Bypassing the Intractable Problem of Student Modelling

John Self

1990

AAI/AI-ED Technical Report No.41

(in C. Frasson and G. Gauthier (eds.), *Intelligent Tutoring Systems: at the Crossroads of Artificial Intelligence and Education*, 107-23, Norwood, N.J.: Ablex)

Computing Department
Lancaster University
Lancaster
LA1 4YR
UK

aai@comp.lancs.ac.uk
http://www.lancs.ac.uk/computing/research/aai-aied/
ftp.comp.lancs.ac.uk (148.88.8.9) /pub/aai/

BYPASSING THE INTRACTABLE PROBLEM OF STUDENT MODELLING

John A. Self

Computing Department

Lancaster University, Lancaster, LA1 4YR, U.K.

Abstract -- This paper attempts to rehabilitate student models within intelligent tutoring systems. Recently, some researchers have questioned both the need for detailed student models and the practical possibility of building them. We regard it as axiomatic that any *intelligent* tutoring system needs a student model. This paper suggests some practical guidelines and changes in philosophical approach which may help in building effective student models.

1. Introduction

In a review of the 1987 "Artificial Intelligence and Education" Conference, Sandberg (1987) summarised a general opinion that

> "detailed user models do not necessarily enhance the capability of an intelligent tutoring system ... good teaching can do without a detailed user model, because in good teaching serious misconceptions are avoided, and errors will be repaired on the spot ... it is debatable whether the cost of constructing very

detailed, complex user models that are runnable and have to be maintained all the time is worthwhile in terms of the gain in teaching efficiency."

The aim of this paper is to rehearse the arguments for student models in intelligent tutoring systems and to present a less bleak prognosis of the possibility of actually constructing them.

Opinions such as the above derive from two sources:

1. preconceptions about the potential roles of student models;
2. theoretical and practical difficulties in building and using student models.

For the former, we offer some alternative, possibly more productive, views of the potential roles of student models; for the latter, we describe some more realistic, practically achievable and useful goals for student modelling. In the next section, the student modelling problem is reviewed and in the following section some possible ways forward are suggested.

2. The Student Modelling Problem

A pure version of the student modelling problem might be the following:

Tutor: What is the integral with respect to x of

$$x^4 / (1 + x^2)$$

Student: $x + x^3 / 3 + \tan x$

Tutor (thinks: How did she get that?): .....

A standard ITS approach might be to identify the set of allowable transformations and to study student protocols to build a catalogue of associated mistransformations. If there are m possible transformations (m ≈ 30 in this case), n mistransformations for each of these (say, n ≈ 5) and up to p steps to a solution (say, p ≈ 10), then there may be up to $(m*(n+1))^p$ paths to analyse (about 10^{24}), which is clearly intractable.

We can, of course, eliminate the combinatorial explosion if we ensure that p=1, as in the Lisp tutor (Anderson and Reiser, 1985), constraining the student to the smallest analysable step, with the consequent imposition of a rigid tutorial style. However, we still have the considerable difficulty of determining an appropriate 'grain' of detail in defining the (mis)transformations. It is relatively easy to interrupt a student if she appears to be putting the clauses of a COND in the wrong order, but much harder to realise that she is doing so because she has confused the 'if-then-else' semantics with a 'production-system-like' semantics, where all the true conditions would be simultaneously acted upon.

Advocates of student models would wish to go beyond the analysis of student performance in terms of surface mistakes. They would like to isolate the underlying misconceptions which are the 'cause' of the mistakes, because remedying such misconceptions might eradicate a whole set of mistakes. But defining, representing and recognising such misconceptions is even more difficult than identifying a procedural mistake.

And then, some would say, the p=1 restriction and the usual ITS analysis of student-input on a sentence-by-sentence basis represents a very weak view of student problem-solving. Students do not always solve problems as finite automata, responding only to the state they are in - they develop goals, plans and strategies. And indeed they should, and our ITS student models should ideally include descriptions of these, to permit discussions with students at this level. Unfortunately, such goals are often mal-formed, idiosyncratic and

difficult to identify automatically. For example, Unix experts know all sorts of hacks for achieving goals not normally associated with the Unix commands actually used.

