The Effectiveness of Open Student Modelling on Learning

November 6, 2001

Danita Hartley Antonija Mitrovic[1]

[1]Supervisor

Abstract

Intelligent tutoring systems (ITSs) are computer tutors that provide individualised instruction by maintaining models of their students. Traditionally, these models have been hidden from the student. However, recent work in the area has suggested educational benefits in exposing the student model. This approach, known as open student modelling, allows the student to inspect their model thereby facilitating reflection, which is known to enhance the learning process. To date, few evaluations have been conducted to determine the effects that open student models have on learning. This is the focus of our work. In particular, we are interested in whether even a simple open model can have a positive effect on learning. For this purpose, we have exposed the student model in an existing ITS and have performed an initial evaluation study. Subjective results from the study are encouraging, although a more extensive study is needed to draw reliable objective conclusions. This report presents the extended ITS, followed by the results from the evaluation study.

Contents

Chapter 1

Introduction

Intelligent tutoring systems (ITSs) are an active research area. The goal is to develop intelligent computer tutors that provide learners with individualised learning environments comparable to one-to-one human tutoring. It is well known that a one-to-one learning environment is more effective than a traditional classroom/lecture style environment because the teacher can adapt to the individual, thereby coaching them in such a way that they will learn more effectively. The high student:teacher ratio [Mit01a] makes it more difficult for students to obtain individual support. ITSs are a potential solution to this problem, although current research is far from reaching this goal.

Development of an ITS is a complicated task as individual student needs differ. Consequently such a system must be capable of dynamically adapting to and monitoring each student. To model individuals, these systems usually maintain models of each student representing at least their performance in the domain. The central problem is how to obtain/maintain a realistic model of each student. If the task is to model a student's knowledge completely and accurately, the process is bound to be intractable. Attempting to model what a student knows correctly is not sufficient, yet attempting to model what a student knows incorrectly is too complicated because of the huge search spaces involved [Mit01a]. However, a student model may be useful even if it is not complete and accurate. This is supported by findings that human teachers are highly effective in what they do, and yet they use only very loose models of their students [MMSM01]. There have been many approaches to student modelling including overlay models, perturbation models and, more recently, constraint based models [Mit01a, MMM02, MMSM01]. Each of these ideas is concerned with the representation of student knowledge.

Traditional approaches to student modelling make the student model invisible to the student. However, recent work in the area has suggested educational benefits in exposing the student model. This approach to modelling, known as *open* student modelling, allows the student to inspect their model, and facilitates reflection.

Student modelling may be classified as *passive*, *active* or *interactive*. These classifications refer to the extent in which the system involves the student in the construction and maintenance of their student model. Passive modelling infers the model of a student without explicit help from the student. In active modelling the student may be asked questions by the system to assist with the modelling task. Interactive modelling [DSB00, Bul98] is a subset of open student modelling. In addition to giving the student access to their model, it focuses on the idea that a user should play an active role in the development and maintenance of their own model. This approach involves the learner explicitly, allowing them to inspect and change their own student model. This has benefits both computationally and educationally. Computationally, explicit student interaction simplifies the maintenance complexity and increases the reliability of the student model. Educationally, student involvement promotes reflection. "When a learner is engaged in a discussion about the learner model he is *reflecting upon his domain knowledge and experience re-calling and re-considering ideas of which he is aware*" [DSB00].

Past research has concentrated on the development of systems to support open student mod-

elling, particularly systems employing interactive techniques, but little has been reported as to the effectiveness of these systems with respect to learning. Our interest lies in the evaluation of simple *open* student models, that is, non interactive ITSs that only allow the student to inspect their model. The goal is to evaluate how such an open student model affects learning. For this purpose, KERMIT, an intelligent tutor for database design has been extended and evaluated by students studying database design at the University of Canterbury. Preliminary subjective results are encouraging and support an open modelling approach, although a more extensive evaluation is needed to draw reliable objective conclusions.

The remainder of this report is organised as follows: Chapter 2 gives an overview of existing systems that support open student models and expands on some of the concepts discussed here. Chapter 3 describes the extension made to KERMIT to expose the student model. The evaluation is discussed in Chapter 4, and finally, Chapter 5 presents the conclusions and proposes suggestions for further work.

Chapter 2

Background

Reflection is the process of reviewing one's knowledge about a topic of which one is aware, and is understood to enhance the learning process. Open student modelling provides a way to promote this kind of reflective activity in an ITS. This research focuses on determining whether students learn any better with an ITS that encourages reflection through a simple open student model.

2.1 Open Modelling

Past research into open modelling has concentrated on representation techniques used to expose the student model to the student and the degree of system/student interaction offered by such systems. Representation techniques tried include conceptual graphs [DSB01, DSB00], tree structures, tables and actual `Prolog` clauses [DSB00]. The degree of interaction in these systems ranges from visual access of the student model with limited or no ability for students to influence changes in their model, to fully interactive systems that facilitate discussion of a student's knowledge state. The following sections review some of the existing ITSs that support open student modelling.

2.1.1 ELM-ART

`ELM-ART` (Episodic learner model - The adaptive remote tutor) [PEG96, ELM] is a web based ITS designed to teach students to program in LISP. It provides reference material, examples and problems in hypermedia form and allows the student to navigate through any part of the hypertext space. The student model uses progress bars to indicate students' progress through the material and visual cues such as icons, colours and fonts to distinguish between several educational states. The states represent a student's knowledge of the current page which can be known by the student, ready to be learnt or not ready to be learnt (the prerequisite knowledge has not yet been learnt). This provides support for student navigation through the hyperspace. The student is also able to modify their model to indicate to the system whether or not they feel that they *know* a selected topic. However, system/student discussion of the student model is not supported. In addition `ELM-ART` can give students prerequisite based help and guidance. For example, a student may request an example and will be provided with a list of links to examples ordered by their relevance based on the student's learning history. This can further support student reflection because it can help the student find relevant examples from their previous experience.

2.1.2 STyLE-OLM

`STyLE-OLM` [DSB01, DSB00] is an interactive open learner modelling component for a terminological domain. The student model is jointly constructed by the student and the system, and incorporates the student's beliefs and misconceptions. Beliefs can be correct, incomplete or erroneous, and are open for inspection and discussion. A graphical representation based on conceptual graphs is used to externalise the student's beliefs. The student is involved actively in discussion

of their beliefs through dialogue games where both the student and the system can ask questions, state propositions, challenge or withdraw the other's propositions, justify their own claims and direct the focus of the discussion.

A small evaluation of STyLE-OLM has been conducted with seven postgraduate students [DSB01]. However, the primary focus of the experiment was on the behaviour of the system and not on how the interactive open modelling component affected learning.

2.1.3 Mr Collins

Mr Collins [BP95, DSB00] is another interactive system that illustrates a joint effort from the student and the system to construct the student model. The system is intended to maintain more accurate models and promote student reflection through discussion of model contents. Its domain is object pronouns in European Portuguese for second language learners. In addition to general performance measures, the system's student model maintains additional information as part of the explicit model. This includes acquisition order of target knowledge, analogy and learning strategies. The model and its benefits are discussed in [BBP95].

Mr Collins maintains separate measures of the student's beliefs and the system's beliefs about the student. This allows the system to identify conflicts between the student and system viewpoints of the student's performance, and enables the student to influence the student model through negotiation.

The communication environment is text based and is restrained to menu options which may confine a student's reflection. The student model is conveyed to the student through tables which maintain domain rules associated with both student and system measures of the student's confidence in those rules. Discussion of the student model may be initiated by either the student or the system. To be sure of its own representation of the student model, the system may request information from the student. In addition, if the student's beliefs become too distant from the system's beliefs, the system may challenge the student. The student may also initiate a discussion at any time to change their own beliefs or challenge the system's beliefs. Although both the student and the system can influence the other's beliefs through discussion, ultimately both parties have the final say in their own beliefs.

A preliminary evaluation involving eighteen participants, conducted on Mr Collins [BP95] has suggested students were willing to inspect their student models, suggest changes and argue when in disagreement with the system. Again, the effect of the open model on learning was not considered.

2.1.4 TAGUS

TAGUS [DSB00] is an independent server that provides external agents with viewers for maintaining student models. Both the student and the educational system are external agents. The system provides a general framework for modelling and maintaining changes in students' beliefs, actions, reasoning strategies and goals. Options are provided to control the information in the student model. These include actions to add new information to the model, resolve contradictions and remove information from the model.

The student model is represented through Prolog clauses, and the student can manipulate their model by selecting from these options and typing Prolog clauses into a control panel. A limitation of the system lies in the representation of the student model as Prolog clauses. It is unreasonable to expect that typical students will have knowledge of Prolog and thus be able to make sense of their model.

2.1.5 UM

UM [Kay, DSB00] is a toolkit for user modelling developed for a variety of co-operating agents. The toolkit is composed of simple components that can be combined for modelling tasks and is flexible in the models that it can represent. For example, [Kay] has demonstrated its use in modelling user

preferences and in a coaching system where the modelling task is to represent students' evolving knowledge of a text editor. The latter modelling task uses the toolkit's model viewer to visualise the student model as a tree structure. The appearance of a component in the tree represents the student's knowledge of that component. For example, colours are used to give the student an indication of whether or not they know the element displayed, and the shape of a component indicates its type, which can be beliefs, preferences or attributes. A nested shape is displayed when the student's knowledge of a component is not known. When a component in the tree is clicked, a menu is offered to the student, giving them the ability to ask for an explanation about that component, request a justification of some assigned value, or alter their model by suggesting a new value for that component. Unlike `STyLE-OLM` (Section 2.1.2) and `Mr Collins` (Section 2.1.3), `UM` does not facilitate negotiation of the student model.

An evaluation of the coaching system is reported in [Kay]. Their motivation was to test the usability of the interface, investigate whether students would examine their model, and to see whether the visualised model was understandable.

2.1.6 `See Yourself Write`

`See Yourself Write` [Bul97] is a system for foreign language writing that is composed of two parts: a template that is used by a *human* teacher to give students feedback on their writing assignments, and an inspectible student model for each student that is generated from the feedback given by the teacher and which is built up over time. The system is intended to be used in place of traditional written feedback. Unlike traditional student models, the model maintained by `See Yourself Write` is intended as a source of information for just the student and not an educational system. The model contains both qualitative and quantitative data and is constructed automatically given any new feedback input from a teacher and all previously provided feedback. The purpose of the system is to promote student reflection on completed writing assignments in such a way that the feedback may be used to improve subsequent work. All aspects of the student model are open for inspection; this includes the qualitative and quantitative information, general information and assignment specific information. In addition, students can record their own notes on any information in the model. To further encourage reflection, the system may ask the student why they believe a given pattern of development in their work has occurred over time. The possibility also exists for the student to disagree with the contents of their model by expressing their views to the teacher through the system, and thus, exercising their skills in self explanation.

2.1.7 `diyM`

`diyM` [Bul98] is a system that allows students to construct their own inspectable student models rather than using system created ones and can be used alone or combined into other learning environments. It has been illustrated with `Mr Collins` (Section 2.1.3) and `See Yourself Write` (Section 2.1.6).

2.1.8 Summary

The systems discussed here all expose the student model in some form, providing the student with the opportunity to reflect on their knowledge. However, they also differ in two main ways:

- The degree of system/student interaction is variable. Systems such as `Mr Collins` (Section 2.1.3) and `STyLE-OLM` (Section 2.1.2) not only allow the student to inspect their model, but also facilitate discussion and/or negotiation of the model contents. In addition to supporting reflection, discussion with the student can assist in the development of a more accurate student model [BP95]. Other systems only allow the student to inspect their model, although systems such as `ELM-ART` let the student freely change their model, but discussion and/or negotiation is not supported.

- Different representations such as trees, conceptual graphs and `Prolog` clauses, among others, have been used to visualise the student model. Structures that can convey the relationship between aspects of a domain, such as trees, can create an awareness of the size and structure of the domain, which may aid students' understanding of the domain.

To date there have been few reported evaluations of these systems, and those that have been conducted have mostly focused on system performance, or whether students are willing to inspect their models. However, the effectiveness of open models on learning remains open. The question of how open a student model should be to optimise the learning process is raised in [ZR01], although experiments are yet to be performed.

2.2 Constraint Based Student Modelling (CBM)

Student modelling is the process of representing the knowledge state of a student through the gathering of relevant information about that student. The task of completely and accurately modelling a student's knowledge state is intractable [MMSM01]. CBM is an approach to student modelling that was originally proposed by Ohlsson [S94] as a way of overcoming the intractable nature of the task. Constraint based models reduce the complexity of the modelling task by focusing only on correct knowledge. Rather than check whether a student is performing correctly by comparing the student's procedure to one or more correct ones, CBM is only interested in the state the student is currently in. That is, what the student has done is irrelevant and the procedure used to solve the problem is unimportant [MMM02].

