Feedback Micro-engineering in EER-Tutor

BSc. Honours Research Report

Konstantin Zakharov, Dr. Antonija Mitrović (supervisor)

Department of Computer Science & Software Engineering
University of Canterbury
November 15, 2004

Abstract

Effectiveness of one-to-one tutoring is the key motivating factor supporting the development of Intelligent Tutoring Systems. These systems are capable of adjusting learning support to the individual needs of their users. This study presents EER-Tutor, an Intelligent Tutoring System for teaching Enhanced Entity-relationship modelling. We use EER-Tutor for testing a hypothesis, that feedback messages designed with the theory of learning from performance errors in mind are more effective than conventional feedback messages. The principles of the Cognitive Architecture suggest that the proposed feedback design should provide long and short-term learning advantages through revision of faulty knowledge in the context of learners' errors.

The overall outcome of this evaluation study is promising. However, a comparison of the experimental and control groups did not reveal a significant statistical difference between the proposed format and conventional format of feedback. At the same time, the analysis based on the Power Law of Learning displayed a consistent advantage of the experimental feedback. The lack of conclusive results suggests the need of further research in this area.

Contents

1 Introduction

Instruction and learning are the very foundation supporting the development of today's society. Education is the central focus of newly established *knowledge society* policies promoted by governments around the world. It is virtually impossible to overestimate the importance of education in the current social environment. However, serious questions revolve around the effectiveness of today's education system. Some theorists go so far as to argue that there a crisis in education, caused by the extremely high student/teacher ratios in most educational institutions [5, 26].

Educational research results published over two decades ago [7] single out one-to-one tutoring as the most effective model of instruction. It has been shown that under one-to-one tutoring conditions the average student performed about two standard deviations above the mean of a conventional class with 30 students per teacher. Bloom calls this phenomenon *the 2 Sigma problem*.

These results suggest that most students have the potential of achieving high performance in learning. However, one-to-one tutoring is rarely available on a wide scale and there is no easy way of improving student/teacher ratios. Individual tutoring is simply not a plausible option from either the economic or practical perspectives. Bloom comments: "... an important task of research and instruction is to seek ways of accomplishing this [high level of learning] under more realistic conditions than the one-to-one tutoring, which is too costly for most societies to bear on a large scale."

Learners vary in their amount of prior knowledge, motivation, learning style, natural pace and working memory capacity. Consequently, a uniform predefined instructional sequence cannot provide the optimal learning environment for all learners [15]. Success of one-to-one tutoring is based on the ability of human tutors to adjust their feedback based on their interaction with the learner.

Research in applications of Artificial Intelligence (AI) in Education offers alternative ways of bridging the 2 Sigma gap with the help of computer-based instructional environments. The purpose of incorporating AI into instructional environments is to make them responsive to the individual learner. However, building such environments is not easy. The difficulties arise in the computational complexity of the task.

Although the use of computers in Education began in the 1950s with Computer-Aided Instruction (CAI) systems, the statically defined content and order of instruction in these systems did not offer any significant advantage over a self-directed study performed by reading a text book. In the 1960s a new generation of CAI systems appeared, often referred to as *branching programs*. Branching was based on comparison of student's answers to predefined solutions. A lack of adaptiveness and domain expertise did not allow these systems to provide individualised learning support at the required level.

The first attempts of making CAI programmes *intelligent* started in the 1970s. Today these systems are known as Intelligent Tutoring Systems (ITSs). They shift the focus from knowledge transfer to knowledge construction by actively adapting the instructional process to the needs of an individual student. There has been a number of ITSs developed for well-defined domains such as Algebra, Geometry and Programming Languages [3]. ITSs are known to improve learning performance by 0.3–1.0 standard deviations. For example, Lisp-Tutor, an ITS for teaching Lisp programming language, claims improvement of 1 standard deviation [39]. In addition to this, students who use the tutor take 30% less time to master the domain. SQL-Tutor, an ITS for teaching Structured Query Language (SQL) for databases, improves performance by 0.65 standard deviations in just two hours of interaction with the system [26]. Atlas, a tutoring system

for teaching Physics, improves performance by 0.9 standard deviations [11]. Between 20 and 25 hours of interaction with SHERLOCK, a tutor for technical troubleshooting in avionics, is equivalent to four years of on-the-job experience [19].

The area of ITSs is an interdisciplinary research field that draws on knowledge from Artificial Intelligence, Computer Science, Education, Human-Computer Interaction, Linguistics, Statistics and Cognitive Psychology, the latter of which possibly carries the pivotal significance in that ITSs are designed to support learning, which is the process taking place in the human mind. The central component of Cognitive Psychology is the Cognitive Architecture, which focuses on the organisation and workings of a human mind. The goal of ITSs research is to design and implement educational environments capable of delivering individualised instruction comparable to one-to-one tutoring. While the development of ITSs opens new attractive horizons, more in-depth research is required to improve the educational effectiveness of ITSs by solidifying existing theories and putting them into practice.

1.1 Research Objectives

The goal of this study is to evaluate the effectiveness of feedback constructed strictly according to the principles of the Cognitive Architecture and theory of learning from performance errors [32]. We expect that the proposed method of delivering feedback will improve both short-term and long-term performance of learners. Based on of the recently proposed successful approaches of ITSs design, Constraint-Based Modelling (CBM) [27], we have developed EER-Tutor, an ITS that supports learning and practising principles of conceptual database modelling, in particular Enhanced Entity-relationship (EER) modelling [9]. EER design is a complex knowledge domain that allows for creativity and variability in the solution space. Although EER modelling does produce an outcome defined in abstract terms, there is no single correct procedure for obtaining that outcome. Consequently, EER modelling presents a considerable learning challenge. The specifics of EER modelling make it difficult to identify errors at the early stages of learning. In many cases, errors made during the conceptual design stage may remain hidden until the later stages of a database life cycle. We use EER modelling as the domain in which we conduct our research.

1.2 Report Structure

Chapter 2 presents the key background concepts of research in Cognitive Architecture and theory of learning from performance errors. We next introduce ITSs and CBM. Further in this Chapter we provide a description of two systems for Entity-Relationship (ER) modelling, Kermit and ER-Tutor, developed by the Intelligent Tutoring Systems Group[1] (ICTG). The EER-Tutor, our experimental system has been developed on the basis of ER-Tutor.

A detailed description of the design of EER-Tutor is included in Chapter 3, documenting the changes and extensions made to the base system, ER-Tutor. The Chapter outlines the design and programming tasks we completed in order to prepare EER-Tutor for the current experiment. The Chapter also contains a detailed description of EER-Tutor feedback design, which is the central focus of our study.

Chapter 4 describes the evaluation of EER-Tutor and the experimental results. Beginning with the experiment design, we go on to describe the analysis of performance measures. Further we discuss the observed results in the context of our study. Finally, Chapter 5 presents the conclusion of this study and attempts to explain the outcomes, while making suggestions for future work on EER-Tutor.

[1] `http://www.cosc.canterbury.ac.nz/~tanja/ictg.html` — Intelligent Computer Tutoring Group

2 Background and Prior Research

This chapter contains an overview of the key concepts supporting this research. Section 2.1 is dedicated to the Cognitive Architecture. Section 2.2 introduces ITSs along with the core concept behind them, Student Modelling. We briefly present various approaches to Student Modelling. Section 2.3 gives special attention to CBM, which is the most promising approach to Student Modelling. Finally, Section 2.4 is dedicated to the two predecessors of EER-Tutor, Kermit and ER-Tutor.

2.1 Cognitive Architecture

Theories of the Cognitive Architecture can be roughly divided into two categories: those motivated by the digital computer architecture and those based on an associative architecture. The first category is commonly referred to as Symbolic Cognitive Architecture [40]. The theory behind this approach claims that the mind performs cognitive tasks by computing. Thus cognition is modelled as a dynamic unfolding of computational processes.

The second type of the Cognitive Architecture is referred to as Connectionist Cognitive Architecture [40]. This approach describes the mechanisms of human cognition through the use of simulated networks of simple, neuron-like processing units. Connectionist models are most often applied to what might be called natural cognitive tasks, involving perception of the world of objects and events and interpreting them for the purpose of organised behaviour.

We base our research on the Symbolic Cognitive Architecture, and we refer to it as Cognitive Architecture further in the report. The theory suggests that the model of the Cognitive Architecture includes a few types of memory, namely working memory, long-term memory and rule memory (see Figure 2.1) [40, 2, 33]. Sensory organs and motor capabilities are also a part of the Cognitive Architecture, although these components are de-emphasised. Long-term memory stores declarative knowledge represented as *chunks*. Declarative knowledge is most commonly obtained by reading or other similar processes. Rule memory (or procedural memory) stores a set of procedures, also called *production rules* for historical reasons. Procedural knowledge is obtained by practising.

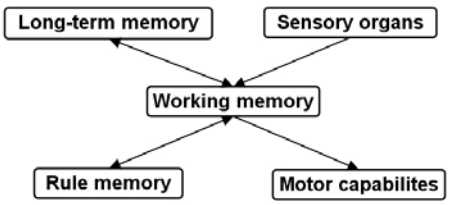


Figure 2.1: Cognitive Architecture

Assumptions, forming the basis of the Cognitive Architecture, suggest that human action (a) is sequential, i.e. typically consists of a coordinated sequence of smaller actions, (b) serves some goal on the part of the actor, and (c) is adapted to the situation in which the action occurs. These assumptions joined together form three-way associations, formally referred to as production rules. They can be written in the following general format:

$$R\text{: } Goal, Situation \rightarrow Action$$

The Cognitive Architecture puts forward two assumptions about how production rules are executed. First, there is only a single goal current at any moment in time. Second, the mind "remembers" which goal preceded the current goal by storing it on the *goal stack*, so that when a current goal is completed, the previous goal can be reinstated in the active state.

The hypothesis underlying the cognitive architecture suggests that a human mind (or brain) is constantly identifying the rules relevant to the current goal and situation, selecting which rule to execute. The cognitive architecture operates in repeated cycles:

1. Evaluate: match all rules
2. Select rule: conflict resolution
3. Execute: fire the selected rule
4. Repeat the cycle

During the first step, the *situation* components of all rules are matched against the current situation and the *goal* components are matched against the current goal. During the conflict resolution step, when more then one rule is relevant for the current goal and situation, declarative memory determines the order of precedence of production rules. After execution, the cycle is repeated again with an updated current situation and goal.

This architecture represents the innate structure of the mind and, in contrast to the rule set, does not undergo modification. However, the production rules become more specialised [33, 32]; this explains the performance improvement acquired through practice. The ACT-R theory of cognition [3] is based on the Cognitive Architecture. The fundamental assumption of the ACT-R is that cognitive skills are realised by production rules. In order to support students learning a specific task, that is, to learn a specific set of production rules that will enable students to perform the tasks correctly, cognitive tutors teach the underlying production rules.

2.2 Intelligent Tutoring Systems and Student Modelling

Development of an ITS capable of accommodating individual students' needs is a complicated task. Such a system must be capable of dynamically adapting to and monitoring each student. The key step in the AI approach to individualised instruction is to equip the instructional system with models of both the target knowledge domain and the learner [31, 15, 4]. The response of the system at each moment in time is computed on the basis of the student model and the expert module. This feature makes ITSs stand above other types of computer-based instructional systems.

Student modelling is the process of representing the knowledge state of a student through the gathering of relevant information about that student. To model individuals, ITSs usually maintain models of each student representing at least their performance in the domain. There have been many approaches to student modelling including Overlay models, Perturbation models, models based on Machine Learning, and more recently, Constraint-based models [31, 15, 24, 25].