This would imply that our student model needs access to quite specific information about the student's prior knowledge. This prior knowledge may be such that an ITS would not have any *a priori* reason for associating it with the topic of the tutorial. For example, Shrager's studies of how people learned by experimentation how to operate a programmable toy showed that they drew analogies with other programmable devices, clocks, and so on (Shrager, 1987). It is difficult to imagine ITSs having access to commonsense knowledge of clocks and thousands of similar objects just in case a student should happen to draw an analogy with them.

We know also that students' learning is influenced not only by their general prior knowledge but also by the more immediate learning context. For example, students attempting physics problems draw surface analogies with immediately preceding problems (Kolodner, 1983). This, of course, implies that student models should ideally maintain an episodic memory in order better to provoke productive analogies and to understand the source of mistaken analogies.

And not only will students have different prior knowledge and learning experiences, but they will have personal learning preferences, styles and strategies. Again, ideally these should be modelled, so that an ITS may present material in a way appropriate to the individual's learning abilities and perhaps to address weaknesses in those abilities.

Perhaps the student has particular interests, or an unusual social background, or some personality characteristic, which, if it were represented within the student model, might be used to individualise the instruction.

In this way, the 'student modelling problem' expands - from computational questions, to representational issues, through plan recognition, mental models, episodic memory to individual differences - to encompass, it would seem, almost all of cognitive science. One reaction to this is to conclude that ITS research, and especially student modelling, is important precisely because many fundamental cognitive science questions have to be addressed.

Another reaction is to conclude that the student modelling problem is overwhelming difficult, that with the current state of knowledge there is no possibility of satisfactorily addressing most of these cognitive science questions, and therefore that ITS development had best proceed without student models at all. I hope to show that this last conclusion is not justified.

3. Bypassing the student modelling problem

It is not essential that ITSs possess precise student models, containing detailed representations of all the components mentioned above, in order to be able to tutor students satisfactorily. If we back off from the grand vision and adopt more realistic aims, then solutions for some aspects of the student modelling problem are practically attainable and useful. This approach will be described under four 'slogans':

Slogan 1: Avoid guessing - get the student to tell you what you need to know

The problem illustrated above with the symbolic integration question - that is, of inducing a student's (mis)conceptions from a problem-solution pair - may be, for ITS, a non-problem. Using pencil and paper, students do indeed occasionally, through bravado or a misplaced sense of style, write solutions down 'by inspection', although this is very much against the recommendations given in modern classrooms. And with the older teletype-like interfaces to ITSs, students may well have been sorely tempted to omit intermediate steps to avoid laborious, error-prone typing. But imagine how a student would solve symbolic integration problems with a modern WIMP interface.

She might be presented with a menu of transformations to apply. With any transformation (e.g. integration by parts), she might be able to map the general form of the transformation onto the specific example by selecting parts of the example using the mouse, for example, to let $u = 1 / (1 + x^2)$ of the above example. Maybe there would be a 'do_it' key to ensure there were no clerical slips. Perhaps the steps of the solution would be displayed in an appropriate tree structure, making it possible to see how the various steps relate to one another and to return to previous steps, if desired.

The process of problem-solving is so much easier than with older interfaces (and with pencil and paper) that we may expect students to be much less inclined to omit intermediate steps. But, more importantly, with careful design the interaction may provide the ITS with precisely the information that it needs to model the problem-solving process, e.g. to understand how the student proposes to apply integration by parts - information that is often difficult to infer even from a complete step-by-step solution. Ideally, the information is provided by the student, naturally and voluntary, while problem-solving (and not in response to some interventionist ITS).

So, rather than attempting to develop better ways of inferring missing steps, a better goal for student modelling research might be to design interactions through which the information needed for building student models is provided non-intrusively as an intrinsic part of problem-solving.

It might immediately be objected that this would only provide us with information about individual steps and not information about the 'goals and strategies', which we have indicated it is more important and useful to model. An experiment reported by Singley (1987) shows that this is not necessarily so.

He designed a system to help students learn to solve algebra word problems about rates of change. He provided a menu of available operators and a solution window. With the first version of the system it was found that students performed badly at solving problems, mainly because they became muddled as to where they were. As a result, a 'goal window' was included in which a student was required to 'post' the goal(s) she was working on before selecting operators. For example, in Figure 1(a) the "dp/dt : t" at the root of the tree denotes that she is trying to find dp/dt in terms of t. The 'chain rule' is the proposed operator which, before it can be applied, needs two sub-goals to be satisfied, namely "dp/ds : c" and "ds/dt : t". An open box around a node indicates the goal currently being worked on, and a closed box (Figure 1(b)) indicates that a goal has been satisfied.

