Student diagnosis in practice; Bridging a gap

Eva L. Ragnemalm
E-mail: evalu@ida.liu.se

Abstract

This paper presents a novel framework for looking at the problem of diagnosing a student's knowledge in an Intelligent Tutoring System. It is indicated that the input and the conceptualisation of the student model are significant for the choice of modelling technique. The framework regards student diagnosis as the process of bridging the gap between the student's input to the tutoring system, and the system's concept and representation of correct knowledge. The process of bridging the gap can be subdivided into three issues, data acquisition, transformation and evaluation, which are studied further. A number of published student modelling techniques are studied with respect to how they bridge the gap.

This work has been financed by NUTEK.

Accepted for publication in User Modeling and User–Adapted Interaction.

IDA Technical Report 1995
LiTH-IDA-R-95-36
ISSN-0281-4250

Department of Computer
and Information Science,
Linköping University,
S-581 83 Linköping, Sweden

Student diagnosis in practice; Bridging a gap

E. L. Ragnemalm

Department of Computer and Information Science, Linköping University, S-581 83 Linköping, Sweden
E-mail elu@ida.liu.se

Abstract: This paper presents a novel framework for looking at the problem of diagnosing a student's knowledge in an Intelligent Tutoring System. It is indicated that the input and the conceptualisation of the student model are significant for the choice of modelling technique. The framework regards student diagnosis as the process of bridging the gap between the student's input to the tutoring system, and the system's conception and representation of correct knowledge. The process of bridging the gap can be subdivided into three phases, data acquisition, transformation and evaluation, which are studied further. A number of published student modelling techniques are studied with respect to how they bridge the gap.

Key words: Student diagnosis, Student modelling, Intelligent Tutoring Systems

1 Introduction

One of the important components in an Intelligent Tutoring System, ITS, is a *Student Model*. This is a collection of data about the student that is used by the other components in the system in a number of different tasks. Lists of such tasks can be found in student modelling literature (e.g. VanLehn, 1988, Nwana, 1991, Holt et al., 1994) and include such items as planning the sequence of instruction, remediating misconceptions, generating feedback and explaining the reason for an error made by the student.

In this paper *student diagnosis* is defined as the abstract process of gathering information about the student and turning that information into the basis for instructional decisions made in the ITS. Sometimes this is called student modelling in the literature. Here, *student modelling* is used to refer to the concrete process of initially constructing a student model in an ITS and maintaining it during runtime.

There is little consensus in the literature on how to do student diagnosis or modelling. Many techniques have been implemented and used successfully but there are no general rules for how to select a technique in a specific problem situation. Verdejo (1994) summarises previous work into 12 dimensions for characterizing user models in general. These dimensions describe properties of user models themselves, but only one refers to things outside the model (that is *Use*, which can be either *descriptive* or *predictive*). The 12 dimensions by Verdejo include among others the three dimensions presented by Rich (1983) and the fourth that was added by Sleeman (1985). Vassileva (1991) adds two "technical" dimensions that have some similarity to that presented here. The distinctions are discussed in section 2.1. Dillenbourg and Self (1992a) describe a framework for student

modelling that takes into account the distance, or *discrepancies*, between the learner, the system and the system's model of the learner. Self (1994) presents a formal description of student modelling that regards student modelling as supporting the interaction between the belief systems of two agents. While frameworks such as these are useful for describing existing models and techniques, what is missing is a consideration of the problem situation. Van Lehn (1988) begins to take the problem situation into account by proposing input to the system as one characteristic, but divides the types of student models differently from the framework presented here. This paper presents a framework that takes the problem situation into consideration in order to provide some assistance in selecting student diagnosis and modelling technique in a given situation.

Two characteristics of the problem situation appear to influence the type of technique chosen. They are *available input* and *type of model to be produced*. Also, the availability of pedagogic expertise appears to influence the choice. The presented framework enables the positioning of existing techniques with respect to these characteristics. This framework does not claim to be the one and only framework for analysis of student diagnosis, but illustrates the importance of these characteristics for the choice of student modelling technique.

Constructing a student model is a complex task for a number of reasons. In principle it parallels what a teacher does when she[1] evaluates a statement from a student in order to critique the student's effort or replan instruction. The task can actually be seen as the process of bridging the gap between the student's input and some conception of correct knowledge. Existing student modelling techniques provide different means of closing this gap.

This paper will briefly discuss the content of a student model and then introduce the concept of the gap. A subdivision of the student modelling process is then presented, and the corresponding subproblems described. A number of existing student modelling techniques are then examined to show how they bridge the gap, and the dependence on the input and type of model is analysed.

Two general kinds of knowledge will be discussed in this paper. They are *subject matter knowledge* and *pedagogic content knowledge* (Shulman, 1986, Wilson et al., 1987, Wasson, 1990). The term *subject matter knowledge* will be used to indicate the knowledge that the student is supposed to learn (the subject of the lessons), including the knowledge necessary for applying the new knowledge (but which the student is expected to know). An example of subject matter knowledge in the domain of LISP programming is that a recursive function always has a base case and a recursive case.

Pedagogic content knowledge is the pedagogical knowledge necessary to convey the subject matter to the student. Here pedagogic content knowledge is taken to include both organization of the subject matter knowledge (e.g. issues or lessons to be learned) and diagnostic knowledge (i.e. how to relate a student's answer to his underlying reasoning). Examples of pedagogic content knowledge in the domain of LISP programming are: "an item to teach is how to analyse CDR recursion" and "if a student can not locate a bug in the base case he probably does not know how to analyse CDR recursion". When referring to subject matter knowledge and pedagogic content knowledge collectively, the term *Tutor's knowledge* will be used.

---

1. In this paper, "she" was arbitrarily chosen to signify teachers and "he" students.

2 Content of a student model

The purpose of the student model is to describe the student in some manner. It is used to guide decisions concerning how to adapt the tutoring, how to provide a tailored explanation or hint, or to provide the right level of coaching. The exact use depends on the design of the rest of the tutoring system. The types of student modelling techniques considered in this paper are *internal* (not embedded in the functionality but explicitly represented) in Verdejo's terms (Verdejo, 1994).

What information the student model should be designed to contain depends on the set of instructional tasks to be supported as well as on the characteristics of the subject matter knowledge. For example, if the system is to adapt its instruction to the student's preferred learning style (such as holistic vs serialised approaches or explorative vs directed learning), it needs to know his preferences. If the system is to increase the student's motivation it needs to know what motivates the particular student. For instance, people react differently to challenge and competition, and for some curiosity and confidence are important motivators (del Soldato, 1992). This paper deals with models containing information about the student's knowledge of the subject matter.

2.1 Knowledge content

Some researchers propose that to be really useful a student model needs to represent the student's actual beliefs, his mental image of the tutoring domain (e.g. Baril et al., 1991).[2] This implies that the student model should be made up of subject matter knowledge (or be overlaid on the ITS's subject matter knowledge). Thus a *subject–matter model* contains a knowledge base of some kind, for instance a production rule model or a frame-based model. An example of a system with such model is the LISP Tutor (Corbett and Anderson, 1992). Representing the knowledge itself also implies that no evaluation has been made of its correctness (it is done later). In this approach generative techniques can be used to build the knowledge, which means that the student's misconceptions can always be represented (Self, 1988). These models are also runnable (or could be), and can for instance be used to limit the search for an explanation by predicting the student's answer to questions.

Many approaches model the student in terms of the pedagogic content knowledge instead of subject matter knowledge. A *pedagogic content model* is organised according to instructional goals, and the information stored pertains to the student's mastery of these goals. Since what is stored is the student's mastery, these models store evaluations of the student's knowledge (using symbolic, binary or numeric values). These models are descriptive: they contain descriptions of the knowledge instead of the knowledge itself. They are not runnable. For example, West (Burton, 1982) models the student's strength or weakness in the different skills that are needed for playing "How the West was Won". In these models only misconceptions predicted at design time can be modelled, since the pedagogic issues to be modelled are predetermined.

Pedagogic–content models may be either numeric or symbolic while subject–matter models are always symbolic. Subject–matter models can be said to *contain* the functionality of the student's knowledge while pedagogic–content models *describe* the functionality.

2. A student model actually represents *the system's beliefs* about *the student's beliefs*. In the following, this is assumed and not explicitly mentioned.

