Artificial Intelligence Review 7, 227—240, 1993.

Student modelling in intelligent tutoring systems

MARK ELSOM-COOK

Electric Brain Company

Abstract. Student modelling is a special type of user modelling which is relevant to the adaptability of intelligent tutoring systems. This paper reviews the basic techniques which have been used in student modelling and discusses issues and approaches of current interest. The role of a student model in a tutoring system and methods for representing information about students are discussed. The paper concludes with an overview of some unresolved issues and problems in student modelling.

Key Words: adaptive systems, student models, intelligent tutoring systems, knowledge representation, learner based modelling.

1. INTRODUCTION

One of the central assumptions underlying research on Intelligent Tutoring Systems is that a teaching interaction should adapt to individual differences between learners. This is normally taken to mean that the system can vary the content and form of its presentations and responses according to the state of the learner. State is generally taken to mean cognitive state, and it is assumed that this state is identified by the system and stored in a student model for future use.

In this paper we will review the basic techniques which have been used in student modelling and highlight some of the issues and approaches which are of current interest. The paper does not focus on an exhaustive description of systems, because most systems use small variations on the same techniques and have little to contribute to the development of student models. Most of the advances in student modelling have been concerned with the way that we understand the problem, rather than the way we solve it.

In the second section of the paper we will examine the role that a student model is expected to play (or could potentially play) in more detail. The third section of the paper reviews methods of representing the student, starting with the simplest and leading to techniques which are the focus of current research. Finally, in section 4 we identify a number of issues and problems which exist in this field of research.

2. WHAT IS THE ROLE OF A STUDENT MODEL?

For the purposes of this paper, we will assume that tutoring systems are constructed around what has become known as the 'trinity' of components (Self 1988c): knowledge about the domain being taught, about how to interact with the learner, and about the student. While this division has been questioned recently, most ITSs have been constructed in this framework. The division is, to some extent, an artificial one since the components are intimately connected: the form of domain representation and the form of the student model typically being very closely bound together.

It is normally assumed that knowledge about the domain and about interaction are fixed components of the system, so a generative, adaptive interaction relies on the presence of the student model to enable selection of aspects of the other components. In general terms, we wish to select the content and form of an interaction segment, i.e. to decide what to say next, and how to say it. There are many competing views on how this adaptation can be engineered, and on the way in which a student model is utilized. Self (1988b) reviews existing systems and finds 6 main uses of student models. These are:

1) corrective: feedback intended at repairing a misunderstanding of the student. In this case the model must identify a difference between the student's understanding and the 'correct' knowledge, and provide this information to other parts of the system.
2) elaborative: extending the knowledge of the student. In this case, the model should identify areas where the student can be introduced to new material, or a refinement of her current understanding.
3) strategic: changing the approach to teaching at a higher level than local tactics. This requires the student model to provide more general information about the student, such as her success rate with the current teaching strategy as opposed to a previous teaching strategy.
4) diagnostic: analysis of the state of the student. In some sense, all aspects of student modelling are diagnostic. What is meant here is the explicit use of the student model to refine information about the student in order to make a decision. If, for example, the tutor wishes to introduce a new topic, but the student model is unable to indicate whether the current level of understanding of the student is adequate, the model can be requested to generate diagnostic examples which can be given to the student.
5) predictive: using the model to anticipate the effect of an action upon the student. This requires the student model to act as a 'simulator', allowing the tutor to pose such questions as "If I do ⟨action⟩ what is the likely change to the student's state?".
6) evaluative: Providing an assessment of the level of achievement of the student. This requires the system to make some aggregation across the information that it has.

There are other taxonomies, and different levels of detail can be described, but for our current purposes we will accept this one as typical. Given this range of activities to be supported, we now need to ask how we can adapt the system in

this way. Can these roles be satisfied without a student model, and if not, what are the essential properties of such a model.

The simplest form of adaptation to the student is the sort of branching CAI program in which an action of the student may correspond to a branch point in the program. Which branch is selected depends on the response of the student. Such a mechanism can, in some sense, support the corrective and elaborative roles listed above, as well as giving a crude form of assessment. However, the local nature of the decision-making, and the lack of longer term structures precludes strategic, diagnostic or predictive use of the mechanism. Even for those things which can be supported to some extent, the lack of a representation of the student means that the level of adaptation would necessarily be fairly large-grained and localized. There is no sense in which a mechanism such as this can be called a model, since there is no representation of anything (i.e. a student).