<Figure 1 about here>

The hypothesis is that, for example, when she sees the two closed boxes then she will have less difficulty in recalling that she was trying to apply the chain rule, which will improve problem-solving and learning performance. But it may be that the overhead of 'goal-posting' actually interferes with problem-solving. Experimental studies showed that, under the conditions Singley specified,

(a) goal-posting improved problem-solving performance, in that operator selections were made faster, there were fewer 'illegal' selections, and the solutions found were shorter;

(b) performance was improved even after the goal window was removed - i.e. students had not just benefitted from a temporary technological crutch but had learned some problem-solving skills;

(c) the improvement transferred to other kinds of (admittedly similar) problem, indicating that the skills learned were not problem-specific tricks but were more generally useful.

One possible conclusion from this experiment is that we should include a goal window in our problem-solving environments and that this will facilitate student learning, thereby reducing the need for any 'intelligent tutoring' and hence student modelling. However, the interpretation I would make is that it may be possible to design such environments so that students provide precisely the information ITSs need for student modelling and have difficulty in inducing from performance (namely, information about goals).

There are two main research questions (which my colleague Michael Twidale and I are investigating):

(a) how can the information provided by interactions with a goal window be used by an ITS to carry on discussions about the student's goals?

(b) what kind of language should be provided for the student to communicate her goals?

We have implemented a prototype logic tutor, a screen image of which is shown in Figure 2. As usual, we have a menu of operators (corresponding to rules of inference) and a solution window.

In addition, we provide a set of plan schema (one or more with each operator, indicating what goal the operator may help achieve, and some, e.g. "contradiction", which are independent of particular operators). Pointing to a plan schema shows an abstract form of the plan, which can be instantiated to the problem at hand if the student wishes (in the Figure the student may be about to instantiate the X of the plan schema to Q and the Y to R). Selecting a plan causes the instantiated form of the plan to appear in the 'goal window'. The student may then select components of this plan to work on, which may of course call up further sub-plans. The student's progress through the plans is automatically marked in the goal window (the tick indicating achieved plans, the arrow pointing to the current goal).

<Figure 2 about here>

The student's selection and use of plans is monitored by the system. The system may intervene if a plan is deemed inappropriate, writing a message in the 'message window'. As usual, it is difficult to say precisely what use an ITS should make of its knowledge of student goals, when it should intervene and how it should express its interventions.

The use of plan schemas is not a general solution to the problem of providing a language through which the student may express her goals. We had previously experimented with the use of a menu of (about 30) stock phrases (e.g."in order to", "it follows that", etc.) from which the student could select and build up a natural language plan of any desired complexity. We believed that the selection of phrases would considerably ease the parsing problem. It turned out that students could so express plans in this domain, and that they could be parsed relatively easily, but that there were unfortunately too many ambiguous phrases (e.g. "from x and y", meaning 'from lines x and y of the proof so far', or 'using proof rules x and y', and so on). However, we are optimistic that the technique of phrase selection to express logical arguments may be adequate in other domains. For example, we are looking at the possibility of using the technique to express causal arguments in

economics, where clauses are usually of the form "x leads to y", with numerous paraphrases.

Slogan 2: Don't diagnose what you can't treat

The grand ambition to build high-fidelity student models can easily obscure the fact that, in practical terms, student models by themselves achieve nothing. Student models are merely data for the tutoring component of ITSs. It follows that there is no practical benefit to be gained from incorporating in our student models features which the tutoring component makes no use of, and there is no point the ITS labouring to identify such features.

However, this assumes that our tutoring component is given and that we have only to inspect its code to discover the features that our student model needs to identify. This is not so: our understanding of tutoring expertise is not deep and it is indeed a prime function of student modelling to drive this understanding. The division of an ITS into the standard 'subject, student, tutor' modules is an explicatory device, not a guideline for ITS implementation. Mobus and Thole (1988) seem to regret the fact that "their identification in current systems is difficult or even impossible to obtain because their knowledge bases are often not as clearly separated as theory postulates." In my view, the student model and tutoring procedure should be developed in tandem, not separately. Any proposed feature of a student model should be explicitly linked with a proposed processing of it by the tutoring component, and ideally this processing should be linked with existing educational evidence which justifies it.

