Individualizing Self-Explanation Support for Ill-Defined Tasks in Constraint-based Tutors

Amali Weerasinghe
Intelligent Computer Tutoring Group
Dept. of Computer Science & Software Engineering, University of Canterbury, New Zealand
acw51@cosc.canterbury.ac.nz

Antonija Mitrovic
Intelligent Computer Tutoring Group
Dept. of Computer Science & Software Engineering, University of Canterbury, New Zealand.
tanja@cosc.canterbury.ac.nz

Abstract. We present the first phase of a project with the goal of developing a general model of self-explanation support, which could be used in constraint-based tutors for both well- and ill-defined domains. We studied how human tutors provide additional support to students learning with an existing intelligent tutoring system designed to help students learn an ill-defined task (database modeling using the ER model). Although the tutors were not given specific instructions to facilitate self-explanation, there were instances when self-explanation support was provided. Analysis of these interactions indicates that they have helped the students to improve their understanding of database design. These findings will serve as the basis for defining the self-explanation model. We also discuss directions for future work.

Keywords: self-explanation, intelligent tutoring systems, student modeling, ill-defined tasks

INTRODUCTION

Studies indicate that some students acquire shallow knowledge even in the most effective Intelligent Tutoring Systems (ITS) (Aleven et al., 1999). Self-explanation (SE) is described as an "*activity of explaining to oneself in an attempt to make sense of new information, either presented in a text or in some other medium*" (Chi, 2000), and has been shown to facilitate the acquisition of deep knowledge (Chi et al., 1989). There are several ITSs that facilitate self-explanation, most of them teaching well-defined tasks. For example, SE-Coach (Conati, and VanLehn, 2000) prompts students to explain solved physics examples. In the PACT Geometry Tutor (Aleven et al., 1999), students explain solution steps by selecting definitions and theorems from a glossary. NORMIT (Mitrovic et al., 2004) is an ITS for data normalization, in which students are expected to self-explain while solving problems. All these domains are closed-ended, as problem solving is well structured, and therefore self-explanation expected from learners can be clearly defined. Database design is an open-ended task: the final result can be defined in abstract terms, but there is no algorithm to find it. Constraint-based tutors have demonstrated their effectiveness in teaching ill-defined tasks, such as database design and querying (Mitrovic et al., 2004), and software design (Baghaei et al., 2005). We have extended the database design tutor with a SE facility (Weerasinghe and Mitrovic 2006). Even though all the above mentioned ITSs facilitate self-explanation, only SE-Coach supports adaptive self-explanation customised the student's knowledge and self-explanation skills.

Our long-term goal is to develop a model of self-explanation which will provide adaptive support to learners for both well- and ill-defined tasks. Since we previously implemented self-explanation support for the database design tutor, the initial work on this project started with the same tutor. We are currently developing an SE model, which will be incorporated into EER-Tutor (Zakharov et al., 2005). In order to develop this model, we need to consider three basic decisions: when to prompt for self-explanation, what to self-explain and how to obtain self-explanations from learners. As the first step, we conducted an observational study to investigate how students interacted with EER-Tutor, while getting additional help by a human tutor through a chat interface.

A brief discussion of database design is given in the following section. Section 3 discusses the functionality of EER-Tutor, followed by a description of the observational study. Analysis of the student interactions are presented in the Observations Section. We then discuss how the findings from the study can be incorporated in the self-explanation model. Future work and conclusions are presented in the final section.

DATABASE DESIGN

Database design is a process of generating a description of a database using a specific data model. Most database courses teach conceptual database design using the Entity-Relationship (ER) model, a high-level data model originally proposed by Chen (1976). The ER model views the world as consisting of *entities*, and *relationships* between them. The entities may be physical or abstract objects, roles played by people, events, or anything else data should be stored about. Entities are described in terms of their important features (*attributes),* while relationships represent various associations between entities, and also may have attributes. There is no algorithm to use to derive the ER schema from a given set of requirements. The learner needs to decide on the appropriate constructs to use, such as types of attributes/entities. For example, the learner might be given a problem illustrated in Figure 1 (note that this is a very simple problem). From the problem text, it is obvious that *students* and *groups* are of importance. Therefore, the learner might start by drawing the entities first. Each student has an id, and the learner needs to use his/her world knowledge to realize that ids are unique, and therefore represent that attribute as a key attribute (shown on the diagram with as underlined). The number assigned to each group is unique, and therefore it should also be a key attribute. In Figure 1, the student has made a mistake by showing GROUP as a weak entity, and group number as a partial key. Next, the learner has to think about the relationships between identified entities. In the problem shown in Figure 1, students work in groups, and for each possible association between a student and a group, it is necessary to represent the role. The Role attribute describes the association, and therefore it should be an attribute of the relationship. The student also needs to specify other integrities, such as cardinality ratios (shown as N on the diagram) and participations (shown as single or double lines).

