Facilitating deep learning through self-explanation in an open-ended domain

Amali Weerasinghe* and Antonija Mitrovic
Intelligent Computer Tutoring Group, Department of Computer Science, University of Canterbury, Private Bag 4800, Christchurch, New Zealand

Abstract. Self-explanation has been used successfully in teaching Mathematics and Physics to facilitate deep learning. We are interested in investigating whether self-explanation can be used in an open-ended, ill-structured domain. For this purpose, we enhanced KERMIT, an intelligent tutoring system that teaches conceptual database design. The resulting system, KERMIT-SE, supports self-explanation by engaging students in tutorial dialogues when their solutions are erroneous. The results of an evaluation study indicate that self-explanation leads to improved performance in both conceptual and procedural knowledge.

1. Introduction

Many Intelligent Tutoring Systems (ITS) have shown significant learning gains for students [23] particularly in the domain of Mathematics [14], Physics [15] and Computer Science [17,22]. However, some empirical studies indicate that students acquire shallow knowledge even in the most effective systems [1]. As a result, students have difficulties in transferring knowledge to novel situations, even though they obtain passing grades on tests. Researchers are therefore interested in finding methods that overcome the shallow learning problem. Self-explanation 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” [9], and has been shown to facilitate the acquisition of deep knowledge [10]. Since explaining instructional material to oneself facilitates the integration of new information into existing knowledge, self-explanation can be viewed as a knowledge construction activity [9,30]. According to the Chi’s study [9] self-explanation facilitates the process of repairing one’s own mental model of the domain, promoting reflection, a meta-cognitive skill lacked by many students [19]. Self-explanation studies also show that the majority of students do not spontaneously self-explain. However, students begin self-explaining more frequently when they are guided [7] or even simply prompted to do so [11]. These results suggest that it may be greatly beneficial to integrate computer-based support to problem solving through self-explanation.

KERMIT (Knowledge-based Entity Relationship Modelling Intelligent Tutor) [27,28] is a problem-solving environment that supports students learning database (DB) modelling. In this paper, we describe how we enhanced KERMIT to support self-explanation. Section 2 describes related work. KERMIT is briefly introduced in Section 3 and KERMIT-SE, its enhancement that facilitates self-explanation, is presented in the next section. The results of the evaluation study are presented in Section 5. The conclusions and directions for future research are given in the final section.

2. Related work

Self-explanation has been used in several ITSs to enhance learning. These systems are discussed in this section.

2.1. Self-explanation with SE-Couch

SE-Couch supports self-explanation by prompting students to explain solved examples [13]. It is im-
plemented within ANDES [15], a tutoring system that teaches Newtonian Physics. The first level of scaffolding in the SE-Coach’s interface is provided by a masking mechanism that covers different parts of the example with grey boxes, each corresponding to a unit of information. When the student moves the mouse over a box, it disappears, revealing the content underneath. The second level of scaffolding produces specific prompts to self-explain. Students are prompted to self-explain only when the tutor decides it is beneficial. To determine when to intervene, SE-Coach relies on a probabilistic student model, that monitors how well the student understands the domain by capturing both implicit self-explanations and self-explanations generated through the interface [12]. The results of the empirical evaluation of SE-Coach reveal that the structured scaffolding of self-explanation can be more beneficial in the early learning stages.

ANDES has been further enhanced by incorporating ATLAS [29], a module to conduct self-explanation in a natural language. 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. A limited study (with ten participants) revealed that the students interacting with ATLAS learnt significantly more than students who interacted with ANDES [16]. However, larger studies are needed to obtain more reliable results.

2.2. PACT Geometry Tutor

Aleven and Koedinger [1] investigated self-explanation in the PACT Geometry Tutor. The students in the experimental group were expected to provide correct explanations for solution steps by citing definitions and theorems used. A glossary of knowledge in the form of definitions and theorems was provided in order to help students to explain the solution steps. The study revealed that explaining reasoning steps results in improved problem solving skills. Students who explained solution steps attempted significantly fewer problems than their peers who provided only the answers, although both groups spent the same amount of time with the tutor. However, there was evidence that self-explainers required fewer problems to reach the tutor’s mastery level criterion.

PACT Geometry Tutor has been further enhanced to facilitate self-explanation through natural language dialogue [3]. In Geometry Explanation Tutor, students explain in natural language, and the system evaluates their explanations and provides feedback. The system contains a hierarchy of 149 explanation categories [4], which is a library of common explanations, including incorrect/incomplete ones. The system matches the student’s explanation to those in the library, and generates feedback, which helps the student to improve his/her 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 [2]. The results revealed that students who explained in natural language did not learn better than those who self-explained through menu selection.

2.3. AUTOTUTOR

AUTOTUTOR [17] 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 tutor attempts to understand the student input by segmenting contributions into speech acts and matching them with the expectations. Latent semantic analysis (LSA) [18] is used to compute these matches.

