Towards a negotiable student model for constraint-based ITSs

David Thomson

November 6, 2008

Department of Computer Science & Software Engineering

University of Canterbury, Christchurch, New Zealand

Supervisor: Antonija Mitrovic

Abstract

Much research has been done on open student models within adaptive educational systems. It has been shown that opening up the student model, and allowing the student to view their model is useful in the learning process. Open student models help support meta-cognitive process, such as self-reflection. Negotiable student models take this a step further, and allow students to negotiate and potentially modify their model. A few negotiable student models have been implemented, but only in relatively simple systems, and not integrated into a complex ITS. As such, it is not clearly known if negotiable student models pose a significant advantage over the traditional open student models. This research implements a basic negotiable student model into a version of a complex and internationally deployed ITS. Subjective evaluation is performed, and shows promising results. Participants felt the negotiable student model was both useful for learning, and enjoyable to use. With a few improvements, this negotiable student model implementation could be used in a wide-scale objective analysis to help determine the usefulness of negotiable student models.

Contents

1 Introduction

Conventional education is currently in a crisis. High student-to-teacher ratios in classrooms mean that few students get sufficient individualised attention [29, 30]. Most classrooms have students across a very wide range of ability levels, and teachers must teach at a level most appropriate for the whole class, rather than being able to cater for individuals. Students with lower ability will often struggle to keep up, and ultimately get behind. If the teacher slows down to accommodate these students, students with higher ability will quickly get frustrated and bored, a situation which must be avoided if students are to learn effectively. This continual conflict of interest increases the stress on teachers dramatically. The most desirable solution would be to have one teacher per student [4], but this is obviously unfeasible. A common approach to try to solve this problem is to use computers as a teaching device, but conventional education programs do not provide any more individualised teaching than a teacher in a classroom [1]. This is because conventional education software has no way to adapt to the user's ability level.

Intelligent Tutoring Systems (ITSs) are computer based tutors that aim to provide the same level of student specific help as a human tutor [6]. This is achieved through Artificial Intelligence, student modeling, and other methods [1]. In an ITS, the system tracks the student's actions, and builds a model of their knowledge. This model is then used to influence pedagogical decisions, such as which problem to suggest to the student next. This allows ITSs to adapt to students of differing ability levels: a below average student will get different recommendations, and possibly different feedback than an above average student. ITSs aim to give feedback appropriate to students of all abilities, so a struggling student will be given substantial assistance while a competent student will be given much less help. Recommending questions based on the students ability means that low level ability students will not get overwhelmed, and high ability students will not get bored. ITS can therefore cater to different learning styles, although some are shown to be better (in respect to the amount of material covered correctly) than others [18].

At least four different levels of visibility for the student model exist: Hidden, Open, Editable, and Negotiable. Most often in ITSs the student model is hidden from the student, and is used only by the ITS itself. It has however, been shown that allowing the student to inspect their model increases the students learning [8]. Open student models allow the student to view their model, editable models allow the student to update their model, and negotiable models allow a form of editing, but the student must convince the system that their knowledge is correct before any changes are made. The purpose of opening up the student model is to get the student to be actively involved in their learning and self assessment. In performing these meta-cognitive processes, the student is likely to learn more from the ITS [8], as improved meta-cognitive skills lead to an improvement in learning. Research has been done on how to best display an open student model [3, 14, 22], but few ITSs implement a negotiable student model.

2 Background

2.1 Intelligent Tutoring Systems

There is a real need today to support teachers in the classroom. High student numbers and low human resources mean that teachers are often stretched and overworked. When this occurs the quality of learning decreases. Possible high achievers may get missed, and fail to reach their potential. Struggling students may get left behind completely. This is all because the teacher does not have the time to teach each student in a specific, individual manner. The most desirable solution to this problem is to have one expert tutor per student. This allows each student to be taught in a way and pace that is most suited to them. Wide-scale one-on-one tutoring is practically infeasible though, as there is simply not enough tutors.

One attempt to try to solve this problem is through the use of ITSs. ITSs are computer based tutoring systems that try to imitate one-on-one human based tutoring [4]. The benefit of one-on-one tutoring is that each student can be taught at their own pace and style, so for an ITS to be effective it should be able to adapt to the student who is using it [6], and provide different experiences to students of differing skills.

To adapt to a student an ITS must have some sort of knowledge about the student. This usually takes the form of a student model, which is a representation of the students knowledge, in the particular domain of the ITS. Representing a person's knowledge is not trivial; it is not enough to model just correct knowledge, but it is too hard to model all correct and incorrect knowledge [7]. One approach to model student (and domain) knowledge is Constraint Based Modeling (CBM) [9]. CBM models the domain as a large number of constraints, which are in effect small bits of declarative knowledge. A student model then contains a history of times the student has come across each constraint when answering a question. For each of these occurrences, the student either satisfied or violated the constraint, which is recorded.

The resulting student model is both useful and easy to implement [9]. The ITS can use this model to influence pedagogical decisions (teaching decisions). The most common use of the student model is in recommending what problem to suggest to the student next. An ITS can, from the student model, calculate a question that is in the zone of proximal development [5] for the student. This means the question will be hard enough to extend the student, but not too hard that the student will get frustrated and give up. Using this approach means that struggling students will be recommended easy questions, whereas competent students will be recommended harder questions. In some domains, the system can also recommend questions that are relevant to concepts that the student is relatively weak in, as in SQL-Tutor [15]. This helps to ensure that the student has a thorough understanding of the domain, and is not just really good at some components or concepts of it.

2.2 Open Student Models

Student modeling can be defined as "*The process of gathering relevant information in order to infer the current cognitive state of the student, and to represent it so as to be accessible and useful to the pedagogical module.*" [10]

Although computing a complete and correct student model is intractable [7], a useful student model can still be implemented effectively. This can be achieved when it is realised that the usefulness of the model is more important than its completeness. When constructing a student model, the ITS should avoid guessing, not bother to diagnose what it can not treat, and empathise with the student [7]. This results in dynamic student models which are as accurate as needed, and can be used by the system (specifically the pedagogical module) to make pedagogical decisions.

Open, inspectable, or viewable student models extend the purpose of a student model from a source of information for the system to a source of information for the student (and the system) [19]. An open student model reflects to students feedback on their progress and overall performance in the system. Usually, the student model is broken up into categories, or concepts. Student performance and progress is shown for each concept. This allows the student to see their strengths and weaknesses within the domain on a finer level. The student will be able to see which concepts they are good at, and therefore do not need to focus so much on, and also the concepts they are bad at, and could do with more work. This could be useful in an examination situation, as the model would show which concepts the student should focus their studies on.

As well as passively suggesting learning material to the student, an open student model aims to promote self reflection and assessment through inspection of the model. It has been suggested [23] that the process of identifying and trying to explain errors can help correct incorrect knowledge.

Various representations for an open student model exist. Figure 1 shows the open student model from E-KERMIT [17] (an enhanced version of KERMIT [24]). The model has been broken down into a hierarchical view giving detailed, low-level information on the student's progress in the domain.