One implication of this suggestion is that the development of the tutoring component is no longer to be left as an intuitive afterthought. Its contents are to be brought out into the open, so that we may assess their implications for student modelling and their educational rationale. This, indeed, is happening. For example, Clancey (1987), Mizoguchi, Ikeda and Kakuso (1988) and Woolf (1988) all give details of tutoring procedures. In general, the implications are that, in practical terms, we can be much less demanding of our student models and that the educational rationale is somewhat ad-hoc.

Imposing the <feature : processing : evidence> test, suggested above, would, I suspect, render many of the discussions on student models irrelevant. For example, the original genetic graph proposal (Goldstein, 1982) suggested overlaying a description of the student's "learning preferences" on the links, where these preferences seem to be described in terms of "his need for repetition, his degree of forgetfulness, and his receptivity to advice." What use the tutoring component was to make of this information, and on what basis, was left to intuition.

The same difficulty arises with most proposals to incorporate some description of 'learner styles' in student models. These proposals invariably quote Pask's holist/serialist distinction (Pask, 1976) but are quite unable to point to any other learner styles which can be reliably identified by ITSs and which can be associated with demonstrably effective differential treatments by ITSs.

As regards the idea that other individual differences, describing intellectual abilities, cognitive styles, academic motivation and personality characteristics, should be represented within student models, then the literature on aptitude-treatment interactions, reviewed by Corno and Snow (1986), is not encouraging. They conclude that further research is needed to identify likely aptitude variables and that, in any case, "practical systems can use no

more than a few control variables for adaptive purposes" (e.g. two, perhaps intelligence and motivation, for human teachers).

Slogan 3: Empathise with the student's beliefs, don't label them as bugs

The general perception, reflected in the Sandberg quote above, that student models in ITS are for *remediation* presents a serious philosophical problem. It is the arrogant, 'tutor knows best' style of ITSs which alienates classroom teachers more than any technical shortcomings. The standard ITS approach of first defining a body of certified knowledge and then devising ways to correct students' understanding so that it conforms to it does not accord with the philosophies of epistemologists, with or without an educational orientation. For example, Piaget urged us to appreciate that a child's understanding was never merely wrong but that it made sense to him in his own terms, if we as teachers could but understand those terms. And philosophers of science, such as Popper, have argued that the history of the development of scientific knowledge demonstrates that we ought to regard all knowledge as provisional and potentially falsifiable.

The idea that student models are for remediation is implicit in both standard approaches to student modelling, the 'mal-rule approach' and the 'expert system overlay approach'. Here I will concentrate on the former.

In the mal-rule approach, two sets of productions are defined: a 'correct' set, which if applied to the problem at hand gives a correct solution, and a 'mal' set, which consists of deformed versions of the correct productions. If a member of the mal set is used instead of its associated correct rule, then an incorrect solution will (usually) be obtained. Of course,

the aim of the ITS is to eradicate any mal-rules which may exist in the student's head (anything 'mal' clearly needs treatment).

Apart from philosophical misgivings, there are practical and theoretical problems with this approach:

1. To label something as incorrect and in need of remediation, an ITS needs to know what is to be deemed 'correct'. Consequently, the mal-rule approach is, in principle, only possible where such a knowledge representation can be specified, i.e. in closed-world, formal domains (such as subtraction and Lisp programming), and even there there may be many equivalent effective representations.

2. A mal-rule is to be understood as one which is systematically applied instead of the correct rule. But how 'systematic' does a mal-rule have to be to qualify as a mal-rule? In an attempt to discover whether it is in fact the case that "there is great systematicity in the appearance of [algebra] errors" (Resnick, Cauzinille-Marmeche and Mathieu, 1987), Payne and Squibb (1987) analysed student errors in solving linear algebraic equations. While they considered that they could identify 99 different mal-rules, over 90% of them were used on less than 50% of the occasions when they could have been used (and by the original Brown and vanLehn (1980) guidelines would not have qualified as mal-rules at all).