If we extend this type of adaptation slightly, we arrive at programs which maintain a data structure that aggregates across a number of interactions with the student. In traditional CAI such mechanisms were used to keep track of the number of questions asked, the material which had been presented, the number of questions which were correctly answered by the student etc. This information was used for such purposes as deciding whether to increase, or decrease the difficulty of questions, and to determine when the student understood the material sufficiently well for teaching to be terminated. The aggregated data allows a more global modification to the behaviour of the system, but it is still very much at a surface level. The types of parameters discussed above are basically aspects of the interaction, rather than the student, and since it is descriptive in nature, this object can be thought of as an interaction history.

The first attempt at a student model was that embodied in the system by Carbonell (1970). This system used a semantic network to represent a domain, and associated tags with the nodes of the network which affected the behaviour of the system. These tags were used to decide whether to ask further questions about a topic etc. but essentially the approach was the same as the sort of interaction histories discussed above. This was particularly the case because these types of systems did not clearly separate the 'material' and the knowledge representation. The teaching material for a particular node was associated with that node (or generated for that specific context), and the tags simply represented whether the material had been used. In later systems, where knowledge representation and material do not have this isomorphism, the same basic technique is more interesting. The tags represent whether some piece of knowledge is understood, rather than whether the student has been asked about it, or some other interaction level aspect. Such a model is sufficient to fulfill all the roles given above, although it is not necessarily a model of the student. Whether it fulfills that role depends on how the information arrives in the model in the first place, and the semantics of a statement such as 'The student knows K' expressed in terms of the model.

Having decided what we mean by 'student model', the following section will examine the representation techniques which are used.

3. STANDARD MODELLING TECHNIQUES

The first part of this section reviews expert-based modelling techniques, which are commonly used (with small variations) across the majority of teaching systems. In the second part we will examine some more speculative techniques which are currently being pursued.

3.1. *Expert based modelling*

Most existing methods of student modelling can be characterized as expert-based. This means that the domain representation of the tutoring system is taken to be a representation of the expert's knowledge, and the student model is constructed in terms of that representation. This approach also brings with it assumptions: namely, that there is a definite objective to the learning, in terms of a specific body of knowledge to be acquired, and that the representation which the expert has is the same as that which the learner has, the difference being in terms of completeness (see Fig. 1). It is also assumed that not only the knowledge, but the way in which that knowledge is structured, is the same within the tutor as within the student. We will investigate these assumptions in more detail below.

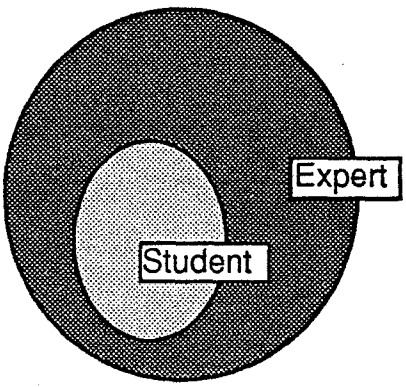

Fig. 1. The student model as a subset of the expert's model.

Essentially, expert-based modelling methods can be divided into two categories: subset-based methods and pertubation-based methods. Each of these can be applied to a variety of representations.

A subset-based approach makes the assumption that the knowledge representation used in the tutor has divided the expertise into units which are sufficiently simple to be considered atomic, i.e. a student either knows or does not know a certain unit: it is not possible to partly know something. It is also assumed that the only knowledge (about this domain) that can be present in the student's head is represented by these units. There is no possibility of allowing the student to have different conceptions of the domain from that of the expert.

