Towards Individualized Dialogue Support for Ill-Defined Domains

Amali Weerasinghe, Antonija Mitrovic, Brent Martin, Intelligent Computer Tutoring Group, University of Canterbury, Christchurch, New Zealand
Amali.weerasinghe@pg.canterbury.ac.nz, Tanja.mitrovic@canterbury.ac.nz, Brent.martin@canterbury.ac.nz

Abstract. One of the critical factors contributing to the effectiveness of human tutoring is the conversational aspect of the instruction. Our goal is to develop a general model for supporting dialogues with menu-based input that could be used in both well- and ill-defined instructional tasks. We have previously studied how human tutors provide additional support to students learning with an existing intelligent tutoring system. On the basis of these findings we developed a model for supporting dialogues, which we present in this paper. We used this model in a Wizard-of-Oz study to provide adaptive support. The results show that students did learn the relevant domain knowledge and that human tutors mostly agreed with the interventions generated from the model.

Keywords. Meta-cognitive skills, tutorial dialogues

INTRODUCTION

Collaborative dialogues between students and tutors are a prominent component of effective tutoring (Graesser, Person & Magliano, 1995). These dialogues provide opportunities for students to reflect on existing knowledge and to construct new knowledge. Such skills are of special importance in ill-defined domains, which are underspecified: they are based on incomplete/imprecise theories, and/or lack prescriptions on how to solve problems.

We start by discussing related work in Section 2, followed by a description of database design, the domain from which we started this project. Then we present a preliminary study of human tutors. This study provided the information for designing our model for supporting adaptive dialogues with menu-based input, which is presented in Section 5. We conducted a Wizard-of-Oz study, using the model to generate dialogue prompts to students in order to evaluate the effectiveness of the model prior to full implementation. This study is presented in Section 6, together with the findings and further improvements to the model. We present the conclusions and the plans for future work in the final section.

RELATED WORK

Some of the dialogue-based tutoring systems that have been developed are Why2-Atlas (Jordan et al., 2006), AutoTutor (Graesser, VanLehn, Rose, Jordan, & Harter, 2001), Why2-AutoTutor (Jackson, Ventura, Chewle, Graesser, & Tutoring Research Group, 2004), CIRCSIM-Tutor (Millis, Evens, &
Freedman, 2004) and Geometry Explanation Tutor (Aleven, Ogan, Popescu, Torrey, & Koedinger, 2004). Why2-Atlas, AutoTutor and Why2-AutoTutor use the dialogues as the main activity to help students learn the domain knowledge. The other systems provide problem-solving environments as the main activity and use tutorial dialogues as a way of remediating errors in student solutions. For example, CIRCSIM-Tutor is a natural language (NL) tutor that helps students solve a set of problems in cardiovascular physiology relating to regulation of blood pressure. The students are engaged in multi-step dialogues based on two experienced human tutors. The dialogue planning is done within the APE framework (Freedman, Rose, Ringenberg, & VanLehn, 2000). Freedman’s approach for the tutorial model is a production system that is focused on having a hierarchical view of the dialog. The performance of 50 first-year medical students who interacted with CIRSCIM-Tutor improved significantly.

Geometry Explanation Tutor, an extension of PACT Geometry Tutor (Aleven, Koedinger & Cross, 1999), incorporates natural language understanding (Aleven, Popescu, & Koedinger, 2001) to enhance the learning experience. In Geometry Explanation Tutor, students explain in natural language, and the system evaluates their explanations and provides feedback. The system contains a hierarchy of approximately 200 explanation categories that represent partial or incorrect explanations commonly used by novices (Aleven et al. 2004). The system parses the student's explanation to generate a semantic representation which is classified according to the hierarchy of explanation categories. The dialogue management system decides the feedback to be presented to the student based on the classification of the student’s explanation. An empirical study was carried out to investigate whether self-explanation facilitated through natural language enhances learning better than self-explanation through menu selection. Even though the students who explained in natural language did not learn more than those who explained through menu selection, they did learn better to state explanations.

Atlas-Andes is the product of integrating the Andes physics tutoring system (Gertner & VanLehn 2000) with the Atlas tutorial dialogue system (Freedman et al., 2000). Andes is a model-tracing tutor that presents quantitative physics problems to students. Each problem-solving step entered by a student is highlighted in either red or green to indicate the accuracy of that step. Atlas enhances the learning experience of Andes by leading students through directed lines of reasoning to teach conceptual physics knowledge. When Atlas recognizes an opportunity to encourage deep learning, it initiates a natural language dialogue with the student. The main objective of these dialogues is to facilitate knowledge construction: hence, the dialogues are known as knowledge construction dialogues (KCDs). KCDs provided by Atlas are currently limited to teaching domain principles. An empirical study revealed that the students interacting with Atlas learnt significantly more than students who interacted with ANDES (Rose et al., 2001)