As can be seen from this simple case, there are many things that the student has to know and think about when designing databases. The student must understand the data model used, including both the basic building blocks available and the integrity constraints specified on them. In real situations, the text of the problem would be much longer, often ambiguous and incomplete. To identify the integrities, the student must be able to reason about the requirements and use his/her own world knowledge to make valid assumptions.

Database design, similar to other design tasks, is an ill-defined task, because the start/goal states and the problem-solving algorithm are underspecified (Reitman, 1964). The start state is usually described in terms of ambiguous and incomplete specifications. The problem spaces are typically huge, and operators for changing states do not exist. The goal state is also not clearly stated, but is rather described in abstract terms. There is no definite test to decide whether the goal has been attained, and consequently, there is no best solution, but rather a family of solutions. Design tasks typically involve huge domain expertise, and large, highly structured solutions.

Although design tasks are underspecified, Goel and Pirolli (1992) identify a set of 12 invariant features of design problem spaces, such as problem structuring, distinct problem-solving phases, modularity, incremental development, control structure, use of artificial symbol systems and others. Problem structuring is the necessary first phase in design, as the given specifications of a problem are incomplete. Therefore, the designer needs to use additional information that comes from external sources, the designer's experience and existing knowledge, or needs to be deduced from the given specifications. Only when the problem space has been constructed via problem structuring, problem solving can commence. The second feature specifies three problem-solving phases: preliminary design, refinement and detail design. Design problem spaces are modular, and designers typically decompose the solution into a large number of sparsely connected modules and develop solutions incrementally. When developing a solution, designers use the limited-commitment mode strategy, which allows one to put any module on hold while working on other modules, and return to them at a later time.

In previous work, we have shown that constraint-based tutors are highly effective in teaching ill-defined tasks such as database design (Suraweera & Mitrovic, 2004) and query definition (Mitrovic & Ohlsson, 1999; Mitrovic et al., 2004). Our tutors compare the student's solution to a pre-specified ideal solution, which captures the semantics of the problem, thus eliminating the need for a problem-solver, which is difficult (or even impossible) to develop for such instructional domains. The constraint-based tutors are capable of identifying alternate correct solutions as constraints check that the student's solution contains all the necessary elements, even though it might be different from the ideal solution specified by the teacher. Goel and Pirolli (1988) argue that design problems by their very nature are not amenable to rule-based solutions. On the other hand, constraints are extremely suitable for representing design solutions: they are declarative, non-directional, and can describe partial or incomplete solutions. A constraint set specifies all conditions that have to be simultaneously satisfied without restricting how they are satisfied. Each constraint tests a particular aspect of the solution, and therefore supports modularity. Incremental development is supported by being able to request feedback on a solution at any time. At the same time, CBM supports the control structure used by the designer (student), as it analyses the current solution looking at many of its aspects in parallel: if a particular part of the

solution is incomplete, the student will get feedback about missing constructs. CBM can be used to support all problem-solving phases. Therefore, we believe that CBM can be applied to all design tasks.

EER-TUTOR: ENHANCED ENTITY RELATIONSHIP TUTOR

EER-Tutor is aimed at the university-level students learning conceptual database design. For a detailed discussion of the system, see (Zakharov et al., 2005); here we present some of its basic features. The system complements traditional instruction, and assumes that students are familiar with the ER model. The system consists of an interface, a pedagogical module, which determines the timing and content of pedagogical actions, and a student modeller, which analyses student answers and generates student models. EER-Tutor contains a set of problems and the ideal solutions to them, but has no problem solver. In order to check the student's solution, EER-Tutor compares it to the correct solution, using domain knowledge represented in the form of more than 200 constraints. It uses Constraint-Based Modelling (Mitrovic, et al, 2004) to model the domain and student's knowledge. The interface (illustrated in Figure 1) is composed of three windows tiled horizontally. The top window displays the current problem and provides controls for stepping between problems, submitting a solution and selecting feedback level. The middle window is the main working area, in which students draw ER diagrams.

Feedback from the system is grouped into six levels according to the amount of detail: *Correct*, *Error Flag*, *Hint*, *Detailed Hint*, *All Errors* and *Solution*. The first level of feedback, C*orrect*, simply indicates whether the submitted solution is correct or incorrect. The *error flag* indicates the type of construct (e.g. entity, relationship) that contains the error. For example, when the solution in Figure 1 is submitted, *error flag* provides the message *Check your entities, that's where you have some problems*. This is associated with the error GROUP being modelled as a weak entity instead of a regular entity. *Hint* and *Detailed Hint* offer a feedback message generated from the first violated constraint. For the solution in Figure 1, the hint message is *Check whether all the weak entities are necessary. Check whether some of your weak entities should be represented using some other type of construct*. On the other hand, the corresponding detailed hint is more specific: GROUP *should not be an entity. It may be extra or you may want to represent it using some other type of construct*, where the details of the erroneous object are given. Not all detailed hint messages give the details of the construct in question, since giving details on missing constructs would give away solutions. A list of feedback messages on all violated constraints is displayed at the all errors level (as indicated in the right-hand pane in Figure 1). The ER schema of the complete solution is displayed at the final level (solution level).