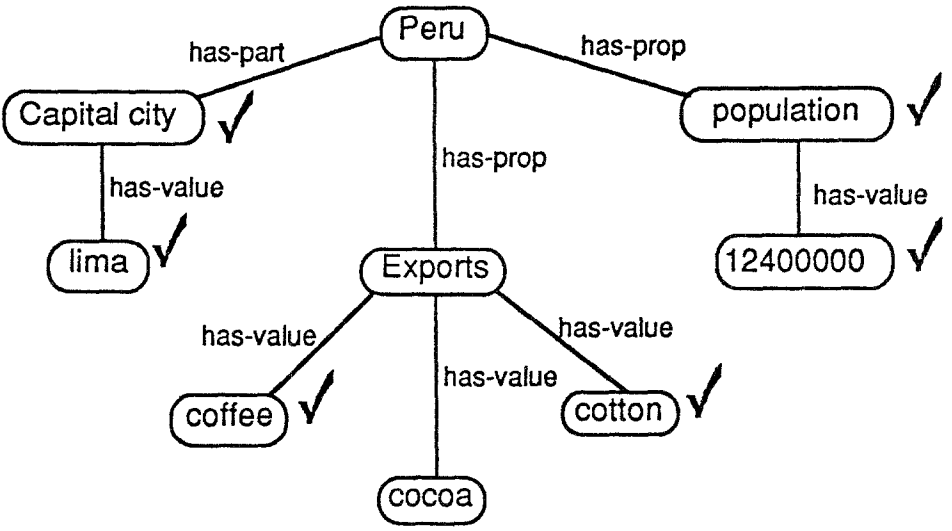

Fig. 2. The subset approach applied to a semantic network.

This technique has been applied to network representations and to rule-based representations.

The above diagram (Fig. 2) illustrates the typical application of a subset approach to a semantic network. This method has been used in such systems as Scholar (Carbonell 1970) and TRILL (Cerri *et al.* 1990). The diagram shows ticks beside some nodes, which correspond to the 'tags' indicating that certain things are known. These tags form a subset of the network corresponding to the knowledge state of the student. What a tagged node means depends on the exact form of the representation used. It is meaningless, in the above example, to say that the student knows 'coffee', but it may be reasonable to assume that the diagram indicates that the student knows that coffee is an export.

With rule-based representations, subset-based modelling becomes more interesting. In these approaches, the knowledge (normally procedural in nature) is represented by a set of rules which constitute a runnable model. This means that a representation of subtraction, as, for example, in Fig. 3, can actually perform the subtraction task. In tutoring systems, rule-based approaches are usually (but not always) production-rule based. Hence, if we delete a rule from our rule-base, the system will still be capable of trying to perform the task, but may fail in different ways depending on which rule is deleted.

This is the interesting feature of subset modelling in conjunction with a rule-based representation. By deleting rules from the 'expert' representation, we produce other systems which may still produce behaviour, but where that behaviour is not the 'correct' one. This provides the tutoring system with a direct performance measure of the effectiveness of its model. If, by deleting rules from the expert behaviour, the tutor can imitate the behaviour of the student, then the corresponding rule-set is a candidate model of that student. We will return to this issue later in the section.

```
A - B and B < A then do(A-B)

( A - B and B > A then borrow )

A - B and A = B then 0
```

Fig. 3. Rules for subtraction.

Let us now turn to examine perturbation based modelling. In this approach, it is still assumed that the expert knowledge representation consists of atomic units, but it is acknowledged that the student may have atomic units of knowledge which are not present in the head of the expert. These additional units are assumed to be flawed versions of expert knowledge units and are termed misconceptions or bugs (see Fig. 4). The tutor must identify and eliminate these misconceptions as well as adding the correct conceptions to the understanding of the student. While the idea of perturbations is general enough to apply to any representation, it has only been applied to rule-based representations in the tutoring systems literature, and hence it is often called mal-rule modelling.

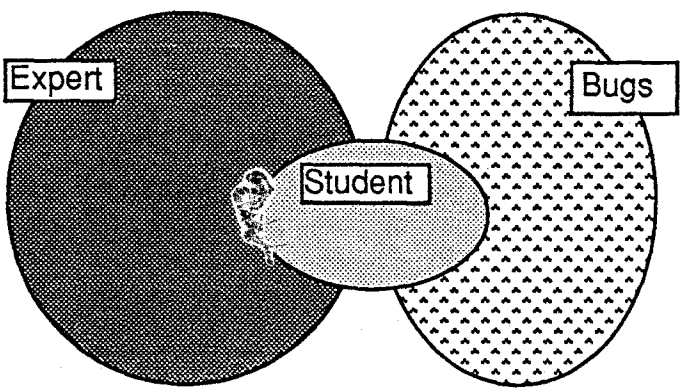

Fig. 4. Buggy modelling.

If we consider the example of subtraction given in Fig. 3, the deletion of the borrowing rule could lead to a variety of behaviours, depending on which other rules are represented in our expert model. There are several behaviours to which it will not lead, however. Our expert will not make the result 0 in cases where she should have borrowed, nor will she simply subtract the smaller of the two numbers from the larger. These are not part of the correct behaviour and are therefore not represented in that part of the system. Unfortunately, these behaviours are two of the most common ones that children demonstrate if they

do not know how to borrow. In constructing a mal-rule model of a particular student, we would delete the correct borrowing rule and insert a flawed version of the same rule (a mal-rule).

