and this subject in particular, were mostly Ph.D. students used to work with (and control) computers.

necessarily of a different quality to that between a student and a "machine teacher", this raises the issue of whether tactics which work well in a human-human interaction will always translate into the human-machine context.

Conclusion

This paper presented:

1. a comparison between the behaviour of a typical domain-based instructional plan and a hypothetical "motivational planner", which reacts according to a set of motivational tactics originated from theories of instructional motivation;
2. the elaboration and formalisation of techniques to detect the learner's state of confidence and independence and the degree of effort spent in every task (the motivational modeller);
3. the formalisation, as a set of production rules, of the reactive motivational tactics previously discussed (the motivational planner);
4. the elaboration and formalisation of a set of instructional primitives (*prob*, *help*, *answer*, etc.), representing the instructional interaction through objects and properties which are manipulated by the motivational modeller and the motivational planner;
5. the elaboration and formalisation of a simple performance modeller and a basic domain-based planner;
6. the elaboration and formalisation of mechanisms to negotiate between the motivational plan and the domain-based plan (the negotiation planner).

Although the tactics selected to be included in the motivational planner were extracted from the Instructional Science literature, many features involved in the actual formalisation/implementation had to be elaborated based on the characteristics and limitations of current ITSs. For example, the techniques to model the student's motivational state were largely constrained by the limitations of typical interface devices.

The actual performance of the motivational planner is shaped (and limited) by the set of instructional objects presented here and the efficiency of the motivational modeller. Moreover, the whole behaviour of the system relies on the priorities defined in the negotiation planner and the precision of the performance modeller, since the application of motivational tactics depends both on the motivational state and the performance state.

The formalisation of the system in terms of explicit and variable parameters (such as the thresholds for confidence and independence state, for instance) allows a degree of manipulation of the motivational features, which can override the priorities set in the motivational and negotiation planner. Coping with other limitations of the resulting system would require, however, the inclusion of extra features in the set of instructional objects. For instance, this work focusses on aspects of motivational tactics concerning problem-solving-based domains. A richer domain representation, including e.g. a wider variety of links between topics, would provide space for further formalisation of motivational tactics: for example, the formalisation of curiosity-related tactics based on the delivery of incomplete information in order to stimulate the learner seeking the "missing" topics.

It was mentioned previously in this paper that several systems include an *implicit* theory of motivation. In particular we mentioned WEST's pedagogical tactic of preventing two

interventions in sequence (apparently WEST assumes that the learner's independence decreases after the first intervention). MORE, on the other hand, deals with the excess of interventions by having an *explicit* independence model, which dynamically decreases after any intervention from the tutor. In other words, WEST avoids two interventions in sequence because it could affect the learner's feeling of independence, whereas MORE represents the learner's independence through an explicit model and prevents excessive interventions when the *independence value* in the model is considered *low*. In most cases, MORE's behaviour will result in avoiding a sequence of two interventions. There are cases, though, when the learner's independence model is so low that even the tutor's first intervention should be skipped, or cases when the learner's feeling of independence is very stable and two interventions do not compromise an excess of control from the tutor. The explicit independence model represented in MORE allows the motivational planner to *reason* about the learner's feeling of control, instead of reducing to a single and automated behaviour the tutor's effort to encourage the learner's control.

The explicit distinction between the motivational and the domain-based planners provides an opportunity to insert the motivational planner in current systems, although it is necessary to investigate in the future whether the domain-based and the motivational tactics should actually be represented in separated modules, or if merging both planners into a unique integrated planner would result in a more efficient system.

References

Arshad, F. (1990). *The Design of Knowledge-based Advisors for Learning*. Ph.D. thesis, School of Education, University of Leeds, UK.

Brna, P., Hernandez, E. R., & Pain, H. (1993). Learning Prolog debugging skills. In *Proceedings of PEG'93* (pp. 561-568). Edinburgh: Scotland, UK.

Burton, R. R., & Brown, J. S. (1982). An investigation of computer coaching for informal learning activities. In D. H. Sleeman & J. S. Brown. (Eds.), *Intelligent Tutoring Systems* (pp. 79-98). London: Academic Press.

