A DIAGRAM DRAWING TOOL FOR SEMI–AUTOMATIC ASSESSMENT OF CONCEPTUAL DATABASE DIAGRAMS

F.Batmaz and C.J.Hinde

A Diagram Drawing Tool for Semi–Automatic Assessment of Conceptual Database Diagrams

F.Batmaz and C.J.Hinde
Research School of Informatics
Computer Science
Holywell Park
Loughborough University
F.Batmaz@lboro.ac.uk
C.J.Hinde@lboro.ac.uk

Abstract

The increased number of diagram based questions in higher education has recently attracted researchers to look into marking diagrams automatically. Student diagrammatic solutions are naturally very dissimilar to each others. However, it has been observed that there are a number of identical diagram components. This observation forms the basis of our semi–automatic assessment. Identifying identical diagram components in student diagrams needs contextual information about each component. This paper proposes a diagram tool which obtains the contextual information of each component in a conceptual database diagram.

Introduction

Automatic marking of student conceptual database diagrams is a difficult problem like free text marking [1]. However, the assessment process can be altered to make it suitable for automation as long as that alteration is justified educationally. This research investigates requirements of the assessment environment, which can help the examiner during the marking by analysing the existing manual assessment in order to computerise it as much as possible. It is believed that this approach will form the foundation for fully automated assessment. In addition, the research results have some immediate practical uses.

This research focuses on semi-automatic diagram marking. The aim of semi-automation is to reduce the number of sub-diagrams marked by the examiner. This requires identifying and grouping identical sub–graphs in student solutions. This is a similar approach to the Assess by Computer (ABC) Project [2], however the approach used for grouping the diagrams in our research is very different from the ABC Project. The ABC project defines identical components by using those component's attributes (e.g. label, type, Adjacent Boxes). In our research, identical components are defined by the references

to the text describing the scenario. A similar approach is used for intelligent tutoring system in the KERMIT project [3].

The ABC and KERMIT projects have developed their own diagram editors to capture student diagrams. This research also requires its own diagram editor, which is discussed in the diagramming tool section. A prototype of the diagram editor has been tested on students. Results from this may be found in the experiment sections and further work is described in the final section.

Related Work

There are four other recent studies known [1,2,3,5], which are concerned with automatic assessment of conceptual database diagrams. However, there have been many other studies on automatic production and integration of conceptual diagrams. These could be directed at automatic assessment, but are not addressed here.

The DEAP Project [1] at The Open University uses statistical techniques to grade student exam scripts. This work likens imprecise diagrams to free-form text. The associated commercial intelligent free-form text assessor uses latent semantic analysis for marking [4]. In this analysis, to perform a semantic matching between student text and ideal solution, the semantic of a word is determined from the paragraph in which that word occurs. The DEAP Project looks for suitable keywords in student answers to mark free-form text. It has considered a “relationship" in E-R diagram equivalent to a word in text and applied the same statistical technique to grade the diagrams. Their initial results show that the automatic grading of simple diagrams is feasible.

The ABC Project [2] aims to present student design to the human marker after filtering out diagrams which are identical so that the speed and quality of the marking process can be improved. ABC uses graph isomorphism with some heuristics for local metrics of matching diagrams. It is reported that the approach works well on large, artificial, examples, but tests with real examination data produced some unexpected results. The results have shown some matches which are not actually valid (over-match). In their approach, matching is largely dependant on the component labels.

DATsys [5] is part of the Ceilidh system and provides a customizable environment to create various kind of diagrams. Model answers and student diagrams are captured by DATsys and then another Ceilidh module marks the diagrams. The Ceilidh system was originally designed for assessing programming. The system marks, for instance, a student flowchart diagram by first converting the diagram into a BASIC program and then checks the program against the test data. DATsys hasn’t been used to assess ER Diagrams yet. There is some very early stage research of adapting DATsys for ER diagram marking [6].

KERMIT [3] is an intelligent tutoring system aimed at the university-level students learning conceptual database design. KERMIT contains a set of problems and ideal solutions to them. Unlike traditional ITS, it hasn't got a problem solver. The system compares the student solutions to the ideal solution using domain knowledge represented in the form of constraints, which are classified into syntactic and semantic ones. The semantic constraints enable the system to deal with alternative student correct solutions. Correspondences between the components of the student and the ideal solution are found by forcing the student to highlight the word or phrase in the text whenever a new part is added to the diagram. These correspondences are used to fire the appropriate production rule/s in the semantic constraints. In the case of violation of any of these constraints, feedback is generated.