Another tutoring system that engages students in natural language dialogue is Graesser et al.’s AutoTutor (Graesser, Wiemer-Hastings, Wiemer-Hastings, Kreuz, & Tutoring Research Group, 1999). It is used in an introductory course in computer literacy. The system improved the students’ learning by 0.5 standard deviation units when compared with a control group of students who read the same chapters from a book. AutoTutor requires students to provide lengthy explanations for the How, Why and What-if type of questions. This approach encourages students to articulate lengthier answers that exhibit deeper reasoning instead of short answers, which may lead to shallow knowledge. A continuous multi-turn dialogue between the tutor and the student takes place throughout the session. The natural language processing components of the tutor are based on Latent semantic analysis (LSA) (Landauer, Foltz & Laham, 1998).
Why2-AutoTutor (Jackson et al., 2004) and Why2-Atlas (Jordan et al., 2006) were developed to facilitate the comparison between the LSA approach used in AutoTutor and the symbolic approach used in Atlas-Andes. Both systems expect the student to write a short essay on a qualitative physics problem. Then the systems analyse the essay and use it as a basis for a tutorial dialogue addressing any misconceptions identified. The systems also provide a critique of the essay and help the student to rewrite it. An empirical study was conducted to test the hypothesis that even when the content is equivalent, students who engage in more interactive forms of instruction learn more. To test this hypothesis, the performance of students who received human tutoring after reading a short text was compared with students who interacted with Why2-Atlas and Why2-AutoTutor. The results revealed that the students learn equally well in all three conditions when the content is at an appropriate level for the student.

The presented systems use different approaches to support tutorial dialogues, depending on the domain and the target student group. CIRCSIM-Tutor, Geometry Explanation Tutor and Atlas-Andes facilitate learning in domains of cardiovascular physiology, Mathematics and Physics. All these domains have well-defined domain theories and the instructional tasks presented to the student are also well-defined (Mitrovic & Weerasinghe, 2009). Therefore, these tutors coach problem-solving in well-defined tasks and use dialogues as a way of remediating student errors. Even though AutoTutor, Why2-AutoTutor and Why2-Atlas support learning in different domains, instructional task supported is eliciting natural language explanations to learn the declarative knowledge. This set of tutors use dialogues as the main method of teaching conceptual knowledge. However a general framework has not been developed to facilitate tutorial dialogues in both well-defined and ill-defined instructional tasks. Our long-term goal is to develop a general model for supporting dialogues via menu-based input that will provide adaptive support to learners across domains. Since we previously incorporated dialogues (Weerasinghe & Mitrovic, 2006a) into a database design tutor (Suraweera & Mitrovic, 2002, 2004), the initial work on this project started with the same tutor. We discuss this domain in the following section.

DATABASE DESIGN: AN ILL-DEFINED TASK

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. Even though there is a well-defined domain theory (Mitrovic & Weerasinghe, 2009), 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 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.

Fig. 1. Interface of the enhanced version of EER-Tutor used in the study.
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 can problem solving 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. Even though there can be a family of solutions that are all equally good, the teacher often has a good pedagogical reason for preferring one solution over the others. For example, in database design there are situations where a single higher-order relationship (which has at least three participating entities) or multiple binary relationships (which has two participating entities) can be used. In such a situation, the teacher may prefer one of the solutions. Therefore, it is possible to nominate one “ideal” solution without compromising the quality of the whole ITS, as long as the ITS is capable of identifying other alternative solutions students may come up with as correct.

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.

**HOW DO HUMAN TUTORS SUPPORT ERROR-REMEDICATION?**

As the first step towards designing a general model to support tutorial dialogues, we conducted an observational study (Weerasinghe & Mitrovic, 2006b), focusing on how students interacted with EER-Tutor (Zakharov, Mitrovic & Ohlsson, 2005), while getting additional help from a human tutor through a chat interface.
The Experimental Set up

The study was conducted in August 2005 at the University of Canterbury, and involved student volunteers enrolled in an introductory database course and professional tutors. In this discussion we refer to professional tutors as tutors, and to EER-Tutor as the system or the ITS hereinafter. All the tutors had several years of tutoring experience providing assistance on request to students in labs and/or teaching small groups. As EER-Tutor provides a problem-solving environment and complements classroom instruction, the study was scheduled after the relevant learning material was taught in the classroom.

The version of EER-Tutor used in the study 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 very similar to that between the student and the ITS. Therefore, tutors could observe only the students’ interactions with the ITS. Participants interacted with the system in one room and the tutors observed their interactions in another room. In the dialogue-enhanced EER-Tutor, dialogues will be used as an additional component to assist problem solving and they would not replace the typical feedback that the system currently provides. Therefore, the participants received both the typical feedback by the EER-Tutor and the additional feedback by the tutors.

We asked the tutors to guide the students towards solutions using appropriate methods like asking questions and so on. However, they were not given any specific instructions on providing assistance. Student participants were not told that a human tutor 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 participants sat an online pre-test, and then interacted with EER-Tutor. The participants were free to end the session whenever they wanted. All learner interactions were recorded. Although initially we wanted the participants to sit a post-test immediately after the study, it was not possible due to another evaluation study that 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. Two tests of comparable difficulty were scheduled to be used as pre and post-tests. Each test contained 7 questions. Each correct answer scored one mark and the total score was given out of seven.

We analysed the recordings to investigate how students were prompted by different tutors and which interactions triggered these prompts. In the second phase, whenever possible, we discussed the recordings with the tutors to clarify how they decided on the timing and the level of feedback provided through the chat interface. This experimental set up varies from previous studies of tutorial dialogue in a number of ways. First, the human tutor in this study provided support in addition to the feedback given by the system. The tutors also responded to learners’ questions. This contrasts with those studies of Chi, Siler, Jeong, Yamauchi, and Hausmann (2001) and Graesser, Person and Magliano (1995), in which the tutor is expected to lead the dialogue through a series of questions. Second, the learner interacted both with the system and the tutor. Although Merrill, Reiser, Raney, and Trafton (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, & Pain, 2002). However this setup is similar to the study conducted by Rose and her colleagues (Rose, Fumer, Aleven, Robinson, & Wu, 2006).