3. The empirical data will indicate that some mal-rules are more common than others. It requires a theory of 'mal-rule generation' to explain why this is so. As Payne and Squibb (1987) point out, it is difficult to see how syntactic mechanisms (such as repair theory (Brown and vanLehn, 1980)) can explain why the transformation:

A: $Mx + N \rightarrow [M + N]x$

(e.g. $3x + 4 = 5x + 2 \rightarrow 7x = 5x + 2$)

is more common than:

B: $Mx + N \rightarrow [M + N]$

(e.g. $3x + 4 = 5x + 2 \rightarrow 7 = 5x + 2$)

[This is so even when there are multiple x's in the equation: the bias is to be expected when there is only one x.] The intuition that A is 'more sensible' amounts to a belief that student models need to include descriptions of (partially formed) conceptual knowledge in addition to purely procedural knowledge.

4. How consistent are mal-rules across different populations? Payne and Squibb (1987) found very little overlap between the most common mal-rules at three different schools (in fact, the 5 most common mal-rules at the three schools gave 13 different mal-rules). This suggests that theories of mal-rule generation need to take more account of educational experience and context than they currently do.

5. The developers of lists of mal-rules have many decisions to make. In an algebra study, for example, do we abstract over integers (but 0 and 1 are surely special cases), do we abstract over operators, do we consider permutations (e.g. Mx+N, N+Mx) the same, and so on? The tally of mal-rules depends on such decisions. Too many mal-rules leads to a vacuous theory and computational inefficiency: too few mal-rules obscures data which may be of theoretical importance.

6. In general, there is a difficult knowledge representation question which has been pre-empted by the mal-rule terminology. Production rules may be suitable for modelling procedural skills in closed, formal domains, but it is difficult to see how, even in principle, the mal-rule approach could be applied to address misconceptions deriving from inappropriate analogies, such as those identified by Shrager (1987), mentioned above.

7. Finally, and referring back to slogan 2, even if we could reliably identify mal-rules, what should an ITS do about them? According to Pintrich, Cross, Kozma and McKeachie (1986), "once a bug has been accurately diagnosed, an instructional prescription follows naturally", but recently, Sleeman (1987) has discovered that "even though [his system] has a model for a student's problem solving it has not so far proved possible to remediate very effectively."

If student models are not primarily for remediation, then what other role may they have? In a recent review (Self, 1988), I identifed twenty different uses that had been found for student models in existing ITSs. The largest class of uses was indeed to do with remediation - but the next largest was what I called 'elaborative', i.e. to do with leading the student to elaborate or refine his current knowledge, not necessarily because a 'bug' had been identified and not necessarily towards some pre-specified target knowledge. This is the role that I would encourage for student models.

Sandberg's assertion that "good teaching can do without a detailed user model, because in good teaching serious misconceptions are avoided" is doubly misleading. First, good teaching involves much more than the avoidance of misconceptions. Secondly, teaching activities other than remediation, e.g. provoking a student to question her own beliefs, would benefit from having a detailed user model.

To escape from the remediationist view of student models, it may perhaps help if we consider that student models are intended to describe not what students 'know' but what they 'believe'. The former encourages ITS designers to impose value judgements about the correctness or otherwise of this 'knowledge'. Beliefs, on the other hand, are always provisional and liable to be changed if their justifications are seen to be inadequate or their implications seen to be unreasonable. The student modelling problem then becomes one of identifying what a student believes, and if possible why, the beliefs being represented in their own terms, not with respect to some target knowledge. The role of student models would then be to help ITSs to provoke students to consider the justifications and implications of their beliefs.

Slogan 4: Don't feign omniscience - adopt a 'fallible collaborator' role

The proposed change of ITS style, from knowledgable remediator to empathetic belief elaborator, is not made solely on philosophical grounds. There are also practical reasons for such a change.

For most of the subjects which we would like our ITSs to address, describing a 'correct' knowledge base, together with an adequate description of potential student misconceptions, is a practical impossibility. To attempt to conceal this under a facade of omniscience is a risky business.

Most studies of human tutoring that I have seen have concentrated on elucidating how a tutor makes pedagogic use of his own subject matter knowledge. It would be interesting to know how human tutors help students in situations where they lack subject matter knowledge. (My own experience is that this situation prevails over the first!)