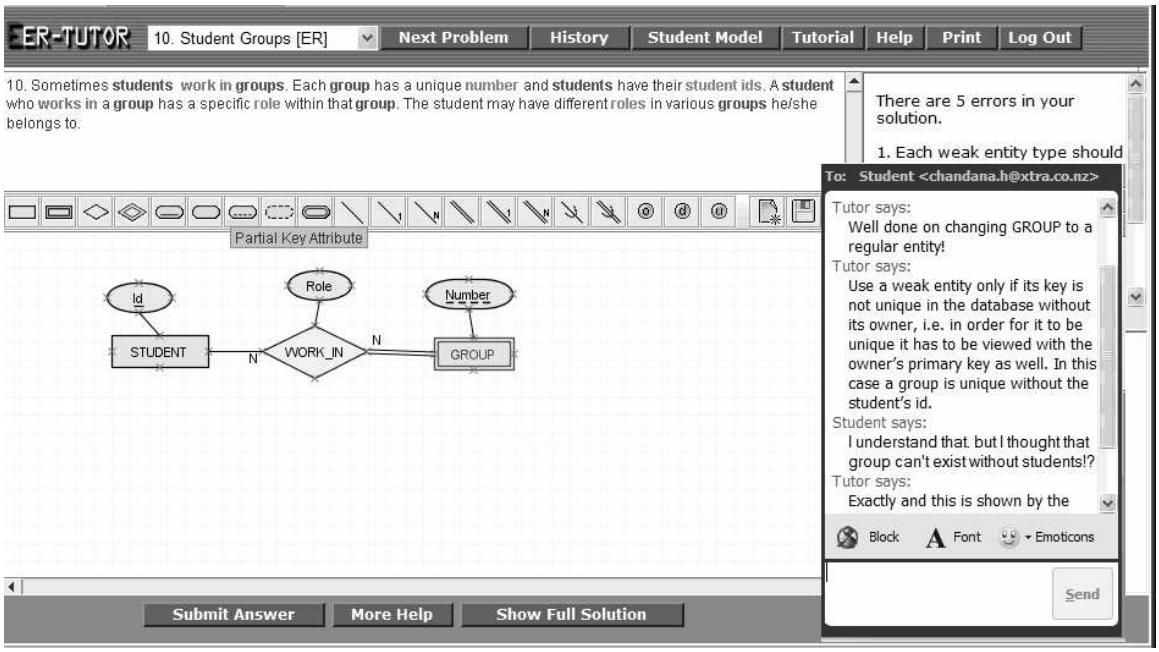

Fig. 1. Interface of the Enhanced EER-Tutor

When the student submits the first attempt at a problem, a simple message indicating whether or not the solution is correct is given. The level of feedback is incremented with each submission until the feedback level reaches the detailed hint level. In other words, if the student submits the solutions four times, the feedback level would reach the detailed hint level, thus incrementally providing more detailed messages. Automatically

incrementing the levels of feedback is terminated at the detailed hint level to encourage the student to concentrate on one error at a time rather than all the errors in the solution. The system also gives the student the freedom to manually select any level of feedback according to their needs.

PRELIMINARY STUDY

The study was conducted in August 2005 at the University of Canterbury, and involved students enrolled in an introductory database course and professional tutors. The professional tutors will be referred to as tutors, while EER-Tutor as the system or the ITS hereinafter. All the tutors had several years of experience providing assistance to students in labs and/or teaching small groups. The study was scheduled only after the relevant material was taught in the classroom. The version of EER-Tutor used was enhanced with a chat interface (Figure 1), so that the tutors could provide one-to-one feedback to students. We wanted to make the bandwidth between the student and the tutor to be similar to that between the student and the ITS. As the result, tutors could observe only the students' interactions with the ITS. Participants and tutors were located in separate rooms.

The tutors were expected to guide the students towards solutions using appropriate methods like asking questions etc. However, they were not given any specific instructions on providing assistance. Student participants were not told that a human was involved in the study. They also had the opportunity to initiate intervention through the chat interface or the *More Help* button in the interface.

At the beginning of the study, the students were asked to sit a pre-test online. All learner interactions were recorded. Students were expected to use the system for at least an hour. However, the students themselves decided when to end the session. Although initially we wanted the participants to sit a post-test immediately after the study, it was not possible due to another evaluation study which was conducted simultaneously. Therefore, the post-test was administered later on. All participants were asked to fill out a questionnaire at the end of the session to understand their perceptions about the system and interventions through the chat interface. At the end of each session, the tutors were also interviewed to understand their views on the tutoring experience.

We initially analysed the recordings without the tutors, to investigate how students were prompted by different tutors. Also we wanted to explore which interactions triggered these prompts. As the second step, whenever possible, the recordings were analysed with the tutors to clarify how they decided on the timing and the level of feedback provided through the chat interface.