**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%). The posttest was available online after a pre-specified date. Only two students 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 problems attempted was 11 (sd = 5) and all participants completed all attempted problems. We discuss observations in two different categories: (1) type of feedback provided in the interventions and (2) timing of interventions.

**Type of Feedback Provided**

We analysed the interactions between the tutors and the students in order to identify episodes, each pertaining to a single topic (Chi et al., 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), motivating and praising the student, suggesting trying a more challenging problem, completing the session or helping with technical problems (e.g., web browser suddenly closing). The number of episodes initiated by a tutor per session ranged between 4 and 20. 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 included 4 episodes, the first intervention occurred only in the 19th problem (the student completed 22 problems).

We are mainly interested in 37 episodes that 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.

An example is presented in Figure 2, which occurred while a student was solving the problem in Figure 1. In this dialogue, the student was able to identify and repair the misconception he had with weak entities and total participation.

The highest number of dialogues in a session was 7, while the lowest was 2. As can be expected, the highest number of dialogues occurred in the longest session.

From the collected data, we identified three techniques used by human tutors. Tutors were rephrasing feedback from the ITS, providing problem-independent explanations and stating the tutor’s observations before starting to discuss the problem state. The tutors rephrased feedback to enable the student to understand their own mistake. 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 our model is to repeat the same kind of prompting, it is difficult to ascertain whether it will have the same effect (Lepper, Woolverton, Mumme, & Gurtner, 1993). The second technique was to discuss the current problem state and then provide a problem-independent explanation. Figure 2 represents an example. These explanations provided an opportunity for the student to repair his mental model of the domain and generated further conversation. The third technique 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?”

![Fig. 2. Example of a self-explanation episode.](image)

As the knowledge base in EER-Tutor is represented as a set of constraints, the errors were recorded as violations of constraints (Zakharov, Mitrovic & Ohlsson, 2005). We analysed how frequently constraints were violated after related errors were discussed using dialogues, 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 see whether tutorial dialogues are useful, the participants’ logs were analysed, and each problem-state after a tutor intervention in which a constraint was relevant was identified. These identified occasions are referred to as occasions of application. 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). At the first occasion, all participants had some relevant constraints, whereas only two participants had a constraint relevant at n = 17. This indicates that the number of constraints that were relevant decreases as occasion number increases.

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 of entities. For this student, the tutor provided a problem-independent explanation on total participation to help him identify and repair the misconception he had with weak entities and total participation (Figure 2). 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 that were discussed by the tutors.
Twenty-eight different constraints were discussed in the dialogues. Three students did not violate any constraints in subsequent occasions after the tutor interventions. These students were tutored by three different tutors who followed techniques like rephrasing feedback, providing problem-independent explanations and stating tutor’s observations at the beginning of the discussion. This suggests that all these techniques have been effective in helping the students learn domain concepts.

**Timing of Interventions**

The original version of EER-TUTOR provides feedback on demand, that is, 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 colours. In such a situation, immediate feedback would not have been welcomed by the student as he may have felt the intervention to be 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.

**Prototype of the Model**

The goal of our project is to provide adaptive dialogue support, which means that the model should determine when to initiate dialogues, what to discuss and how to obtain explanations from learners. We designed the model on the basis of our previous work (Weerasinghe & Mitrovic, 2006a) and the findings from the study of human tutors presented in the previous section. The model consists of three parts: an error hierarchy, tutorial dialogues and rules for adapting them. The error hierarchy categorizes all the error types in a domain. At the lowest level an error type is associated with one or
more violated constraints, which form leaves of the hierarchy. The error types are then grouped into higher level categories. Remediation is facilitated through tutorial dialogues, one of which is developed for each error type. When there are multiple errors in a student solution, the hierarchy is traversed to select the error most suitable for discussion and the corresponding dialogue is then initiated. Finally, the adaptation rules are used to individualize the dialogues to suit the student’s knowledge and reasoning skills by controlling their timing and the exact content. In response to the generated dialogue, learners are able to provide answers by selecting the correct option from a list. Each component is now described in detail.

Error Hierarchy

In previous work, we developed a hierarchy of errors students make in the Entity-Relationship (ER) domain (Weerasinghe & Mitrovic, 2006b), which categorizes errors as being syntactic or semantic in nature. A high-level view of the hierarchy is given in Figure 4, showing the top three levels only. In constraint-based ITSs, violated constraints indicate errors in a student solution: violated constraints for each type of error, therefore, form the leaves of the hierarchy. Syntax errors are generally simple, each requiring only one feedback message to be given to the student rather than initiating a dialogue; for that reason, every syntactic error tends to correspond to a particular constraint being violated. Twelve constraints are associated with syntax errors for the ER domain; for example, constraint 7 is violated when two entities are directly connected to each other, and forms a single error type. In contrast, the hierarchy for semantic errors is deeper because constraints are often related by some high-level concept. For semantic errors, error types are typically further divided into sub-errors.