Overlay models assume that the domain model contains all concepts the student must learn, and that the student knows a subset of this information. The task of teaching is therefore seen as filling in the gaps in the students' knowledge until they have learnt sufficient domain knowledge to achieve mastery.

Perturbation models recognise that the student may harbour misconceptions, or *buggy knowledge*, which also must be represented [24, 31]. Building a Perturbation model usually requires the underlying domain model to include the mistakes students are likely to make, or *bug libraries* so that they can be identified in individual student' behaviour. However, the task of composing bug libraries is a major

undertaking. Studies have shown that bug libraries do not transfer well to new population of students; if a bug library is developed for a certain group of students, it may not cover the mistakes made by another group of students [34].

Machine Learning models try to dispense with bug libraries and generate student model on-line by searching the space of possible models with the help of Machine Learning algorithms such as ID3 and PRISM [13].

The common challenge in these approaches is how to obtain and 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 [35]. Attempting to model what a student knows correctly is insufficient, yet attempting to model what a student knows incorrectly is too complicated because of the huge search spaces involved.

To overcome the intractability of student modelling Self [35] recommends: "Avoid guessing. Have the student supply information needed by the system. This reduces the system requirements and the likelihood of making decisions based on incorrect assumptions." In other words, the design of the system should be such that the information necessary for building a student model is provided by the student, and not inferred by the system. In the same context, Self suggests: "Don't diagnose what you can't treat. Rather then trying to model everything you can about the student, decide what pedagogical actions you wish to take and build the student model to support it." This means that an ITS should model only what it is capable of using in order to generate remedial or other pedagogical actions.

These recommendations imply that a student model may be useful even if it is not complete and accurate. This is supported by the fact that human teachers are highly effective in imparting knowledge, and yet they use only very loose models of their students [31]. Based on this observation, the CBM approach discussed below, focuses on reducing the complexity of the student modelling task.

2.3 Constraint-Based Modelling

CBM is a technique proposed by Ohlsson [31] as a way of overcoming the intractable nature of Student Modelling. CBM arises from Ohlsson's theory of learning from performance errors [32] and is based on the Cognitive Architecture. A Constraint-based model represents domain knowledge as a set of explicit constraints on correct solutions in that domain [27]. At the same time, constraints implicitly represent all incorrect solutions. In this way, constraints partition all possible solutions into correct and incorrect ones. Unlike cognitive tutors, CBM does not require extensive studies of student bugs for compilation of bug libraries, which is an important trade-off [31].

Each constraint specifies a property of the domain that is shared by all correct solutions. A constraint is an ordered pair (C_r, C_s), where C_r is the relevance condition determining a problem state in which the constraint is relevant, and C_s identifies the state in which the constraint is satisfied. If a constraint is relevant in some state then it must also be satisfied in order for the solution to be correct, otherwise the solution contains an error. Thus, the semantics of a constraint are: *if the C_r condition is true, then the C_s must also be true, otherwise something has gone wrong*. The following example is taken from a well-known problem, the Towers of Hanoi[1]:

$$C_r = <\textit{If disk X is on peg Z and disk Y is on peg Z and X is on top of Y}>,$$
$$C_s = <\textit{then X is smaller than Y}> (\textit{or else there is an error})$$

In this example, the relevance condition, C_r, is the complex clause *disk X is on peg Z and disk Y is on peg Z and X is on top of Y*, and the satisfaction condition C_s, is the clause *X is smaller than Y*.

Generally constraints are divided into two types: constraints of the first type represent syntactic properties of the target knowledge domain; they refer only to the student's solution. Constraints of the second type represent semantic properties of the domain; they operate on the relation between the student's solution and the ideal solution. Of course, the distinction between the two kinds of constraints is not strict and some constraints inspect both the syntax and the semantics of the student's solution [26].

[1] In the Tower of Hanoi problem, a stack of disks with holes in the centre are to be moved from one peg to another in accordance with a set of rules: Only one disk is to be moved at a time, a larger disk cannot be on top of a smaller one, and at the end, the stack of disks should be on a specified peg.

In CBM tutors student solutions are evaluated by matching them against the constraint set. First, all relevance patterns are matched against the problem state. Second, the satisfaction components of constraints matching the problem state in the first step (i.e., the relevant constraints) are tested. If a satisfaction pattern matches the state, the constraint is satisfied, otherwise, it is violated. The short-term student model consists of all satisfied and violated constraints. Long-term student model mainly consists of the list of all constrains used by the student and the history of constraint usage.

There has been a number of constraint-based tutors developed withing the ICTG: SQL-Tutor, a tutor for teaching SQL, a declarative database language [23, 26], CAPIT, a system that teaches the rules of punctuation and capitalisation in English [21], and NORMIT [22], an ITS that teaches data normalisation, which is a procedural task.

2.4 ITSs for Conceptual Database Modelling

KERMIT (*K*nowledge-based *E*ntity *R*elationship *M*odelling *I*ntelligent *T*utor) [37], an ITS that teaches database design, was developed at the ICTG group. KERMIT is a problem-solving environment in which students practise database design using the ER data model. The interface displays the state of the current problem and provides controls for stepping between problems, submitting a solution and selecting the level of detail in the feedback (Figure 2.2). There are six levels of feedback, ranging from most general to most detailed. Feedback is presented on request. KERMIT also contains the main working area in which the student draws the ER diagram. KERMIT contains a set of problems and the ideal solutions to them, but has no problem solver. Evaluation of KERMIT in a classroom experiment showed significant improvement of the participant's performance.

KERMIT was a stand-alone system and recently it has been re-implemented as a web-based tutoring system [28, 23], ER-Tutor. ER-Tutor and EER-Tutor share common web-based architecture and design, described in detail in the following chapter. ER-Tutor has not been formally evaluated.

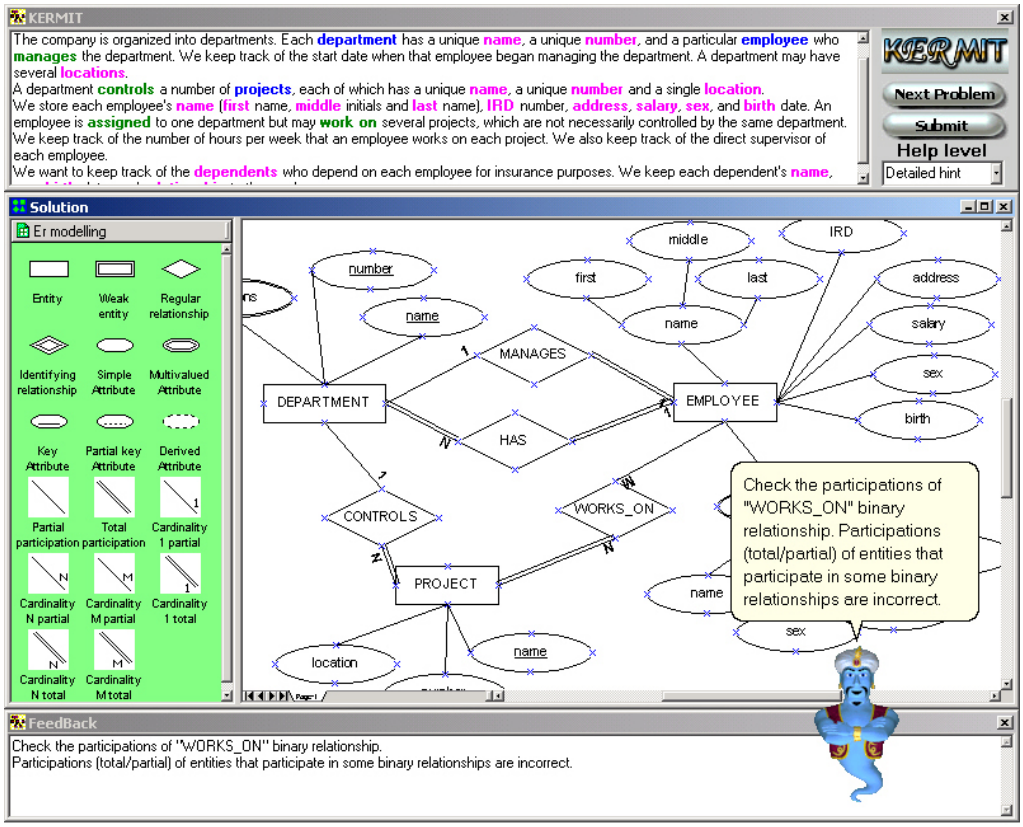


Figure 2.2: Interface of KERMIT

3 EER-Tutor

We have developed EER-Tutor as a tool for exploring the effect of instructional feedback designed according to the principles of the Cognitive Architecture and learning theory. As a software system, EER-Tutor is based on a range of technologies and programming languages: Lisp, Java, HTML, XML, and the ubiquitous application-layer HTTP protocol. The scope of this report does not call for an exhaustive description of EER-Tutor. However, this Chapter provides an overview of the main aspects of EER-Tutor.

Section 3.1 contains a description of the design and architecture of EER-Tutor, accompanied by an outline of the programming tasks and technical decisions that were required for extending the functionality of the base system, ER-Tutor. Section 3.2 describes the proposed feedback design in the context of EER-Tutor. Section 3.2 is dedicated to the explanation of feedback micro-engineering; this Section also contrasts the new experimental approach to feedback against the conventional feedback approach.

3.1 The Architecture of EER-Tutor

As a successor of ER-Tutor and as a typical web-based application, EER-Tutor is divided into a web server and client. The architecture of both systems can be traced back to SQLT-Web, the first web-based CBM tutor [23]. Since ER-Tutor and EER-Tutor share many common features, the following description of the architecture is equally applicable to both systems. The overall architecture of the system is shown in Figure 3.1. The server side is based on HTTP/1.1 [29] compliant Open Source[1] web server, AllegroServe [16], capable of hosting static and dynamic pages. AllegroServe is the optimal platform for a web-based ITS because it is designed as a module that can be loaded into an application to provide a web-based user interface to the application. AllegroServe and the rest of server-side components of EER-Tutor are implemented in Common Lisp [10, 14].

The client-side of EER-Tutor or its interface can be viewed in any common web browser as a set of dynamic HTML pages hosted by the web server. The main page of the interface contains an embedded Java applet [36] (See a screen-shot of the interface in Figure 3.2). The applet embedded in the applet frame provides a set of drawing tools for creating diagrams as solutions to the problems presented by the system. Apart from the applet frame, located in the centre of the browser window, there are navigational frame, feedback frame and submission frame. Similar to the other ITSs, the interface of EER-Tutor acts as a mediator between the student and the system, enabling the student to conduct dialogues with the system.

Communication between the interface and the server is built entirely on HTTP requests and responses exchanged by the web browser and the web server. Depending on the task the requests are initiated either by the applet or the HTML forms embedded in the pages. The interface makes requests using either GET and POST methods [29]. As intended by the HTTP protocol, POST method is used for sending data to the server and GET method is used for retrieving information from the server. For example, when a user clicks on the *Submit Answer* button to submit a solution, a POST request delivers the user's solution to the server and at the same time a GET request is initiated for updating the feedback frame.