The prototypical example of a bug-based modelling system is Buggy (Brown and Burton 1978). In this system a rule-based representation of multi-column subtraction was developed together with a *bug library* based on a large empirical study. The resulting system could take a set of responses to a mathematics test and analyze the errors, producing a model consisting of a mixture of correct rules and bugs, which was a candidate for the student's current subtraction procedure. A comparatively small bug library (of the order of 100 rules) successfully represented the large majority of bugs found in the empirical studies, and during analysis buggy agreed with human experts 80% of the time (Van Lehn 1983a).

It is important to be conscious of the distinction between *errors* and *bugs*. In the sort of tasks which can be modelled with a perturbation approach of this kind, there is a correct procedure (or a set of alternative procedures) to be executed, which produces a specific behaviour (or outcome). If the outcome does not correspond to that which is expected, the difference between expected behaviour and observed behaviour is described as an error. A bug, however, is a variant of the procedure which generates an error. It is possible for the same error to be explained by different bugs.

The prediction of these simple models is that any error a student makes can be explained as either due to a bug or a slip. Any other behaviour corresponds to correct execution of a procedure. An empirical examination of students (Van Lehn 1982) showed that this was not necessarily the case. In particular, examining the bugs of a student over a period of time revealed some important changes in the pattern of bugs of a student which required explanation. In particular, bug migration (the change of one bug into a different, but related, one), highlighted the fact that there appears to be a deeper structure to the sort of bugs that a student generates. Essentially, we can see bugs which are alternative ways of resolving the same difficulty (such as borrowing across zero). An additional observation is that, with a large enough catalogue of bugs, every possible behaviour could be described. This is not a very helpful property if we wish to obtain a concise theory.

For these reasons, Repair Theory was devised (Brown and Van Lehn 1980). This model goes one level deeper than the divide between errors and bugs, suggesting that not only are errors manifestations of bugs, but that bugs themselves are manifestations of a deeper structure called an impasse. The model proposes that there are a number of places in a procedure where the student may not know what to do. At these points, she uses local problem-solving abilities to generate a procedure to carry out, and this procedure is (generally) a bug. For example, if a student is unaware of the correct procedure for borrowing across zero, a 'borrow across zero' impasse occurs, and consequent bugs and errors are generated. This can be illustrated with the following example:

305	305	305
17	17	17
298	198	292

In the first case, the bug is to simply ignore the decrement of the tens column during the borrow, in the second case the bug is to decrement from the nearest number on the left, and in the third case it is to avoid the borrow altogether and take the smaller number from the larger. These are three different bugs, but are each possible ways of resolving the same impasse. Van Lehn (1983a) refined the model of repair theory in a number of ways, identifying 'core procedures' which are the currently stable, unchangeable procedure of the learner. It is situations where these core procedures are inapplicable that give rise to bugs.

Clearly, simply collecting empirical data on repairs is little different from collecting a bug catalogue. The difference is that it is possible to imagine repairs being the result of a local problem-solving mechanism. If such a mechanism is defined, then bugs can be generated, and the need for empirical data avoided.

It should be noted here that both bug-based models and Repair theory have been questioned from a number of perspectives. Empirical work (Payne and Squibb 1986) has called into question the idea that stable bugs or core procedures exist at all, and has also demonstrated problems with the population-independence of mal-rules. Sets of rules which describe one group of children well may explain only a very small percentage of the bugs arising in another group who are theoretically at the 'same level' of understanding. Hennessy (1990) raises some issues about the applicability of repair theory to modelling learners in a real context. She identifies a number of semantic constraints on learning mathematics which were apparent in her own empirical work, and which would negate attempts to model her subjects using a syntactic model such as that underlying repair theory.

Moving on from Repair Theory as a descriptive tool, Van Lehn extended the model in a system called Sierra (Van Lehn 1983a) to make it directly generate bugs. Rather than analyzing student data, as has been the case with Buggy and Debuggy (Burton 1982), Sierra tried to recreate the bugs by analyzing the instructional material. It took as input a 'lesson sequence', essentially consisting of a sequence of examples that would be presented to students, and used machine learning techniques to infer the procedures involved. A solver, based on repair theory, then applied those procedures to test problems to generate bugs.

The task which Sierra attempts to accomplish is to identify the space of bugs which could be caused by a particular lesson sequence, rather than to model a particular student. In common with the earlier systems, its underlying representation is claimed to have some relation to psychological models of procedural knowledge (Van Lehn 1983b). Unlike the earlier systems, however, it includes an inductive learning component akin to that of Mitchell *et al.* (1983). This is a machine learning alorithm with no claim to psychological status, so it is surprising to find that Sierra is remarkably effective at predicting bugs in human learners (Van Lehn 1983a).