Domain knowledge in these models is represented as sets of constraints which define sets of equivalent problem states. A constraint specifies a property of the domain that is shared by all correct solutions. Formally, a constraint is an ordered pair (Cr, Cs), where Cr is the relevance condition which determines those problem states in which the constraint is relevant, and Cs identifies the class of relevant states in which the constraint is satisfied. Consequently, if a constraint is relevant in some state then it *must* also be satisfied in that state in order for the solution to be correct.

An example of a constraint in the domain of ER modelling might be:

```
IF the student's solution contains one or more entities
THEN each entity in the solution must have a key attribute
```

Here the `if` statement is the relevance condition and the `then` part is the satisfaction condition. Hence, whenever the student's solution contains one or more entities, the relevance condition is true, and to be correct, it must be the case that the solution falls into the state defined by the satisfaction condition. If this is not the case then the constraint is said to be violated and the student's solution is incorrect. On the other hand, if the constraint is not relevant in the current state, then the satisfaction condition is trivially satisfied. A correct solution is then one for which *all* constraints are satisfied.

Unlike other student modelling approaches, CBM does not require an expert (or domain) module to solve problems. This is an important advantage because, depending on the domain, a problem solver may be very difficult to develop. The insensitivity of CBM to the procedure used to solve tasks also provides the means for a more flexible system.

2.3 Why KERMIT?

The ICTG (Intelligent Computer Tutoring Group) have developed three constraint based ITSs [MMSM01]. These are: `SQL-Tutor` [MMM02], an intelligent tutor aimed at helping university level students learn `SQL`; `CAPIT`, a punctuation tutor designed for school children aged between nine and eleven; and `KERMIT`, an intelligent tutor for database design aimed at university level students.

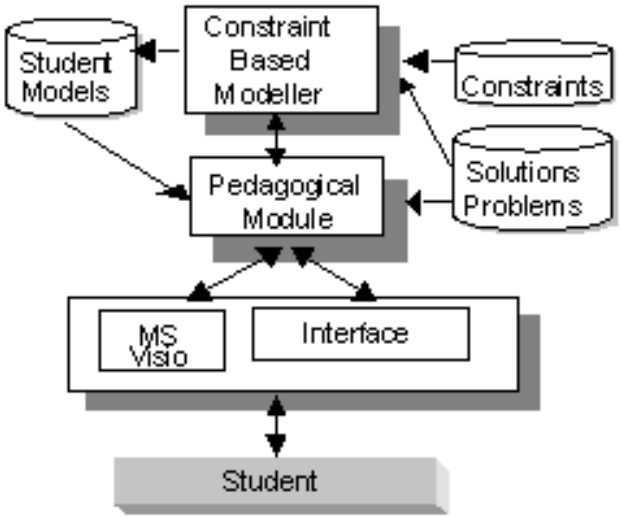


Figure 2.1: The architecture of KERMIT [Sur01]

It was decided to extend `KERMIT` for our research purposes. This choice was influenced by the system's size and the intended educational level of the tutor. We believe open modelling techniques are more likely to benefit high school/university level students because they are more able to interpret what they see. That is, as cited in [WSF99], the ability to reflect is a characteristic of the most advanced stages of cognition. `CAPIT` was thus not considered. `KERMIT` was selected over `SQL-Tutor` mainly due to size. The size of the constraint base in `SQL-Tutor` is over fives times that of `KERMIT`'s and as evaluation is our main objective, an extension of the smaller system in the time given is more suitable.

2.4 Overview of KERMIT

KERMIT (**K**nowledge-based **E**ntity **R**elationship **M**odelling **I**ntelligent **T**utor) is a constraint based tutor developed as a problem solving environment for database design. The system is based on the entity relationship (ER) conceptual data model as described in [EN94]. KERMIT is intended to complement traditional instruction. It assumes students are familiar with the ER design model. KERMIT is an intelligent tutor in that it maintains models of each student, facilitating the ability to provide individualised pedagogical instruction.

2.4.1 The Architecture of KERMIT

The architecture of KERMIT is depicted in Figure 2.1. The constraint based modeller or student modeller maintains the student's knowledge state. Currently it can evaluate a student's solution and record statistics such as the number of times each constraint was relevant and satisfied. This data constitutes the student model and provides individualised instruction.

The pedagogical module is the main driving engine of the system. It determines the appropriate instructions to carry out and the timing of these instructions. Currently this includes problem selection and individualised feedback delivery based on the student's solution to a problem.

KERMIT consists of 92 constraints. These cover both syntactic and semantic knowledge. The syntactic constraints are concerned with syntactic details in a student's solution. An example of such a constraint is "An entity (regular or weak) cannot be directly connected to another entity". Semantic constraints relate the student's solution to the system's ideal solution. "The student's

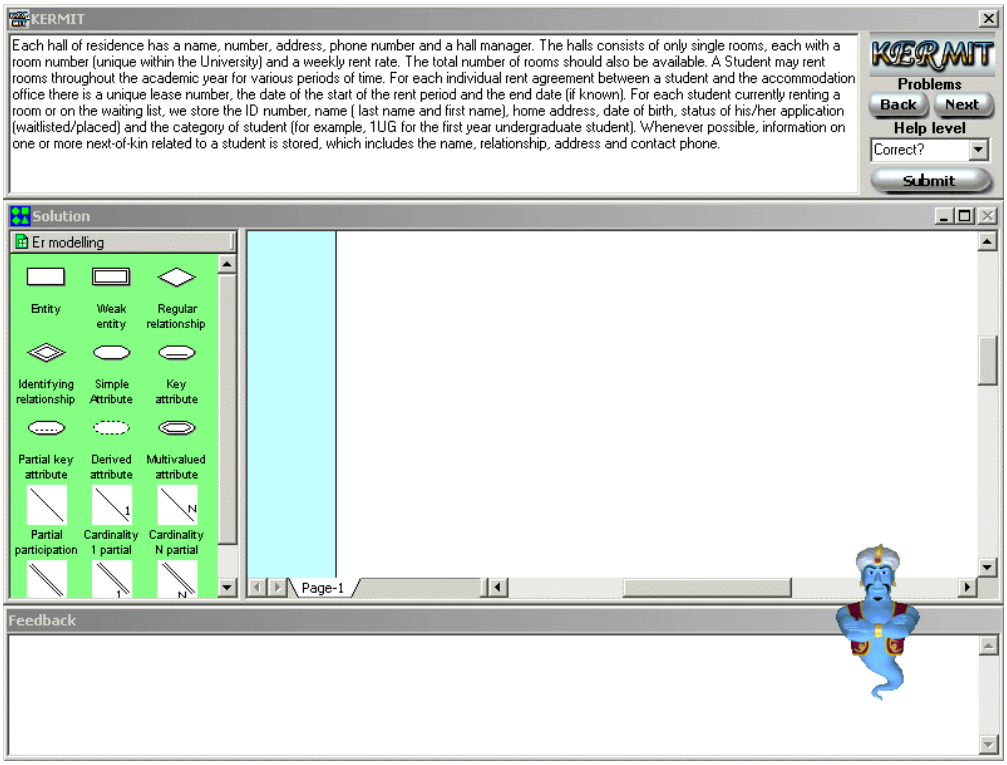


Figure 2.2: The interface of KERMIT

solution should consist of all entities present in the ideal solution" is an example of a semantic constraint.

The interface in KERMIT is composed of three windows tiled vertically (Figure 2.2). The top window displays the current design problem and provides controls for stepping between problems, submitting a solution and selecting the level of feedback (Section 2.4.2). The middle window is the main working area. In this window the student formulates a solution to the current problem using the ER constructs provided by the toolbar on the left side of the window. The lowest window is the feedback window. All feedback generated by the system is displayed here, as well as through an animated pedagogical agent — the genie, which is visible in Figure 2.2.

2.4.2 Learning Support

The interface of KERMIT minimises the working memory load on the student by providing both the current problem description and the toolbar of ER constructs in the one view. This allows the student to concentrate on the actual modelling problem at hand. When creating ER diagrams, students are also forced to highlight the text in the description that corresponds to each new construct they model in their solution. This enables the student to keep track of what they have modelled, reducing the complexity of the problem, and, at the same time, making it easier for the system to inter-relate the elements of the student's solution and the problem specification.

Students are not constrained by the system. That is, the system does not enforce any ordering on the problems or restrictions on the steps involved in solving problems. For example, it does not matter to KERMIT whether a student decides to construct entities before relationships.

Variable level feedback based on the level of detail is supported. Students can select from *Correct*, which indicates whether a solution is right or wrong; *Error flag*, which states the type of construct that contains the error; *Hint* and *Detailed Hint*, which report information on the most

important error from all errors; *All errors*, which lists all errors, and *Solution*, which displays the solution to the current problem.

In addition to reporting feedback in the bottom window (Figure 2.2), an animated pedagogical agent (the genie) is used to gesture and verbalise feedback. Animated agents can facilitate learning by, for example, increasing the motivational levels of students [Sur99].

A current weakness in KERMIT is its lack of support for reflection. Allowing students the opportunity to reflect enables them to consider and question what they know, and is of benefit to the learning process. Reflection can be promoted through the externalisation of the student model, that is, by allowing students access to examine their model. By extending KERMIT in this way, the effectiveness of reflection on learning in the environment of an ITS can be evaluated.

Chapter 3

Extending KERMIT

KERMIT (Section 2.4.2) currently lacks support for students to reflect on their knowledge. We have created E-KERMIT (Enhanced KERMIT) to determine the effectiveness of open student modelling techniques in achieving learning benefits as a result of reflection. This chapter details the extension of KERMIT, describing its design, and providing the major implementation details.

3.1 Design and Implementation

One of the first decisions was how interactive the extension should be, as the degree of interaction varies between open systems (Section 2.1). Consequently, the support for reflection, as well as other factors, such as the intuitiveness of the interface, are likely to influence the additional learning benefit (if any) received from systems incorporating open techniques.

We opted to provide a minimally interactive interface, exposing only the contents of the student model, for three reasons:

1. The time needed to implement a fully interactive open student model would be considerable and our time is limited.
2. Evaluation of such a system may serve as a useful baseline for the development of future interactive systems that intend to achieve benefits from reflection.
3. It allows for an incremental approach. That is, the results gained from the evaluation may serve as useful input in carrying out further extensions and evaluations.

Design of the extension consisted of four stages:

1. A visual representation of the student model was developed.
2. The interface was designed.
3. The internal representation was designed.
4. The algorithm to calculate and maintain student knowledge was developed.

These are discussed in detail in the following sections.

3.1.1 Visual Representation

The question of how to visually present the student model is an important one in the design of any open system. The current student model in KERMIT records the total times each constraint has been relevant, and, of those times, the total number of times the constraint was satisfied. This data can be used to estimate student knowledge of the domain (Section 3.1.4). An external representation of this data must visually communicate the knowledge in such a way that is useful to the

student. To achieve this, a representation that is intuitive and understandable is required, which will encourage students to engage in reflective activities with the system. Without the student *wanting* to reflect, the external representation is of no benefit. This is particularly important for our purposes, as the effectiveness of the open student model on learning can only be examined if students do attempt to reflect on their model.

The existing constraint base in KERMIT currently has no structure, that is, the constraints are not ordered or grouped in any way. However, to be able to effectively communicate student knowledge of these constraints, the constraints must be grouped. Attempting to report statistics on individual constraints is too specific and there are far too many of them to provide any useful information to the student. Hence, a taxonomy was designed to classify every constraint in the constraint base of KERMIT into one or more relevant categories, representing the processes and concepts in the domain of ER modelling. The design of the taxonomy was a mixture of top down and bottom up approaches. The constraints were first examined to find general groupings. It was found that constraints map closely into notation concepts, and the processes of constructing entities, constructing relationships and identifying attributes. General categories were created for these groupings and then the constraints were examined again to partition these categories more specifically. The process was iterated until a suitable set of groupings was obtained. The groups were combined into a root category to form the hierarchical structure shown in Figure 3.1. For example, consider the category `constructing relationships` (see the figure), it is split into the processes of identifying relationships, identifying attributes of relationships, identifying participating entities of relationships and assigning cardinalities and participation constraints, which together form the process of completely constructing a relationship from a design problem. A hierarchical representation was chosen to maximise flexibility, thereby allowing for both fine and coarse grain views of a student's knowledge, and to give students a picture of the structure of the domain.