2.4. NORMIT

NORMIT [20] is designed for university students and provides a problem-solving environment for data normalisation. Students are expected to self-explain while solving problems. In contrast to other ITSs that support self-explanation, NORMIT requires an explanation when an action is performed for the first time. For the subsequent actions of the same type, explanation is required only if the action is performed incorrectly. This approach was utilised because it tends to reduce the strain on more able students by not asking them to provide the same explanation every time an action is performed correctly and also to enable
the system to provide enough situations for students to develop and improve their self-explanation skills. Similar to the PACT Geometry Tutor and SE-Coach, NORMIT supports self-explanation by prompting the student to explain by selecting one of the offered options. For instance, one of the tasks in database normalisation is to specify the candidate keys of a relation. When a student specifies a candidate key for the given problem, the student is asked to explain why the specified attribute(s) make a candidate key, if that is the first time he/she is specifying candidate keys. If the student selects an incorrect explanation, NORMIT will then ask for another explanation. In contrast to the first question, which was problem-specific, the second question is generic. The student will be asked to define a candidate key by selecting one of the options given. A study has been conducted with second year students at the University of Canterbury to assess the effects of self-explanation support in the domain of database normalisation. Due to the small number of participants sound conclusions cannot be made, but the results do indicate that problem solving skills and declarative knowledge increases with self-explanation [20].

2.5. Discussion

These systems use different approaches to facilitate self-explanation, depending on the domain and the target student group. Problem solving activities in these domains are well structured, and the types of self-explanations expected from students can be clearly defined. For example, in Mathematics and Physics, students are expected to explain the theorems that they have used. In computer literacy, the students are expected to explain the definitions, meanings of terms and how a certain task is being carried out. However, it is challenging to incorporate self-explanation in the domain of database design, as it is an open-ended task: there is an outcome defined in abstract terms, but there is no procedure to find that outcome. It is not sufficient to ask the students to explain the concepts of database modelling, as the database design skills can only be developed through extensive practice.

3. KERMIT: A knowledge-based ER modelling tutor

KERMIT is an ITS aimed at the university-level students learning conceptual database design. The architecture of the system is illustrated in Fig. 1. For a detailed discussion of the system, see [26]; here we present some of its basic features. KERMIT is a problem-solving environment in which students practise database design using the Entity Relationship (ER) data model. The system is intended to complement 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. KERMIT contains a set of problems and the ideal solutions to them, but has no problem solver. In order to check the student’s solution, KERMIT compares it to the correct solution, using domain knowledge represented in the form of more than 90 constraints. It uses Constraint-Based Modelling [21, 24] to model the domain and student’s knowledge.

Constraint based modelling [21], based on the theory of learning from performance errors [25], is a student modelling approach that reduces the complexity of the task. CBM focuses on correct knowledge rather than on describing the student’s knowledge as in model tracing [6]. The key assumption in CBM is that the diagnostic information is in the problem state at which the student arrives and not in the sequence of his/her actions. This assumption is supported by the fact that a correct solution cannot exist for a problem that traverses a problem state which violates fundamental ideas or concepts of the domain. Since the space of incorrect knowledge is much greater than correct knowledge, knowledge about a domain is represented as a set of state constraints. A state constraint is an ordered pair \((C_r, C_s)\), where \(C_r\) is the relevance condition and \(C_s\) is the satisfaction condition. The relevance condition identifies the states in which the constraint is relevant and the satisfaction condition identifies a subset of rel-
evant states in which the constraint is satisfied. For instance, violation of constraint 11 (described below) initiated the first feedback message (i.e. Check whether all the entities are necessary. Check whether some of your regular entities should be represented using some other type of construct), which appears on the bottom left window of Fig. 2. This is caused by the error CHAPrER should not be modelled as a regular entity. Constraint 11 is implemented as follows:

Relevance Condition: For each regular entity in the student solution,
Satisfaction condition: there should be a matching regular entity in the ideal solution.

In this example, the relevance condition is satisfied by the regular entity CHAPTER. However, it violates the satisfaction condition because CHAPTER is modelled as a weak entity in the ideal solution.

CBM does not require a runnable expert module to generate pedagogical actions. This is an important advantage as it is very difficult to design an expert module for many domains. CBM is also computationally very simple as student modelling is reduced to pattern matching. During the evaluation of a problem state all relevance patterns are matched against the problem state. In the case where the problem state matches the relevance pattern, it is then checked against the satisfaction condition. If the satisfaction condition is not met, then the constraint is violated, which indicates errors. Moreover, CBM also does not need extensive bug libraries, which model students’ misconceptions about the domain.

Students have several ways of selecting problems in KERMIT. They may work their way through a series of problems, arranged according to their complexity. The other option is a system-selected problem, when the pedagogical module selects a problem for the student on the basis of his/her student model.

The interface 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 this window the student draws ER diagrams. Figure 2
represents the interface of KERMIT-SE, which is very similar to that of the original of KERMIT. The only difference is that bottom window of KERMIT has only one section in original KERMIT.

The 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, Correct, simply indicates whether the submitted solution is correct or incorrect. The error flag indicates the type of construct (e.g. entity, relationship, etc.) that contains the error. For example, when the solution in Fig. 2 is submitted, error flag provides the message Check your entities, that’s where you have some problems. This is associated with the error CHAPTER being modelled as a regular entity instead of a weak entity. Hint and Detailed Hint offer a feedback message generated from the first violated constraint. In the case of solution in Fig. 2, hint provides a general message Check whether all the regular entities are necessary. Check whether some of your regular entities should be represented using some other type of construct. On the other hand, detailed hint provides a more specific message CHAPTER 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 bottom right hand in Fig. 2). The ER schema of the complete solution is displayed at the final level (solution level).