[1] `http://www.opensource.org/` — Open Source Initiative

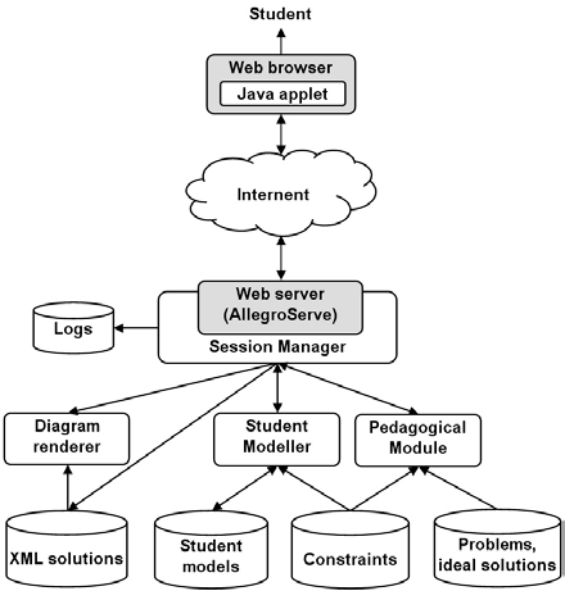


Figure 3.1: Web-based architecture of EER-Tutor

The session manager provides multi-user support for an arbitrary number of concurrent sessions and interacts with the rest of the server-side components. The individual session is established when a student logs on to the system. Sessions can be terminated either by the user of by the *auto time out* feature if the user remains inactive for more than 30 minutes. Session manager records all user actions and system responses in individual log files. Session manager is also responsible for saving students' solutions received through the web server. The solutions are saved in XML format [6]. On the user request the applet retrieves previously saved EER diagrams from the server and displays them in the drawing space for further modification.

Pedagogical module sends submitted solutions to student modeller, which diagnoses the solution, updates the student model and sends the results of the diagnosis back to the pedagogical module. At this stage pedagogical module generates feedback. Ideally Pedagogical module should implement a problem selection strategy in order to prevent the repetition of the curriculum tasks that destroys motivation. In EER-Tutor this feature is yet to be implemented.

On request the diagram renderer module converts a diagram from XML format into a JPEG [2] image, which is later displayed in the browser for viewing or printing.

As previously stated, the basic functionality of EER-Tutor has been inherited from KERMIT and ER-Tutor. Nonetheless, we were required to make modifications to practically all modules of the system. Primarily our work focused on three areas:

- Applet: functionality of the applet was extended to support manipulation of EER diagrams;
- Student modeller: we extended the format of solution representations to allow encoding of EER constructs. We also simplified the way solutions are processed by the server;
- Constraint base: to extend the diagnostic capabilities of the pedagogical module, we added a set of constraints implicitly representing EER concepts;

The details of the modifications and extensions unique to EER-Tutor are described in the following sections.

[2] `http://www.jpeg.org/` —Welcome to JPEG, Joint Photographic Experts Group

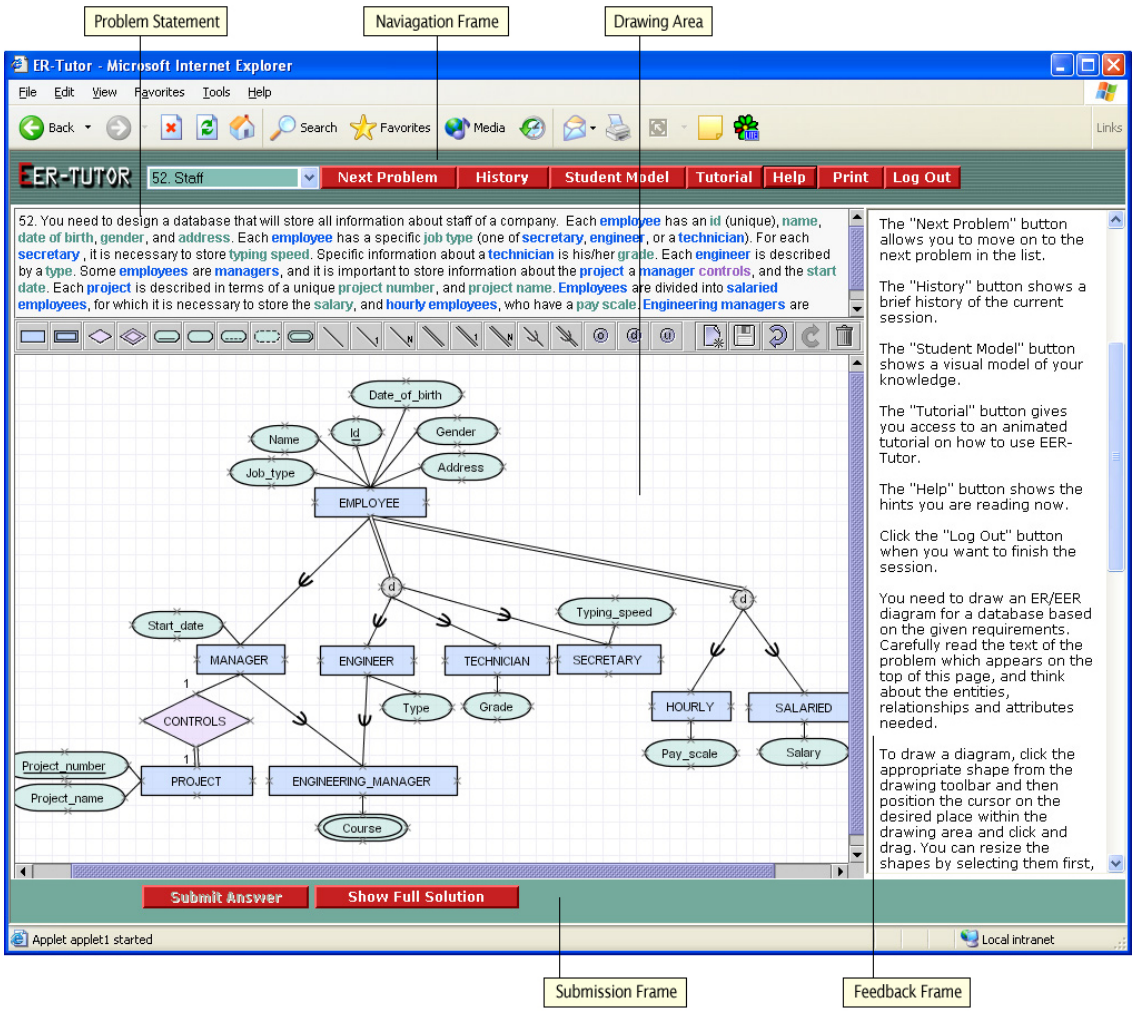


Figure 3.2: Interface of EER-Tutor

3.1.1 Applet Extension

The main part of the interface, namely the drawing applet, has been modified to support drawing EER diagrams. Figure 3.3 shows the drawing tool-bar of EER-Tutor with the added set of tools supporting specialisation, generalisation and categories as required by the EER notation [9].

The set of Java classes that implement the drawable shapes displayed in the diagram space was extended with classes implementing the constructs of the EER notation for constructing specialisation (generalisation) hierarchies, categories and subset connectors. Extensible design of ER-Tutor applet, based on the Object-oriented Design Patterns [12] allowed for efficient implementation of these new features. Figure 3.4 is an example of a generalisation hierarchy diagram shown in the applet drawing space.

3.1.2 Solution Representation Extension

Like ER-Tutor, EER-Tutor, relies on two type of solution representations for different purposes. Solutions in XML format are essential for capturing all structural details of diagrams, so that they can later be restored by the applet for further modification. The other type of solutions in Lisp format is referred to as the *internal representation*. The following two sections discuss the two formats of solutions in more detail and present

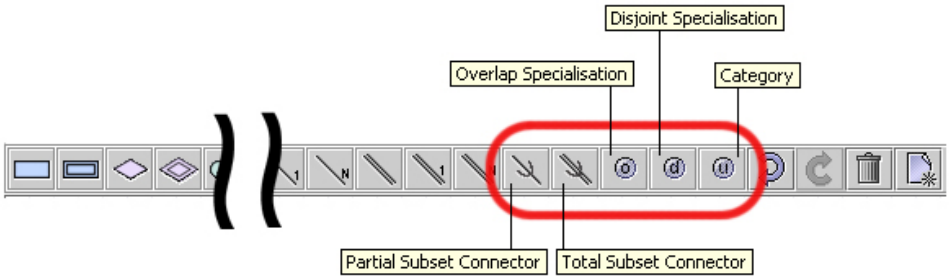


Figure 3.3: Drawing applet tool-bar

the advantages of the new approach.

3.1.2.1 XML Solution Format

The information stored in XML format includes the position, size and type of shapes as well as their labels and attached connectors. Listing 3.1 contains a fragment of XML solution corresponding the the diagram shown in Figure 3.4. We used XML format for storing solutions because of the availability of a wide range of tools for processing data in XML format. For example, we use a SAX parser [17] to restore the drawable shapes in the diagram space of the applet when the user retrieves previously saved solutions from the server.

We extended the format of XML solutions originally used in ER-Tutor by introducing additional XML element tags and attributes corresponding to the new constructs, such as specialisation (generalisation) hierarchies, categories and subset connectors. The complete Document Type Definition (DTD) [6] describing the format of EER solutions in XML format is given in Appendix B.

3.1.2.2 Internal Representation

Listing 3.2 presents a fragment of the internal representation corresponding to the diagram in Figure 3.4. Solutions in this format are used as an input for the student modeller; they contain only high-level repre-