Carbonell, J. R. (1970). AI in CAI: An artificial intelligence approach to computer-assisted instruction. *IEEE Transactions on Man-Machine Systems*, 11(4), 190-202.

Clancey, W. J. (1982). Tutoring rules for guiding a case method dialogue. In D. H. Sleeman & J. S. Brown (Eds.), *Intelligent Tutoring Systems* (pp. 201-225). London, Academic Press.

del Soldato, T. (1992a). Motivational planning. In P. Brusilovsky & V. Stefanuk (Eds.), *Proceedings of East-West Conference on Emerging Computer Technologies in Education* (pp. 293-298). International Centre for Scientific and Technological Information, Moscow, Russia.

del Soldato, T. (1992b). Detecting and Reacting to the learner's motivational state. In C. Frasson, G. Gauthier & G. I. McCalla (Eds.), *Intelligent Tutoring Systems: Proceedings of ITS'92 - Montréal, June 1992* (pp. 567-574). New York: Springer-Verlag.

del Soldato, T. (1994). *Motivation in Tutoring Systems*. Tech. Rep. CSRP 303, School of Cognitive and Computing Sciences, University of Sussex, UK.

Gegg-Harrison (1989). Basic Prolog Schemata. Tech. report, Department of Computer Science, Duke University, Durham, NC 27706, USA,

Keller, J. M. (1983). Motivational design of instruction. In C. M. Reigeluth (Ed.), *Instructional-Design Theories and Models: An Overview of their Current Status* (pp. 386-434). Hillsdale, NJ: Lawrence Erlbaum Associates.

Lepper, M. R., Aspinwall, L. G., Mumme, D. L., & Chabay, R. W. (1990). Self-perception and social-perception processes in tutoring: Subtle social control strategies of expert tutors. In J. M. Olson & M. P. Zanna (Eds.), *Self-Inference Processes: The Ontario Symposium*, Vol. 6 (pp. 217-237). Hillsdale, NJ: Lawrence Erlbaum Associates.

Lepper, M. R., & Chabay, R. W. (1988). Socializing the intelligent tutor: Bringing empathy to computer tutors. In H. Mandl & A. Lesgold (Eds.), *Learning Issues for Intelligent Tutoring Systems* (pp. 242-257). New York: Springer-Verlag.

Lepper, M. R., Woolverton, M., Mumme, D., & Gurtner, J. (1993). Motivational techniques of expert human tutors: Lessons for the design of computer-based tutors. In S. P. Lajoie & S. J. Derry (Eds.), *Computers as cognitive tools*. Hillsdale, NJ: Lawrence Erlbaum Associates.

Malone, T. (1980). What makes things fun to learn? A study of intrinsically motivating computer games. Tech. rep. CIS-7 (SSL-80-11), Xerox Palo Alto Research Center, Palo Alto, CA 94304, USA.

Malone, T., & Lepper, M. R. (1987). Making Learning Fun. In R. Snow & M. Farr. (Eds.), *Aptitude, Learning and Instruction: Conative and Affective Process Analyses*, pp. 223-253. Hillsdale, NJ: Lawrence Erlbaum Associates.

Peachey, D. & McCalla, G. (1986). Using planning techniques in intelligent tutoring systems. *International Journal of Man-machine Studies*, 24, 77-98.

Shuell, T. J. (1992). Designing instructional computing systems for meaningful learning. In P. H. Winne & M. Jones (Eds.), *Adaptive learning environments*. New York: Springer-Verlag.

Schunk, D. H. (1989). Self-efficacy and cognitive skill learning. In C. Ames & R. Ames (Eds.), *Research on motivation in Education: Goals and Cognitions*, Vol. 3. London: Academic Press.

Wasson (Brecht), B. (1990). *Determining the Focus of Instruction: Content Planning for Intelligent Tutoring Systems*. Ph.D. thesis, Department of Computational Science, University of Saskatchewan, Canada.

Woolf, B. (1984). *Context-dependent Planning in a Machine Tutor.* Ph.D. thesis (COINS Tech. Rep. No. 84-21), Department of Computer and Information Science, University of Massachusetts, USA.