Initially, when the student begins to work on a problem, the feedback level is set to the correct level. As a result, the first time a solution is submitted, 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. In the case when there are several violated constraints, and the level of feedback is different from ‘all errors’, the system will generate the feedback on the first violated constraint. The constraints are ordered in the knowledge base by the human teacher, and that order determines the order in which feedback would be given.

KERMIT maintains two kinds of student models: short-term and long-term ones. Short-term models are generated by matching student solutions to the knowledge base and the ideal solutions. The student modeler iterates through each constraint in the constraint base, evaluating each of them individually. For each constraint, the modeler initially checks whether the current problem state satisfies its relevance condition. If that is the case, the satisfaction component of the constraint is also verified against the current problem state. Violating the satisfaction condition of a relevant constraint signals an error in the student’s solution.

The short-term student model consists of the relevance and violation details of each constraint, discovered during the evaluation of the problem state. The pedagogical module uses the short-term student model to generate feedback to the student. The long-term student model is implemented as an overlay model, and stores each constraint’s history. It records information on how often the constraint was relevant for the student’s solution and how often it was satisfied or violated. The pedagogical module uses these data to select new problems for the student.

4. Design and implementation

All systems that facilitate self-explanation prompt students to explain most of the problem-solving steps, requiring students to point out the definitions/theorems used. We believe this approach puts too much burden on able students. Therefore, our tutor prompts for self-explanation only when the student violates a constraint, which indicates missing/erroneous knowledge, or a slip. The tutor is thus able to customise self-explanation based on the student model so that the knowledge construction is facilitated for students who have misconceptions or gaps in their knowledge without disrupting others [8]. For a detailed discussion on how KERMIT was enhanced to support self-explanation see [31]. Some of the important details are discussed here.

Since a student can make several errors in a single submission, the pedagogical module (PM) needs to decide on which error to initiate self-explanation. The constraints are ordered according to traditional ER modelling procedure, starting with modelling entities
first, then relationships and finally the attributes. This
ordering alone is not sufficient to select an error for
self-explanation, as most solutions violate more than
one constraint. At the same time, semantic constraints
are not specific enough to guide self-explanation ef-
fectively. For instance, the constraint *A construct that
should be a regular entity has been represented by an-
other type* is violated when a regular entity is repre-
sented either as a simple attribute or a weak entity. Dif-
ferent approaches are required in these two cases. Self-
explanation in the first case should help the student to
clarify the definitions of entities and attributes so that
the student understands when to use them. In the lat-
ter case, the student should understand the differences
between regular and weak entities, which will enable
the error to be corrected and the correct design deci-
sions made in future. Also, PM should enable students
to build a more comprehensive mental model of the
domain by giving them an opportunity to learn basic
concepts before complicated ones.

4.1. The error hierarchy

We analysed the constraint base of KERMIT and
students’ answers for an assignment in an introductory
database course to develop a list of errors that can oc-
cur in the domain of database modelling. Students’ re-
sponses to pre- and post-tests of a previous evaluation
study conducted for KERMIT [28] were also analysed
to make the list comprehensive. The list of errors com-
plied was then used to develop a hierarchy of errors that
can be used to introduce a high-level structure of con-
straints to the constraint base of KERMIT. The overall
view of the hierarchy of errors is shown in Fig. 3. Er-
rors in this domain can be divided into two categories:
(i) Syntax errors (ii) Semantic errors. Violated con-
straints (which are specified by the constraint numbers
in Figs 3 and 4) for each type of error are represented
as the leaves of the hierarchy. Constraints for the nodes
in Fig. 3 are given in separate lines to indicate that each
constraint deals with a specific type of error.

Our experience in teaching database design in a tra-
ditional classroom setting indicates that it is easier for
students to understand and correct syntax errors than
semantic ones. Therefore, the error hierarchy focuses
on dealing with the syntax errors before the semantic
ones. This is achieved by placing the node *Syntax er-
rors* before the node *Semantic errors* (Fig. 3). Nodes in
this hierarchy are ordered from the basic domain prin-
ciples to more complicated ones so that a student can be
guided systematically towards building and repairing
his/her mental model of the domain. For example, con-
straint 1 is violated when more than one ER construct
is used to model a phrase of the problem statement
whereas constraint 21 is violated when a relationship
does not have at least two participating entities. Thus,
constraint 1 deals with a relatively simpler error than
21.

Selecting an error to initiate self-explanation is cru-
cial for enhancing learning because if error selection
is not effective, it might be difficult for students to
systematically develop a comprehensive mental model
of the domain. Error selection depends on the stu-
dent solution and the error hierarchy. When the hierar-
chy is traversed from left-to-right, top-to-bottom, the
simplest error type for a given student solution can be
found.

Semantic errors are divided into five categories,
which are further divided into sub categories as indi-
cated in Fig. 4. For instance, the node *Using an in-
correct construct type is divided into two child nodes:
(i) Using a completely different type of construct, (ii)
Using a different variation of the correct construct.* In
ER modelling design decisions need to be made using a
two-step process. Initially, the student needs to decide
whether a certain phrase in the problem statement can
be modelled as an entity, relationship or an attribute.
In the second step, he/she needs to decide whether it is
regular or weak in the case of an entity; regular or
identifying in that of a relationship etc. The errors as-
associated with the first step are classified under the node
*Using a completely different type of construct.* The
node *Using a different variation of the correct construct*
deals with the errors of the second step.