3.1.2 Interface

The interface is the medium through which the student has the opportunity to interact and reflect on their knowledge, so it is important that its design be simple and understandable. Figure 3.2 shows the extended main interface of KERMIT[1]. The original feedback window now includes a summary of the student's progress, the number of problems the student has completed, the option of seeing the detailed view of their progress with the `Show Me More` button, and access to a tutorial on understanding the main progress view via the `help` button. Summary statistics are shown in the main interface because the main progress view is too large to display with the main interface. The summary statistics are intended to provide a form of constant feedback on progress and act as an aid to remind and motivate students to further inspect their progress.

Figure 3.3 depicts the window that shows the full hierarchical view of a student's progress in E-KERMIT when the `Show Me More` button in the main interface is clicked. The window is split into two frames. The top frame displays the hierarchical taxonomy of ER knowledge, as described in Section 3.1.1, with progress statistics for each category. For example, consider the highlighted category `Composite` in Figure 3.3. The fraction to the right of the progress bar shows the student's score out of the percentage of material for composite attribute identification covered so far. That is, the student has scored a total of 45% out of a possible 55%, or equivalently, they have got 82% ($\frac{45}{55} \times 100$) of the material covered for this category correct. The progress bar represents the 45% indicating the student's progress through the material so far. These statistics are calculated as outlined in Section 3.1.4.

The lower frame provides a textbox that describes the currently selected category. This is an aid to provide an explanation for any part of the taxonomy that the student may have problems interpreting. In addition, the first time a student accesses their progress they are given a short tutorial on how to interpret what they see. The tutorial is made available at any time by clicking the `Help` button in the summary progress view in the main interface. A screen shot of part of

[1]The original interface is shown in Figure 2.2.

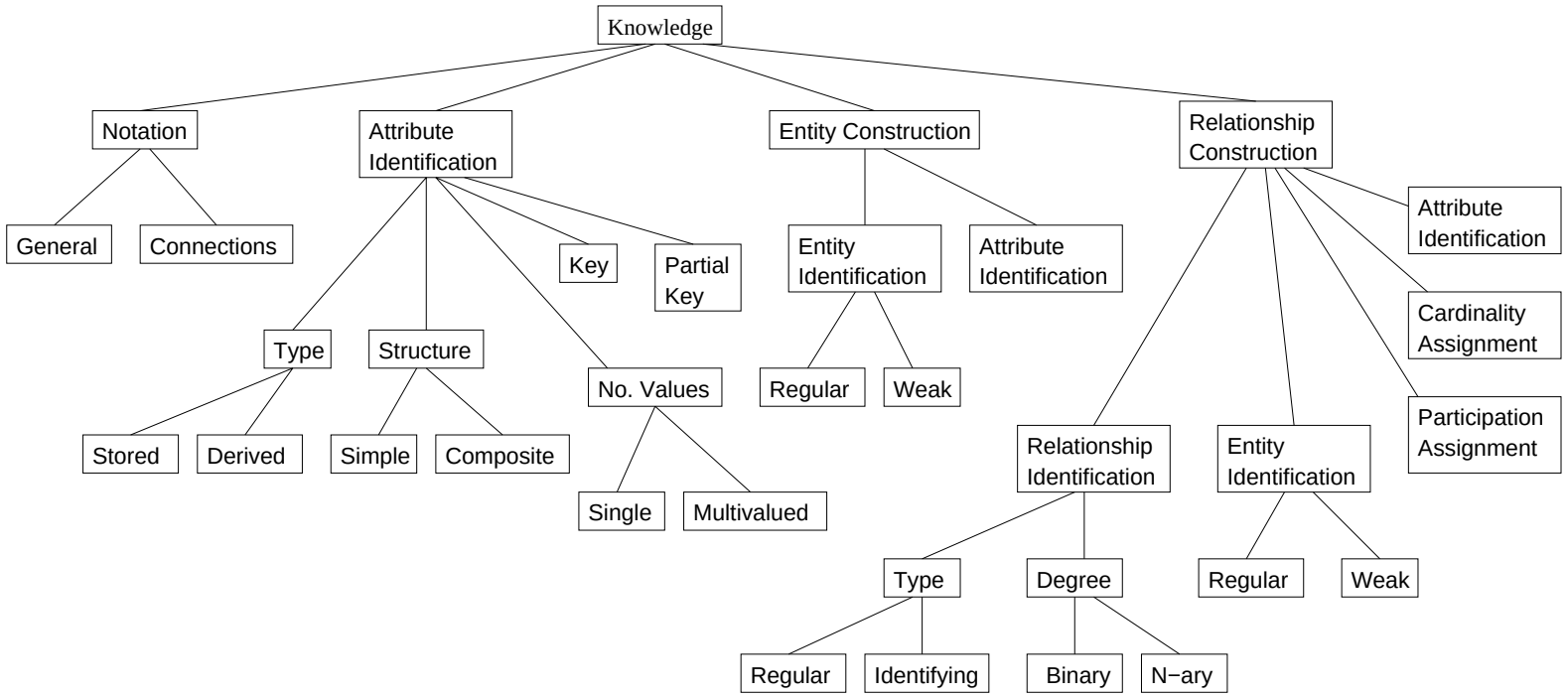


Figure 3.1: Taxonomy of ER modelling concepts and processes

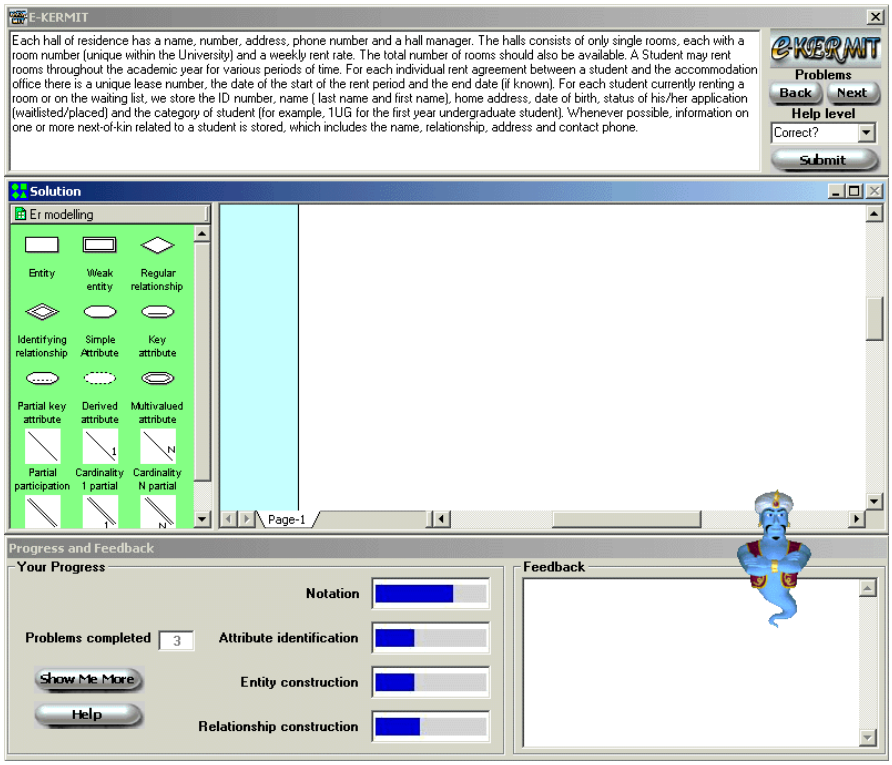


Figure 3.2: The interface of the extended KERMIT system, E-KERMIT

the tutorial is shown in Figure 3.4. The aim of the tutorial is to facilitate a quick understanding by giving students an introduction to how their progress is represented, what the percentages convey, and to create an awareness of the category explanation aid in the lower frame[2]. As stated previously (Section 3.1.1) reflection is dependent on the student wanting to reflect, which is likely to be partially dependent on the interface. That is, students cannot be expected to effectively reflect if they cannot understand what they see as their model. Consequently the approach taken here is to avoid the need for human explanation and have E-KERMIT assist the student as much as possible.

3.1.3 Internal Representation

The current student model of KERMIT lacks any form of structure between individual constraints. To represent student knowledge as defined by the ER knowledge taxonomy described in Section 3.1.1 and shown in Figure 3.1, a hierarchical representation that groups the constraints in the constraint base according to the taxonomy is required. A simple representation can be achieved by implementing each category in the taxonomy as a node in a tree, where each node has a collection of constraints as defined by the category, and a collection of child nodes (if the node is not a leaf). However, a more efficient implementation is possible because it is not necessary in this case for every node to maintain its own constraint collection. The constraints belonging to a parent category are exactly the union of the constraints in the parent's child categories, thus it is sufficient for only the leaf nodes to maintain such a collection.

Our implementation uses the more efficient representation described above, where nodes are one of two types: leaf nodes or non-leaf nodes, such that leaf nodes have a collection of constraints

[2]Previous use of KERMIT had revealed that some students did not become aware of the feedback window at the bottom of the interface (Figure 2.2).

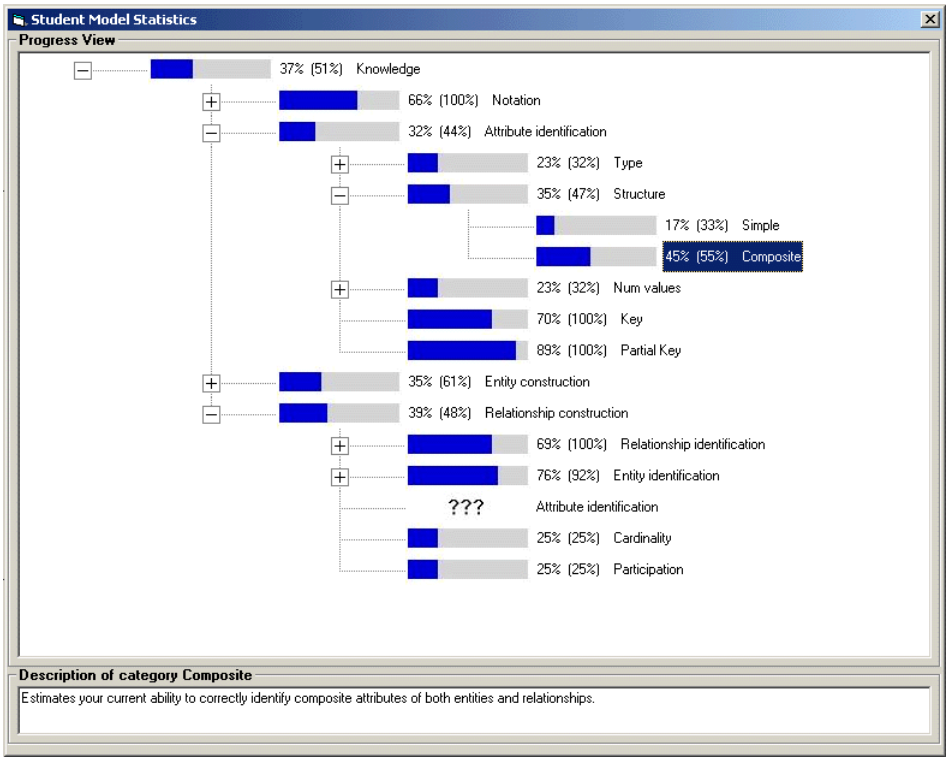


Figure 3.3: The main view of a student's progress in E-KERMIT

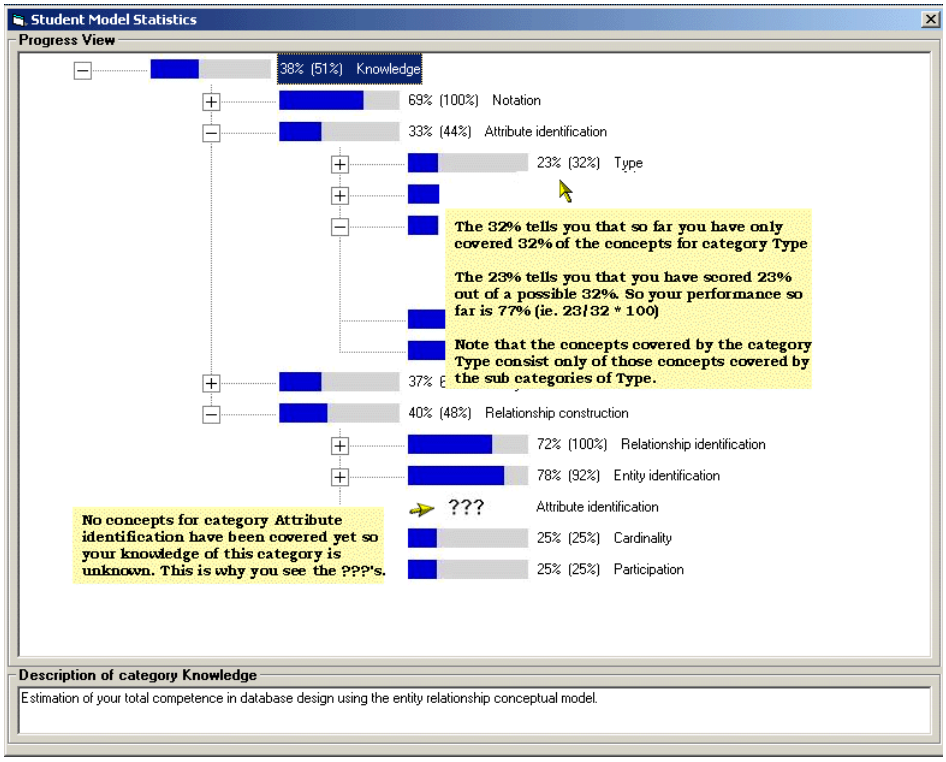


Figure 3.4: A slide from the E-KERMIT progress view tutorial

and non-leaf nodes have a collection of child nodes. In addition to these collections, *every* node maintains a count of the number of constraints it represents and the sum of the probabilities that all constraints represented by that node have been learnt. Storing these additional values as well as maintaining collections of constraints only in leaf nodes makes the implementation more efficient in both time and space, because less time can be spent updating student knowledge and fewer collections need to be stored. Section 3.1.4 outlines how student knowledge is calculated from this representation.