Clearly it is possible for tutors to respond appropriately to questions for which they do not know the answers: "Why are cooling towers the shape they are?" "Are all modern aeroplanes powered by turbines?" "How can mercury be a metal?" and so on. We can respond in a variety of ways, such as giving suggestions about where to find relevant information, joining in an attempt to answer the question by reasoning from shared 'common sense' knowledge, probing the student's preconceptions behind the question, and so on. In many ways, these would be better responses than direct answers even if we were able to give them, because, of course, they go beyond mere factual knowledge to address problem-solving and learning skills which are, one hopes, of general utility. Is it possible for an ITS to play a similar role?

We have been toying with the idea of building an ITS which deliberately does not have the knowledge which the student is endeavouring to acquire. For example, to take a somewhat artificial situation, imagine that the student and the ITS have access to a database giving details of the properties of all the chemical elements, and that the student is trying to discover why certain elements are called 'metals', some 'nonmetals' and others 'semimetals'. The ITS would not itself know the rules (if any) which map sets of properties onto these classifications. The student may ask for specific information ("What is the boiling point of mercury?"), probe hypotheses ("Are all metals solid?"), offer thoughts ("Maybe all nonmetals are soft"), make strategic comments ("Let's look at magnetic properties"), ask for specific help ("List the boiling points we've seen so far"), seek general help ("What next?"), etc.

The role of the ITS is to act as a collaborator in this endeavour, by giving (or indeed taking) strategic advice, derived not from its prior knowledge of the concepts concerned, but from its understanding of how to set about developing such hypotheses in general and its knowledge of the particular information which the student has asked for. The perceived style of the ITS is all-important: if the system were to say "Those last two metals were both shiny - perhaps they all are", this would have to be understood by the student not as an unsubtle hint (the tutor knowing perfectly well that all metals are shiny), but as a genuine, hopefully useful, but possibly mistaken, comment.

How could it be made to work? The ITS needs to be able to work out what a student may reasonably be expected to infer (or not to infer) from the data she sees. In other words, in this case, the ITS needs a machine learning program capable of inferring concepts from examples in a psychologically plausible way. The ITS would use such a program in two ways: first, for building a student model describing the student's beliefs about the concepts (again, as above, these are not to be judged by the ITS as correct or not, but, if necessary, as

'interesting' or justified by the evidence or not), and secondly, to work out which information would enable it (the ITS) and, it is to be hoped, the student to refine those beliefs, in particular, to find information capable of falsifying them.

We adapted the focussing algorithm for concept learning to try to build these student models (Gilmore and Self, 1988), but we encountered three particular problems:

1. We had underestimated the volume of prior knowledge which such an ITS would need. Only with AI machine learning programs does the closed world assumption hold. Students bring to bear all sorts of 'common knowledge' (such as that 'temperatures are precisely given, but colours are imprecisely described in English', 'boiling points are related to melting points but probably not to conductivity') which an ITS would need access to, even if it is not to know specifically what is to be learned by the student. Perhaps the approach of explanation-based learning would have been a better choice than the similarity-based learning of focussing. Explanation-based learning emphasises the role of domain knowledge in promoting learning from a single example, rather than by generalising over several examples.

2. The perceived setting was too sterile: we could not imagine many students being sufficiently motivated to explore the database in the way assumed. Learning concepts is not an abstract game: concepts are learned for a purpose (to help solve problems or to make sense of a story, say). Without knowing why the student is trying to discover what a metal is it is not possible for an ITS to determine whether her beliefs about the concept are adequate for the present purpose. Conceptual learning needs to be embedded within a problem setting.

3. We had little idea of how to support collaborative dialogues, in which both participants are to be seen as of 'equal status', or whether students would, in fact, welcome such a style of interaction with a computer. The benefits claimed for human-human collaboration (e.g. Slavin, 1983) may result more from social and motivational effects than

process effects, and may therefore disappear even if the student-computer collaboration is carefully designed.

However, the exercise was productive in two ways. First, it showed that educationalists were more responsive to the idea of a computer collaborator than to that of a computer tutor. Of course, 'collaborative learning' is very much in vogue in modern classrooms and we may simply have triggered a positive reaction. On the other hand, they may have grounds for believing that students learn more productively by discussing problems with peers than by being tutored by 'experts' such as themselves.