For example, the CHAPTER regular entity in Fig. 2
violates constraints 11 and 14 because it should be mod-
elled as a weak entity. Constraint 11 deals with extra
regular entities in the solution which should be mod-
elled using some other type of construct. Constraint
14 deals with weak entities which are represented as
some other type of construct in the solution. This situ-
uation is described by the node *Using a regular entity to
represent a weak entity* which is a child node of *Using
a different variation of the correct construct* (Fig. 4).
The two constraints (i.e. 11 and 14 ) are combined with
‘and’ to state the requirement that both constraints need
to be violated by the same constraint. i.e. CHAPTER is
an extra regular entity (constraint 11) and needs to be
modelled as a weak entity (constraint 14).

4.2. Designing the dialogues

A dialogue is developed for each error in the error
hierarchy. For simple errors, the dialogue is limited
ALL ERRORS

Syntax errors
1. There should be only one construct that models a word of the problem
2. A database must have some entities
3. Both ends of each connector should be connected to components of the diagram
4. A connector attached to an attribute should be a single line with no cardinality
5. Attributes cannot be directly connected to more than one other attribute, or twice to the same attribute
6. Attributes cannot be connected to an entity as well as another construct.
7. Entities cannot be directly connected to each other.
8. All attributes should be connected to another attribute.
9. Relationships cannot be directly connected to a construct.
10. Relationships cannot be directly connected to another entity.
11. All attributes should be connected to another attribute.
12. All entity and relationship names must be unique.

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. 3. Overall view of the refined error hierarchy.

to a detailed feedback message, as it is sufficient to explain the error. For other errors, a tutorial dialogue that consists of a series of questions is necessary to guide students to self-explain both domain concepts and solution steps. Therefore, dialogues can be classified into two categories depending on the number of nodes in the dialogue: (i) Dialogues with a single node (ii) Dialogues with several nodes.

Single-node dialogues handle errors associated with simple connections. There are 12 such dialogues. An example of a single node dialogue is You have connected an entity A to entity B directly. Entities cannot be directly connected to each other.

On the other hand, multi-node dialogues consist of a series of questions beginning with general questions and moving to more specific ones. Most dialogues consist of four main stages:

(i) Informs the student about the incorrect modelling decision and asked to specify the reason (general rule) for his/her decision;
(ii) Prompts the student to understand why his/her decision is incorrect;
(iii) Prompt the student to specify the correct modelling decision;
(iv) Prompt the student to review the domain concept learned through a review question.

For example, consider the problem statement shown in Fig. 2. The student’s solution contains several errors: the CHAPTER entity should be represented as a weak entity and its partial key is missing, TEXTBOOK is missing its key attribute, CONTAINS should be an identifying relationship and its participation is wrong.

When this student solution is submitted, the student modeller evaluates it against the constraint base and identifies the violated constraints. The pedagogical module then searches for the first tutorial dialogue for the same violated constraints. Since the error types and the corresponding dialogues are ordered according to the complexity of the domain concepts, the pedagogical module selects the dialogue by traversing the hierarchy in a top-to-bottom, left-to-right manner selecting some or all violated constraints. In this situation, the pedagogical module selects the dialogue corresponding to the error CHAPTER represented as a regular entity instead of as a weak entity to start with. Figure 5 presents a sample dialogue that occurs between the student and the tutor. The dialogue is represented internally in a tree from, in which each node contains a question and a list of possible answers to it. The tutorial dialogue ideally starts from the root node and finishes at one of the leaf nodes. The specific path taken depends on the correctness of the answers selected by the student for each question. In other words, if the student selects a
correct answer for a question, the pedagogical module then selects the question associated with the left child of the current node. If the answer is incorrect, the pedagogical module selects the question associated with the right child of the current node.

In the first stage of the dialogue, the tutor informs the student about the incorrect modelling decision and prompts him/her to specify the reason (general rule) for his/her decision (tutor-1). Because self-explanation is initiated by making the student aware of an incorrect action and prompting them to interpret their decision, the student is given an opportunity to reflect on their problem-solving procedure. A list of possible answers is provided. Sometimes it might be difficult for students to select the correct explanation even from a list of possible explanations provided by the tutor. In order to provide an opportunity for students to request self-explanation, the tutor provides phrases like I'm not sure and I need more help. If the student selects one of these phrases or an incorrect explanation, there is evidence that he/she has difficulties in providing the correct explanation. In such situations, the student is given more help by explaining the domain concept or by guiding him/her through simpler questions.

For example, if the student fails to interpret his/her modelling decision correctly, he/she is guided to simple, straightforward questions (tutor-2). If he/she still fails to understand the mistake through simple prompts provided by the tutor, the correct answer is provided as a last resort. In the second stage of the dialogue, the student is guided to explain why his/her modelling decision is incorrect.

If it is difficult for the student to understand why his/her decision is incorrect, the tutor guides the student to search for more information in the problem statement. For instance, when the student fails to understand why modeling CHPAPTEr as a regular entity is incorrect, he/she is asked to search for any key attributes for
Tutor-1: You’ve modelled CHAPTER as a regular entity. Can you tell me the general rule to decide whether something is a regular entity?