3.1.4 Calculating Student Knowledge

To be able to assess student knowledge, a student model that maintains an estimate of student progress is required. The current student model in KERMIT maintains counts for the total number of times each constraint has been relevant and the number of those times each was satisfied. Calculation of student knowledge is based on these values.

Learning Constraints

A simplistic estimate that a constraint has been learnt is to calculate the proportion of times the constraint has been satisfied out of the total times it has been relevant. This estimate is simplistic because it does not *forget* easily. That is, it bases student knowledge of a constraint on the student's entire history of using that constraint, and does not take into account that earlier attempts at applying the constraint may no longer be valid. A better model is to base knowledge of a constraint on the last `x` times the constraint has been relevant. In this case `x` can be used rather than the relevance total, unless this total is less than `x`, in which case the total should be used. This allows the influence of previous attempts to be controlled. The choice of `x` will clearly have an effect on the probability estimate. For example, choosing a value too large will result in knowledge of a constraint being influenced by irrelevant history, but a value too small will disregard previous attempts that are still relevant. Lack of time meant experimentation with values for `x` was not possible. Our choice of `x` was five. This was based on the value used in `Mr Collins` [BP95] for a similar purpose.

One question that arises with either of these approaches is what to report when the constraint has never been relevant. This situation will arise mostly when the student uses the system for the first time. By assigning a probability in such a case, an assumption is made on the student's knowledge. To avoid an incorrect assumption our approach is to report the knowledge of the constraint as unknown.

Calculating Knowledge of Categories

A student's ER knowledge is based on the hierarchical taxonomy given in Figure 3.1. Each category in this taxonomy is defined by a group of constraints, where each parent in the hierarchy contains exactly those constraints in the parent's child categories.

Student knowledge of a single category can be calculated as a combination of the probabilities that the individual constraints of the category have been learnt. A simple combination function is to take the average of the probabilities of the individual constraints. However, a modification is required to allow for the possibility that some or all of the constraints represented by the category could be unknown. Suppose we want to calculate student knowledge of category c, then let n be the number of constraints in c, p_i be the probability that constraint i has been learnt, x be the total number of constraints in c for which probabilities are unknown, and p_c be the knowledge probability for c. Two options were considered:

1. If $x > 0$ then set

$$p_c = unknown$$

else set

$$p_c = \frac{\sum_{i=1}^{n} p_i}{n}$$

2. If $x = n$ then set

$$p_c = unknown$$

else set

$$p_c = \frac{\sum_{i=1}^{x} p_i}{n}$$

Option 1 avoids the problem of unknown probabilities by reporting a category probability iff probabilities are known for *all* of the constraints in the category. Otherwise the probability of the category is said to be unknown. This causes problems when some constraints are rarely relevant. For example, the category `Relationship construction` has around thirty constraints related to all areas of relationship construction. A few of these constraints deal with recursive relationships which are only relevant in a small fraction of problems, as recursive relationships are quite rare. It is expected that such problems would be given to more advanced students. Consequently, unless the student has used the system for long enough to see such a problem, the system would be unable to report on the student's progress on relationship construction even though recursive relationships are only a minor part of the category. To overcome this, our approach uses Option 2, which relaxes the condition that *all* constraint probabilities are required. It does this by assuming that the probabilities for unknown constraints are 0. This allows E-KERMIT to report information on part of a category. The probability that is calculated represents a proportion of the constraint coverage which is defined by $\frac{x}{n}$. For example, if a student had covered 40%[3] of the constraints of a category for which their knowledge estimate was 35% (that is, $p_c = 0.35$), then the student's knowledge is 35% out of a possible 40%, or equivalently, they are currently performing at 87.5%.

Updating Student Knowledge

A student's knowledge is updated each time the submit button in the main interface (Figure 3.2) is clicked. When this happens, the internal structure (Section 3.1.3) of the student's knowledge is updated. This involves recalculating the probabilities for all relevant constraints as outlined in Section 3.1.4 and updating those nodes/categories that represent these constraints as described in Section 3.1.4. The external representation (Section 3.1.1) is adjusted each time a node is updated. To update the nodes in the internal representation tree, all nodes that require evaluation (those for which at least one represented constraint was relevant) are marked for evaluation when the relevant constraint probabilities are recalculated. The node probabilities are then evaluated in a depth first manner starting from the root of the tree (Section 3.1.3). Maintaining the total number of constraints and the sum of constraint probabilities as detailed in Section 3.1.3 avoids unnecessary processing time, because evaluation of a node (and hence, its children) can stop if the node does not require re-evaluation since the previous sum is still valid and available, and thus, eliminates the need to descend to the leaves of the tree to sum up the probabilities for each constraint represented in that node.

3.2 Implementation Specific Details

E-KERMIT is implemented in Microsoft Visual Basic [DDN99, BM99] and uses the ActiveX `TreeView` control to maintain the graphical view of student progress or knowledge.

The main reason for choosing the `TreeView` control rather than implement a new control with similar functionality, was to reduce implementation time, since implemention was not the main focus. However, the use of the `TreeView` control introduced a few restrictions. Firstly, the control cannot accommodate objects other than images and text. This meant the progress bars had to be restricted to images. The implementation uses twenty one images of progress bars ranging from 0% to 100% inclusive, stepping up in increments of 5%, and one image to represent the *unknown* probability (the three question marks as shown in Figure 3.3 for the category `Attribute identification`). Secondly, the `TreeView` control also enforces the ordering that images must

[3]This is the proportion multiplied by 100.

appear to the left of the text. In general, the main problem with the control is its narrow public interface which makes any extension virtually impossible. For example, the control provides no way to access the labels on the view making specialised tool-tips on individual categories impossible also.

Chapter 4

Evaluation

The main goal of this research is to evaluate whether students learn more effectively when given the opportunity to reflect through an open student model. To achieve this, an evaluation of E-KERMIT was carried out with students enrolled in the second year database design course at the University of Canterbury. The evaluation sought answers to the following questions:

- Do students learn more effectively with an open student model?
- Do students investigate their student models?
- Do students feel that an open model assists their learning?
- Does an open model encourage reflective activity?
- Are there any differences between more and less able students with respect to the above points?

The next section outlines the design of the experiment and in the sections to follow, the results are analysed and discussed.

4.1 Experiment Design

The experiment was conducted with Stage 2 students enrolled in the database course, COSC226 (Introduction to Databases) in the Department of Computer Science at the University of Canterbury. The experiment was run during the students' normal lab hours in the department's computer laboratories over the duration of one week. Participation was anonymous and voluntary.

To control the experiment, students were assigned to one of two groups: A control group which interacted with the original KERMIT system, and an experimental group which interacted with the extended version, E-KERMIT. In each session, the groups were assigned to different laboratories to prevent students from being exposed to both systems. Students randomly chose a workstation in one of the laboratories[1], and were not aware of the different versions of the system.

The duration of each session was at most 110 minutes and each student participated in one session during the week. Data collection consisted of four stages: pre-testing, system interaction, post-testing and subjective system assessment. Figure 4.1 shows this process and the number of participants involved in each part[2]. Since participation was voluntary and laboratory attendance varies, it was hard to control group sizes[3]. To assess student knowledge of ER modelling prior to and after using KERMIT and E-KERMIT, each student was asked to complete a pre-test and a

[1] In cases where one laboratory was significantly favoured over the other, some students were asked to move to the other laboratory.

[2] Unfortunately five students in the KERMIT group did not sit the pre- and post-tests.

[3] This study was combined with another one to evaluate KERMIT. This made it more difficult to obtain equal numbers in each group.

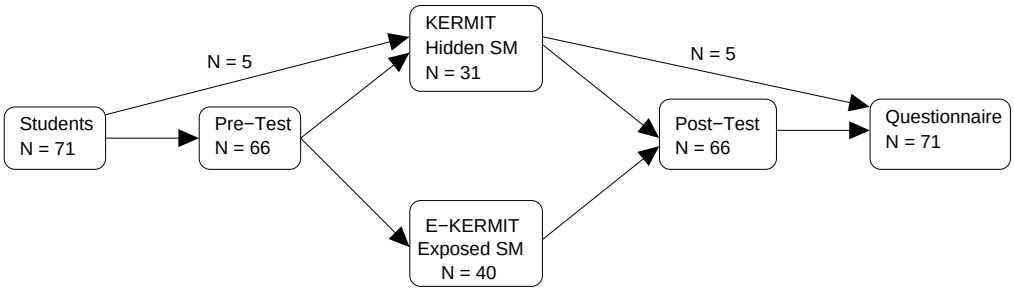


Figure 4.1: Outline of the experiment

	N	Mean		Standard deviation	
		Pre-test	Post-test	Pre-test	Post-test
KERMIT	26	16.12	17.77	1.82	1.45
E-KERMIT	40	16.23	17.13	2.59	2.37

Table 4.1: Mean scores and standard deviations of the pre- and post-tests

post-test. The tests consisted of three questions each, with corresponding questions intended to be of similar difficulty. To minimise any effect resulting from variation in test difficulty the tests were rotated between successive sessions.

On completion of the pre-test, the student was free to interact with the assigned system. Most students had at most just over an hour of system interaction, but actual times depended on student preferences. Students were not explicitly told what to do or how to use the systems unless they asked for help. The actions generated by the students were recorded automatically in log files for each student. Following the end of each student's interaction session, they were given a post-test and questionnaire to complete. The questionnaire given to students differed between groups. The E-KERMIT questionnaire included questions from the KERMIT questionnaire that were most relevant to this study[4], and in addition, contained questions regarding E-KERMIT's open student model.

4.2 Results and Analysis

Data from the evaluation study were collected from three sources: pre- and post-tests, student log files and the questionnaires. The data from these sources were subjected to objective, performance and subjective analyses respectively. The results are discussed in the following sections.

4.2.1 Pre and Post Tests

All students who participated in the evaluation study completed a pre- and post-test[5]. The two tests were rotated between sessions and each consisted of three questions. The maximum mark available was 22 in both cases. Table 4.1 shows the number of students and the averages and standard deviations of the pre- and post-test scores for the KERMIT and E-KERMIT groups respectively[6]. Students on average performed equally well on the pre-test ($T(63.5) = 0.202,\ p > 0.05$) providing evidence that both groups were of comparable competence.

A paired T-test was performed within subjects for both groups to evaluate whether students' performance improved after interacting with the allocated system. In both the control and exper-

[4]Not all of the questions in the KERMIT questionnaire were relevant to this study.
[5]The tests are given in appendix A.
[6]The actual student scores can be found in appendices B and C.

		N	Mean gain	Std dev.
KERMIT		26	1.65	1.72
	Less able	15	2.33	1.40
	More able	11	0.73	1.74
E-KERMIT		40	0.90	2.24
	Less able	22	1.73	2.29
	More able	18	-0.11	1.75

Table 4.2: Mean gains and standard deviations achieved on the post-test

			Mean		Standard deviation	
		N	Pre-test	Post-test	Pre-test	Post-test
KERMIT	Less able	15	14.80	17.13	1.01	1.25
	More able	11	17.91	18.64	0.83	1.29
E-KERMIT	Less able	22	14.41	16.14	1.89	2.36
	More able	18	18.44	18.33	1.20	1.78

Table 4.3: Means and standard deviations of the pre- and post-tests for the more and less able groups

imental groups, the post-test mean was significantly higher than the corresponding pre-test mean ($T(39) = 2.542,\ p = 0.015$ and $T(25) = 4.905,\ p = 0.000$ respectively) revealing that, on average, students did improve their performance as a result of system interaction.