Subsequently, via the Cirrus (Van Lehn and Garlick 1987) and Cascade (Van Lehn *et al.* 1991) systems, this work has continued to develop using machine learning techniques and expanding from the domain of arithmetic into simple physics problems. This research is founded on the idea that student modelling necessarily requires a model of learning processes, and further, that to be of educational use, this model has to be more domain-specific than a general theory of cognition such as ACT* (Anderson 1983) or SOAR (Laird *et al.* 1987). Current machine models of learning are computationally very intensive, and it is Van Lehn's view that this type of student modelling will not be possible in real-time systems for several years. His own research is concentrating on the off-line analysis of instructional material and student protocols in order to improve a priori design of teaching materials.

In the next section we will investigate other approaches to student modelling based upon learning methods.

3.2. *Learner based modelling*

Perhaps the most dubious of the assumptions underpinning the expert based approaches to student modelling is that the student is a mini-expert. There is considerable psychological evidence that conceptions of a domain may radically alter over time and many models of teaching (e.g. Whitehead 1932) are explicitly based around supporting this evolution of understanding.

The realization that these approaches are not supported by the standard modelling techniques led to a move towards the examination of learner-based approaches. The fundamental point about these approaches is that they do not assume that the learner is to be made into a copy of the expert. It is also the case that these approaches are not based on the 'knowledge state' used in the approaches described above. Instead, they place an emphasis on the relationship between pieces of knowledge as they evolve over time. These approaches are therefore concerned with the mechanisms by which the learner acquires knowledge as well as a description of what it is that the student acquires.

This move away from a state-description, or snapshot approach has some interesting consequences for the utilization of the model. Our intention was to use the models to answer such questions as "What do I say next" and "How do I say it". A state model can clearly help us in deciding what knowledge to deal with next, but it is of limited use in helping us to decide how to say it. State information can inform us of knowledge that we can or cannot assume when, for example, generating a description of something, but that is the limit of its usefulness.

If we have a model which incorporates learning mechanisms, then we can say something about why a student has certain knowledge (correct or incorrect) and also say something about what the student might learn from a given intervention. This brings us to a point where we can think of a much closer relationship between teaching strategies and learning strategies. For example, if a student believes that all buses are red because they are the only sort she has seen, the

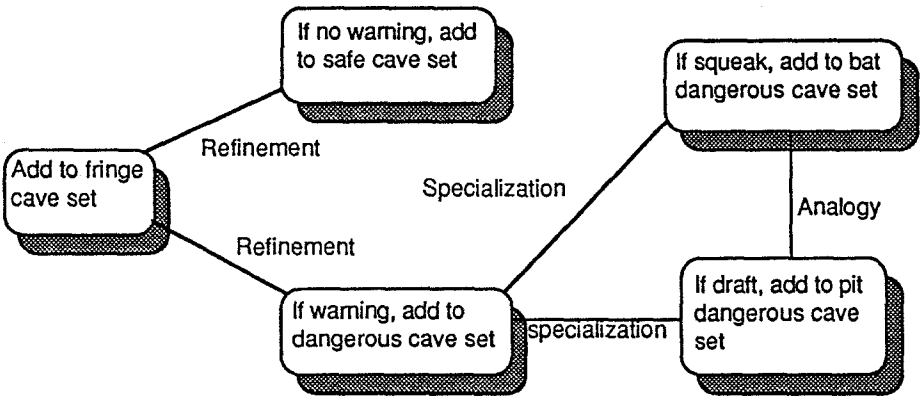

Fig. 5. Modelling using genetic graphs.

sort of tutoring which is appropriate would be very different than if she held the belief because someone had told her so, or because she was colour-blind.

The first approach to a learner-based modelling method was the Genetic Graph (Goldstein 1982). In this system Goldstein analysed the requisite skills for playing a game into atomic components (much as in the subset-based modelling technique) and created links between these components. Unlike the links in expert-based modelling however, these links were links expressing the sort of learning which could take a student from one point to another. If, as in Fig. 5, a student has learnt that a squeak meant that a pit was dangerous, the system could teach it about drafts by analogy, or about dangerous caves by generalization.

While the system did attempt to take the learner's perspective into account, it was not really learner oriented, in the sense that it still relies on the expert's conception of the domain. Indeed, the links are an expert's view of how learning could occur between nodes, and the actual modelling was a standard subset technique. They only real difference is the sort of knowelege on which the system worked.