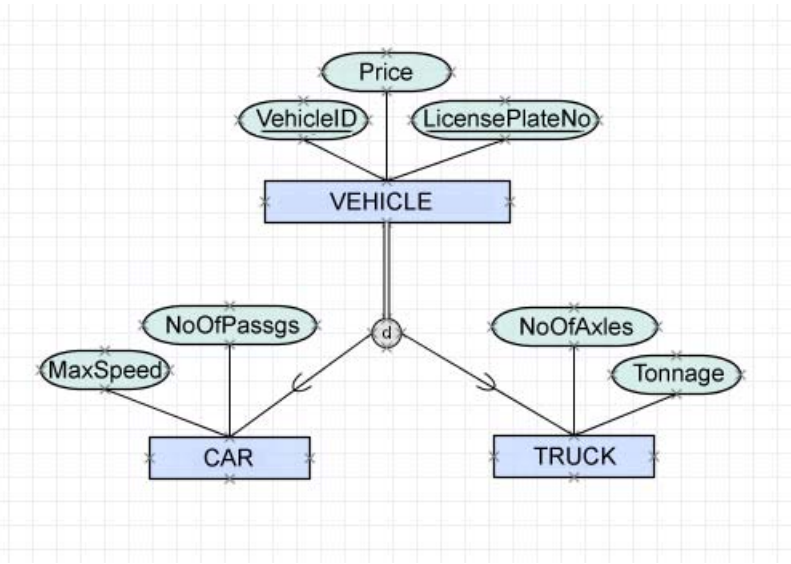


Figure 3.4: Example of a generalisation hierarchy

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<individual_solution user_name="1control" problem_number="99">
  <bound_component hash_code="_24594643" label="TRUCK" tag="TAG_24594643">
    <position xval="416.0" yval="319.0" />
    <dimension xval="164.0" yval="26.0" />
    <entity entity_type="regular" />
  </bound_component>
  <bound_component hash_code="_32999791" label="NoOfAxes" tag="TAG_32999791">
    <position xval="458.0" yval="245.0" />
    <dimension xval="78.0" yval="24.0" />
    <attribute attribute_type="simple" />
  </bound_component>
  <bound_component hash_code="_28681679" label="CAR" tag="TAG_28681679">
    <position xval="211.0" yval="313.0" />
    <dimension xval="126.0" yval="26.0" />
    <entity entity_type="regular" />
  </bound_component>
  <bound_component hash_code="_15961328" label="" tag="">
    <position xval="369.0" yval="238.0" />
    <dimension xval="18.0" yval="18.0" />
    <junction junction_type="d" />
  </bound_component>
  <connector_component hash_code="_10526273" role_name="" tag=""
                       participation="partial" cardinality="_NC" subset="yes">
    <position xval="369.0" yval="247.0" />
    <position xval="274.0" yval="313.0" />
    <component_reference component="_15961328" handle_number="0" />
    <component_reference component="_28681679" handle_number="1" />
  </connector_component>
  ...
</individual_solution>
```

Listing 3.1: Fragment of a sample XML solution file

sentation of the EER data model corresponding to the diagrams produced by system users. During solution diagnosis stage, constraints are matched against solutions represented in this format. For example, line 1 of Listing 3.2 shows that regular entity `TRUCK` identified by the tag `TAG_24594643` is intended to be a subclass of another entity type that has a tag value of `TAG_27134372`. The conceptual EER information about specialisations (generalisations) and categories is contained in the solution in the form of four lists: `SUPERCLASSES`, `SUBCLASSES`, `CATEGORY-SUPERCLASSES` and `CATEGORIES`, shown in the lines 4–7. In our case, the diagram contains only one superclass with two subclasses and no categories. In considering simplicity and performance issues during the system development stage, the option of presenting the EER solution components as the four additional lists has been given preference over the option of adding EER information to the end of the `ENTITIES` list. The complete definition of the internal representation format is given in Appendix C.

3.1.2.3 Extracting Internal Representation with XSLT

In ER-Tutor internal representations of students' solutions were compiled and sent to the server by the applet, but in EER-Tutor solutions in internal representation form are derived from XML solutions with the help of the eXtensible Stylesheet Language Transformation (XSLT) [18, 6]. We adopted this approach because it offers the following advantages:

- It simplifies the applet, by removing the code that traverses the objects in the diagram and extracts the internal representation. This makes the applet light weight, thus reducing the requirements placed on the client-side part of the system.

```
(("ENTITIES" . "@ TAG_24594643 TRUCK regular ...")
 ("ATTRIBUTES" . "@ TAG_32999791 NoOfAxes simple simple TAG_24594643
                 @ TAG_16900472 Tonnage simple simple TAG_24594643 ...")
 ("SUPERCLASSES" . "@ TAG_27134372 d total ( TAG_24594643 ...)")
 ("SUBCLASSES" . "@ TAG_24594643 TAG_27134372 ...")
 ("CATEGORY-SUPERCLASSES" . "")
 ("CATEGORIES" . "")
 ... )
```

Listing 3.2: Fragment of a sample internal representation

- It simplifies the general architecture of the system by cutting out the need for transfer of two types of solutions over the network, thus reducing the coupling of the system modules.

XSLT is commonly used for processing XML data. In many situations, a data set needs to be presented in a different format, that may not be precisely equivalent to the original document. This is achieved by suppressing some data or presenting the data derived from the original set; the latter applies to the transformation of XML solutions into the internal representation format. XSL transformation suppresses the irrelevant details of the XML representation (such as location and size of the shapes in the diagram) and derives information about the roles of the objects in the diagram, i.e. superclass/subclass hierarchies.

Out of the four stylesheet design patterns [18] the stylesheet we developed for our purposes falls under the category of Computational stylesheets, because the conceptual structure (in our case the EER model) in the source of the original document is not explicit in the XML markup. The structure of Computational stylesheets closely resembles functional programming.

```
...
<!-- Entities Template -->
<xsl:template name="get-entities">
 <xsl:for-each select="bound_component/entity">
   <!-- reference hash_code of this entity -->
   <xsl:variable name="entity_ref">
     <xsl:value-of select="parent::bound_component/@hash_code"/>
   </xsl:variable>
   <xsl:text>@ </xsl:text>
   <!-- output tag -->
   <xsl:value-of select="/individual_solution/bound_component[@hash_code=$entity_ref]/@tag"/>
   <xsl:text> </xsl:text>
   <!-- output label -->
   <xsl:value-of select="/individual_solution/bound_component[@hash_code=$entity_ref]/@label"/>
   <xsl:text> </xsl:text>
   <!-- output type -->
   <xsl:value-of select="@entity_type"/>
   <xsl:text> </xsl:text>
 </xsl:for-each>
</xsl:template>
...
```

Listing 3.3: Fragment of the stylesheet

Application of the `get-entities` template starting on line 3 in Listing 3.3 to the XML solution fragment in Listing 3.1 would produce `"@ TAG_24594643 TRUCK regular"` string, which is a part of internal representation shown on the first line of Listing 3.2.

3.1.3 EER Constraints

As previously stated, EER model incorporates *superclass/subclass* relationships and *type inheritance*. Based on these relationships, there are concepts of generalisation, specialisation and categories [9]. By analysing the description of these concepts in [9], we have identified 13 syntax constraints and 26 semantic constraints. Combined with the ER-Tutor constraints, the new constraints explicitly describe a full set of correct solutions to EER modelling problems. Constraint representation used in EER-Tutor is described in [20].

Here we give examples of two EER constraints. A syntax constraint in Listing 3.4 will signal an error if the subset connector in the students' solution has one or more subset connectors pointing in the wrong direction. Correct direction of a connector in a diagram is determined by the subset symbol $\subset$ pointing away from the subclass or category, indicating that subclasses (or categories) represent subsets of their superclasses. The relevance condition (lines 4–6) matches a problem state, when there is a subset connector present in the student solution. The satisfaction condition (lines 7–11) will remain `true` only if the connector goes from an entity type toward a circle; this would be indicated by the presence of a valid binding of the variable `?tag`, followed by the string `"junction"`.

```
(91
 "Check whether subset connectors have the right direction, indicating that subclasses
  (or categories) represent subsets of their superclasses."
 (match SS CONNECTIONS (?* "@" ("partial" "total") "subset" ?*))
 (not-p
  (and
   (match SS CONNECTIONS (?* "@" ("partial" "total") "subset" ?tag "junction" ?*))
   (match SS ENTITIES (?* "@" ?tag ?*)))
  )
 "connections"
 ())
```

Listing 3.4: Example of a syntax constraint

A semantic constraint in Listing 3.5 checks that the participation of subclasses specified by the student in a generalisation (specialisation) hierarchy conforms to the participation type defined in the ideal solution to the given problem. The variable `?p` in the satisfaction condition (line 4) is bound to the participation type (which can be either `"total"` or `"partial"`) in the ideal solution. The satisfaction condition will return `true` only if the string in place of `?p` (line 7) equals the original binding of the `?p` variable.