To assess whether students in the E-KERMIT group learnt any better than those in the KERMIT group, the gain on the pre-test score (difference between the post- and pre-test marks) for each student in both groups was calculated and the gains for both groups were compared. Within each group students were also divided into *more* and *less* able groups based on their performance in the pre-test. The more able group was comprised of those students who scored above the mean in that test and the remaining students formed the less able group. Table 4.2 shows the number of students and the average gains and standard deviations for each group. A 2×2 two-way randomised ANOVA revealed that the system used had no significant effect ($F(1, 62) = 2.277,\ p > 0.05$) on student gain suggesting that students who used KERMIT on average improved their performance just as much as those who used E-KERMIT. The mean gains between the more and less able students in both the control and experimental groups were found to be significantly different ($F(1, 62) = 45.881,\ p = 0.001$), but there was no significant interaction between student ability and the system used ($F(1, 62) = 0.059,\ p > 0.05$). Further analysis of the pre- and post-test scores for the more and less able groups by way of paired T-tests revealed that *only* the less able students in both the KERMIT and E-KERMIT groups achieved a significant improvement ($T(14) = 6.468,\ p = 0.000$ and $T(21) = 3.534,\ p = 0.002$ respectively) after system interaction[7]. Hence, system interaction appears to be more beneficial to the less able students. This was also found to be the case in [Mit01b, WSF99]. Table 4.3 shows the mean scores when the control and experimental groups are further divided into the more and less able groups.

Comments

By requiring all students to complete pre- and post-tests directly before and directly after interaction with the allocated system, the results described above can be attributed to interaction with the given system. However, it is important to realise that testing students' improvement in this way may be confounded by the following factors:

[7]The more able students made no significant improvement, $T(10) = 1.388,\ p > 0.05$ and $T(17) = 0.270,\ p > 0.05$ for KERMIT and E-KERMIT groups respectively.

- Students may not take the tests seriously because they are not tests in the real sense. This is difficult to avoid in a voluntary evaluation. We assumed that the majority of students did make a serious attempt at the tests. This was observed during several sessions when some students asked if they could use their notes[8].
- The questions in the test may not reflect everything that was learnt by the student during system interaction. Attempting to ask questions similar to those asked by the system could reduce this effect, although the amount of material that can be covered by the tests is limited due to the short time available for students to complete the tests.
- It can be hard to design tests of equal difficulty. This effect can be reduced by rotating the tests within subjects. In this study the roles of the two tests were reversed in each session, so that all students in the same session were given say test A as the pre-test and test B as the post-test, but all students in the next session had the tests reversed. The effectiveness of this was reduced because of the unfortunate difference in the numbers of students attending some of the sessions. A more effective solution would have been to rotate the tests between the students in each session rather than alternate over entire sessions.

4.2.2 System Logs

Actions generated by students during system interaction were recorded in log files together with time stamps. These actions included logging in and out of the system, requesting a new problem, submitting a solution and completing problems. On every solution submission the relevant, satisfied and violated constraints were recorded. This data was used to evaluate student mastery of constraints, which provides a more direct measure of student performance than is available with pre- and post-test analysis. In addition, the log files generated by E-KERMIT recorded data on the use of the external student model. The following section provides a general analysis of the log file data, then the analysis of constraint mastery is presented, and finally the E-KERMIT student model data is analysed.

General Analysis

General analysis of the log files revealed that students in both the KERMIT and E-KERMIT groups spent just over an hour interacting with the systems. The average interaction times of the two groups were not significantly different ($T(71) = 0.166,\ p > 0.05$). The total number of problems attempted[9] and completed on average during this time was also insignificant between the two groups ($T(71.68) = 0.998,\ p > 0.05$ and $T(72) = 1.151,\ p > 0.05$ respectively). However, students in the KERMIT group on average attempted more problems that were not completed ($T(72) = 2.637),\ p = 0.010$). The time spent per completed problem in both groups was the same on average ($P(144) = 0.371,\ p > 0.05$), but the mean time spent per uncompleted problem is significant at the $p = 0.01$ level ($T(156) = 1.817,\ p = 0.071$). It is tempting to suggest that students in the E-KERMIT group were willing to spend more time attempting to solve the uncompleted problems than those in the KERMIT group, which could have been influenced by the opportunity to inspect their student model, but a more extensive evaluation is needed to make this claim. Table 4.4 shows the means and standard deviations of these results.

Constraint Mastery

The process of learning in a constraint based tutor corresponds with a reduction in the overall proportion of violated constraints. As cited in [MMM02], plotting students' learning in terms of constraints results in a curve that closely approximates a so-called power law. That is, the degree of mastery of a constraint is a function of the amount of practice on that constraint. As stated in Section 4.2.2, the data from the student log files were used to assess constraint mastery. Figure 4.2

[8] Students who asked this question were told no.

[9] This number is the sum of the number of completed and uncompleted problems.

	Mean		Standard deviation	
	KERMIT	E-KERMIT	KERMIT	E-KERMIT
Interaction time	66.65	67.65	21.35	27.4
Time per uncompleted problem	16.50	20.55	13.31	14.47
Time per completed problem	13.48	13.97	7.23	7.68
No. attempted problems	4.36	3.89	1.45	2.57
No. uncompleted problems	2.61	1.78	1.40	1.25
No. completed problems	1.75	2.11	1.14	1.39

Table 4.4: Mean interaction details (times are in minutes)

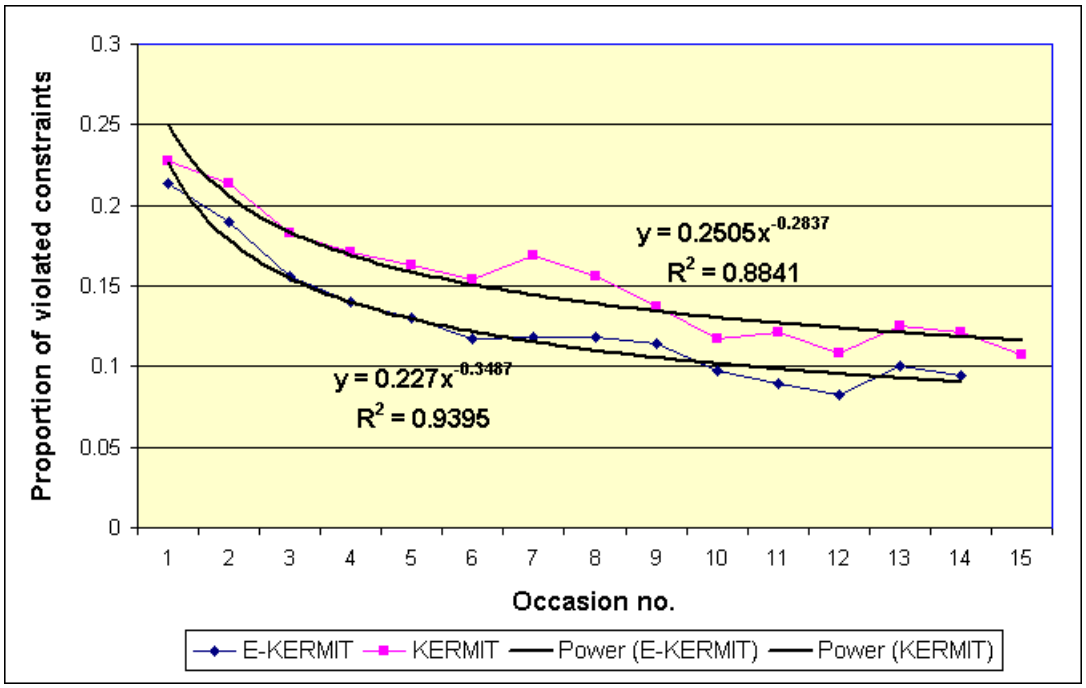


Figure 4.2: Mean proportion of constraints that were violated on the n^{th} occasion

shows the proportion of violated constraints on the n^{th} occasion of application of each constraint averaged across all students and all constraints for both groups. As the occasion of application increases, the set of constraints that were relevant on that occasion decreases in size and the impact of a single failure on the probability increases. To reduce this effect, the analysis was concluded when the size of the set of constraints relevant at occasion n was under two thirds of the original constraint set size. This corresponded to analysing fifteen occasions in KERMIT and fourteen in E-KERMIT. The figure shows very good correlations to the power curve for both groups. The R^2 values for the KERMIT and E-KERMIT groups were 0.8841 and 0.9395 respectively, revealing a better fit for the group who used E-KERMIT.

External Student Model Analysis

Additional log files were generated by the E-KERMIT system to record students' interaction with their models. Data recorded included a rough estimate of the number of times students accessed their full progress model via the `Show Me More` button in the main interface (Figure 3.2), the number of times the tutorial on interpreting the progress model was accessed via the `Help` button

in the main interface, and the categories expanded in the main progress window by each student.

Analysis of this data showed that students, on average, accessed their full progress model 1.36 times[10] during the duration of system interaction. This is possibly an underestimate of the real mean because students may have opened the progress window only once and kept it open in the background. The tutorial was accessed on average 0.53 times. Comparison of the more and less able students revealed no significant difference between the mean number of times both the student model and tutorial were accessed ($T(36) = 1.138,\ p > 0.05$ and $T(36) = 0.989,\ t > 0.05$ respectively). A linear correlation was carried out to see if a relationship existed between the number of times students accessed their progress and their gain on the post-test for both the more able and less able students (Section 4.2.1). The results indicated no real correlation ($r = 0.344$ and $r = -0.146$ respectively), although there was a stronger correlation for the more able students.

Of the 43 students in the group, 22 (51.2%) of them accessed the main progress window. A comparison of the more and less able students showed that 12 of the 18 students (66.7%) in the more able group opened the main progress window, while only 10 of the students (45.5%) from the less able group opened the window. Figure 4.3 shows the proportion of the 22 students who expanded each expandable category in the ER taxonomy described in Section 3.1.1. The numbers in the parentheses from left to right are the total number of less able and more able students who expanded the corresponding category. For example, 45.5% of the 22 students expanded the `Notation` category. Of these, 3 of the students were in the less able group and the remaining 7 were in the more able group. It can be seen that fewer students accessed the more detailed parts of the taxonomy. The figure also indicates that slightly fewer students examined the area of entity construction. This may reflect the evidence that novices tend not to have too much difficulty modelling entities [DR94] and thus, are likely to be less interested in this part of the hierarchy.

4.2.3 Questionnaires

Following system interaction, students were asked to complete a questionnaire for the version of KERMIT they used. The two questionnaires are given in appendices D and E. The responses from the questionnaires[11] provide a subjective opinion on students' interactions and comments on the systems. We examine the commonalities between the two questionnaires, and then the results specific to each questionnaire are discussed in turn.

General Analysis

Students were first asked how much ER modelling experience they had previously had. Valid responses were `only lectures`, `lectures plus some work` and `extensive`. No students in either group said they had had extensive experience; however a surprising number of students (62.5%) in the E-KERMIT group answered with lectures plus work, while only 20% of the students in the KERMIT group answered with this response. Despite the suggested difference in experience between the two groups, results from the pre-tests revealed that the groups were of equal competence (Section 4.2.1).

The time taken to learn the interface was asked next. Possible responses were `less than 5 minutes`, `10 minutes`, `30 minutes` and `most of the session`. To obtain an average time the `less than 5 minutes` option was rounded to 5 minutes and the `most of the session` option was assumed to be 60 minutes. Using these values, the mean times taken to learn the KERMIT and E-KERMIT interfaces were 11.3 minutes and 14 minutes respectively. The Mann-Whitney U test revealed this difference is statistically significant ($z = 0.920,\ p = 0.048$) which is understandable, because the interface of E-KERMIT is more complicated as it includes visualisation of the student model.

[10] There were 43 student log files, but only 40 students in the group, indicating that some students must have logged in more than once with different user names. Since these students cannot be distinguished we ignore this fact and have assumed 43 students in the group.

[11] See appendices F and G.