Vassileva (1991) defines a space of student models that spans from models that contain "the student's actual domain knowledge" (page 203) to models that contain student characteristics, where examples of characteristics include the student's abilities and his learning rate. The division made here is similar but differs in two important respects from Vassileva's. First, characteristics such as learning rate and other cognitive features of the student are not included in the models considered here (indeed, in the work that Vassileva describes later they use two student models, where one represents the student's knowledge and the other the features). Second, Vassileva does not seem to distinguish between representing the student's knowledge and representing his mastery of the knowledge as is done above.

The normative and descriptive student models defined by Laurillard (1993) resemble the subject–matter and pedagogic–content models respectively, in that they represent versus describe the student's knowledge. Laurillard's normative models, however, include only models that are overlays on correct knowledge and appear to exclude models that contain incorrect knowledge. The subject–matter model defined here can contain misconceptions and incorrect knowledge.

Pedagogic–content models become practically indistinguishable from subject–matter models when there is a one–to–one relation between pedagogic issues and the system's representation of the subject–matter knowledge and the model is an overlay on these.

2.2 Granularity

Student models may also vary with respect to the granularity of the knowledge to be represented. Some models have a granularity corresponding to a few major pedagogic issues while some represent the subskills of subskills (for the procedural case). Some models also provide many levels of granularity by representing a hierarchy. The level of granularity in the student model should depend on the desired tutor's instructional capability (cf. Dillenbourg and Self, 1992a). For example, an ITS that teaches algebra would not necessarily benefit from diagnosing the student's understanding of the subtraction required to simplify the equation. Knowing that the student doesn't know something is irrelevant if that knowledge cannot be used. So, presumably, the level of detail necessary in a student model corresponds to the remedial knowledge we are willing to put into the system.

This paper will not go into the issue of the content of the student model any further since that is a complex problem in itself, but will focus on the technical problem of building that model, the student diagnosis process. To provide a base for the further discussion we will assume that the student model is deemed necessary, and that it needs to contain some representation of what knowledge the student is supposed to learn. The model is either represented in terms of pedagogic content knowledge (pedagogic content model) or in terms of the student's (believed) subject matter knowledge (subject matter model).

3 The Student Diagnosis Process

How ever the system designer chooses to represent the Tutor's knowledge and the student model, we still have the problem of diagnosing the state of the student's knowledge at the required level of granularity. Of course the problems of choosing a knowledge representation for the student model and designing the process of updating it are not neces-

sarily (or even preferably) dealt with separately in practice, but the distinction is still conceptually valid. They are two problems, even though the contents strongly influence the construction process and the contents may be modified because of construction problems.

Early instructional systems solved the student diagnosis problem by not representing the Tutor's knowledge in the system. Instead the student's possible or permissible answers were predicted at system design time, along with their implications for the student's knowledge. These implications were then realised as remediating actions, and connected to the answers. In these systems, thus, the student diagnosis is done at design time. A recent example of such a system is the PETAL's (Bhuiyan et al., 1991) use of different tools adapted to different mental models found in a small student corpus. This is sometimes referred to as external student modelling (e.g. Verdejo, 1994).

Later, when explicit representation of subject matter and pedagogic content knowledge was introduced into the ITS's (for some the definition of ITS), a problem arose: How to relate the student's responses to some inspectable representation of Tutor's knowledge. There is a gap between the student's input to the system, and the stored knowledge. The problem of doing this mapping, bridging this gap, is the problem of diagnosing the student's knowledge. Figure 1 illustrates the difference between the information provided by the student and the knowledge stored in the computer, and shows the student diagnosis process which bridges that gap so that instructional decisions can be made.

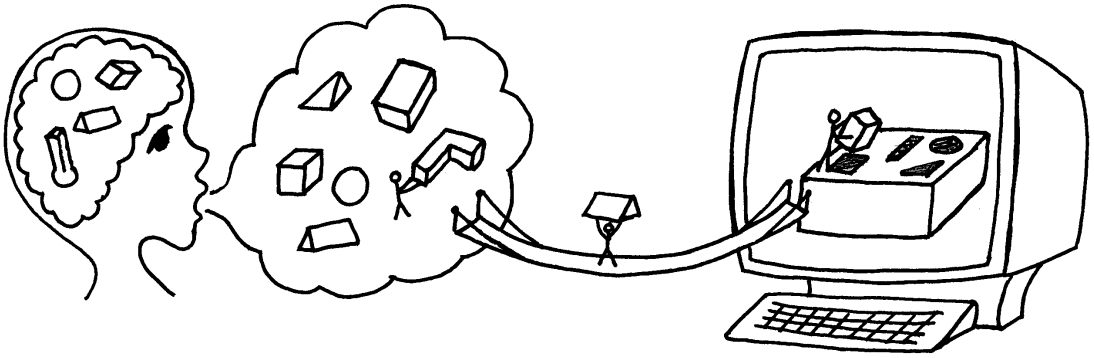

Figure 1. Student diagnosis bridges the gap between student input and Tutor knowledge.

For an illustration of what student diagnosis involves, consider the following interaction between a teacher and a student:

Teacher: *So, if both of us were floating free in space, and you pushed me away from you, how would I move?*
Student: *You'd move in the direction I pushed.*
Teacher: *And how would you move?*
Student: *I'd go nowhere.*

What does this exchange tell the teacher about the student's knowledge?[3] The process of answering that question requires bridging the gap between the student's statements and the teacher's own knowledge of mechanics (how ever it is represented in her brain).

In an ITS, student diagnosis is the whole process of collecting input to the ITS in whatever form it is available, mouse clicks, sound patterns or characters, to the inferences

3. This student probably doesn't know that every action has a reaction.

about what the student knows or does not know, which guide the instructional activities of the tutor. The problem of interpreting data (mouse clicks, sound patterns etc.) into actions or concepts within the domain is often not a trivial problem, but will not be dealt with here. *Student input* as used in the following discussion will refer to the inferred basic actions or concepts that exist in the domain.

Accepting that there is a gap, it can be divided into three subproblems that must be dealt with in a system that is to use its student model for instructional decisions.

4 Phases in bridging the gap

The bridging of the gap between the Tutor's knowledge and the student's input can be divided into three phases, as illustrated in figure 2. The first phase is data acquisition, the second is the transformation, the third is the evaluation with respect to the Tutor's knowledge. Depending on where a system puts its main effort, the other phases are made easier. The actual student model constructed can be located before or after the evaluation step, depending on its content, see figure 2.

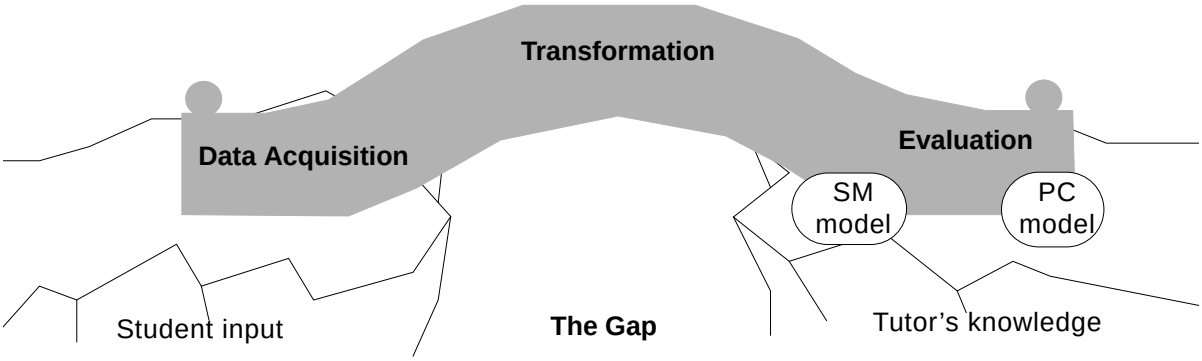

Figure 2. The three issues in bridging the gap and the two different types of student models, subject matter (SM) and pedagogic content (PC), respectively.

If the student model represents the student in terms of subject–matter knowledge (SM model) the actual initialization (and later revision) of the student model occurs between the transformation and the evaluation phase as shown in figure 2. If it is represented in terms of pedagogic content knowledge (PC model), initialization or revision of the student model is done after the evaluation.

The following three sections will discuss data acquisition, the problem of transforming the collected data, and the evaluation problem, respectively.