The experimental set up of this study varies from previous studies of tutorial dialogue in a number of ways. First, the tutors were expected to provide additional support to the feedback given by the system. The tutors were also expected to respond to learners' questions. This contrasts with those studies of Chi. et al. (Chi and Siler, 2001) and Graesser, Person et al. (1995) in which the tutor was expected to lead the dialogue through a series of questions. Second, the learner interacts both with the system and the tutor. Although Merill et al. (1992) have studied tutorial dialogues in the context of problem-solving, the tutor was the only source of feedback for the student as s/he solved problems on paper. Finally, the tutors in our study needed to decide not only how to guide the student but also when. This differs significantly from the study in which the tutors analysed recorded interactions of students to perform motivation diagnosis (De Vicente and Pain, 2002).

OBSERVATIONS

Seven students and four professional tutors participated in the study, with at most two students per tutor. The mean on the pretest was 75.5% (sd=17.9), which was higher than the performance of the whole class (mean=58.1, sd=23.5). We expected this, as the participants were self-selected. Still the range of background knowledge was sufficiently large (ranging from 57% to 100%). Only two students have completed the post-test, hence it is not possible to compare the effect the learner interactions had on performance. The average duration of the sessions was 85 minutes (sd=20). The average number of attempted problems was 11 (sd = 5), and all participants completed all attempted problems. Average number of attempts per problem is 2.8 (i.e. received feedback from EER-Tutor that many times). We discuss observations in two different categories: (i) type of feedback provided in the interventions and (ii) timing of interventions.

Type of Feedback Provided

The interactions between the tutors and the students were analysed to identify different episodes, each pertaining to a single topic (Chi, 2001). There were a total of 69 episodes. In addition to discussing the current problem state, some episodes focused on helping with the interface (such as labeling constructs), motivate and praise the student, suggest to try a more challenging problem, complete the session or help with technical problems (e.g. web browser suddenly closing). The maximum and the minimum number of episodes initiated by a tutor during a session was 20 and 4 respectively. Surprisingly, these 20 episodes occurred in a session of 1.5 hour duration

which is not the longest session (the longest session lasted approximately 2 hours). In the session which consisted of 4 episodes, the first intervention occurred only in the 19th problem (the student completed 22 problems).

We are mainly interested in 37 episodes which discussed the current problem state or the relevant domain concepts. The following statistics were calculated using these 37 episodes. The average number of such episodes per tutor was 9.25. Five episodes contained a single utterance each, which was initiated by the tutor. For instance, a tutor utterance that occurred just after the completion of a problem was "Remember that the participation for weak entity is always total". The longest episode consisted of 9 utterances of which 4 were by the tutor. The student made more utterances than the tutor in only 2 episodes. Furthermore, only 2 episodes were student-initiated. This indicates that the tutor is more likely to be active in the interventions.

Only 20 (54%) episodes were considered to facilitate self-explanation. The criterion to label an episode as facilitating self-explanation is whether it discussed concepts that went beyond the current error, or facilitated justifications for the correct modelling decision. An example is presented in Figure 2, while the student was working on the problem shown in Figure 1. This dialogue contains two episodes, because it covers two different concepts (one related to weak entities, and the other one about total participation). Even though the tutor is leading the dialogue, the student has justified his decision for modeling GROUP as a weak entity (*Student1*). Therefore, the dialogue provided an opportunity for the student to explicitly self-explain his modeling decision. Also, the student was able to identify and repair the misconception he had with weak entities and total participation after the tutor's explanation (*Tutor3*). Student's utterance about learning material during this session (*Student2*) can be considered as evidence of implicit self-explanation.

Tutor1:	Well done on changing GROUP to a regular entity!
Tutor2:	Use a weak entity only if its key is not unique in the database without its owner, i.e. in order for it to be unique, it has to be viewed with the owner's key as well. In this case, a group is unique without the student's id.
Student1:	I understand that, but I thought that group can't exist without students!?
Tutor3:	Exactly, and this is shown by the total participation in the relationship.
Student2:	I have learned something today

Fig.2. A dialogue from the study

The highest number of self-explanation dialogues in a session was 7, while the lowest was 2. As can be expected, the highest number of self-explanation dialogues occurred in the longest session. Even though all tutors initiated interaction episodes, two of them did not have any self-explanation episodes. This may be because the tutors were not explicitly asked to facilitate self-explanation, but to assist the students with problem solving.

When data was analysed to identify different strategies used by tutors, three strategies were prominent. Tutors were rephrasing feedback, providing problem-independent explanations and stating their observations before starting to discuss the problem state. The tutors who did not have any self-explanation eposides in their sessions mainly rephrased feedback to enable the student to understand their own mistakes. For example, the tutor prompted "Does AUTHOR need to be an entity?" or "The cardinalities of BORROWED_FROM needed fixing". Rephrasing feedback may have been effective because most students realised that the additional feedback was provided by a human observing their problem-solving process. If the self-explanation model is to repeat the same kind of prompting, it is difficult to ascertain whether it will have the same effect (Lepper, et al., 1993). The second strategy was to discuss the current problem state and then provide a problem-independent explanation. Figure 1 represents an example. These explanations provided an opportunity for the student to repair his/her mental model of the domain and generated further conversation. The third strategy used was to state the tutor's observations before starting to discuss the problem state. For example, tutor started the dialogue by saying "You seem to be having a few problems with relationships. Think about this. Can a student be enrolled in a course without involving a department?"