Student-1: I don’t know

Tutor-2: Ok! Let’s try a simple thing. What is used to uniquely identify instances of a regular entity?

Student-2: The key attribute

Tutor-3: Great! Now tell me why modeling CHAPTER as a regular is incorrect?

Student-3: I’m not sure

Tutor-4: Let me help you a bit. Check carefully whether any of the attributes of CHAPTER can uniquely identify the instances of CHAPTER? Please press Continue if you want more clarifications.

(When Continue button is pressed) Can you tell me the attribute that you’ve identified as a key attribute for CHAPTER?

Student-4: There’s no attribute that can be a key for CHAPTER.

Tutor-5: Great! Now tell me the best way to model Chapter?

Student-5: A weak entity

Tutor-6: 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.

Student-6: key attribute ,regular

Tutor-7: Excellent. Go ahead and do the changes. Submit the solution when you’re done.

Fig. 5. A sample tutorial dialogue.

the entity type CHAPTER (tutor-4). As students need to be actively engaged in finding out whether there are any key attributes for CHAPTER, the possible answers are not given for the question immediately. Instead, the list of possible answers is displayed only when the student informs the tutor that he/she is ready to continue with the self-explanation process. By delaying the display of possible answers when students are expected to search for information in the problem statement, they are encouraged to actively engage in understanding why their modelling decision is incorrect. Then he/she is given the opportunity to understand that CHAPTER cannot be a regular entity because it does not have a key attribute.

In the third stage, he/she is asked to specify the correct modelling decision (tutor-5). Again, if he/she fails to specify the correct the modelling decision, the tutor provides the correct answer with an explanation. In the final stage of the dialogue, the student is given a review question to provide another opportunity to understand the concept discussed (tutor-6). Various types of review questions are used to increase the student’s motivation. The review questions can be simple questions, fill-in-the-blanks, or true-false questions. If the student has difficulty in answering the review question correctly, the correct answer is provided.

There is evidence that immediate feedback on students’ responses has the potential to enhance learning in an ITS [4]. Therefore, it is important to provide feedback on the accuracy of the self-explanations provided by students during problem-solving. In addition to providing feedback on students’ responses, phrases such as Well done, Great and Good job are used in the dialogues to encourage students to provide more self-explanations. Furthermore, when a student fails to provide the correct self-explanation he/she is encouraged by using phrases such as Let me help you a bit and Think again instead of phrases like Your answer is incorrect. Please try again.

Even though the dialogues contain a series of questions, students can correct a mistake as soon as they realise it without going through the entire dialogue. Therefore, knowledge construction is facilitated for students who have misconceptions or gaps in their knowledge without disrupting others [8].

5. Evaluation

As evaluation is fundamental in all stages of developing an ITS, we conducted two evaluation studies. A pilot study was carried out to investigate the effectiveness of the self-explanation support provided by KERMIT-SE. For details of this study, please see [31]. The system was then refined based on the results of the pilot study.
An evaluation study was conducted in July 2002 with students enrolled in an introductory database course at the University of Canterbury. We wanted to test the hypothesis that self-explanation facilitates acquisition of both procedural and conceptual knowledge. The experimental group used KERMIT-SE, while the control group used a cut down version of KERMIT. Both groups received the list of all errors for their solutions, and could ask for the ideal solution. Even though there were five different levels of feedback in the original KERMIT, only Detailed Hint was available to the control group to make it comparable with the experimental group. As the name suggests, Detailed Hint provides a detailed feedback message on a single error in a student solution.

The experiment was carried out during normal lab hours over the duration of two weeks, and consisted of four phases: pre-testing, system interaction, post-testing and subjective system assessment. The pre- and post-tests consisted of two questions each, of equal difficulty. The first question required students to design an ER model for the given requirements, whereas the second question required them to explain the design decisions for the given ER model. The tests were rotated between successive sessions to minimise any effect resulting from variation in test difficulty. Ideally, each student was expected to spend a total of 4 hours to complete the four stages of the study.

We developed two different versions of the questionnaire: control group was given 12, and the experimental group 15 questions. Initially, students were asked about their previous experience in database modelling, and their impressions of the system and its interface. The experimental group was additionally asked about their perception of self-explanation support. Some of the questions asked were how easy/difficult the dialogues were to understand, and how useful the dialogues were for understanding errors. Students answered on a Likert scale with five responses ranging from very poor (1) to very good (5), and were also allowed to give free-form responses.

5.1. Pre-and post-test performance

Table 1 represents mean pre-and post-test scores and the standard deviation (in parenthesis) for the two groups. The sizes of control and experimental group differ, as they depended on how many students turned out to corresponding lab sessions. The mean score on the pre-test for all students was 72.18% (SD = 18.72%). The difference in pre-test scores for the two groups is insignificant, confirming that the groups are comparable. Examining the logs of the session, we see that only 19 students in the experimental group have gone through at least one dialogue, (they had control over that via the More Help button). We are interested in these students, as the rest of the experimental group has not self-explained, and we summarize statistics for those students separately (column self-explainers in Table 1). The mean score on the pre-test for self-explainers is significantly higher than the mean for the control group. Therefore, we cannot directly compare the control group to self-explainers. However, the other students in the experimental group, who have not gone through any of the dialogues (column non self-explainers in Table 1), are comparable to the self-explainers, as there is no significant difference for these two groups of students on the pre-test.