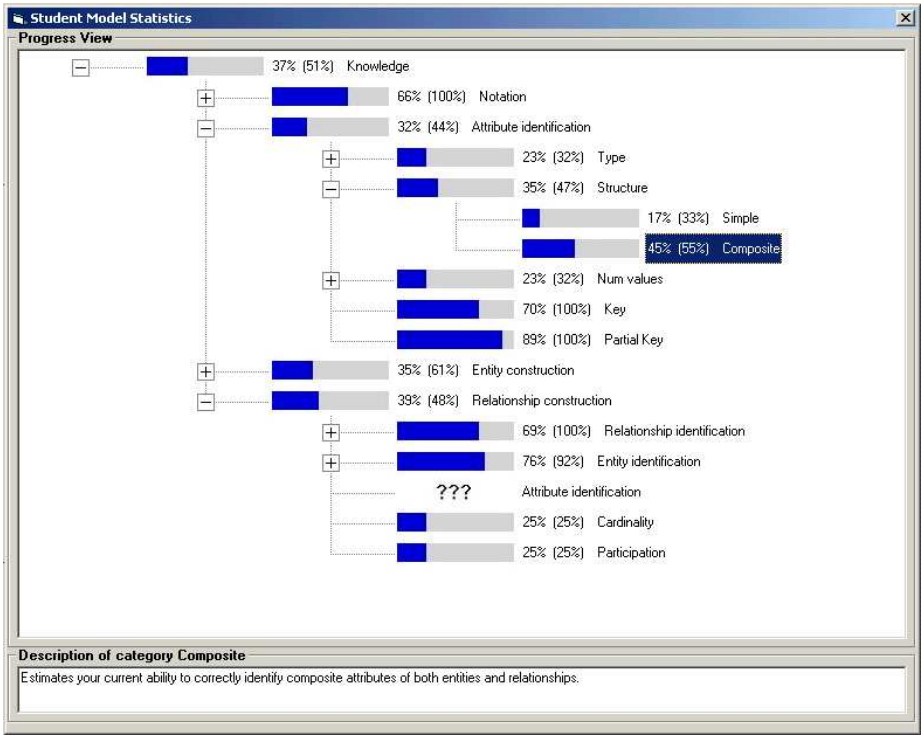

Figure 1: The main view of a student's progress in E-KERMIT

2.3 Related work

2.3.1 CALMsystem

CALMsystem [27] implements a negotiable student model, aimed at primary school students. Students can negotiate their model through a conversational agent, using a natural language interface. The system assesses the student on each topic in the domain with two numerical scores. One score, the system's, is based on the student's performance on questions. The second score, the student's, is given by the student when they first use the system, and subsequently after each question they answer. These two scores, on a scale of zero to one, are converted into "low", "moderate", "good", or "high". Students using the system view these scores (represented by pictures), and can quickly see discrepancies between their beliefs and those of the system. Beliefs are changed through discussion with the agent, and may be initiated either by the system or the student. Students must explicitly rank themselves after every question, and can then compare their rankings with the system's. If the student wishes, they can get justification from the system for the system's beliefs. To change the system's beliefs the student answers questions, and the system will modify it's beliefs based on the responses. The tutor consists of multi-choice questions,

and is separate from the negotiable student model and conversational agent component.

An experimental evaluation was conduced with a tutor loaded with science questions, and 25 UK Primary school children, aged 10-11. The evaluation showed that both the negotiable student model version of CALMsystem, and a version with an inspectable student model help develop student's meta-cognitive processes. The study found that users of the negotiable version of CALMsystem reduced inaccuracies in their self-assessments significantly more than users of the version without negotiation support.

2.3.2 STyLE-OLM

STyLE-OLM [28] integrates a negotiable student model into STyLE, an *"adaptive knowledge based web learning environment aimed at assisting learners from Bulgaria, Romania and Ukraine in acquiring Finance terminology in English."* [28]. STyLE-OLM features a student model jointly constructed by the system and the student, and can contain student's beliefs and misconceptions. Students are shown a graphical representation of their beliefs, as well as textual information. The student is involved in the construction of their model through dialogue with the system, where both the student and system can ask questions, state propositions, and challenge or justify claims. STyLE-OLM maintains just one student model, so the system and student must agree at some level. To achieve this a complex process involving *reasoners* and *initial sets of beliefs* is used.

A small evaluation of STyLE-OLM was conducted, which mainly focused on the behaviour of the system, and not the effectiveness of the student model on learning. In general, STyLE-OLM provided an adequate environment for inspecting and discussing the student model. However, some of the natural language dialogue confused or frustrated some of the users.

2.3.3 See Yourself Write

See Yourself Write [11] implements an inspectable student model. Students must complete foreign language writing assignments, which can be one of a variety of tasks (descriptive essay, translation, or factual reports). The completed assignment is sent to a teacher for evaluation, and based on the feedback from the teacher a student model is constructed, and displayed to the student. The student model includes specific feedback on the errors made by the students. This feedback will help to correct students knowledge, and can be accessed and reviewed later when the student is attempting other assignments. In addition, students can discuss the feedback with the teacher in natural language dialog where the student can try to explain and/or justify their answer. A small evaluation found that from initial indications students would find this sort of student model useful. They found that students did want to discuss their feedback, which often led to misunderstandings being uncovered.

2.3.4 Mr. Collins

Mr. Collins [12] implements a "collaboratively maintained, inspectable student model". In this system the student model is maintained by both the student and the system. Mr. Collins uses a simple student model, which contains two separate confidence measures. The first is provided by the student, and reflects the student's current belief of their knowledge. The second measure is calculated by the system based on the student's performance. These confidence measures take the form of a value from a four point scale (very sure/almost sure/unsure/very unsure). If the two measures differ by a significant amount (more than one value difference on the scale), the student enters a dialog with the system. Here the student can choose to change their own beliefs, or challenge the system's beliefs. When changing the student's own beliefs, the student can ask the system to justify it's decision, which may involve showing the student's last five attempts on the relevant problem. When challenging the system's beliefs, the student may have to justify themselves, which consists of answering a question. The domain for Mr. Collins is object pronouns in European Portuguese for second language learners, and there are twelve rules for pronoun placement in the system. This research investigated whether students would inspect their own student model, and whether they would challenge the contents of the model in cases where they disagreed. Results showed that students did in fact inspect and challenge their model.

2.4 EER-Tutor

EER-Tutor is a web-enabled ITS that teaches the Enhanced Entity Relationship model. It is based on KERMIT [16], which is an ITS that runs as a desktop application, teaching the Entity Relationship model. It uses CBM, and has over 200 constraints. EER-Tutor is used in a database course at the University of Canterbury, and is available online at DatabasePlace[1]. It currently has approximately 5000 registered users.

Enhanced Entity Relationship modeling is an ill-defined, open-ended task. This means the start and end states, as well as operators are difficult to define. The problem solving algorithms are underspecified, and most problems will have more than one correct solution. Regardless, EER-Tutor has been shown to be effective [16], when combined with traditional lectures. EER-Tutor has an open student model, as shown in Figure 2.