4.1 Data acquisition

Data acquisition refers to the process of collecting input for the student modelling process. It is in practice parallelled by a data interpretation or abstraction phase, where the actual interface activities are converted to the concepts used for reasoning. The output of the data acquisition phase is one or a sequence of domain concepts. With respect to the data acquisition, there are three major problems. The first concerns the reliability of the input, the second the level of detail of data, and the third the amount of data available.

The basic question to ask with regard to the reliability of data is "How can we be sure that the student's error is not a slip?"[4]. Slips should not be treated as severely as consistent mistakes, which indicate misconceptions or bugs. This creates a conflict between the

desire to adapt quickly to changes in the student's knowledge, and the desire to ignore minor slips. The problem is often dealt with by varying the manner in which the old student model is updated (i.e. it is not dealt with in the data acquisition phase). Suggestions of solutions to this problem are for instance the use of fuzzy logic (Derry et al., 1989), probabilistic student models (e.g. Martin and VanLehn, 1993, Villano, 1992, Petrushin and Sinitsa, 1993, Sime, 1993) and use of accumulated pro and con arguments of differing strengths (Murray, 1991).

With respect to the level of detail of data, three different levels of granularity of input are defined by VanLehn (1988) as *Final States*, *Intermediate States* and *Mental States*. Input of final states implies that the input reflects the solution to a problem in the form of an answer to a question or an action in the domain. The definition of problem here is the exercise the student is working on in order to apply the skills or knowledge he is to learn. Input at the level of mental states implies that the input reflects actions or concepts at every state transition that the student makes in the problem solving process. Intermediate state input is input at the level of answers to subproblems in the procedure.

For an illustration of the different levels of granularity, consider the following hypothetical example.

Given the problem of adding 12 to 19:
Mental states: (first column 2+9 equals 11)
(11 is one carry and one to write down)
(second column 1+1=2)
(2 and the carry, 1, gives 3)
(second column gave 3 and I had 1, that is 31).
Intermediate states: Carry = 1, Answer = 31
Final state: the answer, 31

The two first levels of input require a detailed study of the cognitive task the student is performing in order to allow representation of the correct set of states.

There is also the problem of the amount of data. What is sufficient naturally depends on the conceptualisation and granularity of the student model. Final states (answers) are often available, while other levels are harder to access. In the case of a fine grained subject–matter model which is to be compared to a cognitive model of expert performance, we would ideally like to tap the student's mental states. In cases where the task is complex and mostly mental, this data is difficult to access, and specific techniques to refine the input can be used. Intermediate states can be derived or deduced from final states, for example Plan Recognition (Genesereth, 1982). Self (1988) proposes that we get the student to supply more information rather than guess, but even so there are a number of ways to do that. One approach is to design a task oriented interface that allows an increase of input to the system which at the same time serves as a tool in the user's problem solving activity. Some examples of this are the planning tool EPIC (Twidale, 1989) in which the user selects subgoals on which to work and the LISP Tutor's structured editor which traces the program construction symbol by symbol (e.g. Corbett and Anderson, 1992). Goodyear and Tait (1991) suggests that the surreptitious diagnosis be abandoned and instead the student model should be openly negotiated with the student. The demands this would put on the system's dialogue capabilities have not been studied.

4. Another source of uncertainty in some systems is the conversion of the actual input to domain concepts, but that problem is beyond the scope of this paper.

4.2 Transformation

Transformation refers to the process of analysing either the input from the data acquisition phase or the Tutor's knowledge in order to extract pertinent information for judging the student's skill. Transformation across the gap can thus be done in either direction. Either the student's input can be converted to a representation closer to the representation of the tutor's knowledge (bringing the input to the representation), or the Tutor's knowledge can be converted to a format closer to the student's input, for instance by producing a correct answer (bringing the representation closer to the input).

The difference between these two directions is illustrated in figure 3 and figure 4.

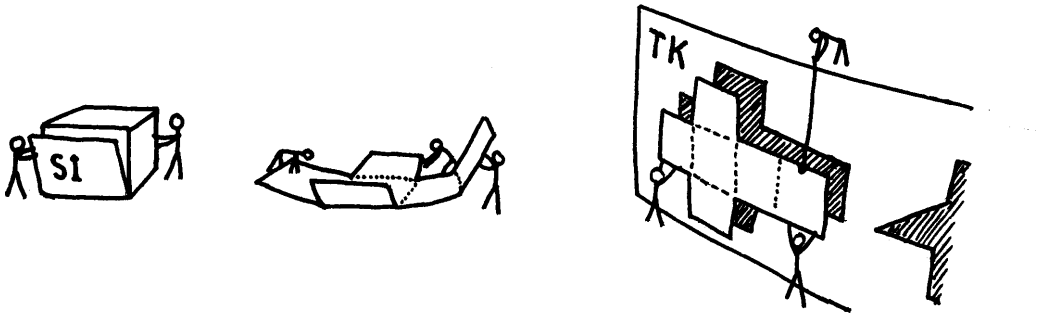

Figure 3. The task of transforming the student's input (SI) and comparing it to the Tutor's knowledge (TK).

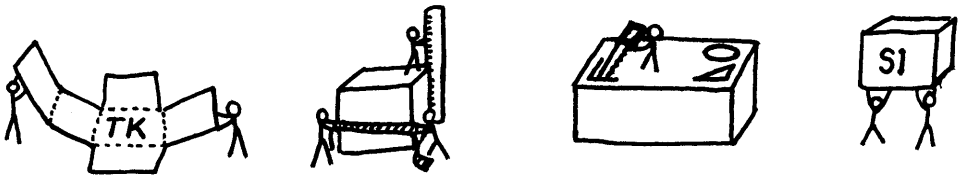

Figure 4. The task of transforming the Tutor's knowledge (TK) into criteria or recognisers for student input (SI).

The first alternative, bringing the input to the Tutor's knowledge, is equivalent to representing the student's beliefs in the same manner as the domain knowledge in the ITS (and then comparing them consists of comparing the knowledge representations). That implies that the student model in this case is a subject–matter model. The other direction, bringing the domain knowledge to the input, would correspond to activating the domain knowledge to analyse the student's input. This can either be by producing an answer to the same problem and comparing it to that of the student, or by producing conditions or recognisers that work on the student's input. The student model in this case can be either a subject–matter model or a pedagogic–content model.

Subject–matter models can thus be constructed through transformations in either direction, while pedagogic content models are only constructed through transforming the tutor's knowledge to conditions (called recognisers or observers in the literature) for the input.

The application of machine learning techniques to the construction of subject matter models has been tried, and seems an interesting avenue to explore. Techniques that have been studied are for instance Induction (Langley and Ohlsson, 1984, quoted in VanLehn, 1988), Theory revision (Baffes, 1994) and Focussing (Self, 1986). A possible problem, though, is the instability of student knowledge. Kono et al. (1993) observe the

necessity of handling contradictions in the student model. Not only do students slowly acquire new knowledge and unlearn old misconceptions, but when initially learning something new the behaviour will be unstable and the new knowledge sometimes applied correctly, sometimes not. Kono et al. (1993) propose the use of truth maintenance systems to deal with the problem of revising these models.

4.3 Evaluation

Evaluation refers to the process of evaluating the student's knowledge or behaviour with respect to some conception of correct knowledge or behaviour. When modeling the student's skill in terms of pedagogic–content knowledge, the transformation and evaluation are easily lumped together. The correctness (or incorrectness) of pedagogic–content knowledge is normally implicit in the definitions. Pedagogic–content knowledge is used to identify the instructional issues in the transformation step, and that reduces the evaluation to merely inspecting the values for the issues in the student model. These values are sometimes probabilistic, and can be updated by using Bayesian Belief Networks to propagate new evidence (e.g. Martin and VanLehn, 1993, Villano, 1992)

When modelling student's subject matter knowledge the student model is evaluated through comparison to the expert model. If the student's knowledge is considered a strict subset of the tutor's subject matter knowledge it is called an *Overlay model*. If we (as system developers) accept the possibility that the student may have knowledge that the tutor does not, we speak of a *Perturbation model* (see e.g. Wenger, 1987, VanLehn, 1988). The evaluation of deviations from the tutor's knowledge is also at issue — do we regard deviations as strictly erroneous or are we open to the thought that the student may possess correct knowledge that the tutor does not?