As the knowledge base in EER-Tutor is represented as a set of constraints, the errors were recorded as constraint violations. We analysed how frequently constraints were violated after related errors were discussed in self-explanation episodes, to see whether tutor interventions helped students to improve their knowledge. If these constraints represent psychologically appropriate units of knowledge, then learning should follow a smooth curve when plotted in terms of constraints (Anderson, 1993). To evaluate this expectation, the participants' logs were analysed, and each problem-state after a tutor invention in which a constraint was relevant was identified. These identified occasions are referred to as *occasions of application* (Mitrovic, et al, 2004). Each constraint relevance occasion was ranked 1 to *n*. For each occasion we recorded whether a relevant constraint was satisfied or violated. We then calculated the probability of violating a constraint on the first occasion of application, the

second occasion and so on, for each participant. The probabilities were then averaged across all participants and plotted as a function of the number of occasions when a constraint was relevant (Figure 3).

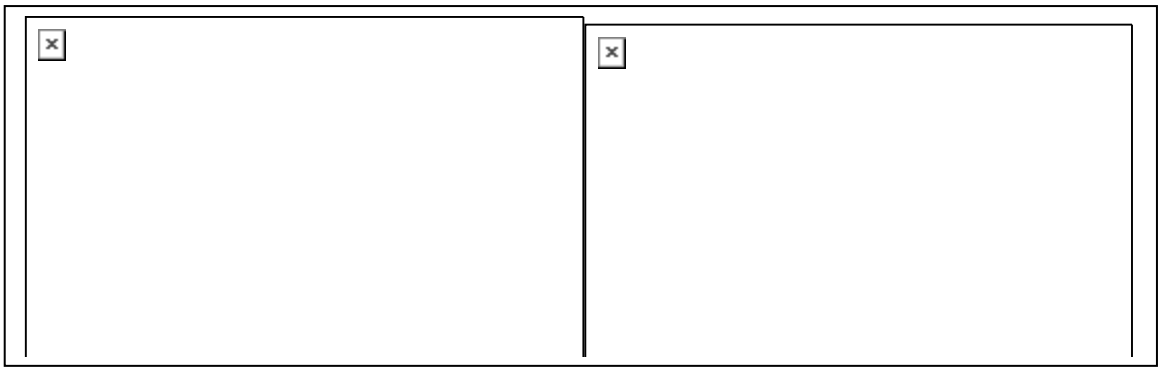

Fig. 3. Probability of violating a constraint as a function of number of occasions when that constraint was relevant

As can be seen from Fig. 3.a, there is an outlier, increasing the probability of violating a constraint in the 4th and the 5th occasions. This is due to a single student violating the constraint dealing with total participation. For this student, the tutor provided a problem-independent explanation to help him identify and repair the misconception he had with weak entities and total participation (Figure 1). The explanation was not related to a problem state later on, as the discussion was a follow-up from another error related to weak entities. This may have been a reason for the subsequent violations of this constraint.

Figure 3.b shows the learning curve with the outlier removed. The probability of 0.22 for violating a constraint at its first occasion of application decreased to 0.02 at its eighth occasion of application, displaying a 90.9% decrease in the probability. The results of the mastery of constraints reveal that students seem to learn ER modelling concepts which were discussed by the tutors.

Twenty-eight different constraints were discussed in the self-explanation episodes. Three students did not violate any constraints in subsequent occasions after the tutor interventions. These students were tutored by three different tutors who followed strategies like rephrasing feedback, providing problem-independent explanations and stating tutor's observations at the beginning of the discussion. This suggests that all these strategies have been effective in helping the students learn domain concepts.

Timing of Interventions

The original version of EER-TUTOR provides feedback on demand, i.e. only when the student submits the solution. The tutors in this study also provided delayed feedback, which was well-received by the participants. Delayed feedback also provided an opportunity for students to correct the mistakes themselves. There were few instances where the student made a mistake and corrected it after referring the problem text again. For example, one of the problems required students to model CAR as an entity and *Colour* as a multi-valued attribute of CAR. The student modelled *Colour* as a simple attribute and then changed it to multi-valued as the last sentence in the problem text indicated that a car can have many colures. In such a situation, immediate feedback would not have been welcomed by the student, as he may have felt the intervention being intrusive.

The important issue with delayed feedback is how the tutors decided that the students needed help. In our study tutors provided help when the student (i) made the same type of mistake repeatedly (ii) asked for more help using the *More Help* button (iii) was inactive for some time, (iv) reacted to feedback, or (v) asked a problem-specific question through the chat interface. These scenarios will be discussed in detail in the next section.

Prototype of the Self-Explanation Model

The self-explanation model will be used to decide when to prompt to self-explain, what to self-explain and how to obtain self-explanations from learners. The model consists of three parts: error hierarchy, self-explanation dialogues and meta-constraints. Each component is now described in turn.