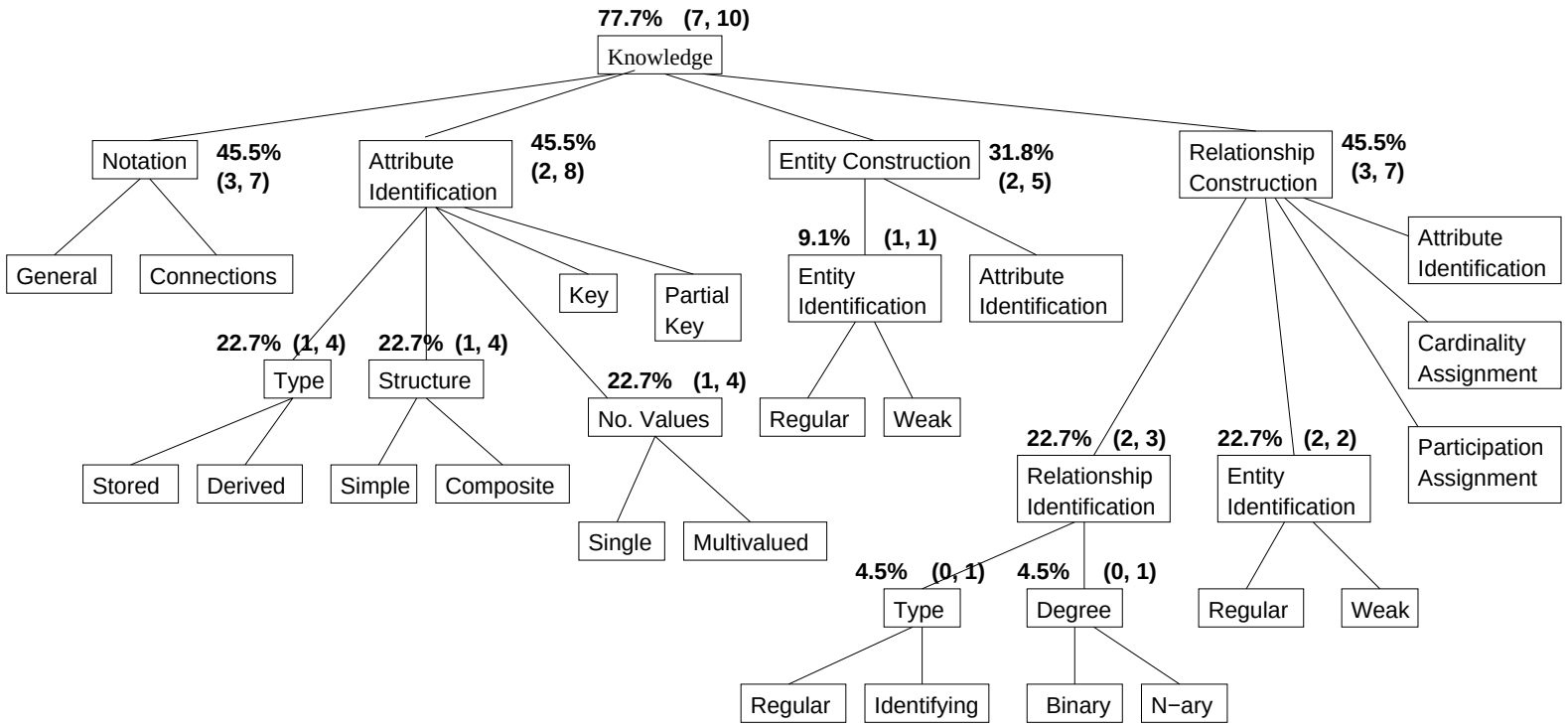


Figure 4.3: Proportion of students who expanded various regions of the ER taxonomy

Students were asked to estimate how much they thought they had learnt from the system they used and how much they enjoyed learning with the system. Responses in both cases were ranked between 1 and 5, with 1 being the lowest and 5 the highest. The mean amount learnt between the KERMIT and E-KERMIT groups was not statistically significant when analysed with the Mann-Whitney U test ($z = 0.709,\ p > 0.05$). The means were 3.2 and 3.1 respectively. Average enjoyment was 3.4 and 3.6 for the KERMIT and E-KERMIT groups respectively. Again the difference was not statistically significant when analysed by the Mann-Whitney U test ($z = 0.389,\ p > 0.05$). It was interesting to compare the more and less able students of both groups to see if the amount these groups thought they had learnt was comparable to their test results[12] (Section 4.2.1). In both the control and experimental group there was no statistical difference between the more and less able students in both how much they felt they had learnt and in the time required to learn the interface ($z = 0.238,\ p > 0.05$ and $z = 1.205,\ p > 0.05$ respectively).

The KERMIT Questionnaire

Students in the KERMIT group were asked to imagine what KERMIT would be like if the system could give them an indication of their current ability to complete ER modelling tasks, and then were asked if they would like to see such a feature in KERMIT. Of the 31 students who completed the questionnaire 1 student said no, 10 were unsure and 20 said yes, thus favouring an open student model approach.

The E-KERMIT Questionnaire

In addition to the questions in common with the KERMIT questionnaire, students in the experimental group were asked a series of questions about E-KERMIT's open student model. On a scale of 1–5, 1 being `never` and 5 being `very often`, students were asked to rate how often they examined their progress. The average response was 3.0 across all students in the group. A Mann-Whitney U test revealed the mean response between the more more and less able groups was not statistically significant ($z = 1.083,\ p > 0.05$). Of the 40 students in the group, 27 of them said they examined their progress mainly by the summary panel in the main interface, 2 said they mostly viewed the main progress window and 8 said they used both views equally[13].

The ease of understanding the open student model is a likely factor contributing to the usefulness of the model. Students were asked whether they thought the progress model was understandable. Of the 36 students who said they examined their progress, 9 students said they had difficulty in understanding their model, 18 had no problems and 9 said the model was understandable after explanation[14].

Students were also asked whether they found the progress views useful, whether they examined their progress to identify weaknesses in their ER knowledge, and whether they felt that the opportunity to examine their progress assisted their learning. Of the 40 students, 20 found the progress model to be useful, 5 were unsure, 4 said no and 10 did not examine their model[15]. The progress model was used to help identify weaknesses in ER knowledge by 22 of these students, thus supporting reflective activity, and 27 students in the group felt that the model assisted their learning. Only 4 students thought the open model did not help their learning and the remaining 9 students were not sure. Table 4.5 summarises the number of responses to the various combinations of these questions. It can be seen from the table that of the 20 students who thought the progress views were useful, 70% also said they examined their model to identify weaknesses in their knowledge and also agreed that the progress model assisted their learning. However, some of the responses appear contradictory. For example, of the 4 students who said they did not think the progress views were useful, 2 of them said they thought the progress model assisted their learning and that they used the model to identify weaknesses in their ER knowledge.

[12] Only 26 students in the KERMIT group had completed a pre- and post-test so only the questionnaires of these 26 students were analysed here.

[13] Three students did not answer this question.

[14] In most cases the explanation came from the system's progress tutorial.

[15] One student did not provide a response.

Progress useful	Identified weaknesses	Assisted learning	Total students
yes	yes	yes	14
yes	yes	don't know	2
yes	no	yes	4
no	no	no	1
no	no	don't know	1
no	yes	yes	2
don't know	no	yes	2
don't know	yes	yes	3
not used	no	no	3
not used	no	don't know	5
not used	no	yes	1
not used	yes	yes	1

Table 4.5: Student responses to three of the open student model questions in the E-KERMIT questionnaire

Several positive comments were made about the open student model in E-KERMIT. These included:

- "It gave a clear graphical representation that is easier to understand."
- "Progress report was pretty good in amount of detail shown."
- "It helped in pointing out areas that I needed further clarification on."
- "Cool."
- "Shows what you need to work on."

4.3 Discussion

The subjective results from this study are quite encouraging. The majority of students who examined their model and understood it at least psychologically found it to be a useful tool to aid in learning. Although the study failed to demonstrate any statistically significant improvements in learning as a result of exposing the student model, students who used E-KERMIT performed at least as well as those who used KERMIT. Given that students were using the systems for the first time and had approximately only one hour's experience with the systems, of which on average, fourteen minutes was used to learn the interface, any benefits of an open student model were not expected to have any significant effect on student performance. Furthermore when E-KERMIT is used by a new user the student model is initially empty because the system does not know anything about that student. Hence, the student needs to use the system for some time before an accurate model of the student's knowledge can be built and visualised to the student. Since only a small number of problems were attempted by students during the study, it is likely that the models of each student in the group were still in the early stages of development, and thus, less informative to the student. Also, the students who participated in the evaluation were enrolled in a database course which required them to learn ER modelling, so it would not be unreasonable to expect students to be primarily focused on solving problems during the short time they used the system.

Students were not told explicitly to explore their model, yet around half of the students who used E-KERMIT did open the main progress window at least once and inspect various parts of their knowledge. This result is not proof that students actually spent time examining their progress, since they may have just been exploring the interface. However, responses from the questionnaire do provide an indication that students did examine their model and reflect on their

progress and domain weaknesses to some extent, although the majority of students said that they consulted mostly the progress summary in the main interface.

This study supports other findings [Mit01b, WSF99] that system interaction benefits the less able students more than the more able ones. In this study the performance of the more able group in both the control and experimental groups on average did not improve after system interaction. It is interesting to note that even though the more able students did not improve on the post test, on a scale of 1–5, the amount that these students thought they learnt, on average, was 3.1, which was the same for the less able group, who, in fact, had improved after system interaction.

Chapter 5

Conclusions and Further Work

Intelligent tutoring systems that support open student modelling are claimed to enhance the learning process because they can promote reflective activity in the student. However, evaluation of these systems with respect to how the open model affects learning is currently lacking. This research has focused on how even a simple open student model can affect learning. For this purpose we developed E-KERMIT (Enhanced KERMIT), in which the student model is open for inspection — an extension of KERMIT, the ER modelling intelligent tutor.

The student model in E-KERMIT is visualised graphically through a hierarchical tree structure which conveys the structure of the ER modelling domain. The hierarchical structure is flexible in that it allows the student to concentrate on specific concepts of the domain, and/or collective higher level domain concepts. Help in the form of a tutorial is provided to assist students in understanding their model, as students cannot be expected to reflect on their model if they cannot understand it.

An evaluation study was conducted with students enrolled in the second year database design course in the Computer Science Department at the University of Canterbury to compare students' learning with KERMIT and E-KERMIT. The study focused on how the open student model in E-KERMIT affected learning, whether students would inspect their model, and students' subjective opinion of their model. The subjective results were generally encouraging, indicating that the majority of students who inspected their model used it to reflect on weaknesses in their ER knowledge, and felt that the model assisted their learning. However, the study failed to demonstrate any statistically significant improvement in learning as a result of the open student model, although, given the short time students used the systems, such an improvement could not be expected.

Empirical evaluation is an important aspect in the design of any ITS. This work is a first step in evaluating a simple open student model with respect to its effect on learning. However, further work is needed to draw any reliable objective conclusions as to whether such a simple open model can contribute to a more effective learning environment. Possibilities also exist to make E-KERMIT more interactive, whereby the system may facilitate discussion and/or negotiation of the student's knowledge state. This may proceed incrementally with an empirical evaluation performed at each stage. If positive objective conclusions can be drawn, a lower bound on the degree of interactivity required to achieve effective learning as a result of open modelling can be determined and used to focus the development of future intelligent tutors.

Acknowledgements

I would like to thank Dr Antonija Mitrovic for all her support and guidance with this research. I thank Pramudi Suraweera for assisting in the understanding of KERMIT and in helping with the evaluation. Thanks must also go to Jane McKenzie for proof reading this report.

Bibliography

[BBP95] Susan Bull, Paul Brna, and Helen Pain. Extending the Scope of the Student Model. In *User Modeling and User Adapted Interaction*, 1995.

[BM99] Julia Case Bradley and Anita C Millspaugh. *Programming in Visual Basic 6.0*. Irwin/McGraw-Hill, 1999.

[BP95] Susan Bull and Helen Pain. "Did I say what I think I said, and do you agree with me?": Inspecting and Questioning the Student Model. In *Proceedings of World Conference on Artificial Intelligence in Education*, 1995.

[Bul97] Susan Bull. See Yourself Write: A Simple Student Model to Make Students Think. In *User Modeling: Proceedings of the Sixth International Conference, UM97*, Vienna, New York, 1997.

[Bul98] Susan Bull. 'Do It Yourself' Student Models for Collaborative Student Modelling and Peer Interaction. In *Lecture Notes in Computer Science, ITS '98*, August 1998.

[DDN99] H M Deitel, P J Deitel, and T R Nieto. *Visual Basic 6 How to Program*. Prentice-Hall Inc., 1999.