[1] http://www.aw-bc.com/databaseplace

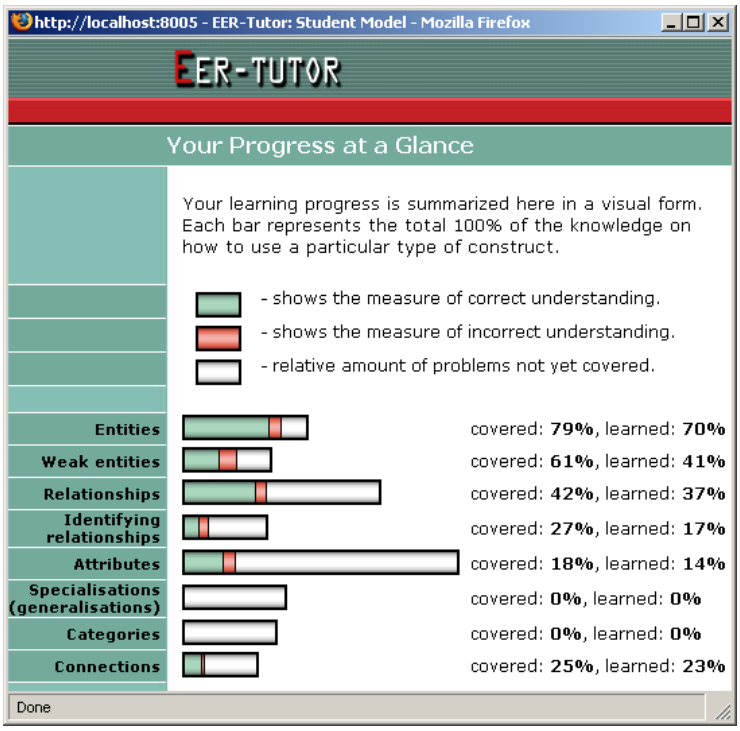

Figure 2: Open student model in EER-Tutor

The domain (Enhanced Entity Relationship modeling) is broken down into eight concepts. The student is able to see which specific parts of the domain they have covered, and to what level of proficiency. The horizontal size of the bar indicates how much material there is on that concept in the tutor. This bar is divided into three distinct sections: correct understanding, incorrect understanding, and material not covered. This allows the student to identify concepts that they struggle with (high amount of incorrect knowledge), and concepts that they have not learned much of (high amount of material not covered). As the student progresses through the tutor, the total material covered (correct plus incorrect) will increase. Hopefully, but not necessarily, the amount of incorrect understanding will decrease, until all the material is covered correctly.

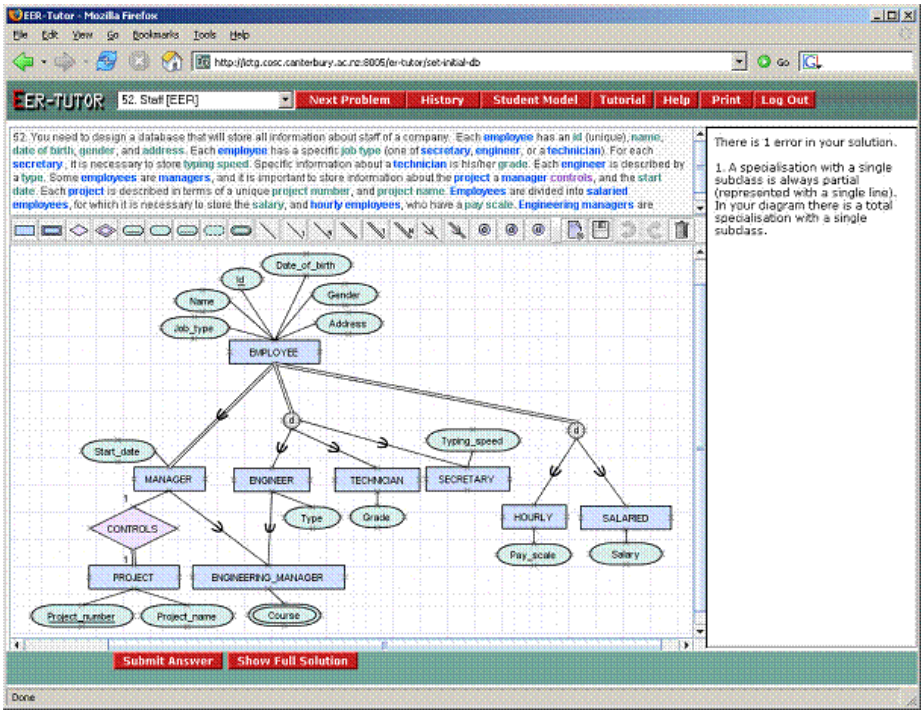

Figure 3: The EER-Tutor interface

Figure 3 shows the EER-Tutor interface. The main area of EER-Tutor is a place for the student to draw EER diagrams. Tool buttons are provided for the different components of an EER diagram, and the question text is always shown. These two features aim to help reduce the cognitive load on students. When the student wishes to, they can submit their diagram. If there are some errors in their solution, feedback will be displayed on the right side of the window. The student can then use this feedback to help correct their solution, before re-submitting. There are also buttons for system actions such as Next Problem, Logout, and a button for the student to view their student model.

Where previous research [12, 27] has implemented a negotiable student model in a simple tutor, this project will use EER-Tutor. This will allow for evaluation of a negotiable student model in a complex ITS that is being used in a university course, and internationally over the Internet.

3 Enhancing EER-Tutor

3.1 An Ideal Negotiable Student Model

A negotiable student model allows the student to edit their model, but there needs to be a form of control on this editing. If the student could arbitrarily change their model, it would defeat the purpose of the model which is to reflect the current knowledge of the student. One way to implement this control is to force the student to first convince the system of their knowledge, before they can modify their model. For example, if the student does not agree with part of their model, they can start a dialog with the system. If the student can convince the system their knowledge is higher than their model suggests, the system will modify the model appropriately.

One way the student can prove their knowledge is by correctly answering questions asked by the system. Figure 4 shows an example of a dialog between the system (EER-Tutor) and a student.

Student: *"I know more than you think I do about entities*"

System: *"OK, when drawing EER diagrams, which sort of words in the problem text are likely to model entities?*"

Student: *"big words"*

System: *"Not quite, the answer is a type of word e.g. verb, adjective, noun etc. Try again.*"

Student: "*nouns*"

System: "*Correct! I'll update your student model now*"

Figure 4: Example dialog with EER-Tutor

At this stage the Entities component of the student model will be increased slightly.

This method relies on natural language parsing, which is still considered quite hard. Although it would be good to implement such a system, natural language parsing is beyond the scope of this project. However, a useful negotiable student model should still be able to be implemented, in a somewhat reduced form, to serve as a tool for evaluation.

A negotiable student model presents two benefits:

The model will be more accurate. If the student is allowed to correct their model, it should result in the model giving a more accurate representation of the student's knowledge. This relies on the fact that the student does in fact have a good understanding of their knowledge (which is not always the case), and that editing the model is controlled. Forcing the student to prove their knowledge before their model is changed is a good way to keep the model accurate. With this approach, students do not edit their model directly, rather, their model gets modified based on information gained about the student's knowledge, from a source other than the domain problems. If editing of the model was not controlled, it would quickly become very inaccurate, and thus useless. The benefit of the model being more accurate is not the main focus, however.

Students will develop meta-cognitive skills. The main purpose of a negotiable student model is to encourage the student to actively think about their progress and learning, as they are using the system. Meta-cognition is defined as the knowledge or awareness of one's cognitive processes (the processes of thought). Increased meta-cognitive skills have been shown [2, 13, 25, 26] to be beneficial to the learning process. A negotiable student model seeks to engage the student in thoughts such as "do I really know that much", or "I think I might know more than that". For example, if the student model displays the students knowledge on a particular concept as relatively low (or high), it might make the student pause and think about how much they think they know on that concept. This is supported in open student models, but the ability to actually modify a negotiable student model should facilitate more of this type of thought.