Secondly, it caused us to re-think the role of a student model. In a conventional ITS, the student model is an internal data structure purporting to describe the student which is used furtively by the ITS to determine tutorial actions. (Incidentally, it is questionable whether such a student model is legal - in the U.K., at least, all data about individuals has to be registered with the Data Protection Registrar and accessible to the individuals concerned!) In a collaborative ITS, there is no need to hide the student model from the student - in a sense, it represents a shared understanding of the problem and with the more 'open' philosophy, it may well help, or challenge, the student to be aware of what the system thinks she believes.

Indeed, it would be a salutary principle to insist that all student models be made open to the student. This might benefit ITS design by reducing the temptation to include crude, ad-hoc classifications, and, more importantly, may lead to educational benefits as it might well provoke the student to reflect on her own understanding. To promote student reflection and to foster collaborative learning it would not be necessary to develop the high-fidelity student models needed for remediation.

A collaborative interaction may also help the student develop a more favourable self-image and a better view of how knowledge is acquired. Instead of being perpetually corrected, with the imputation of incompetence or stupidity, she may see how her own understanding reasonably develops. A 'knowledgeable' ITS commenting that "that is a perfectly reasonable misconception but ..." would be seen as patronising: a collaborative ITS would carry the same message without needing to verbalise it.

In a related exercise, we are considering whether it may be possible for an ITS to understand, and comment upon, a student's strategy, without attempting to understand the subject matter. With the AlgebraLand system (Foss, 1987), a student solves algebra problems and has the search space displayed graphically to reflect upon. AlgebraLand does not attempt to tutor the student. But by analysing the shape of the search space and simple syntactic properties of the nodes of the graph (and *not* by trying to understand the problem-solving process in depth), it is possible to determine potentially useful features of the space, such as when students tend to abandon a solution path, where they tend to back up to, which nodes seem to be a source of difficulty, whether the search is in any way systematic, and so on. Consequently, it may be possible for an ITS to give strategic advice to a student, from the evidence of structural properties of the search space.

We are looking at the ways students use hypertext systems, e.g containing a classical Greek dictionary, to answer questions such as "Describe the main adversaries faced by Jason in the Argonautica". It is clearly not possible to apply machine learning techniques to learn, as students would, from reading the dictionary text. We are hoping to be able to make sufficient sense of their search spaces, without understanding any of the subject matter, to enable an ITS to give strategic advice to students.

However, there is an apparent logical flaw in the argument that ITSs do not need subject knowledge, but rather need knowledge of strategic learning skills, so that ITSs may promote the learning of those strategic skills, which we may consider more important than mere factual knowledge. For, by a similar argument, perhaps ITSs do not need learning skills either, but rather need knowledge of how to develop learning skills (and so on). But for all practical purposes, and I suspect for all theoretical ones too (for the skills at 'higher' levels may not be significantly different from those at lower ones), there are only a very small number (e.g. 2) of such levels.

4. Conclusions

This review of the role of student models in ITSs is intended to show that, while research in student models is potentially capable of embracing almost all the problems in cognitive science and that there is therefore no realistic possibility of building student models which meet all the objectives of ITS designers, there are nevertheless several ways in which, by changing our design principles and our philosophical approach, we may build ITSs in which student models play a significant role.

Several suggestions have been made, perhaps the most important of which are:

1. To design student-computer interactions in which the information needed (especially about the student's goals) by the ITS to build a student model are provided naturally by the student while using the ITS, and does not have to be inferred by the ITS from inadequate data.

2. To explicitly link the proposed contents of student models with specific tutorial actions, ideally supported by educational evidence, in order to clarify what is really needed (and not needed) in the student model.

3. To avoid viewing student models solely as devices to support remediation, which is often perceived as implying a behaviourist philosophy of learning and which often cannot be satisfactorily achieved anyway because of various difficulties with the 'mal-rule' approach to student modelling.

4. To use student models 'constructively' by regarding the contents as representing student beliefs, with no value judgements imposed by the ITS, the ITS's role being to help the student elaborate those beliefs.

5. To make the contents of the student model open to the student, in order to provoke the student to reflect upon its contents and to remove all pretence that the ITS has a perfect understanding of the student (and that ITS designers should build systems which proceed as though they do).