The participation was voluntary, and not all students completed the study. We report the number of post-tests in Table 1. The difference between the post-test scores is not significant. The control group improved significantly on the post-test ($p < 0.01$). However, these students had lowest existing knowledge (lowest pre-test score) and therefore had more room for improvement. Even though the self-explainers did better in the post-test, the improvement is not significant. As the result, the difference in gain scores for the experimental and the control groups was significant ($t = 1.33, p = 0.09$).

The difference between the post-test scores for the self-explainers and non self-explainers is not significant. Although both groups improved after interacting with the allocated system, the improvements of both groups were not statistically significant.
The self-explainers improved only slightly on question 1 but the difference in the post-test performance between the two groups on this question is not significant. The improvement on both questions for self-explainers was not statistically significant.

The difference in pre-test scores for the conceptual question is not significant confirming that the groups are comparable. However, self-explainers performed significantly better on the conceptual question in the post-test ($t = 2.74, p < 0.01$) than their peers.

It is interesting to note that the performance of the non self-explainers decreased after using the cut down version of KERMIT which did not provide any self-explanation support. As the test used in the study were rotated between successive sessions any effect resulting from variation in test difficulty is minimised. Therefore, these results suggest that the self-explanation support provided by KERMIT-SE has significantly contributed towards improving the conceptual knowledge of the students.

5.2. Learning

There is no significant difference between the problem solving time for the control and experimental groups (Table 4). However, self-explainers spent significantly more time on problem solving ($t = 5.01, p < 0.001$) than non self-explainers. This might be due to the self-explanation dialogues, as students needed time to answer the questions. However, the self-explainers also attempted and solved significantly more problems than the rest of the experimental group. Therefore, self-explanation does support problem-solving.

The student logs were also used to analyse the mastery of constraints. The domain knowledge in KERMIT-SE is represented by constraints [26]. If these constraints represent psychologically appropriate units of knowledge, then learning should follow a smooth
Table 4
Mean system interaction details

<table>
<thead>
<tr>
<th></th>
<th>Control</th>
<th>Experimental</th>
<th>Self-explainers</th>
<th>Non-self-explainers</th>
</tr>
</thead>
<tbody>
<tr>
<td>No of students</td>
<td>72</td>
<td>53</td>
<td>19</td>
<td>34</td>
</tr>
<tr>
<td>Problem solving time (min.)</td>
<td>105:21 (44:19)</td>
<td>98:37 (49:34)</td>
<td>133:21 (30:44)</td>
<td>79:13 (47:41)</td>
</tr>
<tr>
<td>No. of attempted problems</td>
<td>7.08 (2.69)</td>
<td>6.34 (3.22)</td>
<td>8.21 (2.42)</td>
<td>5.29 (3.17)</td>
</tr>
<tr>
<td>No. of completed problems</td>
<td>5.25 (2.43)</td>
<td>4.62 (2.63)</td>
<td>6.36 (2.31)</td>
<td>3.65 (2.29)</td>
</tr>
<tr>
<td>No. of post-tests</td>
<td>59</td>
<td>35</td>
<td>18</td>
<td>17</td>
</tr>
<tr>
<td>Pre-test</td>
<td>70.98 (18.47)</td>
<td>75.61 (17.33)</td>
<td>79.32 (13.16)</td>
<td>73.17 (20.47)</td>
</tr>
<tr>
<td>Post-test</td>
<td>79.94 (17.75)</td>
<td>78.11 (14.35)</td>
<td>79.76 (12.22)</td>
<td>77.37 (16.76)</td>
</tr>
</tbody>
</table>

Fig. 6. Probability of violating constraints for the self-explainers and non self-explainers.

The power curve for the non self-explainers has a better fit than for the self-explainers. This might be due to the fact that existing knowledge of non self-explainers is lower than that of the self-explainers and as the result, they have more room for improvement.

5.3. Self-explanation performance

The self-explainers on average went through 6.95 dialogues, ranging from 1 to 21. On average, students completed 78.25% of the dialogues, with an average of 57.61% of correct responses to the questions in the dialogues. To test the second part of our hypothesis (self-explanation results in improved conceptual knowledge), we analysed the student answers to the first question of a chosen dialogue, which prompts students to explain domain concepts. Figure 7 illustrates the correctness of students’ explanations. The probabilities of correct answers on the first and subsequent occasions were averaged over all error types and over all students. The fit to the power curve is very good, indicating that students do learn by explaining.

We also analysed students’ answers to the second question in a chosen dialogue, which expects students to provide problem-specific explanations. Figure 8 il-
Fig. 7. Performance on the first question in the dialogues.

Fig. 8. Probability of providing correct problem-specific explanation.

illustrates the correctness of students’ problem-specific explanations. On the first occasion, 15 students provided a problem-specific explanation, whereas only a single student did so at \( n = 6 \). To reduce impact of a single failure in the probability of providing the problem-specific explanations correctly, we selected an arbitrary cut-off point of \( n = 3 \). A good fit to the power curve indicates that students learn by providing problem-specific explanations. The \( R^2 \) values for the conceptual question and the problem-specific question were 0.79 and 0.61, respectively, revealing a better fit for the conceptual question. This suggests students learn better by explaining domain concepts rather than providing problem-specific explanations.