<table>
<thead>
<tr>
<th>ALL ERRORS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Syntax errors</td>
</tr>
<tr>
<td>Semantic errors</td>
</tr>
<tr>
<td>Using an incorrect construct type</td>
</tr>
<tr>
<td>Extra constructs</td>
</tr>
<tr>
<td>Missing constructs</td>
</tr>
<tr>
<td>Connecting an attribute to an incorrect construct</td>
</tr>
<tr>
<td>Errors dealing with cardinalities and participation</td>
</tr>
</tbody>
</table>

Fig. 4. Overall view of the error hierarchy.

The original error hierarchy (Weerasinghe & Mitrovic, 2006b) was developed for the ER modeling domain, so we were interested whether it could be reused in other domains. With that goal, we tried to fit the errors from a different domain, logical database design, into this structure. This domain involves mapping high-level, conceptual ER schemas to relational schemas using the 7-step mapping algorithm (Elmasri & Navathe, 2007). The task is well-defined due to the deterministic algorithm used. However, both domains (ER modeling and ER-to-relational mapping) involve mapping as the major activity. Due to this similarity, we decided to explore additional domains of different nature, such as data normalization and fraction addition. Data normalization is the process of refining a relational database in order to ensure that all relations are of high quality (Elmasri & Navathe, 2007).

During this investigation, we identified situations when it was not enough to present a single feedback message for some violated syntax constraints: a dialogue was required. Therefore, we
modified the structure of the error hierarchy to divide all error types into two main categories: Basic Syntax Errors and Errors dealing with the main problem-solving activity (Figure 5). Under the new node Basic Syntax errors, we included simple syntax errors such as checking whether the student has filled the required fields, the components used to fill the required fields are valid and so on. Hence it is sufficient to discuss such errors using a single message. The other category requires a dialogue to be conducted. We also decided to change the names of some nodes in the highest level to make the hierarchy more general. For instance, Extra constructs was renamed to be Extra solution components, Missing constructs to Missing solution components and so on.

Another observation was that different clusters of syntax errors were needed in these newly examined domains, as opposed to the flat structure of the syntax error sub-hierarchy in Figure 4. For example, in data normalization, several constraints check whether the student is using valid attribute names in different steps of the algorithm, all of which can be categorized into a single node (Check Validity of attributes), specified as a child node of Basic Syntax Errors.

Another refinement required was to make the two domain-specific nodes Connecting an attribute to an incorrect construct and Errors dealing with cardinalities and participation more general so that the overall hierarchy can be used across domains. As these two nodes deal with associations between solution components, it is appropriate to have a new node Associations (Figure 5). This new node has different domain-specific child nodes.

A final refinement was made based on an observation from the study of human tutors reported in the previous section: some students seemed to be reacting to feedback on errors by making suggested changes without reflecting on other modifications that also needed to be carried out. As an illustration, in ER modeling if a regular entity with a key attribute is changed to a weak entity, a partial key should be specified instead of the key attribute. This may have led to frustration due to the number of attempts that the student had to go through to arrive at the correct solution. A new node Failure to complete related changes was added to the existing error hierarchy, which reminds the student to check whether other changes are necessary (Figure 5). In such cases, the student will be prompted to reflect on other related changes before submitting the solution.

<table>
<thead>
<tr>
<th>ALL ERRORS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Basic syntax errors</td>
</tr>
<tr>
<td>Errors dealing with the main problem solving activity</td>
</tr>
<tr>
<td>Using an incorrect solution component type</td>
</tr>
<tr>
<td>Extra solution components</td>
</tr>
<tr>
<td>Missing solution components</td>
</tr>
<tr>
<td>Associations</td>
</tr>
<tr>
<td>Failure to complete related changes</td>
</tr>
</tbody>
</table>

Fig. 5. Overall view of the refined error hierarchy.
Figure 5 only shows the three highest levels of the refined error hierarchy, as the lower levels deal with domain-specific concepts. We also tested the refined error hierarchy in the domain of fraction addition. Even though this domain is very simple, it is quite different from the domains that we have investigated previously. All the error types in the fractions domain could be specified using the error hierarchy.

Using an incorrect solution component type

**Using a completely different type of solution component**

- Using an entity to represent another type of solution component
  - Using an entity to represent a relationship
    - (27 or 28)
  - Using an entity to represent an attribute
    - (202 or 203 or 204 or 205 or 206 or 202-1 or 205-1 or 206-1)
- Using another type of solution component to represent an entity
  - Using a relationship to represent an entity
    - (13_1 or 14_1)
  - Using an attribute to represent an entity
    - (13_2 or 14_2)
    - 65-5
    - 65-6

**Other representations**

- Using a relationship to represent an attribute
  - 207 or 208 or 209 or 210 or 211_A or 207_1 or 210-1 or 211-B
- Using an attribute to represent a relationship
  - 27_2 or 28-2

**Using a different variation of the correct solution component**

**Entity**

- Using a regular entity to represent a weak entity
  - (14)
  - (67-2)
- Using a weak entity to represent a regular entity
  - (13)

**Relationship**

- Using a regular relationship to represent an identifying relationship
  - (28_1)
- Using an identifying relationship to represent a regular relationship
  - (27-1)

**Attribute**

- Using a different type of attribute
  - (54 or 55 or 56 or 57 or 58 or 59 or 54_1 or 57_1 or 58_1 or 59-1)
  - 65-2

Fig. 6. Detailed view of the node *Using an incorrect solution component type* for Database Design.