6. To develop ITSs which adopt a more collaborative role, rather than a directive one, for then the style corresponds to a better philosophy of how knowledge is acquired and we do not have to seek such a high degree of fidelity in the student model.

"It is not often that any man can have so much knowledge of another, as is necessary to make instruction useful."

Samuel Johnson (1752), *The Rambler*, p87.

Acknowledgements

The U.K. Science and Engineering Research Council has supported the work of Michael Twidale, Steve Payne, Helen Squibb, David Gilmore and myself. Logica (Cambridge) Ltd and British Telecom have been involved in some of the projects. Other members of the University of Lancaster's Centre for Research on Computers and Learning (in particular, Peter Goodyear, Jim Ridgway and Rachel Rymaszewski) will recognise the influence of their thoughts on this paper.

References

Anderson, J.R. and Reiser, B. (1985). The Lisp tutor, *Byte*, 10, 159-175.

Brown, J.S. and vanLehn, K. (1980). Repair theory: a generative theory of bugs in procedural skills, *Cognitive Science*, 4, 379-426.

Clancey, W.J. (1987). *Knowledge-Based Tutoring: the GUIDON Program*, Cambridge, Mass.: MIT Press.

Corno, L. and Snow, R.E. (1986). Adapting teaching to individual differences among learners, in M. Wittrock (ed.), *Handbook of Research on Teaching*, New York: MacMillan.

Foss, C.L. (1987). Productive thrashing in a computerized tutoring system, *Proceedings of the Third International Conference on Artificial Intelligence and Education*, Pittsburgh.

Gilmore, D.J. and Self, J.A. (1988). The application of machine learning to intelligent tutoring systems, in J.A. Self (ed.), *Artificial Intelligence and Human Learning*, London: Chapman and Hall.

Goldstein, I.P. (1982). The genetic graph: a representation for the evolution of procedural knowledge, in D.H. Sleeman and J.S. Brown (eds.), *Intelligent Tutoring Systems*, London: Academic Press.

Kolodner, J.L. (1983). Towards an understanding of the role of experience in the evolution from novice to expert, *International Journal of Man-Machine Studies*, 19, 497-518.

Mizoguchi, R., Ikeda, M. and Kakusho, O. (1988). An innovative framework for intelligent tutoring systems, in P. Ercoli and R. Lewis (eds.), *Artificial Intelligence Tools in Education*, Amsterdam: North-Holland.

Mobus, C. and Thole, H. (1988). Tutors, instructions and helps, Absynt Report 3/88, University of Oldenburg.

Pask, G. (1976). Styles and strategies of learning, *British Journal of Educational Psychology*, 46, 128-148.

Payne, S.J. and Squibb, H.R. (1987). Understanding algebra errors: the psychological status of mal-rules, CeRCLe Technical Report 43, University of Lancaster.

Pintrich, P.R., Cross, D.R., Kozma, R.B. and McKeachie, W.J. (1986). Instructional psychology, *Annual Review of Psychology*, 37, 611-651.

Resnick, L.B., Cauzinille-Marmeche, E. and Mathieu, J. (1987). Understanding algebra, in J. Sloboda and D. Rogers (eds.), *Cognitive Processes in Mathematics*, Oxford: Clarendon.

Sandberg, J.A.C. (1987). The Third International Conference on Artificial Intelligence and Education, AICOM, 0, 51-53.

Self, J.A. (1988). Student models: what use are they?, in P. Ercoli and R. Lewis (eds.), *Artificial Intelligence Tools in Education*, Amsterdam: North-Holland.

Shrager, J. (1987). Theory change via view application in instructionless learning, *Machine Learning*, 2, 247-276.

Singley, M.K. (1987). The effect of goal posting on operator selection, *Proceedings of the Third International Conference on Artificial Intelligence and Education*, Pittsburgh.

Slavin, R.E. (1983). *Cooperative Learning*, New York: Longman.

Sleeman, D.H. (1987). Some challenges for intelligent tutoring systems, *Proceedings of the International Joint Conference on Artificial Intelligence 87*, Milan.

Woolf, B.P. (1988). Representing complex knowledge in an intelligent machine tutor, in J.A. Self (ed.), *Artificial Intelligence and Human Learning*, London: Chapman and Hall.

Figure 1: Two images of the goal-posting window (from Singley, 1987).

Figure 2: Monitoring plans in a logic tutor.