A hierarchy of errors was developed after analyzing different students' errors (Weerasinghe, 2003). A high-level view of the hierarchy is given in Figure 4. Nodes in this hierarchy are ordered from basic domain principles to more complicated ones. Violated constraints for each type of error are represented as leaves of the hierarchy. Some error types are further divided into sub errors. Constraints for some nodes are given in separate lines to indicate that each constraint corresponds to a specific error type. For instance, each constraint assigned to the node *Syntax Errors* specifies a different type of error. The error hierarchy enables the system to

locate the appropriate dialogue to be used in case of incorrect student submission. If there are multiple errors, depth-first search identifies the dialogue to initiate with the student.

Self-explanation is facilitated through tutorial dialogues. We designed a tutorial dialogue for each type of error. Each dialogue consists of 4 stages. In the first stage, the model informs the student about the concept that s/he is having difficulty with, and then asks for a justification of the student's action. The purpose of the second stage is to assist the student in understanding why his action is incorrect. The third stage prompts the student to specify how to correct the mistake. A further opportunity is provided in the fourth stage to review the domain concept learned. An example of a tutorial dialogue is given in Figure 5. Initially, the system identifies the domain concept the student has problems with, and asks the student to explain it (*EERTutor1*). We have yet to decide how to obtain self-explanation from learners i.e. whether to provide a list of possible answers from which the correct one could be selected or to incorporate natural language processing module enabling learns to use free-form questions. If the student fails to provide the correct answer (*Student1*), s/he will be asked a more specific question that provides a further opportunity to understand the fundamental principle that is violated (*EERTutor2*). However, if s/he fails to correct the mistake even after going through a series of detailed questions, as the last resort the tutor will provide an explanation on how to the correct the mistake together with a brief description about the fundamental principle that needs to be learnt (*EERTutor5-7*). The dialogues use various types of interactions such as simple questions (*EERtutor1*), fill-in-a-blank (*EERTutor7*), or true-false questions, to motivate the student to self-explain. When a certain mistake is repeated, the model informs the student of its observations (*EERTutor1*), thereby providing an opportunity to reflect on his/her domain knowledge. As all dialogues facilitate self-explanation by pointing out errors (*EERtutor3*), students are given opportunities to reflect on their problem solving procedure, which is another important meta-cognitive skill.

ALL ERRORS
- Syntax errors
  - 1 – More than one construct is used to model a word of the problem
  - 2 – Database does not have any entities
  - 3 – A connector is not connected to two entities
  - 4 – A connector is used to connect attributes is not a single line with no cardinality
  - 5 – An attribute is connected to more than one other attribute
  - 6 – An attribute is connected to an entity and another construct
  - 101 – An attribute is connected to a relationship and another construct
  - 7 – An entity is directly connected to another entity
  - 8 – An attribute is not connected to any other construct
  - 9 – A relationship is connected to another relationship
  - 10 – Names of entities and relationships are not unique
  - 21 – A relationship is not connected to two entities
- Semantic errors
  - Using an incorrect construct type
  - Extra constructs
  - Missing constructs
  - Connecting an attribute to an incorrect construct
  - Errors dealing with cardinalities and participation

Fig. 4 Overall view of the error hierarchy

Ideal SE behaviour will be represented as a set of meta-constraints, and will enable individualization of the self-explanation dialogues. As we discussed previously, the SE dialogues are pre-specified sequences of questions; however, for each individual student, the SE model will decide on the entry point to the dialogue, or the timing of the dialogue. The observations from the study will be used to develop meta-constraints.

As delayed feedback provided by the tutors in this study was well-received by the student participants, the self-explanation model will also provide delayed feedback. The critical issue then is to decide when it will be beneficial to intervene. As noted earlier, tutors intervened when the student (i) made the same type of mistake repeatedly, (ii) asked for more help using the *More Help* button, (iii) was inactive for some time, (iv) reacted to feedback, or (v) asked a problem-specific question through the chat interface.

When a student made the same error repeatedly, tutors provided a problem-independent explanation of the domain concept they have difficulty with. Some tutors initiated the dialogue by stating their observations. For instance, if it is difficult for the student to identify a weak entity, the tutor's initial response was "*You seem to be having some difficulty with regular entities. Let's look at regular entities in detail*." followed by an opportunity to discuss the corresponding domain concept. One meta-constraint will check for these situations, and will be

violated when the same error is made in the last *n* attempts. In that case, a dialogue corresponding to the mistake will be initiated, but the dialogue would start from the problem-independent question (*EERTutor1* in Figure 5).

As noted in (ii) (asking for more help using the *More Help* button), the student will be given an opportunity to receive more help for each feedback message provided by the system. If more help is requested, then the corresponding self-explanation dialogue will be initiated. For instance, if the student requested more help on CHAPTER being modeled as a regular entity and he has not made the mistake repeatedly, then dialogue will discuss the error within the current context (*EERTutor3* in Figure 5). Hence the self-explanation dialogues will be adaptive based on the student's domain knowledge and the self-explanation skills.