[DR94] Batra D and Antony S R. Novice Errors in Conceptual Database Design. *European Journal of Information Systems*, pages 57–69, 1994.

[DSB00] Vania Dimitrova, John Self, and Paul Brna. Involving the Learner in Diagnosis - Potentials and Problems. Tutorial at Web Information Technologies: Research, Education and Commerce, Montpellier, France, May 2000.

[DSB01] Vania Dimitrova, John Self, and Paul Brna. Applying Interactive Open Learner Models to Learning Technical Terminology. In *UM-2001*, pages 148–157, 2001.

[ELM] `ELM-ART, Episodic learner model - The adaptive remote tutor`. Available at `www.psychologie.uni-trier.de:8000/elmart`

[EN94] Ramez Elmasri and Shamkant B Navathe. *Fundamentals of Database Systems*. The Benjamin/Cummings Publishing Company, Inc., 1994.

[Kay] Judy Kay. The um toolkit for cooperative user modelling. Early draft of paper to appear in UMUAI volume 4. `www.cs.usyd.edu.au/~judy/Research_um/um.ps`

[Mit01a] Antonija Mitrovic. COSC420 lecture notes: Cognitive modeling and Intelligent Tutoring Systems. Computer Science Department, University of Canterbury, 2001.

[Mit01b] Antonija Mitrovic. Investigating students' self-assessment skills. In Springer-Verlag LNAI 2109, editor, *UM-2001*, pages 247–250, 2001.

[MMM02] Antonija Mitrovic, Brent Martin, and Michael Mayo. Using Evaluation to Shape ITS Design: Results and Experiences wiht SQL-Tutor. *UMUAI*, 12, 2002.

[MMSM01] Antonija Mitrovic, Michael Mayo, Pramuditha Suraweera, and Brent Martin. Constraint-Based Tutors: a Success Story. In *IEA/AIE*, 2001.

[PEG96] Brusilovsky P, Schwarz E, and Weber G. ELM-ART: An Intelligent Tutoring System on World Wide Web. In *Intelligent Tutoring Systems (Lecture Notes in Computer Science)*, volume 1086, pages 261–269, 1996.

[S94] Ohlsson S. Constraint-based Student Modeling. In J.E Greer and G.I McCalla, editor, *Student Modeling: the key to Individualized Knowledge-based Instruction*, pages 167–189, 1994.

[Sur99] Pramuditha Suraweera. An Animated Pedagogical Agent for SQL-Tutor. Honours report, Computer Science Department, University of Canterbury, 1999.

[Sur01] Pramuditha Suraweera. KERMIT: A Knowledge-based Entity Relationship Modelling Inteligent Tutor. In *New Zealand computer science research students' conference (nzcsrsc)*, 2001.

[WSF99] Barbara Y White, Todd A Shimoda, and John R Frederiksen. Enabling Students to Construct Theories of Collaborative Inquiry and Reflective Learning: Computer Support for Metacognitive Development. *International Journal of Artificial Intelligence in Education*, 10:151–182, 1999.

[ZR01] Juan-Diego Zapata-Rivera. Supporting Negotiated Assessment Using Open Student Models. In *UM 2001*, pages 295–297, 2001.

Appendix A

Pre- and Post-Tests

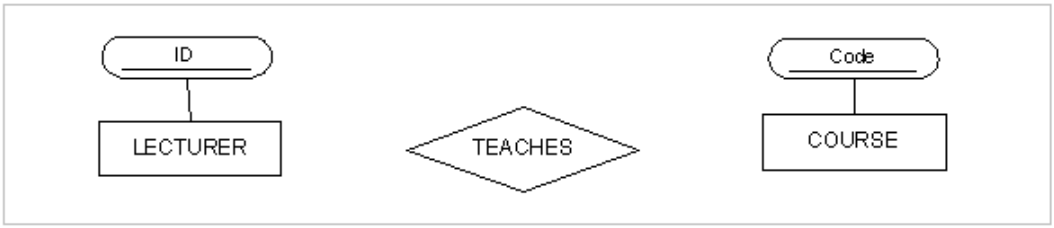
KERMIT

Test A

Please specify your computer name: PC

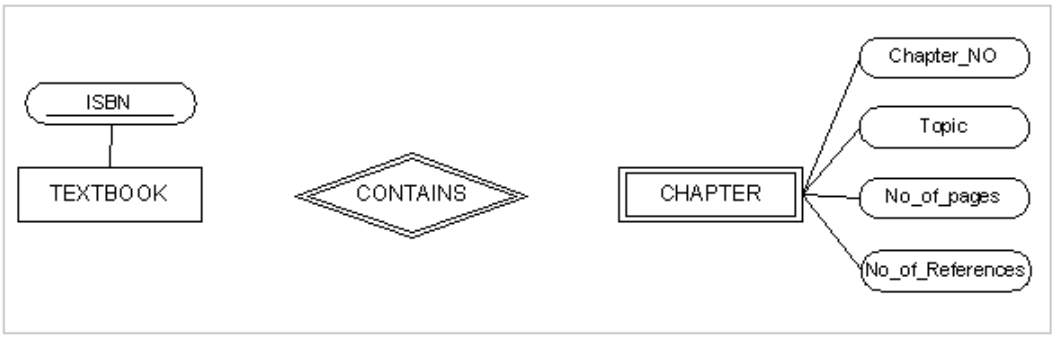
Answer all questions.

1. The following ER model represents the relationship between lecturers and courses. Please specify the cardinality and participation constraints.



2. The following partially completed ER model, represents the scenario described below. Please complete the ER model.

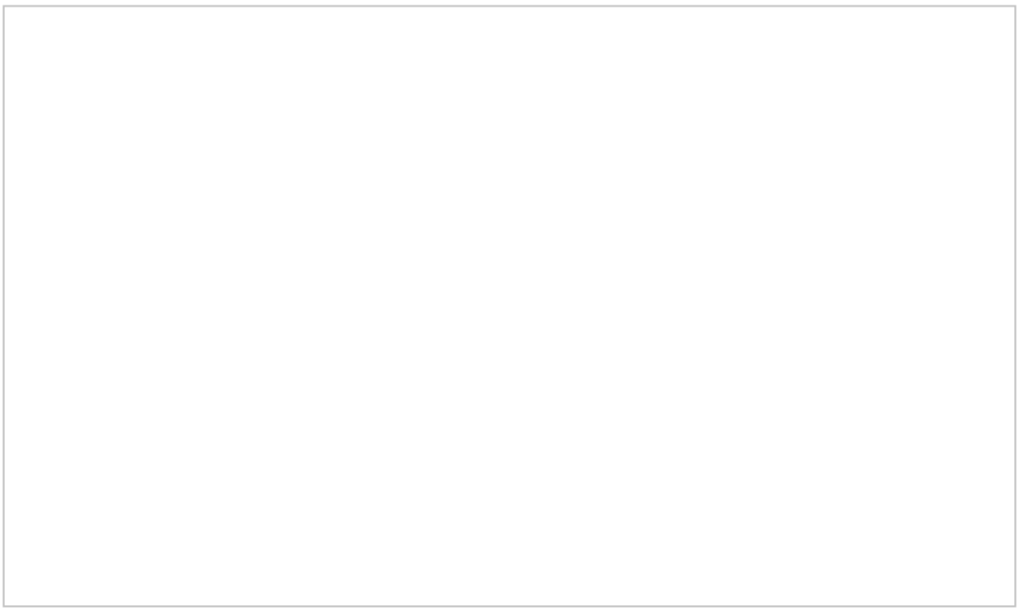
   - A textbook, with a unique ISBN, contains a number of chapters. For each chapter we know its chapter number, topic, total number of pages and total number of references. Different textbooks may cover the same topics.



(P.T.O)

3. Please draw an ER diagram that captures the information given below.

- For each course a student has taken, we need to know the final grade. Each course has a unique course code and a student has his/her student id.



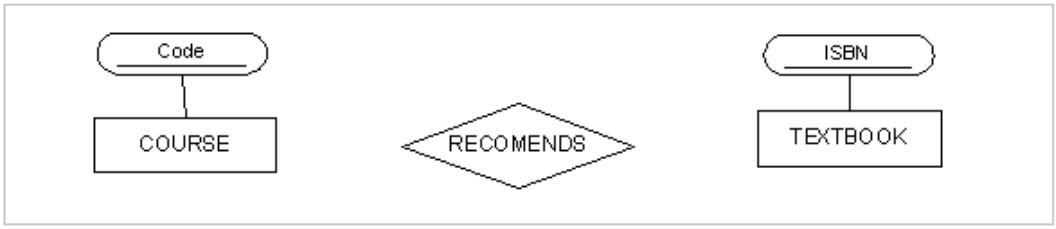
KERMIT

Test B

Please specify your computer name: PC

Answer all questions.

1. The ER diagram below represents the relationship between courses and recommended textbooks. Please specify the participation and cardinality constraints.



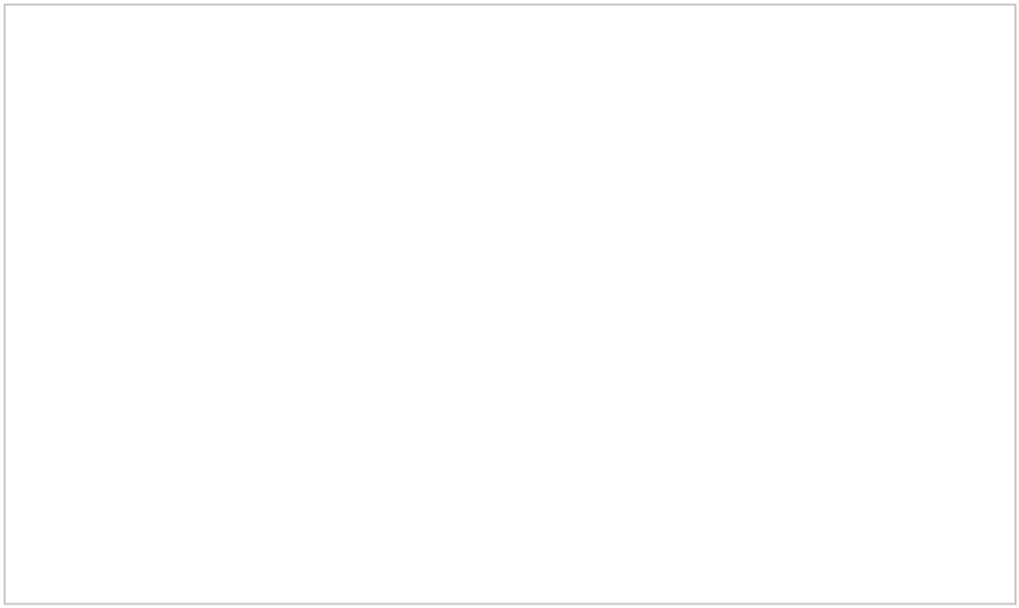
2. The ER diagram below is a partially completed ER model that represents the information given below. Please complete the ER diagram.

- Each course, with a unique course code, consists of several sections. For each section, we know the topic it covers, and the number of lectures and labs it contains. The same topic can be covered in several courses, but the number of lectures and labs will differ in that situation.

(P.T.O)

3. Please draw an ER diagram that models the requirements given below.

- Sometimes students work in groups. Each group has a unique number and students have their student ids. A student may have different roles in various groups he/she belongs to.