Automated Cognitive Modelling (Langley and Ohlsson 1984) is a technique based on machine learning which attempts to construct a model of a student off-line. As with many of the approaches, the resulting model consists of a set of production rules which together mimic the behaviour of a student on a given set of problems. Unlike the earlier approaches, the rules — current or buggy — are not selected from built-in libraries, but are constructed directly from the data.

In essence, ACM constructs productions which are rules for searching the problem space in which the student is operating. If, for example, the student has been presented with the problem 23−17, and has produced the result 14, then our initial state is the problem and our final state is the solution. ACM is provided with a set of operators to move between states (such as add-10, subtract etc.) and a set of predicates which can be used to determine the properties of a state (such as greater-than, equal etc.). ACM first generates the

search space, including the paths to the student's actual solution, and the steps which move off that path at each point. For each operator which has been applied, a set of positive and negative instances of its application are abstracted from this search tree. Positive instances are those which remain on the path towards the student's actual solution, and negative instances are those which diverge from the path. At each choice point, all the predicates are applied to the state. The problem for ACM is then to identify the minimal set of predicates to apply to each operator in order to ensure that only positive instances of operator application are selected. This is a standard discrimination learning task.

Belief-based bounded user modelling (Elsom-Cook 1990) is one of a number of proposals for moving away from representing knowledge to working in terms of beliefs of the student. This approach modifies a standard belief system to permit various forms of justification between nodes (truth-preserving, inductive, abductive, flawed etc.). This permits the system to try and identify the sorts of justification the student has for her beliefs as well as the beliefs themselves. Additionally, the approach maintains consistency only among things of which the student is simultaneously aware (allowing inconsistency of belief), and uses a version spacing machine learning algorithm (Mitchell 1978) to allow uncertainty about the beliefs of a student to be represented by partially ordered sets. In common with the other machine learning approaches, this mechanism is a candidate for further research, rather than a practical technique for real ITSs.

4. PROBLEMS AND ISSUES

It is apparent from the above descriptions of techniques that there are some severe restrictions being imposed on student models. We either have the expert-based approaches, which can be made to work, but are founded in a flawed model of the teaching and learning processes, or the learning based approaches, which are still beset by a host of theoretical problems. In all cases, we have avoided modelling many of the features which influence the learning of the student. There is no attempt to model conative or affective information, for example: the motivation, wants, goals and desires of the students are missing entirely. In human teaching maintaining motivation is observably a major part of the interaction (Bellack *et al.* 1966).

These problems have led a number of researchers to abandon the student modelling problem altogether, claiming either that it is intractable, or that it is unnecessary and systems can be effective teachers without such a model. In a recent examination of this question, Self (1990) produced a slightly more optimistic prognosis. He concluded that there are many difficult problems, but that they are not as significant as they seem if we re-examine exactly what it is we are trying to achieve with a student model. There are two major points to emerge from his review. The first is that student modelling is not about building exact cognitive models. If it were, we would have to solve all the problems of cognitive science, and teach a machine to be a cognitive scientist, before we

could build a student model. We only need to model the student to the level of detail necessary for the teaching decisions we are able to take. If the tutor only has two choices of action, then the model only needs to be accurate enough to distinguish between them. The second point is that there is no need to make the computer an all-powerful machine which can magically infer everything it needs to know from the actions of the student. Much more can be done to enable the student to explicitly give the system the information it needs for modelling, and to enable the student to actively collaborate with the system on building a model.

5. CONCLUSIONS

In reviewing the use of student modelling in Intelligent Tutoring Systems, it is apparent that the field is one in which work has only just begun. The main contribution of the research reviewed here is not to come up with a range of methods and techniques for constructing such models, but rather to change our understanding of the problem. Contributions tend to be theoretical, rather than practical, and it is all too easy to criticise any one of the methods outlined here. I have tried to outline the major assumptions which have been discarded as our understanding of student modelling has developed, and to show the way that has led to changes of perspective. We have perhaps reached the point where we are beginning to talk about student models in the true sense of a model, for the first time.

BIBLIOGRAPHY

Note: This bibliography is not intended to be exhaustive. It lists the key papers in the area. Several of the papers have appeared in slightly different guises as book chapter, conference proceedings and journal article.

Anderson, J. R. (1983). *The Architecture of Cognition.* Harvard University Press, Cambridge, Massachussetts.