These questions actually result in four different views, as illustrated in figure 5, on the comparison between student knowledge and the tutor's subject matter knowledge. The student's knowledge may be perceived as a strict subset of correct subject matter knowledge (*Strict Overlay*, see figure 5a). In this case the student model is incapable of representing knowledge that is not in the subject–matter knowledge base, and student errors are assumed to derive from lack of such knowledge. When a static bug library is used, the student's knowledge is seen as a strict subset of the combination of correct and incorrect domain knowledge. This model can be called *Extended Overlay*, see figure 5b for illustration. Here student errors are assumed to be caused by the known (to the system) buggy knowledge. Both types of overlays implicitly assume that the system possesses all the knowledge that the student may have.[5]

If the student's knowledge is dynamically generated from smaller pieces we may get knowledge that is not recognised by the system. That knowledge may be perceived as automatically wrong, on the assumption that the system does recognise all correct knowledge. In this case the model can be called *Simple Perturbation*, see figure 5c. Koegel et al. (1989) proposes that if the ITS does not take the role of an omnipotent oracle, the status of the unrecognised knowledge is debatable. This type of model can be called *Advanced*

5. If no distinction is made between subject–matter models and pedagogic–content models, what has been described here as pedagogic–content models would also be classified as Overlay models (Strict or Extended depending on the inclusion of misconceptions).

Perturbation, see figure 5d. To be effective this approach probably needs some knowledge of common errors, to be able to rule them out.

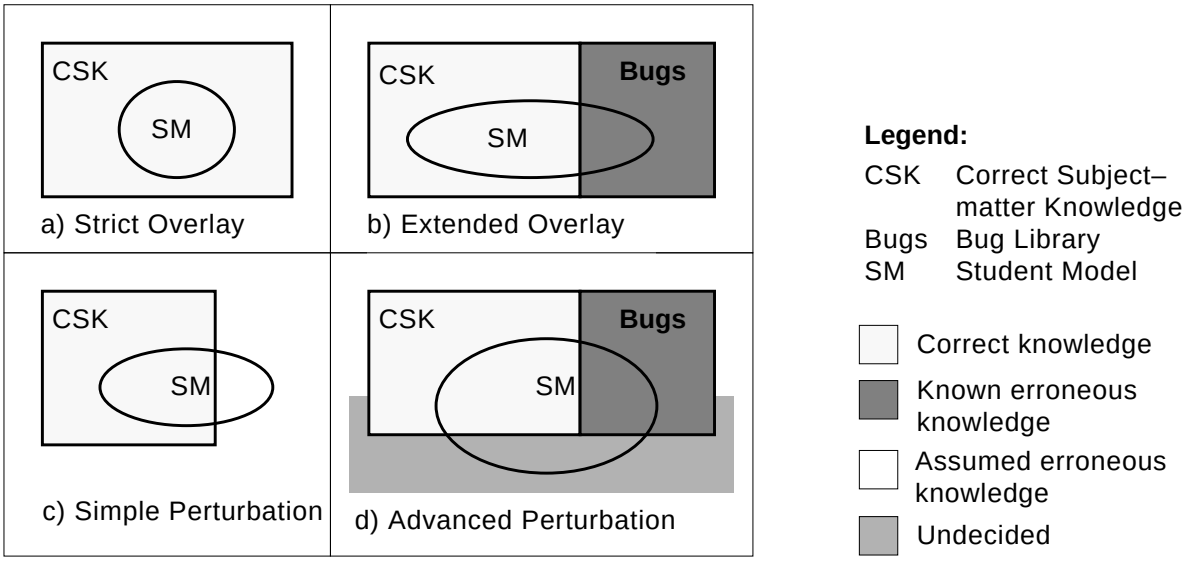

Figure 5. The four different views on the relation between student knowledge and the domain knowledge.

Whether or not the model is a pedagogic content model or a subject matter model, there are also the problems of more or less correct answers, a grey zone. If the student's answer was in the right category but he picked the wrong instance, is this right or wrong? If the student's answer contained two correct components and one erroneous, how wrong is that? Solutions to this have been proposed by using hierarchies of knowledge (Lesgold et al., 1989, Greer et al., 1989). Another aspect of evaluation is when evidence is of unequal strengths — e.g. recognition of a concept is easier than to recall and use it. Judging competence by using arguments of differing strength has been proposed by Murray (1991). There can also be different degrees of belief in the model, with probabilistic values attached to the items in the model.

These three components of the gap, data acquisition, transformation and evaluation must all be bridged in one manner or another before the abstract process of student diagnosis is complete. The ways this can be done appear unlimited, but some general trends can be identified. In the following, a number of existing techniques are examined as to in what manner they bridge the gap, and some trends pointed out.

5 Techniques that bridge the gap

This section will show how a sample of existing student modelling techniques relate to the gap. First the collection of nine general student modelling techniques identified by Van Lehn (1988) will be examined. After that a few more recent techniques will be used to illustrate how different techniques could be combined in bridging the entire gap.

In the context of using the student model, the entire gap is bridged, from some input up to and including the evaluation phase. In studies of student modelling mechanisms in isolation, the entire process is not always included. For instance, when discussing techniques for generating subject–matter models the evaluation phase is often disregarded..

Bridging the gap is done using the same kind of knowledge as is represented in the model. In this context it is not self–evident where knowledge of common misconceptions

or bugs should be placed, but in this work they will be viewed as subject–matter knowledge, albeit erroneous knowledge. The knowledge that they exist is pedagogic content knowledge, while the knowledge of exactly what they are takes the form of subject matter knowledge.

The gap and the important characteristics is illustrated in figure 6. On the side of student input, the data acquisition phase includes the three different levels of input granularity. On the side of Tutor's knowledge, the subject–matter model is constructed before the evaluation phase and the pedagogic–content model after. The fact that subject–matter models are relatively fine grained while pedagogic–content models of all granularities exist will be seen in the following examples.

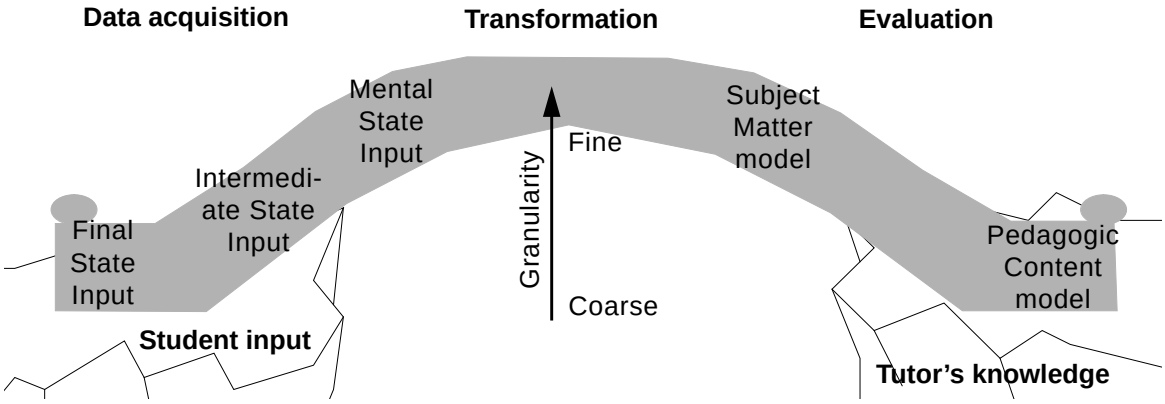

Figure 6. The characteristics of existing student diagnosis techniques with respect to the gap: the data acquisition granularity and the models' type and granularity.

First we will look at a number of more general techniques for student diagnosis in order to show how they contribute to bridging the gap. These techniques are described graphically in figure 7, on page 12. VanLehn (1988) identified nine techniques used for student diagnosis at that time. They can be described as follows:

Model Tracing is a technique that presumes that the input reflects the different steps in the student's reasoning, or the different mental states in the problem solving process. It also assumes that the subject matter knowledge is a cognitive model of the task represented as rules for state transitions. The idea is to find the rule for a state transition that corresponds to the student's state transition, whether it be a correct rule or a mal-rule. In order to reduce the search space, it first predicts what rules the student can possibly apply, and then searches among them. This technique transforms the subject matter knowledge to possible states for comparison to the input mental states. A system that uses Model Tracing is the LISP Tutor (Anderson et al., 1990, Corbett and Anderson, 1992).

- This technique starts at fine grained subject matter knowledge and bridges to mental states, see the arrow Model Tracing in figure 7. It requires a runnable cognitive model of the task.