An important aspect of any ITS is that the interface must be easy to learn and use. This is to reduce the cognitive load on the student, and allows them to focus solely on the domain problems. If the interface is hard to use, students will not learn as effectively because they will be spending unnecessary mental effort just to use the system. It is therefore very important that the negotiable student model is easy to learn and use; if it is not, students will not use it. It should also not detract from solving domain problems; it should be treated as an extra component, and not the main focus of the ITS.

Ideally, the student should be able to start a dialog with the system at anytime. This could be after correctly solving a problem, incorrectly solving a problem, or in the middle of working on a problem. The dialog should be in plain language, and thus be easy to understand. When proving their knowledge, the student should, in general, be able to describe and explain concepts asked about by the system. This interaction aims to simulate a student explaining what they know to a teacher or tutor.

A complete negotiable student model could also have system-initiated dialog. If at certain points, say after correctly solving a problem, the system finds that a certain concept in the student model is at a significantly different level to the other concepts, it would start a dialog with the student about this concept. If the system had over-estimated the students knowledge on that concept, asking them to prove their knowledge would result in a decrease in that section of the

model, leading to a more accurate representation of that student's knowledge. If the student was in fact that competent in that concept, the dialog may lead the student to try to work out why they are so much better at that concept.

The overall aim of the negotiable student model is to encourage and facilitate the development of meta cognitive skills; the actual changes to the student model are less important.

3.2 Design of a Basic Negotiable Student Model

The negotiable open student model in EER-Tutor has been designed as an additional, separate component. The existing student model is still used, with the negotiable student model effectively acting as a layer above the conventional model. When the student views their model, they see a combination of the two models. Changes made to the negotiable model have no effect on the underlying student model, which can no longer be seen by the student. This design was chosen to minimise the effect of adding a negotiable model, and as a result only minimal changes needed to be made to the existing code.

This approach could be used because the main purpose of the student model in EER-Tutor is to be visible to the student. There is no system recommended next problem (a common use for student models) in EER-Tutor; all the problems cover most of the concepts, so they are all equally relevant. This means that the effects of having the student see a different representation of their knowledge to what is actually stored in the student model are negligible.

The display of the combined models, shown in Figure 5, is similar to that of the single model implementation, except that the combined open/negotiable model is always displayed to the student. This was a deliberate change, and should prompt the student to think more about their learning, thus increasing their meta-cognitive processes. The bold red colour used to indicate incorrect knowledge aims to encourage students to correct their knowledge.

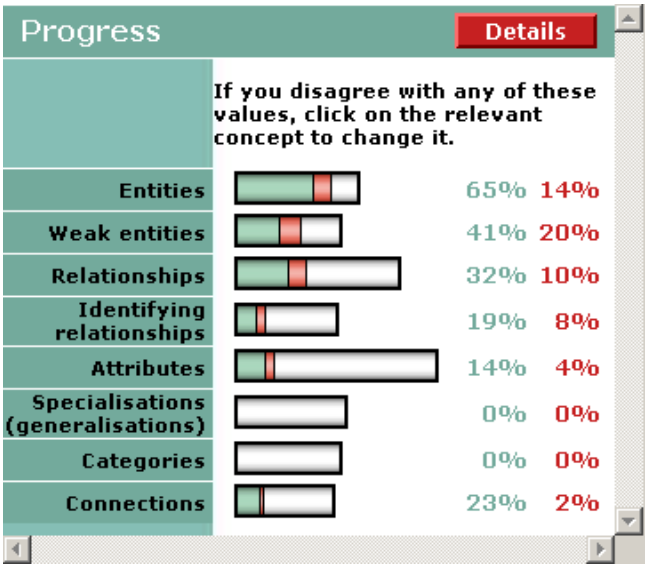

Figure 5: The student model

The first bar (a predominant green colour) represents correct knowledge; the second bar (bold red) represents incorrect knowledge. The amount of material not yet covered by the student is represented by the third (white) bar. When using the negotiable student model, it is only possible to change the green and red components. At best, one can eliminate all the red (incorrect) knowledge. This means that to cover more material it is still necessary to attempt domain problems. This helps to ensure that the negotiable student model does not become the student's focus; they still need to work on domain problems to progress through the tutor.

Certain design decisions had to be made to get around the need for a natural language parser. There are two places were natural language parsing would be used, if implemented. These are when the student initiates the conversation, and when the student answers a question. An example of this is shown in Figure 4 above. These two cases where a natural language parser would be needed were dealt with in different, but similar ways.

A student must first initiate a dialog with the system. When starting the conversation, there is one key variable: the concept. Once this information is obtained it can be assumed that the student wants to be asked a question on that concept. The initial idea was to provide a text entry and look for certain keywords relating to concepts in the users input. This is problematic though as the keyword might not always be found, might be spelled incorrectly, or the timing might be wrong, for example the student is trying to answer a question and the system mistakenly asks another question. Trying to solve the latter problem would lead to cases where the student wanted to be asked another question before answering the current question not being identified correctly.

Due to these problems, a more ridged design was used. When the student model is displayed to the student, each concept is a link which opens a new window, and starts a dialog between the student and the system, on that concept. This solves the problem of the previous design, and allows conversations to be started in a very controlled way, making the implementation simpler.

Answering questions is dealt with in a somewhat similar manner; enforcing more controlled input via the interface. This basic negotiable model uses two types of questions: multi-choice and short answer. The multi-choice questions were used because they eliminate the need for the student to type in their answer; they simply choose an item from the list. This was technically simple to implement, as there was no text parsing involved, thus eliminating the need to check for spelling mistakes or synonyms. While this is simple, it is not very flexible, and one of the key aims of a negotiable student model is to get the student to explain their knowledge in plain English. Due to this, this basic negotiable student model also supports short answer questions, using regular expression pattern matching. For such questions, the student is asked to enter their answer into a text box, which is then checked against the correct answer. To get around the need for a natural language parser, two methods are used: regular expression pattern matching, and multiple correct answers. Regular expression pattern matching can help reduce the effect of trivial mismatches such as extra white space or erroneous capital letters. Multiple correct answers allow the author to specify more than one correct answer, which the student's answer only needs to match to one of. This lets the system somewhat support synonyms, but only if the author adds additional answers manually. These two methods put more burden on the author, but should allow the system to be flexible and useful enough to perform an informative study of the effectiveness of a negotiable student model in EER-Tutor.

Behind the scenes, both multiple-choice and short answer questions are dealt as text questions. For multiple-choice questions the interface generates the text answer corresponding to the item the student selected. This means all questions and answers are modeled in the same way, keeping the design consistent. This has been done to make the future addition of a natural language parser as easy as possible; all that will need to be changed is the answer checking code.

As shown in Figure 4, sometimes the system will give a hint to the student to help them out. For this, each question can have feedback, specified by the author, for specific incorrect answers. If the student's answer matches one of these incorrect answers, the corresponding feedback will be displayed.

3.3 Implementation of a Negotiable Student Model in EER-Tutor

Central to this implementation is the questions and their answers. Figure 6 shows the general format for a question.

```
(
  question-number
  relevant-concept
  (question-text (option1 option2 ...))
  (correct-answer1 correct-answer2 ...)
  ((incorrect-answer1 feedback1)
   (incorrect-answer2 feedback2)
   ...)
  max-number-of-attempts
)
```

Figure 6: General question format

Each question has six components: a question number, relevant concept, the question text, the correct answer(s), incorrect answer and feedback pairs, and a maximum number of attempts.