```
(165
 "Check whether your specialisation should be total or partial."
 (and
  (match IS SUPERCLASSES (?* "@" ?super ("d" "o") ?p "(" ?* ?sub ?*))
  (match SS SUBCLASSES (?* "@" ?sub ?super ?*))
  )
 (match SS SUPERCLASSES (?* "@" ?super ("d" "o") ?p "(" ?* ?sub ?*))
 "specialisation/generalisation"
 (?super ?sub))
```

Listing 3.5: Example of a semantic constraint

3.2 Feedback Micro-engineering

Recall that CBM is based on Ohlsson's theory of learning from performance errors [32]. This theory states that an erroneous action indicates that the knowledge structure that caused this error is faulty. In general, the

basic learning process consists of two parts: *error detection* and *error correction*. Needless to say, errors can not be corrected unless the learner notices them. There are two aspects to error detection. First, the action must generate some perceptible effect, since errors are detected by their effect on the environment. Second, the learner must be able to interpret the effect as a signal of an error. In some situations, errors are easy to detect, but in the others the error signal can be transparent or indirect.

To correct an error means to improve future performance by revisiting the faulty knowledge that caused the error. In the Cognitive Architecture, the function of identifying the faulty rules is called *blame assignment*. A simple explanation of the blame assignment tactics suggests that the undesirable consequence is attributed to the last action before the error signal was observed. However, this simplistic approach does not always work; in many situations the mind has to engage in more involved processing to allocate the blame. The learner might need to revisit a large number of previous steps and detect the one that caused the error. Such processing requires considerable knowledge about the task environment.

If we apply the above assumption to the process of learning conceptual database design with a pen and paper, the problem arises from the fact that usually a novice learner is unable to detect an error because it requires deep theoretical knowledge of the EER domain. In general, this observation emphasises the fact that learning tasks based on conceptual knowledge places the learner in a vicious circle: trying to improve performance in some skill, the learner naturally does not intend to make errors but he or she is unable to detect errors, because of the lack of experience and knowledge. The same problem applies to the error correction stage: the learner must revisit the faulty knowledge, but with open-ended theoretical tasks like EER modelling the learner's mind will have difficulty identifying the relevant declarative knowledge for making suitable corrections.

This is where CBM comes to the rescue. Recall that constraints implicitly represent all possible erroneous solutions in a knowledge domain. So, when an error occurs, the task of error detection and blame assignment is carried out by the system. At this stage, the system should refer the user to the relevant part of the domain knowledge. Consequently, an effective feedback message should tell the user (a) where exactly the error is, (b) what constitutes the error (perform blame allocation), (c) refer the user to the underlying concept of a correct solution (revise underlying knowledge) and (d) possibly tell the user what has to be done to correct the error.

The above observations constitute the central focus of our experimental study with EER-Tutor. The ITSs implemented at ICTG to date did not directly utilise these observations in the feedback generation. Feedback messages in ER-Tutor, as well as the other tutors, merely tell the user to check a certain aspect of the solution and accompany the advice with a suggestion for correcting the problem. For example, consider problem 5:

> *Some students live in student halls. Each hall has a name (unique) and an address. Each student has a number (unique) and a name. Assume that there are students living in every hall.*

Suppose the student submits an attempt containing an incorrectly specified participation of an entity type in a relationship (see Figure 3.5). In response to this solution, ER-Tutor would produce the following error message, associated with the violated semantic constraint that validates the correctness of the participation of the entity types in a relationship:

> *Check the participation of entities in relationships. You have specified partial participation, when it should be total.*

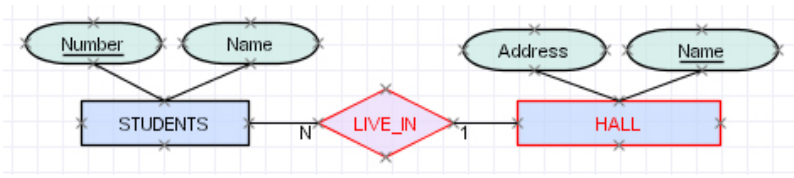


Figure 3.5: An incorrect solution to problem 5

The student erroneously used a partial participation connector due to lack of experience in extracting the modelling requirements from the problem statement. In particular, the phrase *Assume that there are students living in every hall* implies that every entity in the total set of HALL entities must be related to N STUDENT entities, which implies total participation of the HALL entity type in the LIVE_IN relationship. The error message partially allocates the blame and tells the student what has to be done to correct the error. However, after closer examination of the feedback message keeping in mind Ohlsson's theory, we can detect a number of problematic issues with this type of feedback. First, the message is not pointing out the error; this might be confusing to a user dealing with a larger diagram containing, for example, a dozen entities and half as many relationships. Second and most importantly, the message does not offer help with the revision of underlying faulty knowledge. The message simply tells the learner what to do in order to correct the solution; this is insufficient for successful learning.

The following message, theoretically, would generate a greater impact on the user:

> *The participation constraint specifies whether each entity of a specific type has to appear in the relationship type (total participation) or not (partial participation). The participation of the connector between the highlighted constructs is incorrect.*

Referring to the same erroneous solution in Figure 3.5, the above message starts with the general concept which, most likely, has not been internalised by the user. This is the step aimed at specialising the corresponding rule in the procedural memory of the learner, so that next time when a similar situation arises, the user will hopefully be able to differentiate correctly between choosing partial or total participation. The second sentence of the message ties the concept to the situation at hand, simultaneously pointing out the error and allocating the blame. The error correction information is not essential for the given problem, since there are only two options for specifying participation in a relationship. The careful engineering of every feedback message should theoretically have strong influence on the performance. For the purposes of our study, we have defined a feedback message for each constraint in the new format.

We want to draw the line between the two types of feedback and compare the possible effective differences from a theoretical point of view. First, we suspect that the old format of feedback might result in *shallow learning*, which refers to failure in internalising the knowledge and poor knowledge transfer. In other words, the user might learn how to arrive at a point of producing solutions that are correct from the system's points of view. However, having to apply the underlying concepts in a different environment may result in poor performance. This point is supported by research proving that learning how to play an educational game does not necessarily imply learning the target instructional domain [8]; learning happens only when students actively build the connections between game sequence and underlying knowledge. In this light, we expect that the micro-engineered feedback in EER-Tutor will result in better knowledge transfer and deeper learning [38].

Another argument in support of the new feedback format originates from ACT-R theory, but is equally applicable to CBM: the fourth principle of the ITSs design [2], states that an ITS should promote an abstract understanding of problem-solving knowledge. This principle was motivated by the observation that students often develop overly specific knowledge from particular problem-solving examples; this is also related to shallow learning and poor knowledge transfer.

Overall, the theoretical advantages of the new feedback format favour it over the conventional type of feedback. However, taking into consideration the first motivating factor behind ITS, which is the individuality of learners' needs, we should not forget that the use of different levels of feedback proved its effectiveness. Consequently, for some users, the old type of feedback may still result in greater performance improvement coming about as a result of the more active mental process required in response to less helpful feedback.

4 Evaluation

An evaluation study of EER-Tutor was conducted at the University of Canterbury, Christchurch in August 2004. Second year students enrolled in COSC226, an introductory course on databases taught at the Department of Computer Science and Software Engineering, were offered the opportunity to voluntarily participate in the study. The students had learnt EER modelling concepts prior to the study during three weeks of lectures and had had some practice in EER modelling during two weeks of tutorials.

Section 4.1 contains a description of the experiment design and students' interaction with EER-Tutor. Experimental results and their analysis are presented in Section 4.2, followed by a discussion of the evaluation outcomes in Section 4.3. Subjective evaluation of EER-Tutor in this study was de-emphasised in preference to objective performance measures. We did not use questionnaires; instead the users were given maximum freedom in their on-line interaction with the system.

4.1 Experiment Design

The study focused on a comparison of the performance of the two groups of students using two versions of EER-Tutor differing only in the type of feedback they provided: the proposed experimental feedback and the original feedback. Before starting with EER-Tutor, the system was briefly introduced to the class in a lecture. For most students the first session with EER-Tutor took place during a laboratory session. Students were required to log onto EER-Tutor with their user codes and passwords. At the time of the first log on the system randomly allocated the students to either experimental or control group. The students were free to use EER-Tutor for a two week period, prior to submission of a course assignment on the EER modelling. Curriculum of EER-Tutor consisted of 56 problems ordered approximately in an increasing order of difficulty; the last five problems focused on EER concepts. The users were not restricted in their choice of problems.

To assess students' knowledge before and after practising with EER-Tutor, the first session for every student started with an on-line pre-test and at the end of the two week period the students were offered an on-line post-test. In this way most students sat the pre-test during the first session in a supervised environment, but the post-test was offered to students in an uncontrolled environment.

We designed two tests of approximately equal difficulty that were interchangeably used for pre- and post-tests. In order to reduce any bias, one of these two versions was randomly chosen for each student as a pre-test. The other previously unseen version, was used as a post-test. Each test contained four multi-choice questions and one true/false question. In terms of the test content, there was a question asking students to choose a correct ER schema, a question on entity types, a question on attribute properties, and two questions on EER concepts. Both versions of the tests are given in Appendix A.

In our attempt to maximise the effect of feedback on learning, we introduced three restrictions to the users' interaction with the system:

1. The system provided only one level of feedback, listing the messages of the first three errors at most;

2. The system would not allow users to view ideal solutions to a current problem unless they made at least five attempts at solving the problem;

3. If the user had seen the ideal solution to a certain problem, the system would not allow the user to submit any more attempts for that problem.

4.2 Results

During the two-week period 105 students used EER-Tutor; this constituted approximately 82% of the class. The experimental and control groups contained 52 and 53 students respectively. Table 4.1 contains the general statistics of the students' interaction with the system. The maximum number of attempted problems was 52, while the maximum number of solved problems was 43. The number of attempts per problem ranged from 1 to 17. The time required to solve a problem ranged from 7.2 minutes to 23 hours respectively, however the two highest measures of 11 and 23 hours were, most likely, outliers. Total time spent in the system ranged from 10 minutes to 45 hours.

	Experimental		Control	
	mean	s. d.	mean	s. d.
Attempted problems	15.25	10.69	15.5	11.38
Solved problems	12.92	10.17	13.21	10.32
Attempts per problem	4.24	2.53	4.96	3.83
Time per problem (hours)	1.87	3.77	1.62	1.85
Seen feedback messages	23.48	20.02	24.39	22.06
Time spent in the system (hours)	15.90	10.53	16.87	12.61

Table 4.1: Basic details of the study

Results of T-tests comparing the difference between the groups based on the mean values for the number of solved problems, amount of time and number of attempts required to solve a problem were not significant (at $\alpha = 0.05$). Two-way ANOVA tests were used to compare the two groups (a) based on the number of attempts per problem and number of solved problems and (b) the number of seen feedback messages and number of solved problems. These tests did not reveal significant difference between the groups. The following sections present a deeper analysis of system logs and student models.

4.2.1 Pre-test and Post-test Results

In total there were 105 pre-test submissions (52 in experimental group and 53 in control group), with the mean or 61.9% and standard deviation of 27.66%. There were 91 post-test submissions (45 and 46 in experimental and control groups respectively), with the mean 24.83% and standard deviation or 33.51%.

Table 4.2 shows the results for pre- and post-tests. The difference between pre-test means for the two groups is not significant, which indicates that the two groups are comparable in their prior knowledge levels. However, we cannot rely on the obtained post-test results. It is apparent from the log files, that the majority of students did not take their time to even read the post-test questions. For example, the following fragment of a log file shows that the student clicked the *Submit* button on the post-test page in 14 seconds after the system displayed the post-test questions:

```
...
20:22:41 26/08/2004 User logged in
20:22:55 26/08/2004 Post−test submission
20:22:58 26/08/2004 Changing subset to SET1, problem number is 1
...
```

Even when the time between login and post-test submission is more substantial, we can not tell apart the situations when students did not answer questions at all or answered them incorrectly. The reason for this is that in the encoding scheme for the post-test results, both a no answer submission and an incorrect submission were recorded as zero. Consequently, comparison of the pre- and post-test results between the experimental and control groups does not contain conclusive difference.

	Experimental			Control		
	N	mean	s. d.	N	mean	s. d.
Pre-test	52	59.62	28.69	53	64.15	26.71
Post-test	45	26.54	33.77	46	16.60	30.32
Assignment	52	68.65	9.35	53	68.16	7.30
Exam	48	52.96	22.12	51	55.62	20.74

Table 4.2: Pre-, post-test, assignment and exam results

Intending to compensate for the lack of reliability in the post-test results, we analysed the assignment and exam results, shown in Table 4.2. In the assignment the students were required to design a comprehensive EER model. There were 52 and 53 assignment submissions in experimental and control groups respectively. There was no significant difference between the groups on the assignment results. However, comparison of improvement inside the groups between the pre-test and assignment grade revealed significant improvement in the experimental group ($t = -2.42$, $p = 0.01$).

Exam results in Table 4.2 are based on the scores for three questions on EER modelling. The exam was taken by 48 and 51 students from the experimental and control groups respectively. There was no difference between pre-test and exam results for experimental group, however for the control group the difference indicates worse performance in the exam questions ($t = 2.06$, $p = 0.04$).

We found significant difference ($t = 5.23$, $p = 4.13e^{-5}$) in assignment grades for students who used the system (both experimental and control groups, 105 students, mean of 68.41%, standard deviation 5.35) and vs. the rest of the class (14 students, mean 54.07%, standard deviation 9.79). However, this result does not have much conclusive power because it compares two self-selected groups. There is a high chance that these groups were not comparable in the level of prior knowledge and motivation.

	Experimental			Control		
	N	mean	s. d.	N	mean	s. d.
Pre-test (less able)	33	42.94	21.53	29	46.1	21.12
Assignment (less able)	33	67.54	10.65	29	67.21	7.92
Pre-test (more able)	19	88.43	10.14	24	86.67	9.63
Assignment (more able)	19	70.58	6.36	24	69.33	6.45

Table 4.3: Post-hoc results comparison

Another type of analysis involves a post-hoc division of the experimental and control groups into two subgroups: more able students (pre-test score above the class mean of 61.9%) and less able students. Groups in the same categry, i.e. more able groups and less able groups are comparable, since there is no significant difference in the pre-test results. Again, we did not find significant difference comparing the scores of assignments between the two types of experimental and control groups, but there is significant difference inside the groups (Table 4.3). It is apparent that the less able students in both experimental and control groups displayed significant improvement ($t = -6.78$, $p = 5.5e^{-6}$ and $t = -5.31$, $p = 5.93e^{-6}$ respectively). On the other hand, the assignment results of the more able students in experimental and control groups are significanlty less then the pre-test results ($t = 8.56$, $p = 4.63e^{-8}$ and $t = 9.6$, $p = 8.2e^{-10}$ respectively). These results have two important implications:

1. It is quite possible that less able students made better progress in their learning, relative to their initial knowledge level. The overall assignment results of the more able group are still higher than in the

less able group, but it may be possible that the relative progress of the more able group is of less significant in comparison to the less able group.

2. There may be a chance that the pre- and post-tests used for this study do not provide a reliable way of assessing students' knowledge. This is not very surprising, since five multi-choice questions can cover only a small fraction of the domain knowledge.

4.2.2 Learning Curves

To compare the effect of two kinds of feedback on constraint acquisition we performed analysis based on the Power Law of Practice [30]. If constraints represent appropriate units of domain knowledge, the learning should follow a smooth curve [1]. Variations of the power curves fits to the experimental data can be explained by a large number of interfering factors and noise in the data, however in some cases, poor fits may indicate that some constraints might not represent pedagogically equivalent units of knowledge. We analysed two types of learning curves: the first type evaluates the probability of constraint violation on a per problem basis, while the second type calculates the probability of violations on a per attempts basis.

Based on the user logs, we identified a state containing all relevant constraints for every problem attempted or solved by the user. Each constraint relevance occasion (whether the constraint was satisfied or violated) was rank-ordered from 1 up. We calculated, for each participant, the probability of violating each individual constraint while solving the first problem, second problem and so on. The probabilities were averaged across all the constraints in order to obtain an estimate of the probability of constraint violation for each solved problem. These probabilities were averaged across all the participants in the experimental and control groups and plotted as a function of the order of solved problem (Figure 4.1(a)). For the first problem the were a total of 4825 and 4872 constraints used by experimental and control groups respectively.

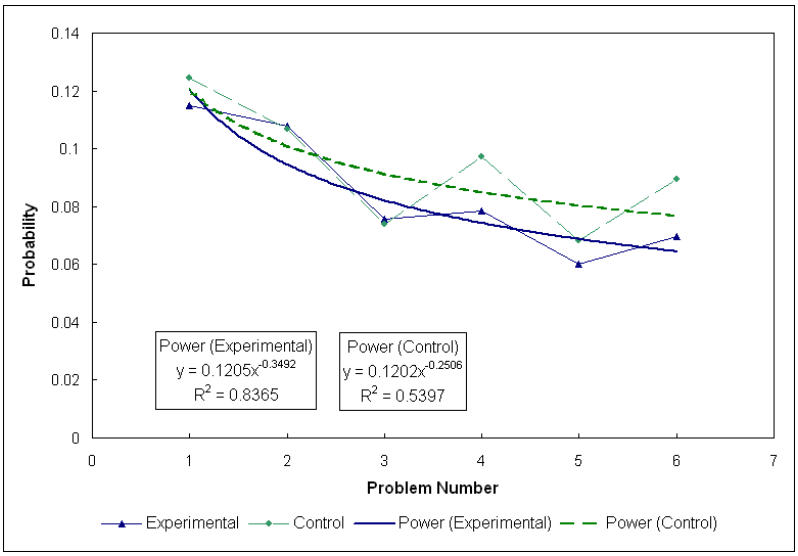
To reduce individual bias for all of the learning curves described in this section we identified the cut-off points at 50% of the initial number of relevant constraints.

The general trend of the data-points indicates the decreasing probability of constraint violation in both experimental and control groups as the participants solve more and more problems. The power curves for the experimental and control groups indicate lower probabilities of constraint violation in the experimental group. For example, on the 6th problem the probability of constraint violation for the experimental group is 0.065 vs. 0.08 for the control group; this constitutes a 19% improvement. The experimental power curve also has a better fit of $R^2 = 0.84$ vs. $R^2 = 0.54$ for the control group, indicating a more stable process of constraint aquisition.

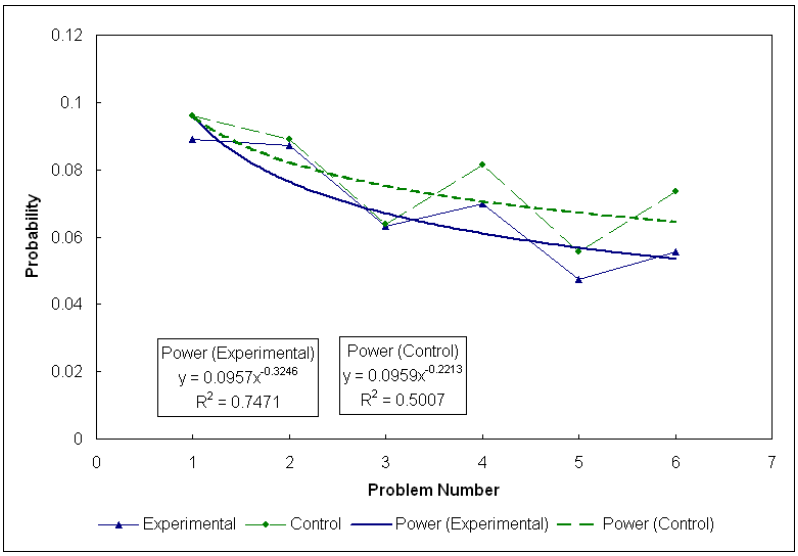
Both sets of learning curves in Figure 4.1 have distinctive sets of local maximums and minimums, which can be explained by the non-uniform increase of problem difficulty. A comparison of the relative complexity of the first few problems indicates that problem 4 is considerably more involved then problems 3 and 5.

We took a similar approach for plotting the learning curves shown in Figure 4.1(b), except in this case we calculated the probabilities only for the constraints whose feedback messages has been seen by each individual user. As described in Section 4.1, the feedback was limited to the messages associated with the first three violated constraints for each attempt. Consequently, only these constraints are accounted for. There were 4735 and 4773 constraints for the experimental and control groups respectively. Again the curves show faster decline of constraint violation probability for the experimental group. On the 6th problem the experimental power curve shows the probability 0.055 vs. 0.065 for the control group, which constitutes an 18% improvement. The experimental power curve has a better fit of $R^2 = 0.74$ vs. $R^2 = 0.5$ for the control group.

The second approach to calculating learning curves has a finer granularity; it focuses on distinct attempts. For every attempted recorded in the logs we identified a state containing all relevant constraints. Similar to the previous set of learning curves calculations, each constraint relevance occasion was rank-ordered. We calculated for each participant the probability of violating each individual constraint on the first attempt, second attempt and so on. The probabilities were averaged across all constraints in order to obtain an estimate of the user violating a constraint on every successive attempt. These probabilities are plotted as a function of the attempt rank (Figure 4.2(a)). There were 4817 and 4865 constraints used by the experimental and control groups respectively. In this case, the fits of the curves spanning 20 attempts for



(a) All relevant constraints



(b) Previously seen constraints

Figure 4.1: Learning curves calculated on per problem basis

experimental and control groups are very close ($R^2 = 0.913$ and $R^2 = 0.911$ respectively), but the curve for the experimental group is declining faster then for the control group.

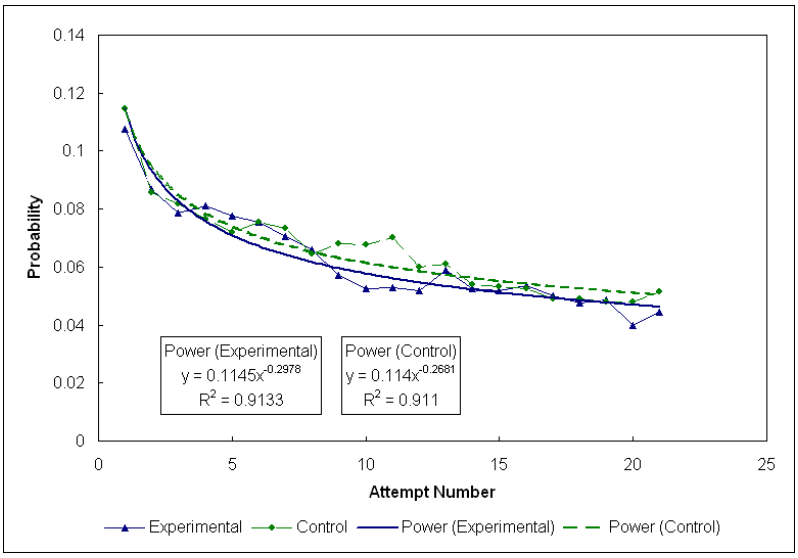
Learning curves based on constraints whose feedback messages were displayed to the participants are given in Figure 4.2(b). This set of learning curves again highlights the difference between the experimental and control groups: $y = 0.049x^{-0.1915}$ vs. $y = 0.0482x - 0.1498$ indicates better learning performance in the experimental group, however the curve fits are considerably lower for both curves. There were 4815 and 4851 constraints used by the experimental and control groups respectively.

The comparison between Figures 4.2(a) and 4.2(b) may be interpreted as a proof of the hypothesis that knowledge acquisition is possible even in the absence of feedback or constraint violation [32]. In the context of EER-Tutor, the knowledge base contains a number of constraints teaching pedagogically equivalent units of knowledge, differing only in some secondary details. For example, Listing 4.1 shows two constraints correcting the same error: derived attributes cannot have components. Constraints 39 and

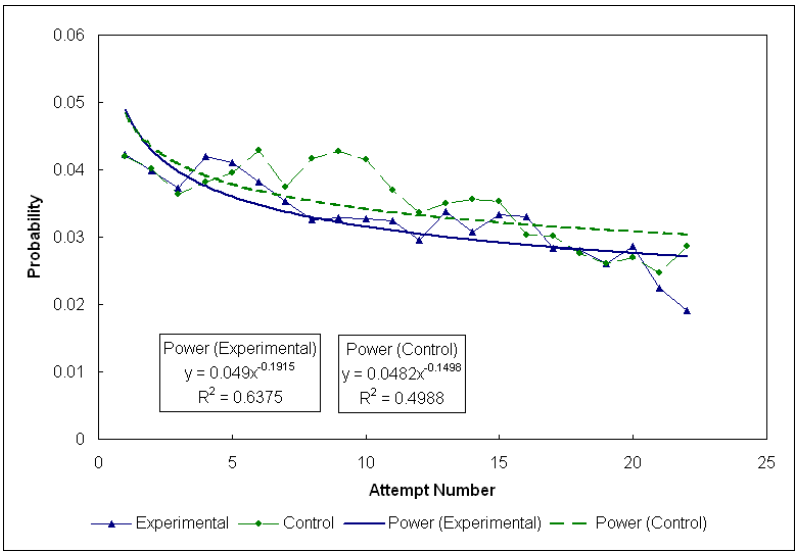
39_1 check that entity types or relationship types do not have derived attributes with components. There is some likelihood that after reading the feedback message for constraint 39, the user might deduce that a derived attribute cannot have components, whether it is associated with an entity type or a relation type. Thus constraint 39_1 might be acquired by the user without prior violations.

4.2.3 Learning Gain Analysis

The third type of trace data analysis we performed is aimed at a direct comparison of the amount of learnt constraints. By examining constraint usage history in the student models, we identified for each user the number of known constraints at the end of the experiment. A constraint was considered known by the student if the last five entries in the constraint history indicated successful application of this constraint in 80% or more relevance cases. For each individual user, the number of constraints known to the user



(a) All relevant constraints



(b) Previously seen constraints

Figure 4.2: Learning curves calculated per attempt

```
(39
 "Check your derived attributes that belong to entities. They have components.
  Derived attributes cannot have components."
 (and (match SS ATTRIBUTES (?* "@" ?tag ?label ?type "composite" ?tag_own ?*))
      (match SS ENTITIES (?* "@" ?tag_own ?*)))
 (not-p (test SS ("derived" ?type)))
 "attributes"
 (?tag))