Even though all tutors intervened when a student has been inactive for about a minute, they tended to wait longer when the student is in the initial problem-solving phase (i.e. student has not submitted his solution and has not received any feedback from the system so far). One of the meta-constraint will identify that situation, by checking whether the student has made any attempts at the current problem, and has been inactive for a specified period of time. (such as 1.5 minutes, the time period we observed in the study). Violation of this constraint will initiate an evaluation of the student's solution even though it has not been submitted yet, and also prompt the student to identify which concept he is having difficulty with. For instance, if the student is having difficulty with the regular entity CHAPTER, and the evaluation of his solution identifies that it needs to be modeled as a weak entity, then the self-explanation model will help the student to understand the correct modeling decision through a series of questions. Figure 5 presents a sample dialogue that occurs between the student and the tutor in this case.

A student seems to be just reacting to feedback if he makes a single change without reflecting on the other changes that need to be carried out as a result. In ER modeling, certain changes trigger a number of other changes in order to develop a syntactically correct model. For instance, if a regular entity with a key attribute is changed to a weak entity, then a partial key should be specified instead of the key attribute. Also one of the relationships of the regular entity needs to be changed to an identifying relationship etc. Sometimes students tend to make a single change suggested by the system and submit again for more feedback. This may lead to frustration and a feeling of exhaustion due to the number of attempts that the student has to go through to arrive at the correct solution. In such a situation, the student will be prompted to reflect on other related changes before submitting the solution. An example meta-constraint which identifies such situations is given below; when this constraint is violated, the system will remind the student to check whether other changes are necessary.

> Relevance Condition: If the student has changed the solution to correct one mistake (i.e. violated constraint),
> Satisfaction Condition: Then he should also make related changes.

For scenario (v) (asking problem-specific questions through the chat interface), the self-explanation model needs to enable learners to ask free-form questions while interacting with the system. We are currently investigating different alternatives. One alternative is to use TFIDF (Term Frequency Inverse Document Frequency) vector weighting scheme (Salton and Buckley, 1987) to retrieve information from a pre-defined question database. TFIDF is currently being used in another project by our group developing a question asking module while solving database mapping problems with the ERM-Tutor (Milik, et al., 2006). Another alternative is to incorporate a natural language processing module to the EER-Tutor.

EERTutor1: You seem to be having some difficulty with regular entities. Let's look at regular entities in detail. Can you tell me the general rule to decide whether something is a regular entity?
Student1*: I don't know*
EERTutor2: Ok! Let's try a simple thing. What is used to uniquely identify instances of a regular entity?
Student2*: The key attribute*
EERTutor3: Now tell me what is unique about *CHAPTER* regular entity?
Student3*: I'm not sure*
EERTutor4: Let me help you a bit. Check carefully whether any of the attributes of *Chapter* can uniquely identify the instances of *Chapter?*
{After sometime} Please tell me whether you managed to find any attributes
Student4: *No*
EERTutor5: Fine. It doesn't have any attributes that can uniquely identify instances of *Chapter*. So now can you tell me why modeling *Chapter* as a regular entity is incorrect?
Student5 : *Because it doesn't have a key attribute.*
EERTutor6: Great! Now tell me the best way to model *CHAPTER*?
Student6*: A weak entity*
EERTutor7: Well Done! Now select the correct pair of words to fill in the blanks so that you'll remember what you learnt.
A ….. is used to uniquely identify instances of a …….. entity.
Student7*: key attribute ,regular*
EERTutor8: Excellent, Go ahead and do the changes. Submit the solution when you're done.

Fig. 5. A sample tutorial dialogue

The short-term student model in EER-Tutor consists of a list of satisfied and a list of violated constraints for the student's solution, while the long-term model records the history of each constraint (Zakharov et al., 2005). In the enhanced version of EER-Tutor, the long-term model will additionally record details of the self-explanation skill of a student in terms of types of errors made (in terms of meta-constraints) and the level of prompting the student needed to correct his mistake for every constraint.

Conclusions and Future Work

Self-explanation is an effective learning strategy to facilitate deep learning. This research focuses on developing a self-explanation model for both ill- and well-defined tasks. As the first step, we conducted a preliminary study to observe how tutors prompt students to guide them towards solutions while using EER-Tutor, a constraint-based tutoring system for learning Entity-Relationship modelling. In addition to the feedback received by the system, the students were prompted by the tutors through a chat interface. Students also had the opportunity to initiate a dialogue with the tutor either through the chat interface or using the *More Help* button.

The interactions between the tutors and the students were analysed to identify the different episodes, each pertaining to a single topic. Only 20 (54%) of these episodes that discussed the current problem state or the relevant domain concepts were considered to facilitate self-explanation. These episodes either facilitated the discussion of domain concepts that went beyond the current error, or prompted justifications for the correct modelling decision. The user logs were analysed to investigate how frequently a certain error occurred after each self-explanation episode. The analysis indicated that in spite of different tutoring strategies, the tutor interventions helped the learners to improve their understanding of ER modelling concepts.