5.4. Subjective analysis

A summary of the responses to the user questionnaires is given in Table 5. Both groups required approximately the same time to learn the interface. This was expected, as there was not much difference between the two interfaces. The experimental group found it significantly easier to use the interface (\( t = 2.17, p = 0.01 \)). We were encouraged to see that students in the experimental group felt that the interface was easier to use even though KERMIT-SE has more features than the version used by the control group. The difference in mean responses on the amount learnt and the enjoyment were not significant.

Control group students found feedback to be useful, even though they only had limited feedback. Although student on average find questions in the dialogue difficult to understand, they rated the usefulness of the dialogues higher than the rank of feedback by the control group.

The user responses to the questionnaires were compared for self-explainers and non self-explainers. As expected, there was no significant difference between the time needed by the two groups to learn the interface. When asked to rate the amount learnt on a scale 1 to
Table 5
Means for the questionnaire responses

<table>
<thead>
<tr>
<th></th>
<th>1 to 5 on Likert scale</th>
<th>Control group</th>
<th>Experimental group</th>
<th>Self-explainers</th>
<th>Non self-explainers</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time to learn the interface</td>
<td>N/A</td>
<td>14.27 (13.99)</td>
<td>11.82 (11.64)</td>
<td>11.67 (13.39)</td>
<td>12.00 (9.59)</td>
</tr>
<tr>
<td>Amount learnt</td>
<td>Nothing to Very much</td>
<td>3.35 (0.82)</td>
<td>3.33 (0.78)</td>
<td>3.22 (0.73)</td>
<td>3.40 (0.83)</td>
</tr>
<tr>
<td>Enjoyment</td>
<td>Very much to Not at all</td>
<td>2.84 (0.98)</td>
<td>2.72 (1.14)</td>
<td>3.12 (1.11)</td>
<td>2.27 (1.03)</td>
</tr>
<tr>
<td>Ease of using interface</td>
<td>Very much to Not at all</td>
<td>2.78 (0.98)</td>
<td>2.34 (0.87)</td>
<td>2.47 (0.94)</td>
<td>2.27 (0.79)</td>
</tr>
<tr>
<td>Usefulness of feedback/dialogues</td>
<td>Not at all to Very much</td>
<td>3.45 (1.09)</td>
<td>3.50 (0.82)</td>
<td>3.41 (0.94)</td>
<td>3.61 (0.65)</td>
</tr>
<tr>
<td>Ease of understanding the questions in the dialogues</td>
<td>Very easy to Not at all</td>
<td>N/A</td>
<td>2.65 (0.95)</td>
<td>2.50 (1.21)</td>
<td>2.86 (0.53)</td>
</tr>
</tbody>
</table>

5 (Nothing to Very much), non self-explainers claimed that they learnt more (the ranking of non self-explainers is higher by 0.18). Though the non self-explainers’ claim seems surprising without them going through at least one dialogue, the difference was not statistically significant. Non self-explainers’ claim of learning more suggests that students think problem solving using general hint messages helped them to acquire a robust knowledge of the domain concepts. This is consistent with the control groups’ claim that they have learnt more than the experimental group even though they received limited feedback (Table 5).

When asked to rate the ease of using the interface on a scale 1 to 5 (Very much to Not at all), the non self-explainers found the interface easier to use than the self-explainers. This can be expected, because non self-explainers did not interact with some of the interface controls, as they did not use the self-explanation support. However, the difference in the mean rating was found to be statistically insignificant ($t = 0.66$, $p = 0.25$).

Even though non self-explainers did not go through any of the dialogues, they gave a ranking indicating the ease in understanding the questions in the dialogues and the usefulness of the dialogues to understand mistakes. As they gave rankings for two aspects of the self-explanation support that they did not use, it is not meaningful to compare the mean ratings. However, self-explainers gave a mean rating of 3.41 for the usefulness of dialogues to understand mistakes.

68% of the control group students indicated that they would recommend KERMIT to other students while the percentage of experimental group students who had the same opinion was lower (60%). 60% of the control group students preferred more details in the feedback whereas, only 17% indicated that they do not want more details. As it is not effective to consider the response given by the non self-explainers in the experimental group about the effects of the dialogues, we consider only the response of the self-explainers. When asked how many questions they had to go through on average to realise a mistake in their student solution, the majority (66%) of the self-explainers indicated that 2 to 3 questions were needed. Only 5% of the self-explainers indicated that they needed to go through the entire dialogue. This suggests that being able to resume problem solving in the middle of a dialogue might have increased the usability of KERMIT-SE. Moreover, 56% of the self-explainers felt that the questions in the dialogues assisted them in understanding the domain concepts.

5.5. Discussion

Results of the evaluation study indicate that students’, performance improved by self-explaining although not significantly. Furthermore, results reveal that students learn by explaining domain concepts and providing problem-specific explanations. As students have the freedom to initiate the self-explanation dialogues (via the More Help button), the experimental group cannot be considered a true experimental group. This factor can be considered an experimental flaw. However, if students were forced to go through self-explanation dialogues, it might have had a detrimental effect on their motivation to use the system for a considerable period of time. Students’ use of the dialogues suggest that they were willing to self-explain while problem solving. As many students did not go through the entire dialogue before resuming problem-solving, giving the freedom to resume problem-solving at any point within the self-explanation dialogue might have improved their motivation to use the system. Students who self-explained attempted and solved significantly more problems, thus the self-explanation process seems to support problem solving.