(39_1
 "Check your derived attributes that belong to relationships. They have components.
  Derived attributes cannot have components."
 (and (match SS ATTRIBUTES (?* "@" ?tag ?label ?type "composite" ?tag_own ?*))
      (match SS RELATIONSHIPS (?* "@" ?tag_own ?*)))
 (not-p (test SS ("derived" ?type)))
 "attributes"
 (?tag))
```

Listing 4.1: Equivalent constraints

at the end of the experiment was plotted as a function of received instruction, which was estimated as the total number of seen feedback messages (Figure 4.3(a)). The slopes of the linear trend lines indicate the exerimental feedback resulted in more efficient constraint aquisition.

A clearer result was obtained through a more restrictive analysis the learning gain. This analysis took into account only those constraints that were not known to the user at the start of the experiment, but were learnt by the end of the experiment. A constraint was considered not known at the start of the experiment if the first three times it was used unsuccessfully in 66% or more cases. Constraint status at the end of the experiment was calculated in the same way as described above, based on the five last relevance cases. This conservative approach includes only those constraints whose usage histroy was long enough to fit the criterion described above.

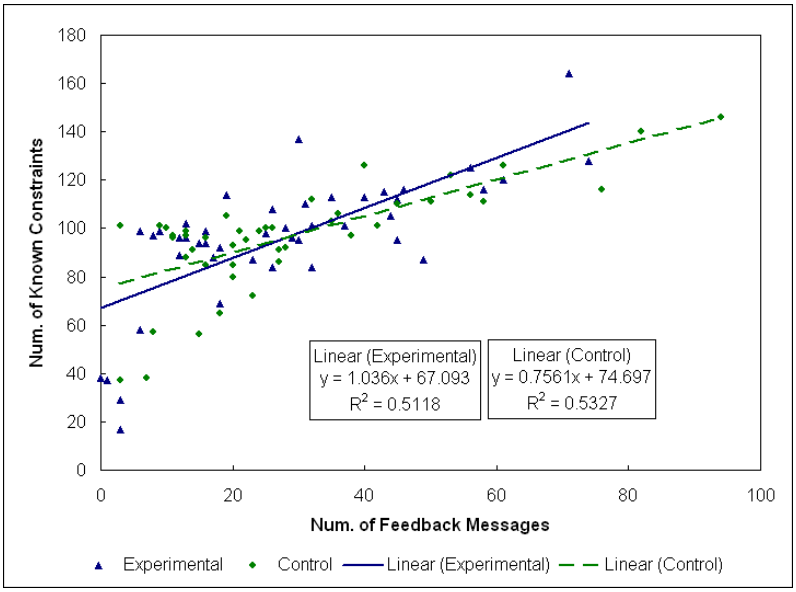
Figure 4.3(b) presents a comparison of learning gains. From both experimental and control groups we chose the students who, according to our analysis, learnt at least one constraint (27 and 32 students in experimental and control groups respectively). In the experimental subgroup, the number of feedback messages ranged from 12 to 74, while the maximum number of learnt constraints was 16. In the control subgroup, the number of feedback messages ranged from 10 to 82, while the maximum number of learnt constraints was 12. There are two distinct clusters of data-points associated with the experimental and control groups. Comparison of the slopes of the trend lines (0.16 vs. 0.01) indicates a higher learning gain in the experimental group. However, a T-test comparing the number of learnt constraints for the two groups was not significant.

4.3 Study Reflections

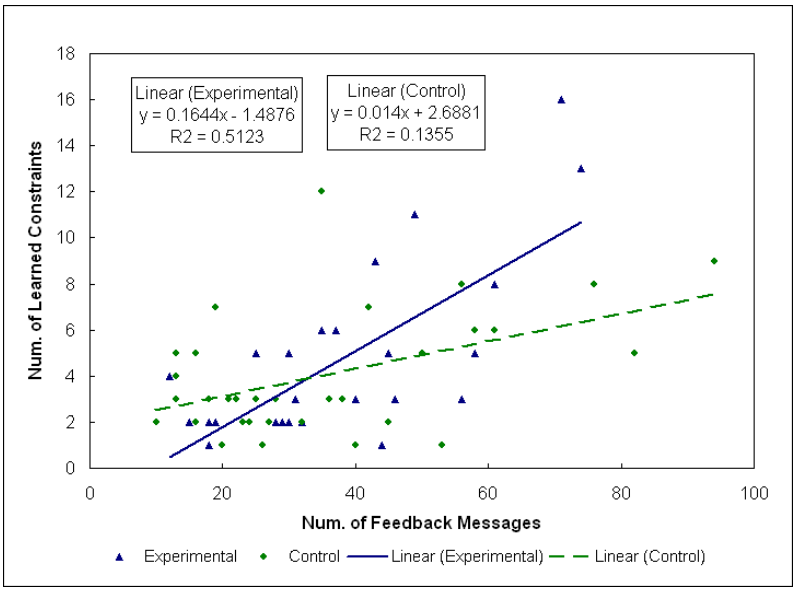
The learning curves (Section 4.2.2) and instruction dependent learning gain analysis (Section 4.2.3) seem to indicate the advantage of the experimental feedback; however the lack of significant statistical difference between the groups based on their performance does allow us to make definitive conclusions. In retrospect, we believe that the difference of performance measures was dampened by a number of objective and subjective factors and noise interference.

First, the lack of validity in the post-test results impaired a clean-cut comparison of the experimental and control groups. Possibly the post-test results would have been more reliable if the post-test was conducted in a supervised environment. Second, the pre- and post-test should possibly contain a larger number of questions covering a more extensive part of the target domain. This would help to obtain more accurate estimates of the participants' knowledge levels before and after the experiment.

Another possible component of interference is the fact that during the experiment run, EER-Tutor was



(a) Constraints known at the end of the experiment



(b) Learning gain calculated at the end of the experiment

Figure 4.3: Correlation of amount of instruction and learning

not the only source of the students' learning. Needless to say, confining the participants to EER-Tutor as the only source of learning in the University environment would not be a realistic option.

From visual examination of student models and logs we conclude that most students did not have a chance to practise application of more complex constraints, especially those constraints that covered EER concepts. It seems that most students were solving problems in sequential order, which would involve a lot of similar repetitive tasks. This factor could have a negative influence on the students' motivation. The appropriate solution to this issue lies in the problem selection strategy implemented by the pedagogical module. This would give the students a chance to identify and learn a wider scope of incomplete or untouched units of the target domain.

5 Conclusions and Future Work

One-to-one tutoring is known to be the most effective model of instruction because an expert tutor is capable of adjusting the flow and direction of instruction according to the needs of individual learners. Effectiveness of one-to-one tutoring is the key motivating factor supporting the development of ITSs. Trying to replicate the behaviour of human tutors, ITSs developed for a variety of knowledge domains help their users achieve higher learning goals. However, development of such systems is a complicated task because it needs to take into account the complexity of the human mind.

The theory of learning from performance errors can be used by the ITSs to the advantage of individual learners by providing them with extensive support during the learning process. In this report we presented EER-Tutor, an ITS for teaching EER modelling. This ITS was used for testing a hypothesis that feedback messages engineered with the principles of the Cognitive Architecture in mind are more effective than conventional feedback messages. In other words, we suggest that the two components of learning, error detection and error correction, expertly addressed by feedback messages in EER-Tutor provide better learning support. The combination of the general domain knowledge relevant to the student's error, along with the specific details of the error in the given situation should provide long-term and short-term learning benefits through simultaneous revision of faulty knowledge and strategies.

A comparative analysis of performance measurements between the two groups did not reveal significant statistical difference between the experimental feedback format and conventional format. Partially this result is due to lack of reliability of the post-test results. However, it is apparent that the experimental feedback resulted in statistically significant improvement of performance inside the experimental group. In addition the learning curves displayed a consistent advantage of the experimental feedback over conventional feedback. Similarly, a comparison of learning gains as a function of amount of received instruction also suggests that the experimental feedback has a stronger influence learning performance. The overall outcome of this evaluation study is promising, but a certain lack of conclusiveness suggests that further research in this area is required.

EER-Tutor users would significantly benefit from a problem selection strategy, which is an essential feature for every ITS, because adaptive curriculum sequencing is another factor of success on one-to-one tutoring. There is a chance that problem selection strategy could make the difference between the two types of feedback discussed in this report more apparent.

Although we attempted to maximally enrich the experimental feedback design with the theory of learning, there is still more work that needs to be completed in this direction. The situation-specific feedback provided by the system as a part of the feedback message in EER-Tutor can be specialised even further to help the learner build the missing links. At this stage, feedback messages in EER-Tutor are statically defined in the source code. However, dynamic generation of the specific part of feedback messages, referring to a concrete EER problem through the vocabulary used in problem statement would make the gap between the user's mind the instructional environment even smaller.

Acknowledgements

I would like to thank my supervisor Antonija Mitrović for the invaluable guidance, unending enthusiasm and patience. I would also like to thank my co-supervisor Brent Martin for the ever-ready spot-on advice and the code for ER-Tutor. Thank you to Pramudhi Suraweera for developing KERMIT. Thank you to the past and present members of the Intelligent Computer Tutoring Group for contributing in one way or another to this research. A special thank you to Stellan Ohlsson for the foundation of this research and COSC420 filled with the insightful theories and ideas. Finally thank you to the students who on average gave about 16 hours (s. d. 11.5) of their time to participate in the evaluation study of EER-Tutor. :)

Bibliography

[1] J. R. Anderson. *Rules of the Mind*. Lawrence Erlbaum Associates, 1993.

[2] J. R. Anderson, A. T. Corbett, K. R. Koedinger, , and R. Pelletier. Cognitive Tutors: Lessons Learned. *The Journal of the Learning Sciences*, 4(2):167–209, 1995.

[3] J. R. Anderson and C. Lebiere. *The Atomic Components of Thought*. Lawrence Erlbaum Associates, Mahwah, New Jesry, 1998.

[4] J. Beck, M. Stern, and E. Haugsjaa. Applications of AI in Education. `http://www.acm.org/crossroads/xrds3-1/aied.html`, 1996.

[5] F. Bennett. *Computers as Tutors: Solving the Crisis in Education*. Faben, 1999.

[6] M. Birbeck, J. Diamnod, J. Duckett, O. G. Gudmundsson, P. Kobak, E. Lenz, S. Livingstone, D. Marcus, S. Mohr, N. Ozu, J. Pinnock, Keith, Visco, A. Watt, K. Williams, and Z. Zaev. *Porfessional XML 2nd Edition*. Wrox Press Ltd., 2001.

[7] B. S. Bloom. The 2 Sigma Problem: The Search for Method of Group Instruction as Effective as One-to-One Tutoring. *Educational Researcher*, 13(6):4–16, 1984.

[8] C. Conati and X. Zhao. Building and Evaluating an Intelligent Pedagogical Agent to Improve the Effectiveness of an Educational Game. In *Proceedings of the 9th International Conference on Intelligent User Interface*, pages 6–13. ACM Press, January 2004.

[9] R. Elmasri and S. B. Navathe. *Fundamentals of Database Systems*. Addison-Wesley, fourth edition, 2003.

[10] Franz Inc. Franz Inc: Allegro Common Lisp and Common Lisp Products. `http://www.franz.com/`, 2004.

[11] R. Freedman. Atlas: A Plan Manager for Mixed-Initiative, Multimodal Dialogue. *International Journal of Artificial Intelligence in Education, Workshop on Mixed-Initiative Intelligence*, 1999.

[12] E. Gamma, R. Helm, R. Johnson, and J. Vlissides. *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley Longman Publishing Co., Inc., 1995.

[13] D. Gilmore and J. Self. The Application of Machine Learning to Intelligent Tutoring Systems. In J. Self, editor, *Artificial Intelligence and Human Learning: Intelligent Computer-Aided Instructions*. Chapman and Hall, 1988.

[14] Guy L. Steele Jr. *Common LISP: the Language*. Digital Press, second edition, 1990.

[15] P. Holt, S. Dubs, M. Jones, and J. Greer. The State of Student Modelling. In J. E. Greer and G. I. McCalla, editors, *Student Modelling: The Key to Individualised Knowledge-based Instruction*, volume 125 of *Computer Systems and Sciences*, pages 3–35. NATO ASI, Springer-Verlag, 1994.

[16] John Foderaro, Franz Inc. AllegroServe — a Web Application Server. `http://opensource.franz.com/aserve/index.html`, 2004.

[17] E. Jung, A. Cioroianu, D. Writz, M. Akif, S. Brodhead, and J. Hart. *Java XML Programmer's Reference*. Wrox Press Ltd., 2001.

[18] M. Kay. *XSLT Programmer's Reference 2nd Edition*. Wrox Press Ltd., 2001.

[19] S. P. LaJoie and A. Lesgold. Apprenticeship Training in the Workplace: Computer-coached Practice Environment as a New Form of Apprenticeship. *Machine-Mediated Learning*, 3:7–28, 1989.

[20] B. I. Martin. *Intelligent Tutoring Systems: The Practical Implementation of Constraint-based Modelling*. PhD thesis, University of Canterbury, 2001.

[21] M. Mayo and A. Mitrović. Optimising ITS Behavior with Bayesian Networks and Decision Theory. *International Journal of Artificial Intelligence in Education*, 2001.

[22] A. Mitrović. NORMIT: a Web-enabled Tutor for Database Normalization. In R. Kinshuk, K. Lewis, R. Akahori, T. Kemp, L. Okamoto, C. Henderson, and H. Lee, editors, *Proceedings of the International Conference on Computers in Education*, volume 2, pages 1276–1280, 2002.

[23] A. Mitrović. An Intelligent SQL Tutor on the Web. *International Journal of Artificial Intelligence and Education*, 13(2–4):173–197, 2003.

[24] A. Mitrović, K. R. Koedinger, and B. Martin. A Comparative Analysis of Cognitive Tutoring and Constraint-Based Modelling. In P. Brusilovsky, A. Corbett, and F. de Rosis, editors, *Ninth International Conference on User Modeling*, pages 313–322. LNAI 2702, Springer-Verlag, 2003.

[25] A. Mitrović, B. Martin, and M. Mayo. Using Evaluation to Shape ITS Design: Results and Experiences with SQL-Tutor. *International Journal of User Modeling and User-Adapted Interaction*, 12(2–3):243–279, 2002.

[26] A. Mitrović, M. Mayo, P. Suraweera, and B. Martin. Constraint-based Tutors: a Success Story. In J. V. L. Monostori and M. Ali, editors, *Proceedings of the 14th International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems*, pages 931 – 940. Berlin: Springer, 2001.

[27] A. Mitrović and S. Ohlsson. Evaluation of a Constraint-based Tutor for a Database Language. *International Journal of Artificial Intelligence and Education*, 10:238–256, 1999.

[28] A. Mitrović, P. Suraweera, B. Martin, and A. Weerasinghe. DB-suite: Experiences with Three Intelligent, Web-based Database Tutors. *Invited paper for the special issue of Journal of Interactive Learning Research (JILR), on Computational Intelligence in Web-based Education*, 15(4):409–432, November 2004.

[29] Network Working Group. RFC 2616, Hypertext Transfer Protocol, HTTP/1.1. `http://www.w3.org/Protocols/rfc2616/rfc2616.html`, 2004.

[30] A. Newell and P. S. Rosenbloom. Mechanisms of skill acquisition and the law of practice. In *The Soar Papers: Research on Integrated Intelligence*, volume 1, pages 81–135. MIT Press, 1993.

[31] S. Ohlsson. Constraint-Based Student Modelling. In J. E. Greer and G. I. McCalla, editors, *Student Modelling: The Key to Individualised Knowledge-based Instruction*, volume 125 of *Computer Systems and Sciences*, pages 167–189. NATO ASI, Springer-Verlag, 1994.

[32] S. Ohlsson. Learning From Performance Errors. *Psychological Review*, 103, 1996.

[33] S. Ohlsson. Learning and Instruction: An Introduction. COSC420 Course Reading Material, July 2004.

[34] S. J. Payne and H. R. Squibb. Algebra Mal-rules and Cognitive Accounts of Errors. *Cognitive Science*, 14:445–481, 1990.

[35] J. A. Self. Bypassing the Intractable Problem of Student Modeling. In C. Frasson and G. Gauthier, editors, *Intelligent Tutoring Systems: at the Crossroad of Artificial Intelligence and Aducation*, pages 107–123. Ablex Publishing Corporation, Norwood, NJ, 1990.

[36] Sun Microsystems, Inc. Java Technology. `http://java.sun.com/`, 2004.

[37] P. Suraweera and A. Mitrović. KERMIT: A Constraint-Based Tutor for Database Modelling. In S. A. Cerri, G. Gouardères, and F. Paraguaçu, editors, *Proceedings of the Sixth International Conference on Intelligent Tutoring Systems ITS 2002*, pages 376–387. LNCS 2363, Springer-Verlag, Berlin, 2002.

[38] K. VanLehn, R. Freedman, P. Jordan, C. Murray, R. Osan, M. Ringenberg, C. Rosé, K. Schulze, R. Shelby, D. Treacy, A. Weinstein, and M. Wintersgill. Fading and deepening: The next steps for andes and other model-tracing tutors. In G. Gauthier, C. Frasson, and K. VanLehn, editors, *Proceedings of Fifth International Conference on Intelligent Tutoring Systems*, Lecture Notes in Computer Science, pages 474–483, Montréal, Canada, 2000. Springer-Verlag Telos.

[39] G. Weber and M. Specht. User Modeling and Adaptive Navigation Support in WWW-based Tutoring Systems. In *International Conference on User Modeling*, 1997.

[40] R. A. Wilson and F. C. Keil, editors. *The MIT Encyclopedia of the Cognitive Sciences*. The MIT Press, 1999.

Appendix A

Pre- and Post-tests

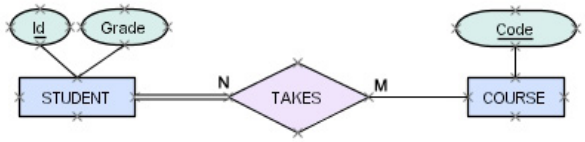
The following test were used in the evaluation study of EER-Tutor.

A.1 Version 1

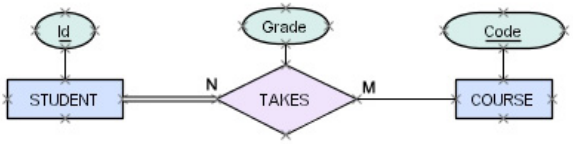
Question 1

Which of the given ER diagrams corresponds best to the following requirements?

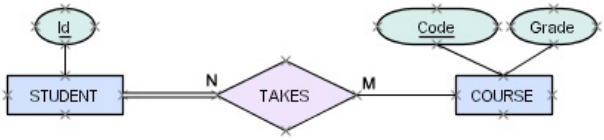
> *For each course a student has taken, we need to know the final grade. Each course has a unique course code and a student has his/her student id.*



(a)



(b)



(c)

Question 2

If an entity type has a multivalued attribute, then

(a) Each entity of this type can have one of several values for that attribute.

(b) There are some entities of this type that have more that one value for that attribute.

(c) Each entity of this type has more than one value for that attribute.

(d) There are many valid values for that attribute.

Question 3

The values of the completeness constraint are:

(a) Disjoint or partial.

(b) Disjoint or overlapping.

(c) Partial and total.

(d) Overlapping or partial.

(e) Total or disjoint.

Question 4

If there is a specialisation with a single subclass, the specialisation must be total. True or false?

Question 5

A weak entity type participates in the identifying relationship type

(a) Always totally.

(b) Always partially.

(c) Either totally or partially.

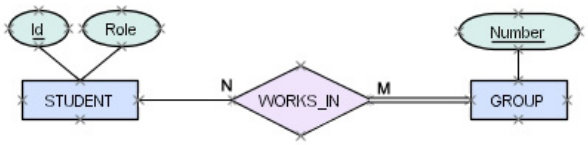
(d) You did not answer this question.

A.2 Version 2

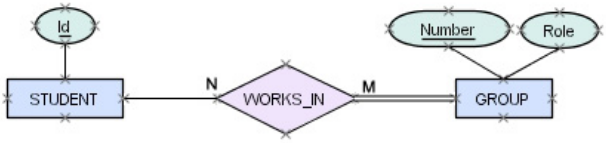
Question 1

Which of the given ER diagrams corresponds best to the following requirements?

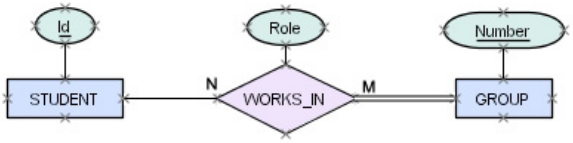
> *Sometimes students work in groups. A student may be a member of several groups, but he/she may have different roles in different groups. Each student has an id, and each group has a unique number.*



(d)



(e)



(f)

Question 2

A regular entity type can have more than one key attribute. False or true?

Question 3

A disjoint specialisation means that:

(a) Every entity belonging to the superclass must belong to exactly one subclass.

(b) Every entity from the superclass may belong to one or more subclasses.

(c) Some entities from the superclass will appear at the level of subclasses.

(d) An entity from the superclass may belong to at most one subclass.

Question 4

A derived attribute is

(a) An attribute whose values do not exist for every entity.

(b) An attribute that has several components

(c) An attribute whose values are optional.

(d) An attribute whose values can be derived from other attributes/relationships.

Question 5

A shared subclass is a subclass of two or more superclasses that

(a) Have different keys.

(b) Have the same key.

(c) Either have the same key or have different keys.

Appendix B

DTD for XML Solutions

The following DTD is the formal description of the EER diagrams in XML format produced by EER-Tutor.