Figure 6 represents the detailed view of the node *Using an incorrect solution component type* for database design. This node is further divided into two categories: *Using a completely different type of solution component* and *Using a different variation of the correct solution component*. When a relationship in the ideal solution (IS) is represented as an entity in the student’s solution (SS), this
error is categorized under the first sub node (Using a completely different solution component). For instance, constraint 27 is violated when an entity is used to model a regular relationship in the IS. On the other hand, constraint 28 is violated when an entity is used to represent an identifying relationship. The second sub node (Using a different variation of the correct solution component) deals with more subtle errors such as a weak entity being represented as a regular entity in the SS or a regular relationship being represented as an identifying relationship. The content of the error hierarchy (the number of levels and the nodes) depends on the domain. Figure 7 represents the detailed view of the node Using an incorrect solution component type for the fraction addition domain. As can be expected, the subtree for this node for fraction addition is much simpler than that of the ER modeling. The node Associations forms the largest tree with 8 levels for the ER modeling whereas the node Extra solution components forms the smallest with 3 levels.

<table>
<thead>
<tr>
<th>Using an incorrect solution component type</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Using an incorrect least common denominator (LCD)</strong></td>
</tr>
<tr>
<td>- Higher multiple of the correct LCD</td>
</tr>
<tr>
<td>- 10-gse</td>
</tr>
<tr>
<td>- Incorrect LCD</td>
</tr>
<tr>
<td>- 0-gse</td>
</tr>
<tr>
<td><strong>Using an incorrect denominator</strong></td>
</tr>
<tr>
<td>- Fraction1</td>
</tr>
<tr>
<td>- 5-gse</td>
</tr>
<tr>
<td>- Fraction 2</td>
</tr>
<tr>
<td>- 4-gse</td>
</tr>
<tr>
<td>- Sum</td>
</tr>
<tr>
<td>- 2-gse</td>
</tr>
<tr>
<td><strong>Using an incorrect whole number</strong></td>
</tr>
<tr>
<td>- 7-gse</td>
</tr>
</tbody>
</table>

Fig. 7. Detailed view of the refined node Using an incorrect solution component type for fraction addition.

The common feature in all these tasks is that the syntactic and semantic accuracy of a solution can be completely evaluated by the components of the solution and its associations. However, there are exceptions. For instance, in reading and comprehension, where learners are asked to answer questions based on a paragraph, the accuracy of an answer cannot be evaluated by checking only for the correct words according to the grammatical rules. We also need to understand the implicit semantic meaning of the sentence. Therefore, our error hierarchy is not useful in such cases. In summary, we have been able to use this hierarchy in four different types of tasks: thus we believe it would be sufficiently general to be used for different types of instructional tasks only when the solution can be completely evaluated by the components of the solution and its associations.

If the error hierarchy is developed for an existing constraint base, then the hierarchy can also be used to understand/verify how comprehensive the constraint base is. When the hierarchy was developed for the fraction addition domain, several missing constraints were identified. For example, two new constraints were needed when a student has correctly converted the denominator but failed to do the same for the numerator of the given fractions fraction1 and fraction2. When we were exploring the data normalization domain, several constraints that needed to be made more specific were
identified. For example, when simplifying functional dependencies (fds) one of the initial checks was to compare the number of fds of the IS with that of the SS. This constraint can be violated either by having too many or too few fds in the SS. As a result, this constraint can be categorized under both *Missing solution components* or *Extra solution components*, leading to two different dialogues. Therefore, this constraint needs to be made more specific so that it deals with only one type of error.

**Tutorial Dialogues**

In our model, error remediation is facilitated through tutorial dialogues. A dialogue is designed for each error type (i.e., each leaf node in the hierarchy). As the domain model of constraint-based tutors is represented as a set of constraints, violations of constraints indicate errors in a student solution. In other words, an error in a student solution indicates the domain concept that the student has difficulty with. Each dialogue therefore discusses the domain concept associated with that error.

Each dialogue consists of four stages. In the first stage, the dialogue informs the student about the concept that s/he is having difficulty with, and then asks for the justification of the student’s action. The purpose of the second stage is to assist the student in understanding why the performed action is incorrect. The third stage prompts the student to specify how to correct the mistake. In the fourth stage, the student can review the domain concept learned.

Different stages of the dialogue address different aspects of a domain concept. The first stage facilitates the understanding of the concept whereas the final one provides an opportunity to reinforce it. The second stage helps the students to understand how s/he has applied the concept incorrectly to the current context. The third stage provides an opportunity to apply the domain concept correctly.

Dialogues are developed as tree structures and authored manually. The top-level prompt is problem-independent. It consists of an observation (“You seem to be having some difficulty with regular entities” - *EERTutor1* in Figure 8) and an opportunity to discuss the reasoning behind the student’s problem solving step. Starting the discussion by stating the domain concept that the student has difficulty with is one of the strategies used by the observational study described previously. Starting the discussion with such an observation is appropriate as the top-level prompt is used only after the student has made the same error three times repeatedly. If the student’s response is incorrect, the ITS guides the student through a remediation recipe. Remediation recipes can take different forms. In simple cases, a brief description of the domain concept is presented to help the student acquire missing information (*EERTutor2* in Figure 8). In other cases, a more specific prompt is used to guide the student to provide the correct answer. After going through the remediation recipe, the ITS will move to the next stage. A correct response at each stage will activate the next level prompt of the dialogue.