Question Number. Each question has a unique number that identifies it from the rest. This is used by the system to keep track of questions the student has answered, so it can ask the student questions they have not already answered (if possible).

Relevant Concept. Each question is relevant to exactly one concept. Such concepts include "Entities", "Relationships", "Specialization (generalization)", and others.

Question Text. This is the text displayed to the student when the dialog is initiated. If the question is multi-choice, then after the question comes a list of options. There can be any number of options, but there should be at least two for a multi-choice question. If the question is short answer the list is not specified.

Correct Answer(s). Each question can have any number of correct answers, and the student's answer only needs to match one of them. If the question is multi-choice, one of the options from the question text component must be a correct answer.

Incorrect Answer Feedback Pair. If the author wishes they can specify feedback to give if the student enters incorrect answers. Different feedback can be written for common errors the student may make. If the answer the student gives is not in this list, no additional feedback will be given.

Maximum Number of Attempts. For some questions, such as True/False questions (which are a special sort of multi-choice), it only makes sense to allow the student to have one attempt. For other, more complex questions, the student may be allowed multiple-attempts. The maximum number of attempts the student can make on the question before the system makes a judgment on their knowledge is specified here. Once again this is to provide flexibility and extensibility to the implementation.

```
(3
  ‘‘identifying relationships’’
  (‘‘An attribute of a weak entity type that is used to identify
entities of this type in combination with the key of the owner is
called a _________’’ nil)
  (‘‘partial key’’)
  ((‘‘key’’ ‘‘Incorrect.  A weak attribute does not have a key
attribute.’’)
   (‘‘unique’’ ‘‘Incorrect.  A weak attribute does not have any
unique attributes.’’)
   (‘‘primary key’’ ‘‘Incorrect.  A weak entity type does not have a
primary key attribute.’’))
  2
)
```

Figure 7: Question number 3

Figure 7 shows an example of a question from the negotiable student model in EER-Tutor. It is relevant to the “identifying relationships” concept, and is a plain text question with an answer of “partial key”. If the student enters “key”, “unique”, or “primary key”, they will get some custom feedback. Students are allowed two attempts at this problem; if their second attempt is incorrect, the system will decrease the “identifying relationships” concept in their student model by a small amount. This amount is set to 0.01, but can be changed easily to give more or less weighting to the negotiable student model questions.

```
(6
  ‘‘attributes’’
  (‘‘A simple attribute is an attribute that:  (choose one)’’
   (‘‘Has more than one value at any one time for each entity.’’
    ‘‘Cannot be broken down into smaller parts.’’
    ‘‘Has a unique value for each entity.’’
    ‘‘Cannot be derived from other attributes.’’))
  (‘‘Cannot be broken down into smaller parts.’’)
  ((‘‘1’’ ‘‘Incorrect.  A simple attribute cannot be broken down
into components.’’)
   (‘‘3’’ ‘‘Incorrect.  This is true for key attributes!’’)
   (‘‘4’’ ‘‘Incorrect.  This is the definition of stored
attributes!’’))
  2
)
```

Figure 8: Question number 6

Figure 8 shows another example of a question. The question is multi-choice, with four possible choices. The correct answer is the second choice (*Cannot be*

broken down into smaller parts.). When specifying incorrect feedback for multi-choice questions it is possible to specify the incorrect answer as the number of the answer in the option list. This question is relevant to the "attributes" concept, and has a maximum of two attempts allowed.

3.3.1 Question Selection

Currently there are 48 questions in the negotiable student model component, distributed over concepts as shown in Table 1.

Concept	Questions
Entities	8
Weak entities	6
Relationships	8
Identifying relationships	1
Attributes	10
Specialisations (generalisations)	10
Categories	0
Connections	5

Table 1: Questions per concept

The distribution is somewhat arbitrary, notably *Identifying relationships* and *Categories* are rather lacking. This initial set of questions was converted from another source, to save development time. Before this system is used for anything more than a simple evaluation, more questions should be added.

Questions are selected by a simple algorithm:

- With a list of all the questions relevant to the selected concept:
  - Iterate over the list until a question is found that the student has not answered correctly. When one is found, select it. If the student has answered all the questions, go to the next step.
  - Iterate over the list again, and for each question generate a random number between zero and one. If this number is greater than 0.7, select the question.
  - If no question has yet been selected, select the first relevant question from the list.
- If there are no relevant questions, select a dummy question that alerts the student there was no question relevant to the concept.

3.3.2 Negotiable Student Model Representation

The negotiable student model is stored in a simple manner. For each concept, a numerical value is recorded, which can be positive or negative. A positive

value means the student has answered more questions (from the negotiable student model component) correctly than incorrectly. Conversely, a negative value means the student has answered more questions incorrectly than correctly. A list of the questions the student has answered correctly is also stored. Figure 9 shows an example of a negotiable student model after a reasonable period of use on the system. In this example, for the "Weak entities" concept, the student has answered one more question incorrectly than correctly, giving a value of -0.01. The actual number of questions answered for the concept is not recorded. As such, there is no distinction made between one correct and two incorrect answers, or two correct and three incorrect answers; both situations will give the value -0.01.

```
((''Entities'' 0.07)
 (''Identifying relationships'' -0.01)
 (''Weak entities'' -0.01)
 (''Connections'' 0.03)
 (''Relationships'' 0.05))
(8 2 3 1 48 47 46 45 22 28 27 26 25 24 44 43 42 41 20)
```

Figure 9: A negotiable student model

To display a single representation to the student, the negotiable student model needs to be combined with the standard student model. When displaying the combined model, for each concept three variables need to be calculated: material learned (correct), material to learn (incorrect), and material not covered. They are calculated as follows:

correct_ val = min(material covered, max(0, learned + negotiated))

*material learned = correct_ val * 100*

*material to learn = (material covered - correct_ val) * 100*

*material not covered = 100 - (material covered * 100)*

correct_ val is a temporary value used to simplify the later calculations. This gives *material learned*, *material to learn* and *material not covered* as percentages, adding up to 100%. *learned* is the competence value from the standard student model, which is between zero and one, and *negotiated* is the numerical value from the negotiable student model. *material covered* is the amount of material covered for the concept, from the standard student model. These calculations ensure that the combined display makes sense:

- The amount of correct knowledge is never more than the amount of material covered.
- The amount of correct knowledge is never less than 0.