Condition Induction also assumes that the input corresponds to sequences of mental states, and that the subject matter knowledge is represented in rules for state transitions. But instead of searching for a suitable rule, it generates a rule that explains the student's state transition in order to compare it to the domain knowledge. Thus this technique transforms mental states to rules that are compared to the subject matter knowledge. A student modelling program that uses Condition induction is ACM (Lang-

ley and Ohlsson, 1984, referred in Wenger, 1987).

- This technique bridges from mental states to fine grained subject matter knowledge, see the arrow Condition Induction in figure 7.

Path Finding is a technique that identifies the path or sequence of mental states that best explain input consisting of final states. This technique transforms input of final states to sequences of mental states, that can then be handled with Model Tracing or other techniques that require input of mental states. A system that uses Path Finding is DPF (Ohlsson and Langley, 1985, referred in VanLehn, 1988).

- This technique bridges final states to mental states, see the arrow Path Finding in figure 7.

Plan Recognition is similar to Path Finding in that it is a front end to Model Tracing. It is especially useful for certain domains, but requires that the knowledge be procedural and hierarchically organised, and that intermediate states are available. It generates a plan that explains the observed student performance, which can then be passed to a Model Tracing module. Thus this technique transforms intermediate states to mental states that can then be handled with a technique like Model Tracing. A system that uses this technique is MACSYMA Advisor (Genesereth, 1982).

- This technique bridges intermediate states to mental states, see the arrow Plan Recognition in figure 7.

Issue Tracing is a technique that takes intermediate states as input, and maps them onto instructional issues (e.g. subskills). The definition of these issues is done at design time using a detailed cognitive task analysis, and considered pedagogic content knowledge. Pedagogic content knowledge is also used to define the recognisers that detect these issues in the input. This technique, therefore, uses pedagogic content knowledge to examine input of intermediate states. West (Burton, 1982) uses this technique.

- This technique bridges more coarse pedagogic content knowledge to intermediate states, see the arrow Issue Tracing in figure 7. Pedagogic expertise is required to define the recognisers at design time.

Expert Systems is a technique that identifies and evaluates the inferences that the student makes on the basis of which ones the expert module would have used. It identifies the inference rules that could possibly have been used and then applies (domain dependent) knowledge about inference rules to judge the correctness of the student's rules. In effect it works in two steps, first it takes the final state information (the student's hypothesis) and derives the possible inference rules, and then uses pedagogic content knowledge (knowledge about how to evaluate the inferences, i.e. recognisers) to evaluate those rules. Guidon is a system that uses this technique (Clancey, 1982).

- This technique bridges from medium coarse subject–matter content (granularity similar to Issue Tracing according to Van Lehn, 1988) to final states, and then from pedagogic content knowledge to subject matter knowledge, see the arrow Expert Systems in figure 7.

Decision Tree, **Generate and Test** and **Interactive Diagnosis** are identical in that the techniques utilise subject matter knowledge (knowledge about bugs, erroneous procedures encountered in students) to predict the final states, the answers. The Decision Tree technique uses predefined recognisers for bugs and common combinations, while the Generate and Test technique generates recognisers (answers) for bug

combinations dynamically. The Interactive Diagnosis technique dynamically adapts the exercises in order to reduce the complexity of the problem of identifying bug combinations to explain the student's answers. These techniques run the subject–matter knowledge on the exercises to get final states that are compared with student input. Systems that use these methods are BUGGY, DEBUGGY and IDEBUGGY (Burton, 1982, Brown et al., 1982). A cognitive model of the domain (including the bugs) is required, in the Generate and Test and the Interactive Diagnosis techniques it must be runnable.

- These techniques bridge from fine grained subject matter knowledge (albeit augmented with knowledge of bugs or bug parts) to final states, see the arrow Decision Trees, Generate & Test, Interactive Diagnosis in figure 7.

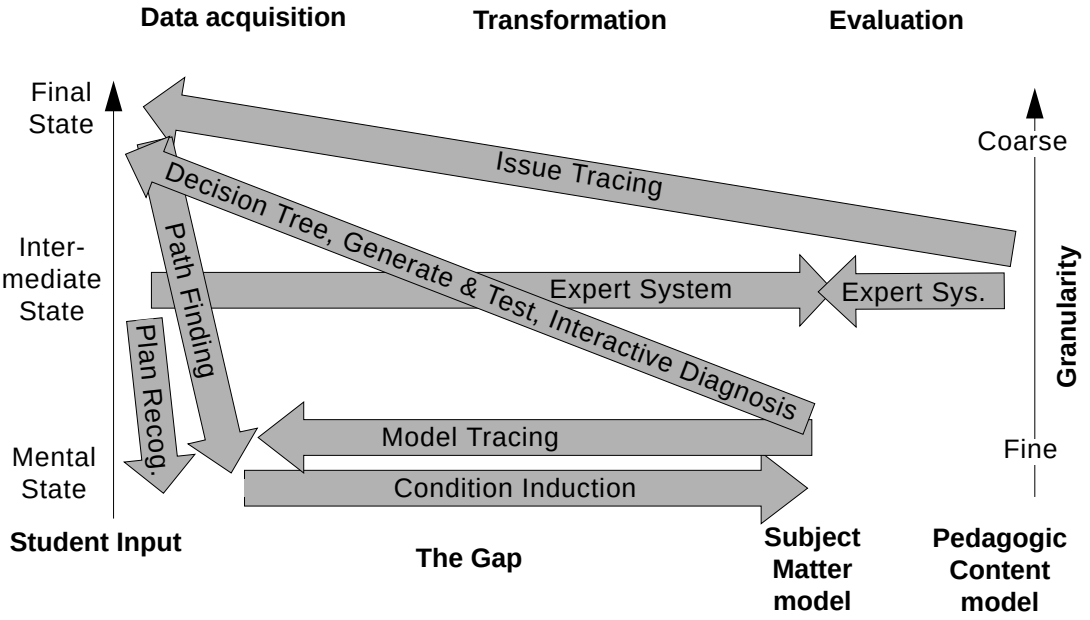

Figure 7. The set of more general techniques do bridge the gap very differently.

In addition to the techniques above, some interesting variations and more general methods have recently been introduced:

ASSERT (Acquiring Stereotypical Student Errors using Revision of Theories; Baffes and Mooney, 1992, Baffes, 1994) is a student diagnosis technique based on the machine learning technique Theory Revision. It uses the student's input in the form of classifications of objects to revise a theory (in this case expressed as rules). The theory is then made consistent with the input, thus creating a theory of student knowledge expressed in rules. These rules can then be compared to the subject matter knowledge. The input can be of any granularity so long as the subject matter knowledge expressed at the same level is a good model of performance. This technique is related to the Condition Induction technique, but uses a different method to generate the representation of student knowledge.

- This technique bridges from any granularity input to the same level of subject matter knowledge, see the hourglass–arrow ASSERT in figure 8.

EPIC, (Explicit Planning and Instantiation by Computer; Twidale, 1989) uses a tool interface technique that allows the user to input intermediate states during problem solving. These intermediate steps are in fact subgoals in the overall plan. It is stated

that this information could be used to provide more data for a student model. This technique thus reduces final states to intermediate states, which can then be directly compared to subject matter knowledge.

- The technique used here bridges from final states to intermediate states, see the arrow EPIC in figure 8.

Else, the Learning Companion approach to student modelling (Ragnemalm, 1994, Ragnemalm, 1995) suggests the use of collaboration as a source of information for the diagnosis process. Else is a computer based Learning Companion (Chan and Baskin, 1988, Dillenbourg and Self, 1992b) which serves as the collaboration partner.

- The technique refines final states to intermediate states, see the arrow Else in figure 8

SCENT's diagnosis module takes as input a student's finished LISP program, and searches it for pieces of code it recognises as evidence for specific programming strategies (Greer et al., 1989). SCENT models the student in a two–dimensional abstraction hierarchy of "strategy objects" each associated with one or many functions that identify that particular strategy from the code (or evidence from other levels in the hierarchy). The abstraction hierarchy ensures that on some level of abstraction, *any* student input will be recognised. Pedagogic–content knowledge, both correct and buggy, is used in the construction of the functions which determine if a strategy is recognised.

- This technique bridges from pedagogic–content knowledge of variable grain size to final states, see the broad–bottomed arrow SCENT in figure 8. Pedagogic expertise is required to define the recognisers at design time.