A hypothetical dialogue for database design is given in Figure 8,\(^1\) shown as a linear sequence instead of the full tree due to space restrictions. It is initiated when a student uses a regular entity in a situation when an attribute should be used. This error is categorized under the node *Using an incorrect solution component type*. Initially, the system identifies the domain concept the student has problems with, and asks the student to explain it (*EERTutor1*). If the student cannot answer (*Student1*), s/he will be given a brief description that provides a further opportunity to understand that domain concept (*EERTutor2*). The dialogues consist of simple questions (*EERTutor1*), fill-in-the-blank (*EERTutor7*), or true-false questions, to motivate the student to explain. As all dialogues discuss errors (*EERTutor2*),

\(^1\) The complete dialogue is available at [http://www.ictg.canterbury.ac.nz/dialogue-enhanced-tutors](http://www.ictg.canterbury.ac.nz/dialogue-enhanced-tutors).
students are given opportunities to reflect on their problem solving procedure, which is another important meta-cognitive skill.

Rules for Adapting Dialogues

Adaptation rules enable individualization of the dialogues by using the student model to decide on the timing, selection and entry point into the dialogue. Table 1 presents the current set of rules, which are based on the observations from the study reported in Section 3. Some of the rules are discussed here.

As these rules do not depend on domain-specific details to individualise dialogues, the rules can be used across domains.

Rule 1 (dealing with timing of dialogues after a period of inactivity) checks whether the student made any attempts at the current problem, and has been inactive for a specified period of time (such as 10 minutes, the time period we observed in the study). If both conditions are satisfied, then student’s
solution is evaluated even though it has not been submitted yet, and a dialogue is initiated to focus on the error most suitable for discussion if multiple errors exist.

Rule 3 addresses the critical issue of selecting a dialogue. Dialogue selection is very important because if it is not effective, it might be difficult for students to systematically develop a comprehensive mental model of the domain. Dialogue selection depends on the student solution and the error hierarchy. When multiple errors exist in a student solution, we would like to focus on the error that the student is most likely to make in subsequent attempts. This is based on the probability of violating a constraint in the last five submissions of the student solution. The probabilities of violating individual constraints are then combined to calculate the probability of making an error corresponding to higher-level nodes in the error hierarchy. These probabilities are updated each time a student solution is evaluated. Rule 3 finds the error type (e.g., node N1, which is a non-leaf node in the hierarchy) that a student is most likely to make in future attempts. As the nodes are ordered from basic domain principles to more complicated ones, the dialogue associated with the left-most leaf node for N1 is chosen as the most suitable dialogue for a set of violated constraints. However, there is one exception: if a sub node which comes under Missing solution components has the highest probability of making a mistake, that indicates the SS has a considerable number of missing solution components. As it is more effective from a pedagogical point of view to discuss what the student has already

MathTutor1: You seem to be having some difficulty calculating the least common denominator (LCD). Let’s look at LCD in detail. Can you tell me why we need the LCD?
Student1: I don’t know
OK! Please remember that LCD is used to convert denominators of the fractions to be added to a common denominator. Now tell me why having 12 as the LCD of 6 and 8 is incorrect?
Student2: I’m not sure
MathTutor3: Let’s see…Check carefully whether there are any common factors for 6 and 8? Please press <Continue> when you are ready.
{After <Continue> is pressed} Please tell me whether you manage to find any common factors for 6 and 8?
Student3: Yes
MathTutor4: What are the common factors?
Student4: It’s 2.
MathTutor5: Good job! Can you also check whether 8 is a higher multiple of 6?
Student5: No
MathTutor6: Good, then do you know why having 12 as LCD is incorrect?
Student6: because you can’t convert denominator of fraction 2 (i.e. 8) to 12
MathTutor7: Great! Now tell me the correct LCD in this case?
Student7: I need more help
MathTutor8: Ok. When you have two numbers with common factors but one is not the higher multiple of the other, step1 is to find the prime factors for each integer. Try to find the prime factors for each integer? Please press <Continue> when you are ready.
{After <Continue> is pressed} pls. tell me the prime factors that you have found?
Student8: 6 = 2 *3; 8 = 2^3
MathTutor9: Great! Now the next step is to find the LCD. It is the product of multiplying the highest power in each prime factor category together. Can you tell me the correct LCD?
Student9: 24
MathTutor10: Well done! Now use the correct word to fill in the blanks so that you’ll remember what you learnt.
…………….. is used to convert the …………. of the fractions to be added to a common …………
Student10: LCD, denominators, denominator
MathTutor11: Excellent, Go ahead and do the changes. Submit the solution when you’re done.

Fig. 9. A hypothetical dialogue in the Fraction Addition domain.
attempted rather than the ones he has not completed, we will be looking for the node with the second highest probability. The errors that are categorized under *Missing solution components* are considered only when the errors that come under other nodes have been corrected.

Dialogues can be more effective if they are adapted to the student’s domain knowledge and reasoning skills. We observed that the tutors tend to discuss the domain concepts relevant for an error if it was done repeatedly. They also tend to state their observations before discussing the domain concept (e.g., “You seem to be having difficulty with regular entities (*EERTutor1* in Figure 3)). Rule 4, which deals with customizing the entry point to the dialogue, is activated when the same error is made in the last $n$ attempts. In that case, a dialogue corresponding to the mistake is initiated, but the dialogue starts from the problem-independent question (*EERTutor1* in Figure 3). If the error was made less than $n$ attempts, then the dialogue will start from the error within the current context (*EERTutor3* in Figure 3).