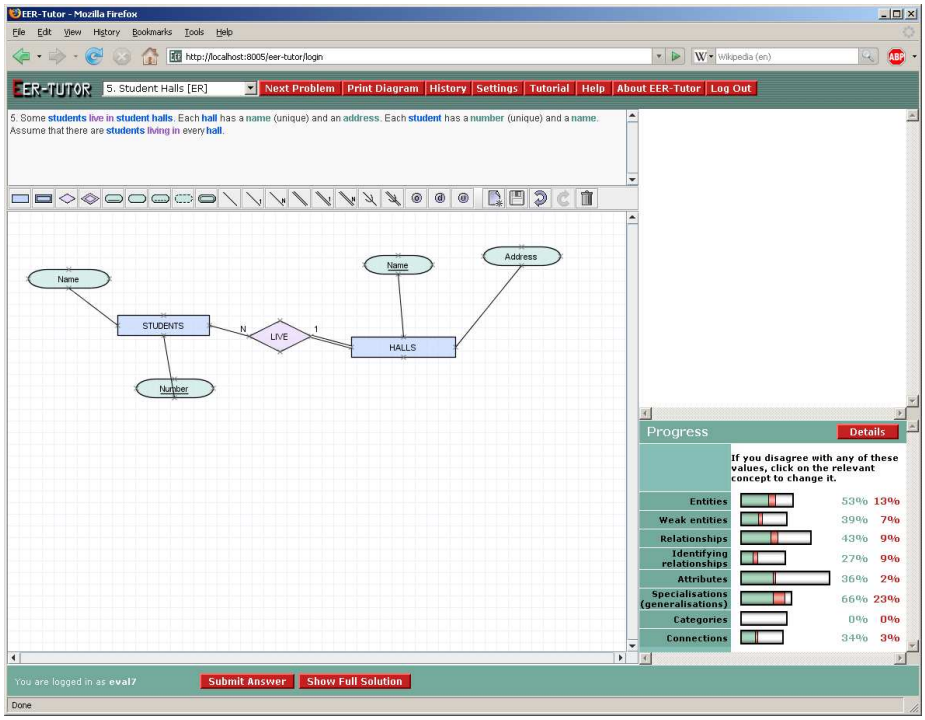

Figure 10: The EER-Tutor Interface

Figure 10 shows the EER-Tutor interface, with the negotiable student model component added. The same colour scheme has been used as throughout the system to maintain consistency of the overall appearance.

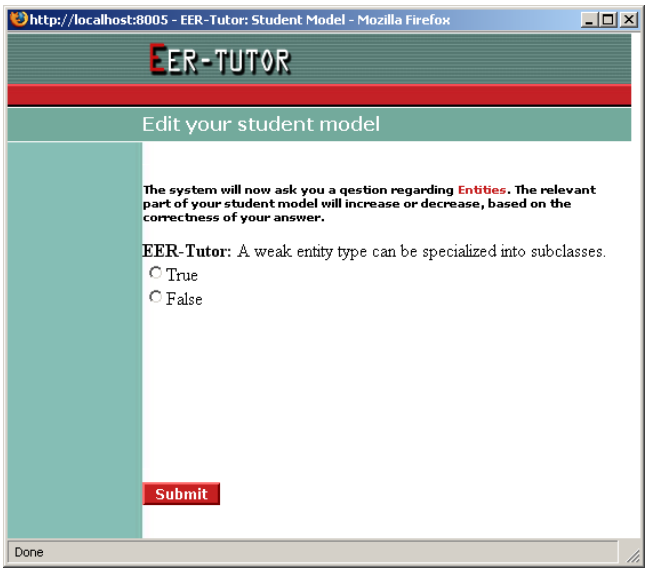

Figure 11: The dialog window

Figure 11 shows the window displayed when the student clicks on one of the concepts from the student model (in this case "Entities").

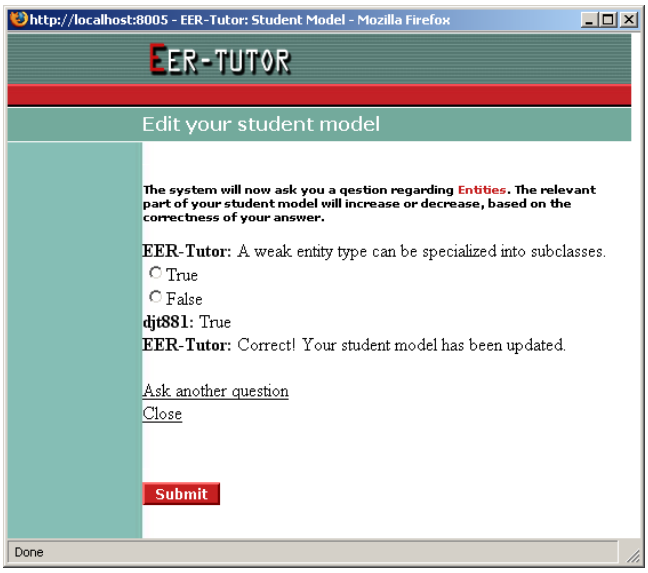

Figure 12: The student correctly answering a question

Figure 12 shows the dialog window after the student has successfully an-

swered the question. The student is informed their model has been updated, and is given the option to be asked another question, or to close the window.

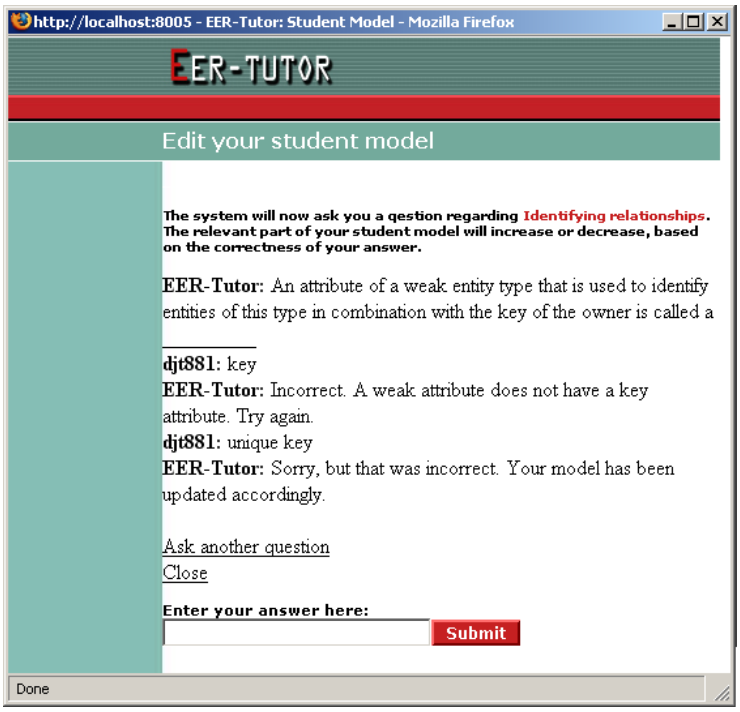

Figure 13: The student incorrectly answering a question

Figure 13 shows the dialog window after the student has incorrectly answered the question (twice). After the first attempt a small piece of advice is given. After the second attempt the student is informed their model has been updated, and once again they are given the option to be asked another question or to close the window. If custom feedback is written for the last incorrect attempt, the feedback will not be displayed, as the student has no chances to answer the question left.

4 Evaluation

To conclusively evaluate this system, a wide-scale, detailed, objective evaluation should be performed. However, due to the timing of this project, and the suitable course, it was not possible to run such an evaluation this year. If desired however, objective evaluation of this system could easily be run in any following year.

4.1 Ideal Objective Analysis

Based on the feedback from the questionnaire described in the next section, some small improvements should be made before performing a large scale evaluation. This would include adding more questions, and making small adjustments to the interface. Refer to the Future Work section for details on these improvements.

When performing the evaluation, the participants (students enrolled in the course) should be placed into one of two groups: Experimental and Control. The experimental group would use the version of EER-Tutor with the negotiable student model; the control group would use the standard version. After a significant period of use, objective results could be obtained from the two groups. This would include time spent on the system, time spent using the negotiable student model component (answering questions), number of problems correctly solved, and number of problems attempted, among others. Learning curves and pre/post-test results for the two groups could also be compared. This would give an objective comparison of the two versions of EER-Tutor, and could be used to determine the effectiveness of the negotiable student model as a learning aid. If the learning curves and differences between pre and post test results for the experimental group were significantly higher than that of the control group, it would mean that the negotiable student model is in fact more useful as a learning aid than a standard open student model.