FITS, (Fraction Intelligent Tutoring System; Nwana, 1991), is a system that teaches the addition of fractions. It uses a two-stage diagnosis approach. First students are presented a pre-test, which is analysed off line using a technique similar to DEBUGGY. At runtime this model is updated using intermediate state input that is evaluated using predefined connections between exercises and domain concepts. An example of a concept is "add equivalent fractions" and there are 24 concepts in total. The predefined connections are constructed from pedagogic content knowledge (each exercise has an associated set of concepts involved).

- The runtime technique used here bridges from pedagogic content of rather coarse grain to intermediate states, see the arrow FITS in figure 8. Pedagogic expertise is required to define the recognisers at design time.

Sherlock II's student modelling facilities uses fuzzy probabilities to describe the system's belief in the student's mastery of knowledge variables (Katz et al., 1992). A knowledge variable can be "ability to use the oscilloscope" and can be aggregated into higher level variables. The system uses predefined recognisers to asses the strength of evidence for or against the lowest level variables based on the student's activities in the simulated test environment.

- This technique bridges from pedagogic–content knowledge of fine granularity to intermediate state input, see the arrow Sherlock in figure 8. Pedagogic expertise is required to define the recognisers at design time.

OLAE, (Martin and Van Lehn 1993), is an assessment system for physics problem solving. It uses a bayesian net to represent a student's problem–solving knowledge, knowledge of facts, use of knowledge and also activities. The representation is descriptive, but the nodes describing problem–solving knowledge correspond to rules in a runnable

cognitive model of performance (which also contains mal–rules). The activities are the leaf nodes, and when observed in the student's behaviour, the probability 1.00 is propagated through the net. The rules in the cognitive model that correspond to nodes in the net are updated with the probabilities from the node. An extension of OLAE also provides a possibility to represent higher level abstractions of the student's skills by analysing the cognitive model. This becomes a pedagogic–content model since knowledge of how to identify the abstract skills from the cognitive model is needed, and the student's skill at these (or the belief in that skill) is represented. It is, however, unlike other pedagogic–content models since the ties to the runnable cognitive model may be kept.

- This technique bridges from subject matter knowledge (the cognitive model), to intermediate states (the actions in the solution). The extension bridges from pedagogic–content to the subject–matter model, see the two arrows named OLAE in figure 8. A cognitive model of the task is required, as well as pedagogic expertise to design the recognisers.

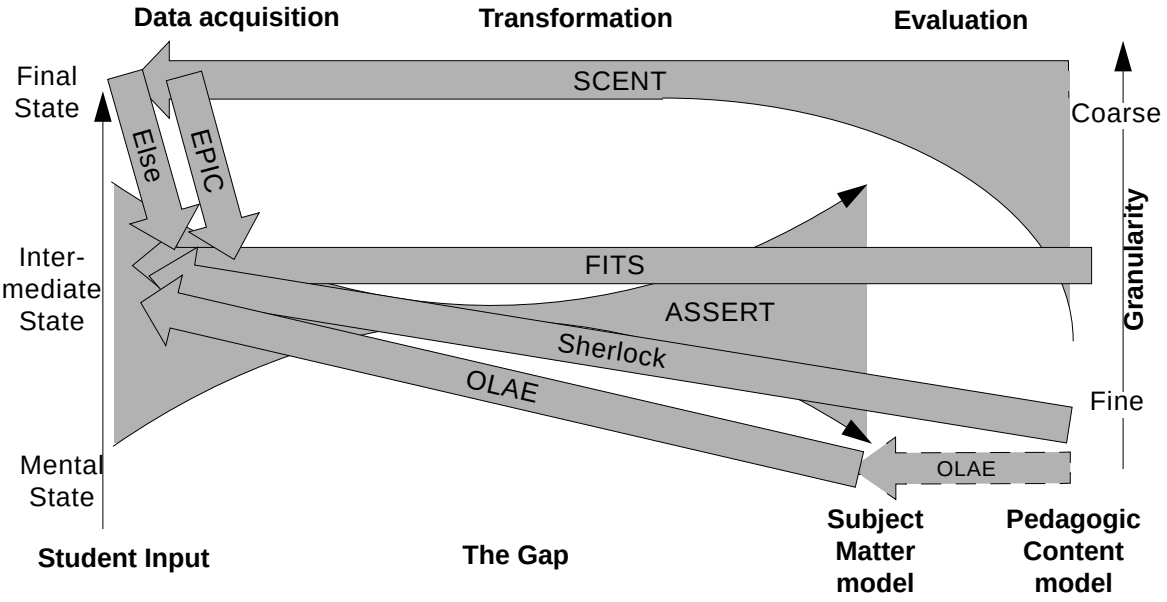

Figure 8. The more recent techniques.

6 Discussion

As can be seen from the illustrations in figure 7 and 8 all techniques do not bridge the entire gap. Some include no transformation, only data acquisition — more precisely data refinement (Plan Recognition, Path Finding, Else and EPIC). Subject–matter models are not evaluated, so the methods producing them do not include the evaluation phase.

If the student model is used to guide instruction, the whole gap is bridged by some component in the system. For example, the LISP-tutor which was used to exemplify the Model Tracing technique uses a tool interface (an editor) to access the mental state transitions, and an extended overlay model for the comparison (Corbett and Anderson, 1992). The FITS system also uses a tool interface to acquire the user's subgoals (intermediate steps) and then predefined recognisers that identify misconceptions, a transition and evaluation similar to Issue Tracing (Nwana, 1991). The MACSYMA Advisor uses Plan Recognition to refine the input and then identifies misconceptions about actions by analysing the

assumptions behind the plan (in effect the transition is similar to Condition Induction, and a Simple Perturbation comparison, Genesereth, 1982).

Some general trends can be identified, and are illustrated in figure 9.

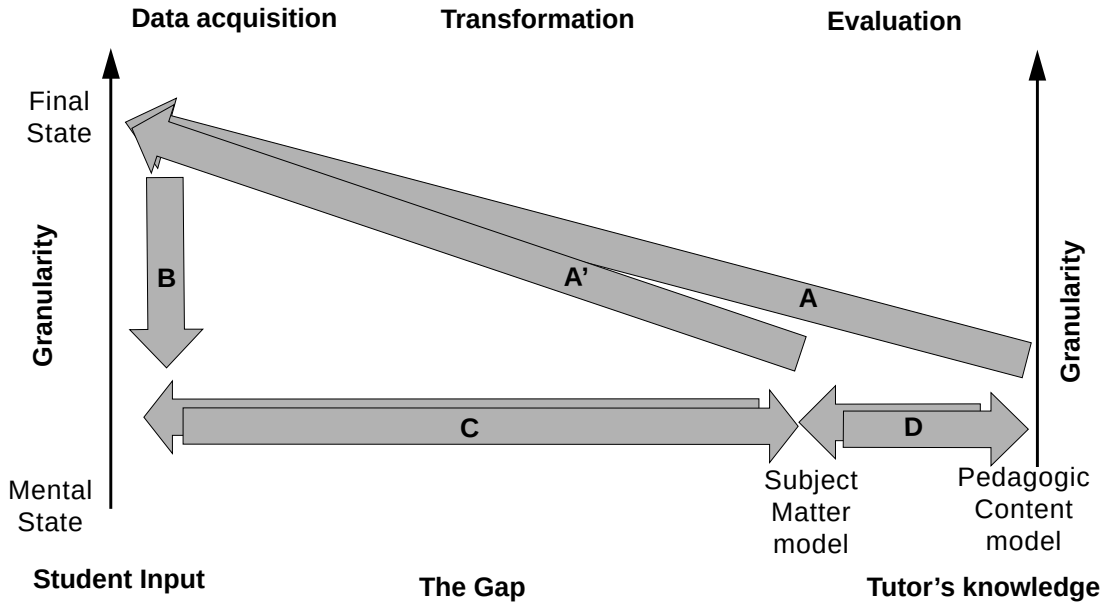

Figure 9. Generalizations of the different techniques

The techniques bridge different parts of the gap between the student's input and the Tutor's knowledge. Some techniques change granularity as they bridge the gap (the A and A' arrows) while some maintain the same granularity (arrow C). Bridging the gap within the same level of granularity can currently be done in ether direction (arrow C, e.g. Condition Induction, Model trace, Expert Systems, ASSERT) while change of granularity at crossing is only done from knowledge to input (arrows A and A'). The left–to–right D–arrow represents the different Overlay and Perturbation type comparisons. The fact that the right–to–left D–arrow only represents two items, the OLAE system and the Expert Systems technique, indicates the scarcity of methods other than comparisons for evaluating subject–matter models.