Even though the dialogues were intended to facilitate error remediation, they might be felt as a burden to some natural explainers. Hence, we do not expect the students to go through the entire dialogue when their solution is erroneous. Rule 5 monitors when they have answered the dialogue prompt correctly indicating that they may have understood their mistake. In such a case, the students are encouraged to resume problem solving.

The short-term student model in constraint-based tutors consists of lists of satisfied/violated constraints for the student’s solution, while the long-term model records constraint histories. We will extend both types of student models to additionally record details of the student’s reasoning skills in terms of types of errors made (i.e., rules that were initiated) and the level of prompting the student needed to correct errors for each domain-level constraint. This additional information can be used to identify whether a student has improved his/her reasoning skills (ideally they need less prompting to understand their mistakes), and the dialogues that are most effective.

Even though this model was developed for constraint-based tutors, it can be used in any ITS providing a problem-solving environment. In such an ITS, a student solution is evaluated and feedback is provided on the errors regardless of the mechanism/methodology used for diagnosis. Therefore, the error hierarchy (the first component of the model) could be developed using the error types of that domain. Tutorial dialogues (the second component of the model) need to be written for each type of error based on the tutorial structure that was discussed in section 2.2. The third component of the model, rules for adapting dialogues, are domain independent, hence it can be used across domains.

**Preliminary Study of the Proposed Model**

We conducted an experiment with the ERM-Tutor (Milik, Marshall, & Mitrovic, 2006) in April 2006 at the University of Canterbury, which involved student volunteers and experienced tutors. Two types of feedback were provided: typical feedback provided by the system, and dialogues initiated by the model. The study was conducted as a Wizard-of-Oz study, in which the first author of this paper simulated the actions of the model. This additional assistance was given through a chat interface (see Figure 10), and will be referred to as interventions hereinafter. The first author used the dialogues from a written script. However, it was not always possible to use the scripted prompts in the later stages of dialogues because the student answers were not constrained as in the proposed system. (In
the proposed system, the students will be given a list of possible answers from which the correct one can be selected.

Table 1
Adaptation Rules

<table>
<thead>
<tr>
<th>Rule Identifier</th>
<th>Adaptation Rule</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>IF SS has not been changed for the last 10 minutes and has not been evaluated at all, THEN evaluate SS and start the dialogue.</td>
</tr>
<tr>
<td>2</td>
<td>IF SS has not being changed for the last 5 minutes and has been evaluated previously THEN evaluate SS and initiate dialogue</td>
</tr>
</tbody>
</table>
| 3               | Selecting the dialogue  
                    IF SS is incorrect  
                    THEN select the left-most dialogue with the highest probability of making a mistake and display the message (about the observation that the student is having problems with a specific sub area). |
| 4               | Selecting the entry point of the chosen dialogue  
                    IF the same mistake is repeated 3 times (this includes errors with the pre-test) within the same session,  
                    THEN start the explanation from problem-independent prompt of the chosen dialogue  
                    ELSE start the explanation from problem-dependent prompt of the chosen dialogue |
| 5               | Moving to the next level of the dialogue  
                    IF the student response for the current prompt is incorrect  
                    THEN move to the next level prompt  
                    ELSE encourage student to resume problem solving |
| 6               | IF the student has changed a problem without completing it (for well-defined tasks, if the problem is changed after completing an intermediate step)  
                    THEN evaluate SS, inform the student that he has not completed the problem, and ask the student if s/he wants to change the problem or wants help. |
| 7               | IF the student has changed 3 problems without completing them consecutively (for well-defined tasks, if the problem is changed after completing an intermediate step)  
                    THEN identify whether they stop when they face the difficulty in a certain sub-area, and state the observation |

Participants interacted with ERM-Tutor in one room, while the first author observed from another room. The participants could initiate interventions through the chat interface or the More Help button. Participants were expected to use the system for at least an hour. However, students themselves decided when to end the session. At the end of the session, they filled out a questionnaire. The first phase of the study involved analyzing the logs to investigate the effectiveness of the dialogues. In the second phase, human tutors were asked to judge the appropriateness of interventions by observing recorded sessions. A time line indicating all the interventions was provided to the judges, who
indicated whether he/she agrees with the timing and the content of interventions. In the case of a disagreement, the judge was requested to provide justifications.

Ten students and five tutors (acting as judges) participated in the study. All students were enrolled in an introductory database course at the University of Canterbury. The judges were the lecturer and the tutors involved in teaching this course. The average session duration was 59 minutes (sd=15.3). In some cases, the ERM-tutor indicated that the student solution was incorrect even though it was actually correct, due to bugs in the system. Such instances were excluded from the analysis. The average number of problems attempted was 11 (sd = 4.6), with 8.4 (sd = 5.2) problems completed on average.