Such an evaluation would take no development effort, as EER-Tutor already supports detailed logging, and pre/post-tests. The benefits of performing an evaluation as part of a university course are that students will be more motivated to actually use the system, and if desired, their exam results can be studied.

4.2 Subjective Survey

4.2.1 Design

Although an objective analysis could not be performed, this version of EER-Tutor with a negotiable student model was evaluated by a small user questionnaire. Eleven participants were involved; three experts (people who had worked on the development of EER-Tutor), and eight volunteers. Participants were asked to use the system for an undetermined period of time, until they had a good feel for the system. They then completed a questionnaire, which consisted of ranking the system on five aspects, and some open-ended questions which aimed to give the opportunity for participants to voice their opinions on

the system. All participants (both experts and volunteers) were postgraduate Computer Science students.

4.2.2 Results

Table 2 shows the average ranking of aspects of EER-Tutor with the negotiable student model (NSM). For each question, participants were asked to select a value on a scale of one to five, with one representing *not at all*, and five representing *very much*.

Question	Average ranking	Std. deviation
Did you enjoy learning with EER-Tutor?	4.00	1.04
Did you find the NSM interface easy?	4.45	0.50
Did the NSM help you to learn?	3.90	0.90
Was the behaviour of the NSM logical?	3.90	0.90
Did you find the NSM distracting?	1.45	0.66

Table 2: Participants ranking of aspects of EER-Tutor with the negotiable student model.

All of these results are positive. Students found the negotiable student model easy to use, helpful, reasonably logical, and not distracting. In addition to these rankings, participants were given the opportunity for comments, and to answer some open-ended questions. The general response was positive, although a few implementation flaws were noted. The comments are summarised as follows:

- Almost all the students felt that the negotiable student model would be beneficial to learning, and many found that it was an interesting challenge.
- The negotiable student model indirectly teaches material, but in a different way than the rest off EER-Tutor. Some students found this to be a welcome change from the at times repetitive nature of EER modeling.
- There was a lot of feedback about repeated questions. For some concepts, notably *Identifying Relationships*, there were not enough questions. After a short period of use the same questions were being asked, which should not occur. There was also a high percentage of True/False questions, which are quite easy to guess, which contradicts the principles of having to prove knowledge.
- A number of participants noted that after you answered a question you could submit your answer again, although nothing would happen.
- Some students were confused about how the student model was updated. When the model shows 100% correct knowledge, answering questions correctly will have no visible effect. Similarly, when there is no material covered on a particular concept, answering questions will also have no visible effect. This behaviour confused some students, although not in a major way.

- One of the aims of the negotiable student model was to not be a replacement for solving domain problems. Although one expert noted they began to "chase bars", rather than read feedback, most students did not find the negotiable student model distracting, and spent almost all of their time working on domain problems.

Overall the feedback was very positive. Any negative comments concerned the number of questions, or a few quirks in the interface. These problems can be fixed easily, which should lead to an enjoyable and useful addition to EER-Tutor.

5 Future Work

The questionnaire described in section 4 highlighted a few areas of possible improvement. As well as these improvements there is the possibility to take the implementation further to provide new features.

5.1 Improvements

These improvements would be simple to implement, and should be made before the system is used for further evaluation.

More Questions. More questions should obviously be added before the system is used for a long period of time. There should be at least ten questions for each concept, with a mix of multi-choice and short answer. Some concepts already have ten questions, but others have few. More questions could also be added slowly over time, eventually resulting in a comprehensive set.

Disable input after answering a question. Once the student has answered a question the maximum number of times allowed, or has answered it correctly, the input should be disabled. For short answer questions the input box should be disabled, and for multi-choice questions the options should all be disabled. This will make it clear to the student that they need to ask for another question, or close the window.

Provide more explanation. Some students were slightly confused by the behaviour of the negotiable student model. This confusing behaviour occurs when the student's knowledge reaches certain bounds. Correct knowledge can not be increased past 100%, nor decreased past 0%, although questions can still be answered. The problem is not with the design, but the fact that this is not explained to the user anywhere. A simple solution would be to provide a dialog box that describes the actions of the negotiable student model, that students can view if they wish.

Improve question selection. The current question selection algorithm is, although functional, not very good. The algorithm should be improved to select truly random questions, while still selecting non-attempted questions over attempted questions, and attempted questions over correctly answered questions.

5.2 New features

These features, among others, could be added to the negotiable student model.

Relate questions to concepts. An interesting possibility, suggested by an expert, in the questionnaire, would be to relate questions for the negotiable student model to constraints from the domain. Doing this would allow the system to ask questions relevant to constraints the student had recently violated.

This however, may not be as good as it sounds, as it means students will most often be asked questions they do not know the answer to. The purpose of a negotiable student model is not to teach new material to the student, so this approach may not always be suitable.

Implement more modes. The negotiable student model implemented in this project is rather limited in modes of operation. An enhancement that could be made would be to implement system initiated dialog. When the system detected certain events, such as a dramatic change in the student model, it could force the student to answer some questions, to verify the model. When implementing this it would be important to realise that the student model will always have more dramatic changes when it is relatively small, and much smaller changes once the student has been using the system for some time.

Tighter integration. The negotiable student model implemented in this project is not tightly integrated into EER-Tutor. An ideal negotiable student model implementation would only have one student model, which is the negotiable one. This model would then be used for all pedagogical decisions. As stated, this was not an issue for this research, due to how the student model is used in EER-Tutor.

Natural language parser. An ideal enhancement to this project would be a natural language parser. To answer a question the student should be able to explain their knowledge, in plain English. The system should then infer whether the student has successfully answered the question from their explanation. The system has been designed to make this enhancement as easy to integrate as possible. The interface, dialog control, student model functions and question format will all stay the same. The only change required would be in answer evaluation; the current naive pattern matching would be replaced by a natural language parser.

6 Conclusion

This project aimed to design and implement a negotiable student model in EER-Tutor. Much research has been done on open student models [11, 12, 20, 21], but no negotiable student model has been implemented and evaluated in a large scale ITS. This project sets the stage for this evaluation to take place. A negotiable student model has been implemented, and evaluated subjectively with a user questionnaire. Although the questionnaire pointed out a few improvements that are needed, feedback was very positive. These required improvements are documented in this report, and will be trivial to implement. Once these improvements have been made, this negotiable student model can be thoroughly, objectively analysed, to determine if it is beneficial to the learning process. If it is shown that a negotiable student model does help students learn, this research could be used as a basis for implementing a negotiable student model in other ITSs.

The negotiable student model implemented in this project was designed to be simple enough to be implemented in a short time frame. As a result, it is by no means a feature-complete negotiable student model. Many enhancements and new features could be added, some of which have been described in this report.

Although no objective data has been collected, subjective results have been very positive. Almost all participants felt the negotiable student model would help them learn, and some noted it was a nice break from the ITS, and encouraged them to correct their knowledge. This enjoyment and added motivation in itself is important in any learning situation. Negotiable student models are welcomed by users, and should be considered for any ITS.

References

[1] Beck, J., Stern, M., Haugsjaa, E. Applications of AI in Education. Crossroads (special issue on AI), vol 3(1), pp. 11-15, 1996.

[2] Dewey, J. How we think: a restatement of the relation of reflective thinking to the educative process, Boston, Heath and Co, 1933.