```
<!-- individual_solution.dtd -->

<!ELEMENT individual_solution (bounded_component | connector_component)*>
<!ATTLIST individual_solution
        user_name CDATA #REQUIRED
        problem_number CDATA #REQUIRED>

<!ELEMENT bounded_component (position, dimension, (entity | relation | attribute | junction))>
<!ATTLIST bounded_component
        hash_code ID #REQUIRED
        label CDATA #IMPLIED
        tag CDATA #IMPLIED>

<!ELEMENT position EMPTY>
<!ATTLIST position
        xval CDATA #REQUIRED
        yval CDATA #REQUIRED>

<!ELEMENT dimension EMPTY>
<!ATTLIST dimension
        xval CDATA #REQUIRED
        yval CDATA #REQUIRED>

<!ELEMENT entity EMPTY>
<!ATTLIST entity
        entity_type (regular | weak) #REQUIRED>

<!ELEMENT relation EMPTY>
<!ATTLIST relation
        relation_type (regular | ident) #REQUIRED>

<!ELEMENT attribute EMPTY>
<!ATTLIST attribute
        attribute_type (simple | key | multi | partial | derived) #REQUIRED>

<!ELEMENT junction EMPTY>
<!ATTLIST junction
        junction_type (d | o | u) #REQUIRED>
```

```
<!ELEMENT connector_component (position, position, component_reference, component_reference)>
<!ATTLIST connector_component
        hash_code ID #REQUIRED
        role_name CDATA #IMPLIED
        tag CDATA #IMPLIED
        participation (total | partial) #REQUIRED
        cardinality (_NC | _1 | _N) #REQUIRED
        subset (yes | no) #IMPLIED>

<!ELEMENT component_reference EMPTY>
<!ATTLIST component_reference
        component IDREF #IMPLIED
        handle_number CDATA #IMPLIED>
```

Appendix C

Internal Representation Format

The following is the formal description of the internal solutions representation used in EER-Tutor.

```
(("CONNECTORS" .
                <participation ["partial" | "total"]>
                <cardinality | "subset">
                <tag1 | "junction">
                <tag2 | "junction">
                <optional rolename tag>)

 ("ATTRIBUTES" .
                <tag>
                <label>
                <type ["key" | "simple" | "partial" | "derived" | "multi"]>
                <structure ["simple" | "composite" | "nested" | "unresolved"]>
                <parent_tag>)

 ("RELATIONSHIPS" .
                <tag>
                <label>
                <type ["regular" | "ident"]>)

 ("ENTITIES" .
                <tag>
                <label>
                <type ["regular" | "weak"]>)

 ("SUPERCLASSES" .
                <tag>
                <disjointness ["d" | "o"]>
                <completeness ["total" | "partial"]>
                (<subclass_tag_1>
                 <...>
                 <subclass_tag_N>)

 ("SUPERCLASSES" .
                <tag>
                <disjointness ["single"]>
                <subclass_tag>)

 ("CATEGORY-SUPERCLASS" .
                <tag>
                <category_tag>)
```

```
("SUBCLASS" .
              <tag>
              <superclass_tag>)

("CATEGORIES" .
              <tag>
              <completeness ["total" | "partial"]>
              (<category−superclass_tag_1>
               <...>
               <category−superclass_tag_N>)))
```