Beard, M. H., Lorton, P. V., Searle, B. W. and Atkinson, R. C. (1973). Comparison of Student Performance and Attitude under Three Lesson Selection Strategies in Computer Assisted Instruction. Tech. report 222. Institute for Mathematical Studies in the Social Sciences, Stanford University.

Bellack, A., Kliebard, H., Hyman, R. and Smith, F. (1966). *The Language of the Classroom.* Teachers College Press, Columbia University, New York.

Brown, J. S. and Van Lehn, K. (1980). Repair Theory: A Generative Theory of Bugs in Procedural Skills. *Cognitive Science* **4**, 379—426.

Brown, J. S., Burton, R. R., Hausman, C. L., Goldstein, I. P., Huggins, B. and Miller, M. L. (1977). Aspects of a Theory of Automated Student Modelling. Technical report 3549. Bolt, Beranek and Newman, Cambridge.

Brown, J. S. and Burton, R. R. (1978). Diagnostic Models for Procedural Bugs in Mathematical Skills. *Cognitive Science* **2**, 155—192.

Burton, R. R. (1982). Diagnosing Bugs in a Simple Procedural Skill. In Sleeman, D. H. and Brown, J. S. (Eds.), *Intelligent Tutoring Systems.* Academic Press, London.

Burton, R. R. and Brown, J. S. (1976). A Tutoring and Student Modelling Paradigm for Gaming Environments. ACM SIGCUE topics, Vol. 2.

Carbonell, J. R. (1970). Mixed-initiative Man-computer Instructional Dialogues. Tech. report No. 1971. Bolt, Beranek and Newman, Cambridge.

Carr, B. and Goldstein, I. P. (1977). Overlays: A Theory of Modelling for Computer-aided instruction. AI lab memo 406. Massachusetts Institute of Technology, Cambridge.

Cerri, S., Elsom-Cook, M. T. and Leoncini, M. (1990). TRILL: The Rather Intelligent Little Lisper. *Intelligent Tutoring Media* **1**(1).

Clancey, W. J. (1986). Qualitative Student Models. In Traub, J. F. *et al.* (Eds.), *Annual Review of Computer Science* **1**, 381—450.

Cumming, G. D. and Self, J. A. (1991). Learner Models in Collaborative Intelligent Educational Systems. In Goodyear, P. (Ed.), *Teaching Knowledge and Intelligent Tutoring.* Ablex.

Elsom-Cook, M. T. (1988). Guided Discovery Tutoring and Bounded User Modelling. In Self, J. A. (Ed.), *Artificial Intelligence and Human Learning.* Capman and Hall, London.

Elsom-Cook, M. T. (1990). Belief-based Bounded User Modelling. In *Learning Technology in the European Communities.* Kluwer, Dordrecht.

Evertsz, R. and Elsom-Cook, M. T. (1990). Generating Critical Problems in Student Modelling. In Elsom-Cook, M. T. (Ed.), *Guided Discovery Tutoring.* Paul Chapman, London.

Gilmour, D. and Self, J. (1988). The Application of Machine Learning to Intelligent Tutoring Systems. In Self, J. (Ed.), *Artificial Intelligence and Human Learning.* Chapman and Hall, London.

Goldstein, I. P. (1982). The Genetic Graph: A Representation for the Evolution of Procedural Knowledge. In Sleeman, D. H. and Brown, J. S. (Eds.), *Intelligent Tutoring Systems.* Academic Press, London. Also in *International Journal of Man-Machine Studies* **11**, 51—77.

Hennessy, S. (1990). Why Bugs Are Not Enough. In Elsom-Cook, M. (Ed.), *Guided Discovery Tutoring.* Paul Chapman, London.

Johnson, W. L. (1986). *Intention-Based Diagnosis of Novices Programming Errors.* Pitman, London.

Laird, J. E., Newell, A. and Rosenbloom, P. S. (1987). SOAR: An Architecture for General Intelligence. *Artificial Intelligence* **33**.

Langley, P. and Ohlsson, S. (1984). Automated Cognitive Modelling. In *Proceedings of AAAI — 84,* pp. 193—197.

Mitchell, T. (1978). Version Spaces: An Approach to Concept Learning. Ph.D. Dissertation, Stanford University.

Ohlsson, S. (1985). Some Principles of Intelligent Tutoring. *Instructional Science* **14**(3—4), 293—326. Also in Lawler, R. and Yazdani, M. (Eds.), *Artificial Intelligence and Education.* Ablex, Norwood, N.J.