The findings from the reported study are being used to develop the self-explanation model for the EER-Tutor. The next step is to incorporate the model into the EER-Tutor, and evaluate it in an authentic classroom environment. We will then implement the same SE model in an ITS for a well-defined task.

References

Aleven, V., Koedinger, K. R., Cross, K. Tutoring Answer Explanation Fosters Learning with Understanding. In: Artificial Intelligence in Education, Lajoie, S.P. and Vivet, M.(eds.), IOS Press (1999) 199-206.

Anderson, J. R., Rules of the mind. Erlbaum, Hillsdale, NJ, 1993.

Baghaei, N., Mitrovic, A., Irwin, W. A Constraint-Based Tutor for Learning Object-Oriented Analysis and Design using UML. In: C.K. Looi, D. Jonassen, M. Ikeda (eds), ICCE 2005, 11-18.

Chen, P. (1976). The ER Model - Toward a Unified View of Data. ACM Transactions Database Systems, 1(1), 9-36.

Chi, M. T. H. Self-explaining Expository Texts: The dual processes of generating inferences and repairing mental models. Advances in Instructional Psychology, (2000) 161-238.

Chi, M.T.H., Bassok, M., Lewis, W., Reimann, P., Glaser, R., Self-Explanations: How Students Study and Use Examples in Learning to Solve Problems. Cognitive Science, 13 (1989), 145-182.

Chi, M. T. H., S. A. Siler, et al. Learning from human tutoring. Cognitive Science 25 (2001), 471-533.

Conati, C., VanLehn, K. Toward Computer-Based Support of Meta-Cognitive Skills: a Computational Framework to Coach Self-Explanation. Int. J. Artificial Intelligence in Education, 11 (2000) 389-415.

Goel, V., Pirolli, P. Motivating the Notion of Generic Design with Information Processing Theory: the Design Problem Space. AI Magazine, 10 (1988) 19-36.

Goel, V., Pirolli, P. (1992) The Structure of Design Problem Spaces. Cognitive Science, 16, 395-429.

Graesser, A. C., Person, N. et al. Colloborative dialog patterns in naturalistic one-on-one tutoring. Applied Cognitive Psychology, **9**, (1995) 359-3

Lepper, M.R., Woolverton, M. Mumme, D. L., and Gurtner, J.L. Motivational techniques of expert human tutors: Lessons for the design of computer-based tutors. In Lajoie, S.P. and Derry, S. J. (eds.) Computer as Cognitive Tools, Lawrence Erlbaum, Hillsdale, New Jersey, (1993)75 -105.

Milik, N., Marshall, M., Mitrovic, A. Teaching Logical Database Design in ERM-Tutor. In: M. Ikeda & K. Ashley (eds) Proc. ITS 2006.

Mitrovic, A., Ohlsson, S., Evaluation of a Constraint-Based Tutor for a Database Language. Int. J. Artificial Intelligence in Education, 10 (1999), 238-256.

Merrill, D. C., Reiser, B. et al. Effective tutoring techniques: A comparison of human tutors and intelligent tutoring systems. Journal of the Learning Sciences 2(3) (1992) 277-305.

Mitrovic, A., Suraweera, P., Martin, B., Weerasinghe, A. DB-suite: Experiences with Three Intelligent, Web-based Database Tutors. Journal of Interactive Learning Research, 15(4), (2004) 409-432.

Reitman, W.R. (1964) Heuristic Decision Procedures, Open Constraints, and the Structure of Ill-defined Problems. In: M.W. Shelly, G.L. Bryan (eds) Human Judgements and Optimality. New York, Wiley.

Salton, G., Buckley, C., Term Weighting Approaches in Automatic Text Retrieval. Technical Report #87-881 Computer Science Dept, Cornell University, Ithaca, NY, 1987.

Suraweera, P. and Mitrovic, A., An Intelligent Tutoring System for Entity Relationship Modelling. *Int. J. Artificial Intelligence in Education*, v14n3-4, 375-417, 2004.

Weerasinghe, A. Exploring the effects of Self-Explanation in the Context of a Database Design Tutor. MSc Thesis, University of Canterbury, 2003.

Weerasinghe, A., Mitrovic, A. Facilitating Deep Learning through Self-Explanation in an Open-ended Domain. Int. J. of Knowledge-based and Intelligent Engineering Systems, 10(1),(2006) 3-19.

Zakharov, K., Mitrovic, A., Ohlsson, S., Feedback Micro-engineering in EER-Tutor. In: Looi, C-K., McCalla, G., Bredeweg, B., Breuker, J. (eds.) Proc. Artificial Intelligence in Education AIED 2005, IOS Press, (2005) 718-725.

De Vicente, A., Pain, H. Informing the detection of the students' motivational state: An empirical study. In: Cerri, S.A., Gouarderes, G., Paraguacu, F. (eds.), Proc. 6th Int. Conf. Intelligent Tutoring Systems (2002) 933-943.