[3] Bull, S., Cooke, N., Mabbott, A. Visual Attention in Open Learner Model Presentations: An Eye-Tracking Investigation. In C. Conati, K. McCoy & G. Paliouras (Eds.): User Modeling 2007: 11th International Conference, Springer-Verlag, Berlin Heidelberg, pp. 187-196, 2007.

[4] Bloom, B.S. The 2 Sigma Problem: The Search for Methods of Group Instruction as Effective as One-to-One Tutoring. Educational Researcher, 13, pp. 3-16, 1984.

[5] Vygotsky, L.S. Mind in society: the development of higher psychological processes. In M. Cole, V. John-Steiner, S. Scribner & E. Souberman (Eds.): Cambridge, MA. Harvard University Press, 1978.

[6] Mitrovic, A., Martin, B., Suraweera, P. Intelligent tutors for all: Constraint-based modeling methodology, systems and authoring. IEEE Intelligent Systems, special issue on Intelligent Educational Systems, vol. 22, no. 4, pp. 38-45, July/August 2007.

[7] Self, J. A. Bypassing the intractable problem of student modeling. In C. Frasson & G. Gauthier (Eds.): Intelligent tutoring systems: At the crossroads of artificial intelligence and education, pp. 107-123. Norwood, NJ: Ablex, 1990.

[8] Mitrovic, A., Martin, B. Evaluating the Effect of Open Student Models on Self-Assessment. Int. J. Artificial Intelligence in Education, vol 17, pp. 121-144, 2007.

[9] Ohlsson, S. Constraint-Based Student Modeling. Int. J. Artificial Intelligence in Education, vol 3(4), pp. 429-447, 1992.

[10] Mitrovic, A. COSC420 Lecture Slides, 2008.

[11] Bull, S. See Yourself Write: A Simple Model to Make Students Think. In A. Jameson, C. Paris and C. Tasso, (Eds.): User-Modeling: Proceedings of the Sixth International Conference (UM97). New-York: Springer, pp. 315-326, 1997.

[12] Bull, S., Pain, H. Did I say what I think I said, and do you agree with me?: Inspecting and Questioning the Student Model. In Y. Greer (Ed.): AIED Proceedings, pp. 501-508, 1995.

[13] Baghaei, N., Mitrovic, A. From Modeling Domain Knowledge to Metacognitive Skills: Extending a Constraint-based Tutoring System to Support Collaboration. In C. Conati, K. McCoy and G. Paliouras (Eds.): 11th Int. Conference on User Modeling, Corfu, Greece, pp. 217-227, 2007.

[14] Bauer, M., Gmytrasiewicz, P.J., Vassileva, J. Supporting Negotiated Assessment Using Open Student Models. UM2001, LNAI 2109, pp. 295-297, 2001.

[15] Mitrovic, A. An intelligent SQL tutor on the Web. Int. J. Artificial Intelligence in Education, vol. 13, no. 2-4, pp. 173-197, 2003.

[16] Suraweera, P., Mitrovic, A. An Intelligent Tutoring System for Entity Relationship Modeling. Int. J. Artificial Intelligence in Education, vol. 14, no. 3-4, pp. 375-417, 2004.

[17] Hartley, D., Mitrovic, A., Supporting learning by opening the student model. In: S. Cerri, G. Gouarderes and F. Paraguacu (Eds.): Proc. 6th Int. Conference on Intelligent Tutoring Systems ITS 2002, Biarritz, France, LCNS 2363, pp. 453-462, 2002.

[18] Mathews, M., Mitrovic, A., Thomson, D. Analyzing high-level help seeking behaviour in ITSs. In W. Nejdl et al. (Eds.): AH 2008, LNCS 5149, pp. 312-315, 2008.

[19] Bull, S., Dimitrova, V., McCalla, G. Preface for Special Issue (Part 1) Open Learner Models: Research Questions. Int. J. Artificial Intelligence in Education, vol 17, pp. 83-87, 2007.

[20] Bull, S., Kay, J. Student Models that Invite the Learner In: The SMILI Open Learner Modelling Framework. Int. J. Artificial Intelligence in Education, vol 17, pp. 89-120, 2007.

[21] Lazarinis, F., Retalis, S. Analyze Me: Open Learner Model in an Adaptive Web Testing System. Int. J. Artificial Intelligence in Education, vol 17, pp. 255-271, 2007.

[22] Van Labeke, N., Brna, P., Morales, R. Opening up the Interpretation Process in an Open Learner Model. Int. J. Artificial Intelligence in Education, vol 17, pp. 305-338, 2007.

[23] Hausmann, R., Vanhehn, K. Explaining Self-explaining: a contrast between content and generation, In: R. Luckin, K. Koedinger, Y. Greer (Eds.): Proc. Artificial Intelligence in Education, pp. 417-424, 2007.

[24] Suraweera, P., Mitrovic, A. KERMIT: a Constraint-based Tutor for Database Modeling. In S. Cerri, G. Gouarderes and F. Paraguacu (Eds.): Proc. 6th Int. Conference on Intelligent Tutoring Systems ITS 2002, Biarritz, France, LCNS 2363, pp. 377-387, 2002.

[25] Swanson, H.L. Influence of meta-cognitive knowledge and aptitude on problem solving. Journal of Educational Psychology, vol 82, pp. 306-314, 1990.

[26] Morales, R., Pain, H., Conlon, T. Effects of inspecting learner models on learners abilities. In J.D. Moore, C.L. Redfield, W.L. Johnson (Eds.): AIED: AI-ED in the wired and wireless future, Amsterdam, IOS Press, pp. 434-445, 2001.

[27] Kerly, A. & Bull, S. Children's Interactions with Inspectable and Negotiated Learner Models. In B.P. Woolf, E. Aimeur, R. Nkambou & S. Lajoie (Eds.): Intelligent Tutoring Systems: 9th International Conference, Springer-Verlag, Berlin Heidelberg, pp. 132-141, 2008.

[28] Dimitrova, V. STyLE-OLM: Interactive Open Laerner Modelling. Int. J. Artificial Intelligence in Education, vol 13, pp 35-78, 2003.

[29] Parker, L. Little wonders. Australian Educator, Spring 2008, pp. 18-20, 2008.

[30] Bennet, F. Computers as tutors: solving the crisis in education. Faben, 1999.

Appendix

User Questionnaire

1. Did you enjoy learning with this system (EER-Tutor) in general? (Please circle one)

Not at all				Very much
1	2	3	4	5

Comments:

2. Did you find the interface for the negotiable student model (NSM) component easy to use? (Please circle one)

Not at all				Very much
1	2	3	4	5

Comments:

3. Did you think the NSM component helped you learn? (Please circle one)

Not at all				Very much
1	2	3	4	5

Comments:

4. Did you find the behaviour of the NSM logical (easy to understand)?

Not at all Very much

1	2	3	4	5

Comments:

5. Did you find the NSM component distracting? (Please circle one)

Not at all Very much

1	2	3	4	5

Comments:

6. What did you like in particular about the NSM component?

7. Is there anything you found frustrating about the NSM component?

8. Do you have any suggestions for improving the NSM component?

Answers

Participant	Q1	Q2	Q3	Q4	Q5
1	5	5	5	5	1
2 (expert)	5	4	3	3	2
3 (expert)	5	5	4	5	3
4	4	4	4	3	1
5	1	5	5	5	1
6	4	4	4	3	1
7	4	4	4	3	2
8	4	5	3	3	1
9	4	4	2	4	1
10	4	5	5	4	2
11 (expert)	4	4	4	5	1