6.1 Direction and domain dependence

Techniques that bridge from the knowledge side to the input side appear to rely on recognisers of different kinds. These recognisers are rules or functions that are evaluated in the context of the exercise and the student's answer. They are often predefined using pedagogic expertise (see Arrow A, e.g. Issue Trace, FITS, SCENT, Sherlock and arrow A', e.g. Decision Tree). Recognisers can also be dynamically generated from a runnable cognitive model of the task applied to the exercise given the student (Arrow A', e.g. Generate and Test, Interactive Diagnosis, OLAE and also Model Trace from arrow C). Even for the dynamic case the interesting characteristics in the answer must be defined beforehand.

The predefined recognisers make the A-type techniques (represented by the arrow A) distinctly domain dependent. The use of a runnable cognitive model of the task in the A' types limits the domain dependence to that model, but also imparts very fine granularity to the model. A study of how recognisers are constructed would be a contribution to the field. Can a domain independent theory for recognisers be constructed for the A–type

techniques? Or would it be more feasible to define generalised recognisers along the right–to–left D–arrow which represents OLAE's and the Expert System's analysis of the student's cognitive model and inferences, respectively?

Techniques that bridge from input to knowledge appear to be inspired by machine–learning techniques. These use a constructive approach, building or refining the knowledge from smaller pieces. Machine–learning techniques are theoretically domain independent but are very fine grained. The problem for machine–learning is that the student's knowledge is not stable, thus the algorithm must adapt on the basis of very little evidence.

6.2 Input and type of model

With regard to the characteristics mentioned in the introduction, the input granularity and the type of model, the following observations can be made:

Pedagogic–content models are built using quite coarse–grained input. They also require pedagogic–content knowledge, not only because they are represented as instructional goals and issues, but because the design of the recognisers that are applied to the input requires this knowledge. This knowledge of the learning task can be collected from an articulate expert with experience, or from studies of students. The frequency of this type of models indicates that the identification of recognisers is not extremely difficult (e.g. Issue tracing, Expert Systems, SCENT, FITS). Pedagogic–content models appear to be of all grain sizes.

Subject–matter models appear to be built from intermediate or fine levels of input. They can be built in either direction across the gap, but the techniques that bridge from knowledge to input can only be used for evaluations of the Overlay comparison types (cf. section 4.3). When the constructive techniques are used (e.g. ASSERT, Condition Induction), Perturbation comparisons can be made. These techniques require that the input be as fine grained as the target model. Building subject–matter models from coarse grained input thus requires the use of input refinement techniques.

The possibility of combining input refinement techniques (arrow B, Plan Recognition, Path Finding, Else, EPIC) with other techniques has great potential when fine grained models are required. The C–type techniques generally require quite fine–grained input, as do some of the A and A' techniques. Using B– and C–type techniques together allows for less domain dependent techniques even when input is coarse, since some B techniques are generalizable. This refinement of input can currently be done by deductive methods (Plan recognition, Path Finding) or by providing additional input to the system through a tool interface (e.g. EPIC, FITS, LISP Tutor) or collaborative dialogue (Else).

7 Summary

In an attempt to provide a framework in which to relate different diagnosis approaches to each other, this paper presented student diagnosis as the process of bridging a gap between the knowledge that a tutor possesses and the information available on student knowledge. This gap is what an ITS that adapts to the individual student has to bridge in order to take decisions on how to tailor its instruction to the student.

The process of bridging the gap is divided into three phases which are always performed in a complete ITS, although not necessarily by the component labelled the student model-

ler. The phases are data acquisition, transformation and evaluation. Data acquisition concerns the collection of input data, where the granularity of the input ranges from input corresponding to mental state transitions to input corresponding to final states. The transformation concerns the necessary analysis and conversion of the input. The evaluation phase concerns the evaluation of the student's knowledge in terms of possible relations between what the student is believed to know and the system's conception of correct knowledge. The actual construction and revision of the student model is done after the transformation or after the evaluation, depending on type of model.

A sample of present techniques for student diagnosis were found to bridge the gap in different manners, depending on the input and model to be constructed. They were also found to put different stress on different phases in the process. Subject matter models are built by taking the student's input and reconstructing or identifying the knowledge that produced the input. That knowledge can then be compared to the tutor's knowledge. These methods often require that the granularity of the input be refined to match the granularity of the system's subject matter knowledge. Pedagogic–content models are built by using the tutor's knowledge to analyse the student's answer for evidence of certain pieces of knowledge or misconceptions. These methods often work with coarse grained input directly, using domain dependent "recognisers".

When designing a student diagnosis technique for a specific ITS it was found to be important to know if the student model would be represented in terms of pedagogic content knowledge or in terms of subject matter knowledge. Other aspects of importance are the granularity of input, the granularity of the model and the availability of pedagogic content knowledge. If a coarse model is all that is required, and sufficient pedagogic knowledge of the domain (to create the recognisers) can be provided, a pedagogic content model can be created using techniques such as Issue Trace, Expert Systems, FITS and SCENT. A theory of how to construct recognisers would be very useful in this context.

A problem in designing an ITS for a completely new learning task is that no experience of the teaching of it exists, making it very difficult to construct recognisers.

If the system's domain knowledge is admittedly incomplete or if it is considered importand to represent the student's actual knowledge, a subject–matter model is required. If a runnable cognitive model of the task is available, coarse input can be used with techniques such as Generate and Test, Interactive Diagnosis. Use of techniques like Model Trace and OLAE would require an input refinement technique, for instance Path Finding, Plan Recognition or a tool interface (e.g. EPIC).

If a subject–matter model is desired when no runnable cognitive model of the task is available, the constructive techniques would have to be used. These would also require combination with input refinement techniques to provide fine enough grain.

An important consideration in this context is what instructional activities require what type and granularity of the model. It is possible that instructional planning can be done on the basis of quite coarse models, while delivery of instruction and tailoring of explanations require fine subject–matter models. This is a question that requires further research.

Acknowledgements

This work was financed by NUTEK.

Special thanks to Barbara Wasson, Brant Cheikes and the anonymous reviewers, all of whom provided many constructive comments on previous versions of this paper.

References

Anderson, J., Boyle, C., Corbett, A., and Lewis, M. (1990). Cognitive modeling and intelligent tutoring. *Artificial Intelligence*, 42:7–49.

Baffes, P. and Mooney, R. (1992). Using theory revision to model students and acquire stereotypical errors. In *Proc. of the Fourteenth Annual Conference of the Cognitive Science Society*, pages 617–622, Bloomington, IN.

Baffes, P. T. (1994). *Automatic Student Modeling and Bug Library Construction using Theory Refinement*. PhD thesis, University of Texas at Austin.

Baril, D., Greer, J. E., and McCalla, G. I. (1991). Student modelling with confluences. Technical report 91–3, ARIES, Dept. of Computational Science, University of Saskatchewan, Saskatoon, Canada.

Bhuiyan, S. H., Greer, J. E., and McCalla, G. I. (1991). Characterizing, rationalising and reifying mental models of recursion. Technical Report 91-4, ARIES, Dept. of Computational Science, University of Saskatchewan, Saskatoon, Canada.

Brown, J. S., Burton, R., and de Kleer, J. (1982). Pedagogical, natural language and knowledge engineering techniques in SOPHIE I, II, and III. In Sleeman, D. and Brown, J. S., editors, *Intelligent Tutoring Systems*, chapter 11, pages 227–282. Academic Press.

Burton, R. R. (1982). Diagnosing bugs in a simple procedural skill. In Sleeman, D. and Brown, J. S., editors, *Intelligent Tutoring Systems*, chapter 8, pages 157–183. Academic Press.

Chan, T.-W. and Baskin, A. B. (1988). "Studying with the prince" the computer as a learning companion. In *Proceedings of ITS'88*, pages 194–200, Montreal, Canada.

Clancey, W. J. (1982). Tutoring rules for guiding a case method dialogue. In Sleeman, D. and Brown, J. S., editors, *Intelligent Tutoring Systems*, pages 201–225. Academic Press.