Payne, S. J. and Squibb, H. R. (1986). Understanding Algebra Errors: The Psychological Status of Mal-Rules. Tech. report no. 43. Centre for Research on Computers and Learning, Lancaster University.

Reiser, B. J., Anderson, J. R. and Farrell, R. G. (1985). Dynamic Student Modelling in an Intelligent Tutor for Lisp Programming. *Proceedings of IJCAI 1985.* Los Angeles.

Rich, E. A. (1979). User Modelling via Stereotypes. *Cognitive Science* **3**, 329—354.

Self, J. A. (1974). Student Models in Computer-aided Instruction. *International Journal of Man-Machine Studies* **6**, 261—276.

Self, J. A. (1979). Student Models and Artifical Intelligence. *Computers and Education* **3**, 309—312.

Self, J. A. (1986). The Application of Machine Learning to Student Modelling. *Instructional Science* **14**, 327—338.

Self, J. A. (1987). User Modelling in Open Learning Systems. In Whiting, J. and Bell, D. (Eds.), *Tutoring in Monitoring Facilities in European Open Learning.* Elsevier.

Self, J. A. (1988a). Knowledge, Belief and User Modelling. In O'Shea, T. and Sgurev, V. (Eds.), *Artifical Intelligence III: Methodology, Systems, Applications.* North-Holland.

Self, J. A. (1988b). Student Models — What User Are They?. In Ercoli, P. and Lewis, R. E. (Eds.), *Artifical Intelligence Tools in Education.* North-Holland.

Self, J. A. (1988c). The Use of Belief Systems for Student Modelling. Technical report. Centre for Research on Computers and Learning, University of Lancaster.

Self, J. A. (1989). The Case for Formalizing Student Models (and Intelligent Tutoring Systems Generally). *Proceedings of Fourth International Conference on AI and Education.* Amsterdam.

Self, J. A. (1990). Bypassing the Intractable Problem of Student Modelling. In Frasson, C. and Gauthier, G. (Eds.), *Intelligent Tutoring Systems: At the Crossroads of AI and Education.* Ablex.

Self, J. A. (1991). Formal Approaches to Student Modelling. In McCalla, G. and Greer, J. (Eds.), *Student Modelling.* Springer-Verlag (to appear).

Sleeman, D. (1983). Inferring Student Models for Intelligent Computer-Aided Instruction. In Michalski, R. Carbonell, J. and Mitchell, T. (Eds.), *Machine Learning.* Tioga Publishing Co.

Stevens, A. L. and Collins, A. (1979). Misconceptions in Students Understanding. *International Journal of Man-Machine Studies* **11**, 145—156. Also in Sleeman, D. and Brown, J. S. (Eds.), *Intelligent Tutoring Systems.* Academic Press, London.

Van Lehn, K. (1982). Bugs Are Not Enough: Empirical Studies of Bugs, Impasses and Repairs in Procedural Skills. *Journal of Mathematical Behaviour* **3**, 3—72.

Van Lehn, K. (1983a). Felicity Conditions for Human Skill Acquisition: Validating an AI-based Theory. Tech. Rep. No. C15-21. Xerox Research Center, Palo Alto, CA.

Van Lehn, K. (1983b). On the Representation of Procedures in Repair Theory. In Ginsburg, H. P. (Ed.), *The Development of Mathematical Thinking.* Academic Press, London.

Van Lehn, K. (1988). Towards a Theory of Impasse-driven Learning. In Mandl, H. and Lesgold, A. (Eds.), *Learning Issues for Intelligent Tutoring Systems.* Springer, New York.

Van Lehn, K. (1990). *Mind Bugs: The Origin of Procedural Misconceptions.* MIT Press, Cambridge, MA.

Van Lehn, K. (1991). Two Pseudo-students: Applications of Machine Learning to Formative Evaluation. In Lewis, R. and Otsuki, S. (Eds.), *Advanced Research on Computers in Education.* Elsevier Science Publishers.

Van Lehn, K. and Garlick, S. (1987). Cirrus: An Automated Protocol Analysis Tool. *Proceedings of the Fourth Machine Learning Workshop.* Morgan Kaufman, Los Altos, CA.

Van Lehn, K., Jones, R. M. and Chi, M. T. H. (1991). Modelling the Self-explanation Effect with Cascade 3. In Hammond, K. and Gentner, D. (Eds.), *Proceedings of the Thirteenth Annual Conference of the Cognitive Science Society.* Erlbaum, Hillsdale, NJ.

Whitehead, A. N. (1932). *The Aims of Education.* Benn, London.