From the logs, we identified 65 episodes. An episode is considered to be a multi-turn dialogue pertaining to a single topic (Chi, 2000). In addition to facilitating remediation, some episodes focused on helping with the interface (such as moving to the next step), completing the session or helping with technical problems (e.g., web browser not being able to display the page). The number of episodes per session ranged from 1 to 13, with the mean of 6.5 (sd = 4.3). We are mainly interested in 31 episodes that facilitated error remediation. Six of these episodes contained a single utterance each, initiated by the wizard. For instance, a tutor utterance that helped a student to understand that multi-valued attributes are not mapped in the first step of the algorithm was “think about the color attribute.” The longest episode consisted of 11 utterances, 6 of which were provided by the model (i.e., the wizard). An example is given in Figure 11. In this dialogue, the student is incorrectly applying step 4 (mapping 1:N relationships) to the identifying relationship, while that step should only be applied to regular relationship types. The correct action here is simply to move to the next step. In this situation, the model aims to assist the student to understand that this step is not necessary.

![Fig. 10. The screenshot of the version of ERM-Tutor used in the preliminary study.](image)

![Fig. 11. A dialogue from the study.](image)
In order to investigate whether the dialogues were effective, we analyzed how frequently an error occurred after being discussed in each episode. As the knowledge base in ERM-Tutor is represented as a set of constraints, the errors were recorded as violations of constraints. Thus we analyzed how frequently the constraints that were discussed in the dialogues were violated subsequently. However, some students were able to correct the mistake themselves just before the episode started. In another situation, a student indicated that he did not require any assistance (even after a period of inactivity) when prompted. Also, some violated constraints were not consistent with the actual mistake in the student solution due to a coding problem. It was not possible to specify a constraint for such episodes. Thus, those episodes were excluded for the analysis. The remaining 15 (48.3%) episodes were included in this analysis.

The selected dialogues involved only seven participants. (The dialogues with the other three students were among the ones excluded.) These dialogues were associated with seven different domain-level constraints. Figure 12 illustrates the learning curve for these constraints. The curve is not smooth due to the small sample size (i.e., this analysis involved only seven constraints discussed in 15 episodes with 7 participants), but it does suggest that the probability of subsequently violating a constraint discussed in an episode decreases with occasion number. This indicates that the students seem to learn domain concepts discussed in the episodes, that is, that the dialogues based on the proposed model did not prevent learning. In order to evaluate whether the dialogues facilitated by this model actually enhance learning, we need to compare the performance with a control group of students who interact with the system without the dialogues.

![Fig. 12. Learning from dialogues.](image)

In phase 2, five judges analyzed the interventions and indicated whether they agreed with their timing and content. At the beginning, we informed the judges that the goal of the study was to develop a model to facilitate remediation through dialogues while interacting with a tutoring system. The judges were asked to comment on the appropriateness of the timing and the content of interventions provided through the chat interface. Each judge analyzed two sessions. Due to time constraints, it was not possible to have every session investigated by two judges.

All the episodes were categorized by the rule that initiated them. Five rules were relevant in this study. Rule 1 was violated only once, and rule 2 was violated three times. The judges agreed with the timing and content of these interventions.

Rule 4 was relevant in 21 episodes and had the highest number of disagreements. Judges disagreed in 7 (33.3%) occasions. Timing was the issue in six of these instances and judges wanted to intervene earlier. The judges disagreed with the content in three situations. For instance, a judge
suggested using “Is there a regular 1:N relationship to map in this problem?” instead of the first prompt in Figure 10.

One of the issues to be addressed is how to effectively facilitate remediation when nothing needs to be done in a particular step (Figure 10). According to our model, the initial prompt was “What do you need to do when you're mapping a 1:N relationship?” which may imply that the student needs to perform an action, even if there is nothing to do. It would be better if the prompt were changed to “Do you know which type of relationship needs to be mapped in this step?”. The new prompt discusses a domain concept so it still conforms to the dialogue structure discussed in section 3.2.

Some judges preferred earlier interventions than those suggested by the model. The model waits for 3 repeated mistakes before initiating a dialogue. However, it might be effective to intervene after two repeated mistakes, because it is easier to assess what the student is trying to achieve in this particular domain. As the result, the number of times a mistake needs to be repeated may be domain-dependent. Rule 1, which checks for a period of inactivity may also be domain-dependant; however, there was no disagreement on these.

CONCLUSIONS AND FUTURE WORK

This research focuses on developing a model to support dialogues for error-remediation in both ill- and well-defined tasks. A prototype model was developed based on the findings of a preliminary study using EER-Tutor, an ITS that teaches the ill-defined task of conceptual database design. This paper focuses on the study that used the prototype model with ERM-Tutor, an ITS developed for the ER-to-relational mapping domain. In addition to the feedback provided by the system, dialogues were facilitated through a chat interface. The interventions through the chat interface were based on the model.

Analysis of user logs indicates that students did learn the domain concepts discussed in the dialogues episodes. Human tutors who were asked to analyze the episodes mostly agreed with the interventions generated by the model.

The findings from the reported study are being used to refine the model. The next step is to incorporate the model into both EER-Tutor and ERM-Tutor. The enhanced systems will then be evaluated in authentic classroom environments. The objective of these evaluations is to investigate whether the adaptive dialogues supported by the model are more effective in facilitating deep learning than the non-adaptive dialogues, in which two students (with different domain knowledge and reasoning skills) receive the same dialogue when they make the same types of errors in their solutions.

ACKNOWLEDGEMENTS

This project was supported by the University of Canterbury doctoral scholarship to the first author. We gratefully acknowledge the support of all members of ICTG.
REFERENCES