Corbett, A. T. and Anderson, J. R. (1992). LISP Intelligent Tutoring System: research in skill acquisition. In Larkin, J. H. and Chabay, R. W., editors, *Computer Assisted Instruction and Intelligent Tutoring Systems: Shared goals and Complementary Approaches*, Technology in Education series, chapter 3, pages 73–109. Lawrence Erlbaum Associates.

del Soldato, T. (1992). Detecting and reacting to the learner's motivational state. In *Proceeding of ITS'92*, pages 567–574. Springer-Verlag.

Derry, S., Hawkes, L., Kegelmann, H., and Holmes, D. (1989). Fuzzy remedies to problems in diagnostic modeling. In Bierman, D., Breuker, J., and Sandberg, J., editors, *Proc. of the 4th International conference on AI and Education, Amsterdam*, pages 81–85. IOS, Amsterdam.

Dillenbourg, P. and Self, J. (1992a). A framework for learner modelling. *Interactive Learning Environments*, 2(2):111–137.

Dillenbourg, P. and Self, J. A. (1992b). PEOPLE POWER: A human-computer collaborative learning system. In Frasson, C., Gauthier, G., and McCalla, G. I., editors, *Proceedings of ITS'92*, pages 651–660. Springer Verlag.

Genesereth, M. R. (1982). The role of plans in intelligent teaching systems. In Sleeman, D. and Brown, J. S., editors, *Intelligent Tutoring Systems*, chapter 7, pages 137–155. Academic Press.

Goodyear, P. and Tait, K. (1991). Learning with computer-based simulations: Tutoring and student modelling requirements for an intelligent learning advisor. In *Learning and instruction: European research in an international context*, pages 463–481. Proceedings of the 3rd conference of the European Assoc. for Research on Learning and Instruction, 1989.

Greer, J., Mark, M., and McCalla, G. (1989). Incorporating granularity-based recognition into SCENT. In Bierman, D., Breuker, J., and Sandberg, J., editors, *Proc. of the 4th International Conference on AI and Education*, pages 107–115, Amsterdam.

Holt, P., Dubs, S., Jones, M., and Greer, J. (1994). The state of student modelling. In Greer, J. E. and McCalla, G. I., editors, *Student Modelling: The Key to Individualized Knowledge-Based Instruction*, NATO_ASI Series F, pages 3–35. Springer-Verlag.

Katz, S., Lesgold, A., Eggan, G., and Gordin, M. (1992). Modelling the student in Sherlock II. *Artificial Intelligence in Education*, 3(4):495–518.

Koegel, J., Lakshmipathy, N., and Schlesinger, J. D. (1989). A tutoring system with incomplete domain expertise. In Bierman, D., Breuker, J., and Sandberg, J., editors, *Proc. of the 4th International Conference on AI and Education*, pages 123–128, Amsterdam.

Kono, Y., Ikeda, M., and Mizoguchi, R. (1993). A modeling method for students with contradictions. In Brna, P., Ohlsson, S., and Pain, H., editors, *Proc. of AI in Education (AI-ED 93)*, pages 481–488, Edinburgh, Scotland.

Langley, P. and Ohlsson, S. (1984). Automated cognitive modelling. In *Proc. of the American Association of Artificial Intelligence, Los Altos, CA*, pages 193–197. Morgan Kaufmann.

Laurillard, D. (1993). From learning need to teaching strategy: What is the nature of the link. In Brna, P., Ohlsson, S., and Pain, H., editors, *Proc. of AI in Education (AI-ED 93)*, pages 12–14, Edinburgh, Scotland.

Lesgold, A., Ivill-Friel, J., and Bonar, J. (1989). Toward intelligent systems for testing. In Resnick, L. B., editor, *Knowing, Learning and Instruction*, chapter 11, pages 337–360. Lawrence Erlbaum Associates publishers.

Martin, J. D. and VanLehn, K. (1993). OLAE: Progress toward a multi–activity, Bayesian student modeller. In Brna, P., Ohlsson, S., and Pain, H., editors, *Proc. of AI in Education (AI-ED 93)*, pages 410–417, Edinburgh, Scotland.

Murray, W. (1991). An endorsement-based approach to student modeling for planner-controlled tutors. In *Proc. of International Joint Conference on AI, (IJCAI'91), Sydney, Australia*, pages 1100–1106.

Nwana, H. S. (1991). User modelling and user adapted interaction in an intelligent tutoring system. *User Modeling and User-Adapted Interaction*, (1):1–32.

Ohlsson, S. and Langley, P. (1985). Identifying solution paths in cognitive diagnosis. Technical Report CMU-RI-TR-84-7, Robotic institute, Carnegie-Mellon University, Pittsburgh, PA.

Petrushin, V. A. and Sinitsa, K. M. (1993). Using probabilistic reasoning techniques for learner modeling. In Brna, P., Ohlsson, S., and Pain, H., editors, *Proc. of AI in Education (AI-ED 93)*, pages 418–425, Edinburgh, Scotland.

Ragnemalm, E. L. (1994). Simulator-based training using a learning companion. In Brouwer-Janse, M. and Harrington, T., editors, *Human-Machine Communication for Educational Systems Design*, NATO-ASI Series F, pages 207–212. Springer-Verlag. Proceedings from the 1993 NATO-ASI in Eindhoven, The Netherlands.

Ragnemalm, E. L. (1995). Towards student modelling through collaborative dialogue with a learning companion. Licentiate thesis Thesis No 482, Linköping University.

Rich, E. (1983). Users are individuals: individualizing user models. *Int. J. of Man–Machine Studies*, 18:199–214.

Self, J. (1986). The application of machine learning to student modelling. *Instructional Science*, 1986(14):327–338.

Self, J. (1988). Bypassing the intractable problem of student modelling. In *Proceedings of ITS'88*, pages 18–24, Montreal, Canada.

Self, J. A. (1994). Formal approaches to student modelling. In Greer, J. E. and McCalla, G. I., editors, *Student Modelling: The Key to Individualized Knowledge-Based Instruction*, NATO_ASI Series F, pages 295–352. Springer-Verlag.

Shulman, L. S. (1986). Those who understand: Knowledge growth in teaching. *Educational researcher*, 15(2):4–14.

Sime, J.-A. (1993). Modelling a learner's multiple models with Bayesian belief networks. In Brna, P., Ohlsson, S., and Pain, H., editors, *Proc. of AI in Education (AI-ED 93)*, pages 426–432, Edinburgh, Scotland.

Sleeman, D. (1985). UMFE: A user modelling front–end subsystem. *Int. J. Man–Machine Studies*, 23:71–88.

Twidale, M. (1989). Intermediate representations for student error diagnosis and support. In Bierman, D., Breuker, J., and Sandberg, J., editors, *Proc. of the 4th International conference on AI and Education*, pages 298–306, Amsterdam. IOS, Amsterdam.

VanLehn, K. (1988). Student modeling. In Polson, M. C. and Richardson, J. J., editors, *Foundations of Intelligent Tutoring Systems*, chapter 3, pages 55–78. Lawrence Erlbaum Associate publishers.

Vassileva, J. (1991). A classification and synthesis of student modelling techniques in intelligent computer–assisted instruction. In Norrie, D. H. and Six, H.-W., editors, *Computer Assisted Learning*, volume 438 of *Lecture Notes in Computer Science*, pages 202–213. Springer-Verlag.

Verdejo, M. F. (1994). Building a student model for an intelligent tutoring system. In Greer, J. E. and McCalla, G. I., editors, *Student Modelling: The Key to Individualized Knowledge-Based Instruction*, NATO_ASI Series F, pages 147–163. Springer-Verlag.

Villano, M. (1992). Probabilistic student models: Bayesian belief networks and knowledge space theory. In Frasson, C., Gauthier, G., and McCalla, G. I., editors, *Intelligent Tutoring Systems*, pages 491–498. Springer-Verlag.

Wasson, B. J. (1990). *Determining the Focus of Instruction: Content Planning for Intelligent Tutoring Systems*. PhD thesis, University of Saskatchewan, Dept. of Computational Science.

Wenger, E. (1987). *Artificial Intelligence and Tutoring Systems; Computational and Cognitive Approaches to the Communication of Knowledge*. Morgan Kaufmann publishers Inc.

Wilson, S. M., Schulman, L. S., and Richert, A. E. (1987). "150 different ways" of knowing: Representations of knowledge in teaching. In Calderhead, J., editor, *Exploring Teacher's Thinking*, pages 104–124. Cassel.