It was encouraging to note that many students from both control and experimental groups requested access to use the system after the study to practice for exams. The systems were made available for students until the end of that semester.
6. Conclusions and future work

Self-explanation is an effective learning strategy to facilitate deep learning. This research focuses on incorporating self-explanation into a tutor that teaches the open-ended task of ER modelling. KERMIT-SE supports self-explanation by engaging students in tutorial dialogues about errors they make. Students are asked problem-specific and general questions, and can select answers from menus.

An evaluation study was conducted to investigate whether guided self-explanation would facilitate acquisition of both procedural and conceptual knowledge in the domain of database modelling. The experiment involved second-year University students enrolled in an introductory database course. The experimental group used KERMIT-SE, while the control group used a cut down version of KERMIT. Both groups received the list of all errors for their solutions, and could ask for the ideal solution. Both groups received similar feedback. In addition, the experimental group had the freedom to initiate the self-explanation process. The pre- and post-tests consisted of two questions each, of equal difficulty. The first question required students to design an ER model for the given requirements, whereas the second question required them to explain the design decisions for the given ER model. The first question was used to measure their problem-solving capabilities and the second question their self-explanation abilities.

The results showed that performance of both experimental and control groups improved. Furthermore, a significant improvement was achieved only by the control group. However, their pre-knowledge (pre-test score) was lower than that of the experimental group so they had more room for improvement. Analysis of the participants’ logs indicated that only 35% of the students in the experimental group have gone through at least one dialogue. We compared the performance of these students (self-explainers) with the other students who have not self-explained (non self-explainers). Although both groups improved after interacting with KERMIT-SE, the improvement was not statistically significant.

We measured the pre and post-test performance on the procedural and the conceptual questions separately for self-explainers and non self-explainers. The results indicated that the improvement in procedural knowledge of non self-explainers is better than their peers although it is not significant. This suggests that guiding students towards the ideal solution through general feedback messages provided by the cut-down version of KERMIT helps them to significantly improve their procedural knowledge. However, self-explainers performed significantly better on the conceptual question in the post-test. It is interesting to note that the performance of the non self-explainers decreased after using the cut down version of KERMIT which did not provide any self-explanation support. Therefore, these results suggest that the self-explanation support provided by KERMIT-SE has significantly contributed towards improving conceptual knowledge.

We wanted to investigate whether the system is more beneficial to less able students. We divided the self-explainers and the non self-explainers into more and less able groups based on their performance in the pre-test. The more able group comprised of those students who scored above the mean score for all participants and the remaining students formed the less able group. The performance of less able students in both groups improved significantly whereas there was a decrease in the performance of more able students. Therefore, the system is more beneficial to less able students.

Self-explanation has been successfully used to facilitate deep learning in several well-structured domains. This research is the first attempt to facilitate self-explanation in an open-ended domain like database domain. The evaluation study revealed that self-explanation improved both procedural and conceptual knowledge.

The authors’ expertise and teaching experience in this area has been extremely valuable in implementing the self-explanation process. We spent about a week reviewing the constraint base and students’ answers for an assignment to design the error hierarchy. Even though the time needed to design each dialogue varied, it took approximately 1 month to design all dialogues. Although the time required to integrate self-explanation support into an existing ITS depends on the complexity of the domain and the underlying methodology of the existing system, it might take about 1–2 months to design the self-explanation process.

There are a number of future avenues that can be explored to further improve the effectiveness of KERMIT-SE. Currently, the system provides the same dialogue for the same error for two different students who could possibly have different self-explanation skills. We believe that the effectiveness of the dialogues could be greatly enhanced by incorporating adaptive self-explanation support for students. Currently the short-term student model of KERMIT consists of lists of satisfied and violated constraints for the student’s last submission, while the long-term model records the history.
of each constraint (how often a constraint has been relevant, and how often it has been satisfied/violated). The current student model describes the knowledge level of each student using constraints. As a result, these models can be used to understand how each student acquires domain knowledge, which is represented as a set of constraints. In order to facilitate adaptive self-explanation skill, the current student model needs to be extended to record students’ self-explanation behaviour. The enhanced student model can also be used to provide additional support in acquiring domain knowledge to students who have difficulty in understanding domain concepts.

Providing the opportunity to self-explain in a natural language might potentially enhance the quality of learning as natural language is more expressive than a restricted interface of an ITS used to obtain self-explanations from a student. In addition, it also provides an opportunity to express partial knowledge using natural language. However, a recent evaluation study revealed that students who explained in natural language did not learn better than those who self-explained through menu selection [2]. Incorporating natural language capabilities to KERMIT-SE might be an avenue worth exploring to investigate whether learning can be significantly enhanced using natural language to self-explain in an open-ended domain like database design.

Acknowledgements

We thank Pramuditha Suraweera and Danita Hartley for their help in implementing KERMIT-SE. This research was made possible by the NZODA postgraduate scholarship awarded to the first author.

References