Appendix B

KERMIT Test Results

Kermit	machine	pre-test	post-test	gain
$<$ mean	pc246	15	17	2
$<$ mean	pc248	15	16	1
$<$ mean	pc251	14	17	3
$<$ mean	pc255	15	18	3
$<$ mean	pc256	15	18	3
$<$ mean	pc264	16	17	1
$<$ mean	pc273	14	16	2
$<$ mean	pcm214	16	20	4
$<$ mean	pcm249	15	19	4
$<$ mean	pcm262	13	18	5
$<$ mean	pcm263	14	16	2
$<$ mean	pcm267	13	16	3
$<$ mean	pcm268	15	16	1
$<$ mean	pcm270	16	16	0
$<$ mean	pcm278	16	17	1
$>$ mean	pc195	20	18	-2
$>$ mean	pc253	17	19	2
$>$ mean	pc257	18	18	0
$>$ mean	pc26	18	18	0
$>$ mean	pc269	18	19	1
$>$ mean	pc271	18	19	1
$>$ mean	pcm188	18	21	3
$>$ mean	pcm266	17	19	2
$>$ mean	pcm272	18	16	-2
$>$ mean	pcm276	18	18	0
$>$ mean	pcm279	17	20	3

Appendix C

E-KERMIT Test Results

E-Kermit	machine	pre-test	post-test	gain
< mean	183	15	17	2
< mean	185	12	13	1
< mean	190	12	16	4
< mean	192	11	11	0
< mean	195	13	15	2
< mean	196	16	17	1
< mean	204	15	18	3
< mean	208	16	18	2
< mean	210	15	17	2
< mean	214	16	17	1
< mean	248	16	15	-1
< mean	249	16	15	-1
< mean	255	14	19	5
< mean	258	9	15	6
< mean	261	16	16	0
< mean	262	15	16	1
< mean	263	15	16	1
< mean	264	15	12	-3
< mean	267	16	21	5
< mean	270	15	19	4
< mean	278	15	14	-1
< mean	279	14	18	4
> mean	171	17	18	1
> mean	184	17	16	-1
> mean	188	19	15	-4
> mean	202	19	20	1
> mean	250	21	18	-3
> mean	251	18	19	1
> mean	256	17	19	2
> mean	259	19	20	1
> mean	260	19	18	-1
> mean	268	20	21	1
> mean	269	19	20	1
> mean	271	18	17	-1
> mean	273	17	17	0
> mean	275	19	18	-1
> mean	282	18	16	-2
> mean	282	20	20	0
> mean	285	17	17	0
> mean		18	21	3

Appendix D

KERMIT Questionnaire

Questionnaire

KERMIT : Knowledge-based Entity Relationship Modelling Intelligent Tutor

Thank you for using KERMIT. Your feedback will be crucial for further improvements of the system and we would be most grateful if you could take time to fill this questionnaire. The questionnaire is anonymous, and you will not be identified as an informant. You may at any time withdraw your participation, including withdrawal of any information you have provided. By completing this questionnaire, however, it will be understood that you have consented to participate in the project and that you consent to publication of the results of the project with the understanding that anonymity will be preserved.

Please specify the name you used to log on to the system:

1. What is your previous experience with ER modelling? (please circle one)

(a)	Only lectures
(b)	Lectures plus some work
(c)	Extensive use

2. How much time did you need to learn about the system's functions? (please circle one)

(a)	Substantial time (most of the session
(b)	30 minutes
(c)	10 minutes
(d)	Less than 5 minutes

3. How much did you learn about ER modelling from using the system? (please circle one)

Nothing				Very much
1	2	3	4	5

Please comment

4. Did you enjoy leaning with KERMIT? (please circle one)

Not at all				Very much
1	2	3	4	5

Please comment

5. Would you recommend KERMIT to other students? (please circle one)

(a)	Yes
(b)	Don't know
(c)	No

6. Did you find the interface easy to use? (please circle one)

Not at all				Very much
1	2	3	4	5

Please comment

7. Did you find the feedback from KERMIT useful? (please circle one)

Not at all				Very much
1	2	3	4	5

Please comment

8. Would you prefer more details in feedback? (please circle one)

(a)	Yes
(b)	Don't know
(c)	No

Please comment

9. Imagine that KERMIT could give a visual indication of your progress. That is, the system could indicate your current ability to complete various tasks involved in designing an ER model. An example of such a task may be identifying entities from the problem. Would you like to see such a feature in the system?

(a)	Yes
(b)	Don't know
(c)	No

10. Did you encounter any software problems or system crashes? (please circle one)

(a)	Yes
(b)	No

If yes, please specify which

11. What did you like in particular about KERMIT?

12. Is there anything you found frustrating about the system?

13. Do you have any suggestions for improving KERMIT?

Appendix E

E-KERMIT Questionnaire

Questionnaire

KERMIT : Knowledge-based Entity Relationship Modelling Intelligent Tutor

Thank you for using KERMIT. Your feedback will be crucial for further improvements of the system and we would be most grateful if you could take time to fill this questionnaire. The questionnaire is anonymous, and you will not be identified as an informant. You may at any time withdraw your participation, including withdrawal of any information you have provided. By completing this questionnaire, however, it will be understood that you have consented to participate in the project and that you consent to publication of the results of the project with the understanding that anonymity will be preserved.

Please specify the name you used to log on to the system:

1. What is your previous experience with ER modelling? (please circle one)

(a)	Only lectures
(b)	Lectures plus some work
(c)	Extensive use

2. How much time did you need to learn about the system's functions? (please circle one)

(a)	Substantial time (most of the session
(b)	30 minutes
(c)	10 minutes
(d)	Less than 5 minutes

3. How much did you learn about ER modelling from using the system? (please circle one)

Nothing				Very much
1	2	3	4	5

Please comment

4. Did you enjoy leaning with KERMIT? (please circle one)

Not at all				Very much
1	2	3	4	5

Please comment

5. Would you recommend KERMIT to other students? (please circle one)

(a)	Yes
(b)	Don't know
(c)	No

6. How often did you examine your progress? (please circle one)

Never				Very often
1	2	3	4	5

If you did examine your progress which view did you use the most? (please circle one)

(a)	Summary view (in main interface)
(b)	Full view (by clicking the **Show Me More** button)
(c)	Used both about the same

7. Was the way your progress represented easy to understand? (please circle one)

(a)	Yes
(b)	Only after explanation
(c)	No
(d)	I didn't examine my progress

Please comment

8. Did you find the progress views useful? (please circle one)

(a)	Yes
(b)	Don't know
(c)	No
(d)	I didn't use them

Please comment

9. Did you examine your progress to help you identify any weaknesses in your ER knowledge? (please circle one)

(a)	Yes
(b)	No

Please comment

10. Do you feel that being able to examine your progress assisted your learning? (please circle one)

(a)	Yes
(b)	Don't know
(c)	No

Please comment

11. Do you feel KERMIT correctly reflected your current ability to solve ER design problems? (please circle one)

(a)	Yes
(b)	Don't know
(c)	No

Please comment

12. Did you encounter any software problems or system crashes? (please circle one)

(a)	Yes
(b)	No

If yes, please specify which

13. What did you like in particular about KERMIT?

14. Is there anything you found frustrating about the system?

15. Do you have any suggestions for improving KERMIT?

Appendix F

KERMIT Questionnaire Results

KERMIT	ID	Prev. Experience	Time to learn interface	Amount learnt	Enjoyment	Recommend to others	Ease of using interface	Usefulness of feedback	More detailed feedback	Need progess visualisation?
< mean	pc246	only lect.	10	3	4	yes	4	4	yes	yes
< mean	pc248	lect.+work	60	3	3	yes	3	4	no	yes
> mean	pc253	only lect.	10	3	3	yes	2	4	yes	don't know
> mean	pc255	lect.+work	5	3	4	yes	4	4	no	yes
< mean	pc256	only lect.	10	4	4	yes	3	5	yes	yes
> mean	pc257	only lect.	5	3	4	yes	3	2	yes	yes
> mean	pc26	only lect.	10	2	5	don't know	3	5	don't know	don't know
< mean	pc264	only lect.	5	4	3	don't know	3	2	don't know	don't know
> mean	pc269	lect.+work	5	3	3	yes	3	4	no	don't know
> mean	pc271	only lect.	10	4	3	yes	4	3	don't know	yes
< mean	pc273	only lect.	5	4	3	yes	5	5	yes	yes
> mean	pcm188	only lect.	5	3	3	a	2	2	yes	don't know
> mean	pcm195	lect.+work	5	3	4	yes	4	3	yes	don't know
< mean	pcm214	only lect.	5	4	4	yes	4	4	no	yes
< mean	pcm249	lect.+work	30	4	3	yes	3	4	no	yes
< mean	pcm251	only lect.	5	3	5	yes	4	2	don't know	don't know
< mean	pcm262	only lect.	10	3	3	yes	3	4	yes	don't know
< mean	pcm263	only lect.	10	3	5	yes	4	3	yes	yes
> mean	pcm266	only lect.	30	3	2	yes	3	3	yes	yes
< mean	pcm268	only lect.	10	2	3	yes	4	3	yes	yes
> mean	pcm269	only lect.	10	2	3	yes	2	3	no	yes
< mean	pcm270	only lect.	5	3	3	yes	3	5	yes	don't know
> mean	pcm272	only lect.	30	2	2	no	3	4	yes	yes
> mean	pcm276		5	4	3	yes	1	3	no	yes
> mean	pcm279	only lect.	5	4	4	yes	4	5	don't know	yes
> mean	pcm278	only lect.	10	4	3	yes	4	4	yes	yes
	pcm184	only lect.	10	2	5	no	2	1	yes	no
	pc259	lect.+work	5	3	1	no	1	4	yes	yes
	pc260	lect.+work	10	3	4	yes	3	3	yes	yes
	pcm275	lect.+work	5	4	4	yes	4	3	yes	yes
	pc265	lect.+work	10	3	4	yes	3	2	yes	don't know

Appendix G

E-KERMIT Questionnaire Results

E-KERMIT		experience	learning time	amount learnt	enjoyed E-Kermit	how often progress examined	view used most	progress understandable	progress useful	examine to identify weaknesses	progress assisted learning
< mean	pcm263	lect.+work	5	5	5	4	full view	yes	yes	no	yes
< mean	pc264	only lect.	30	3	4	3	both same	yes	yes	yes	yes
< mean	pc249	lect.+work	10	2	3	3	both same	no	didn't use	no	don't know
< mean	pc258	only lect.	10	3	3	4	summary	no	don't know	yes	yes
< mean	pcm248	lect.+work	10	5	5	4	both same	after explan.	yes	yes	yes
< mean	pc270	only lect.	5	4	4	4	summary	yes	yes	no	yes
< mean	pc279	lect.+work	5	3	5	3	summary	yes	yes	yes	yes
< mean	pc267	only lect.	5	3		4	summary	no	didn't use	no	no
< mean	pcm262	lect.+work	10	3	3	2	summary	yes	yes	yes	don't know
< mean	pc278	only lect.	60	3	3	4	summary	yes	don't know	no	yes
< mean	pc210	lect.+work	10	4	3	1		didn't use	didn't use	no	don't know
< mean	pcm214	only lect.	5	3	3	3	full view	yes	yes	yes	yes
< mean	pc185	lect.+work	5	2	1	3	summary	after explan.	didn't use	no	yes
< mean	pc208	lect.+work	5	3	3	4	summary	yes	yes	yes	yes
< mean	pcm183	lect.+work	10	3	3	3	summary	yes	yes	yes	yes
< mean	pc195	lect.+work	10	2	3	3	summary	after explan.	no	yes	yes
< mean	pc190	only lect.	30	3	3	4	summary	after explan.	didn't use	no	don't know
< mean	pcm196	lect.+work	10	2	2	3	summary	yes	yes	yes	yes
< mean	pcm204	lect.+work	10	3	5	3	summary	yes	yes	yes	yes
> mean	pc275	lect.+work	5	3	5	3	summary	no	don't know	yes	yes

E-KERMIT		experience	learning time	amount learnt	enjoyed E-Kermit	how often progress examined	view used most	progress understandable	progress useful	examine to identify weaknesses	progress assisted learning > mean
	pc282	lect.+work	10	4	5	4	summary	yes	yes	yes	yes
> mean	pcm251	lect.+work	30	3	5	2	both same	no	didn't use	yes	yes
> mean	pcm250	only lect.	10	3	4	3	summary	yes	don't know	yes	yes
> mean	pc273	only lect.	60	5	5	5	summary	after explan.	yes	yes	yes
> mean	pcm271	lect.+work	10	3	4	2	summary	yes	yes	yes	yes
> mean	pc269	lect.+work	10	2	3	4	summary	didn't use	didn't use	no	don't know
> mean	pc268	lect.+work	60	3	2	4	both same	yes	no	yes	yes
> mean	pcm259	only lect.	5	3	4	5	both same	after explan.	yes	yes	don't know
> mean	pc282	only lect.	30	3	4	2	summary	no		no	don't know
> mean	pcm260	only lect.	5	4	4	2	summary	yes	yes	no	yes
> mean	pcm184	lect.+work	10	3	2	1		didn't use	didn't use	no	no
> mean	pcm188	lect.+work	10	3	4	3	summary	yes	yes	no	yes
> mean	pcm171	lect.+work	5	3	3	2	summary	no	no	no	don't know
> mean	pc285	lect.+work	10	2	3	1		didn't use	didn't use	no	don't know
> mean	pcm202	only lect.	10	3	3	3	summary	yes	don't know	no	yes
	?	lect.+work	5	3	4	3	summary	after explan.	yes	yes	yes
	?	only lect.	10	4	3	2	summary	no	didn't use	no	no
	?	lect.+work	5	2	3	2	summary	no	no	no	no
	pc	only lect.	10	3	5	2	both same	after explan.	yes	yes	yes
	pcm187	lect.+work	5	3	3	3	both same	after explan.	yes	yes	yes