Approach

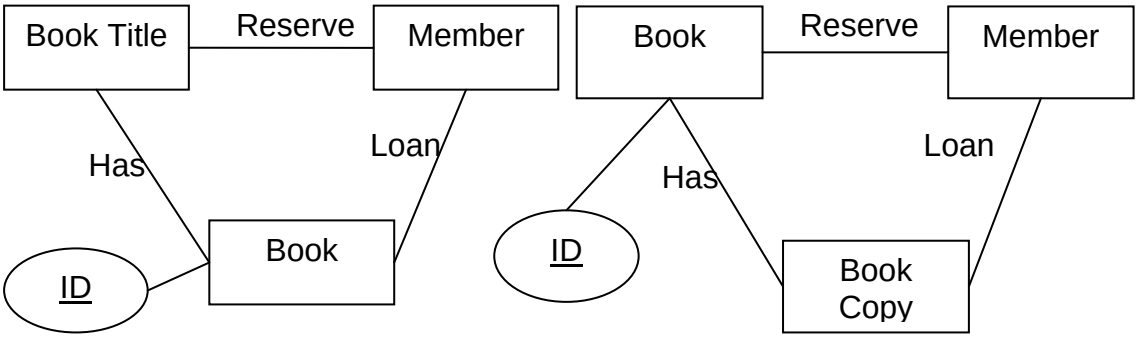
The aim of the semi-automatic assessment is to reduce the number of diagrams marked by the assessor. The system groups identical segments of the student's diagrams and then asks the assessor to approve the correctness of a diagram fragment from the each of the different groups. Therefore the assessor would be involved in the marking process only for the number of diagram groups rather than the total number of student diagrams.

Grouping the diagram pieces not only reduces the marking load but also makes the marking process consistent. The assessor doesn't have to repeat their judgement on the identical diagram pieces from student diagrammatic solutions. This repetition may lead to inconsistency in marking. The approved groups can be automatically graded easily and consistently by the system. Therefore grouping correctly is the key part of the system that enables the system to provide standardised marking.

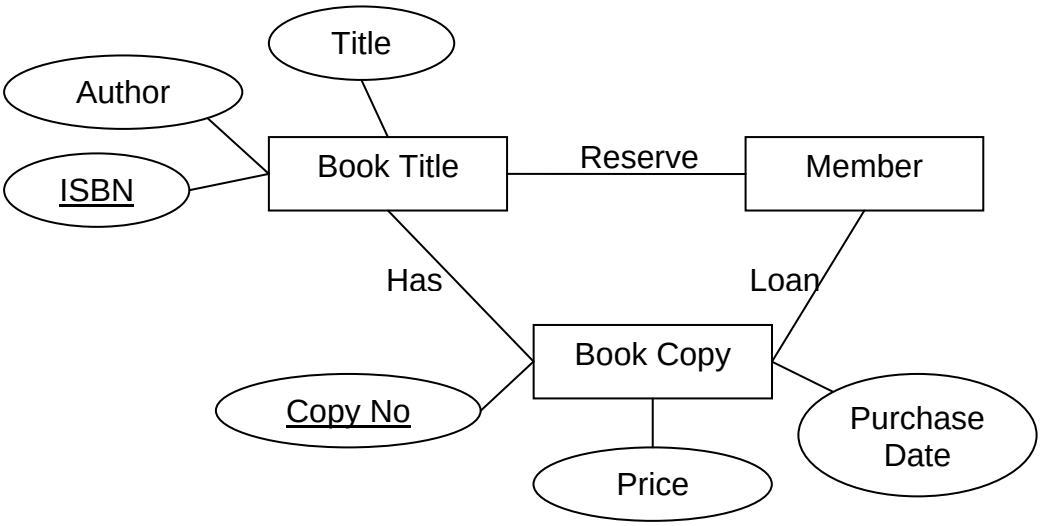
The correctness of the grouping depends on the criteria used to match the diagram pieces. The smallest diagram piece in each group can be either an entity or relationship for a conceptual database diagram. Entities in different diagrams could be considered as matched exactly if they have the same name and the same number of attributes with same name. This initial definition is pretty tight and finding two identical entities among student diagrams may not be trivial. This would increase the number of times the assessor is involved to decide whether the fragment is acceptable or not. However, it might be argued that if the same question is asked many times over the years then it can still be beneficial. Even if we accept this argument, grading a new student diagram by matching previously marked diagram fragments may not work correctly in some cases by using this matching criterion.

The diagrams in figure 1 belong to part of two different student diagrams based on a same scenario. "Book" entity in the first diagram clearly corresponds to "Book Title" with the missing attributes in the teacher solution. However, "Book" entity in the second diagram corresponds to the "Book

Copy" entity. The tool would not get the assessor to mark the second "Book" entity since it matched with the previously accepted "Book" entity by giving it the wrong meaning. Therefore, even the tight definition above is not sufficient for correct entity matching. The definition should also include contextual attributes of an entity. On the other hand, increasing the number of matching criteria required is counter productive.



Student 1 Diagram Student 2 Diagram

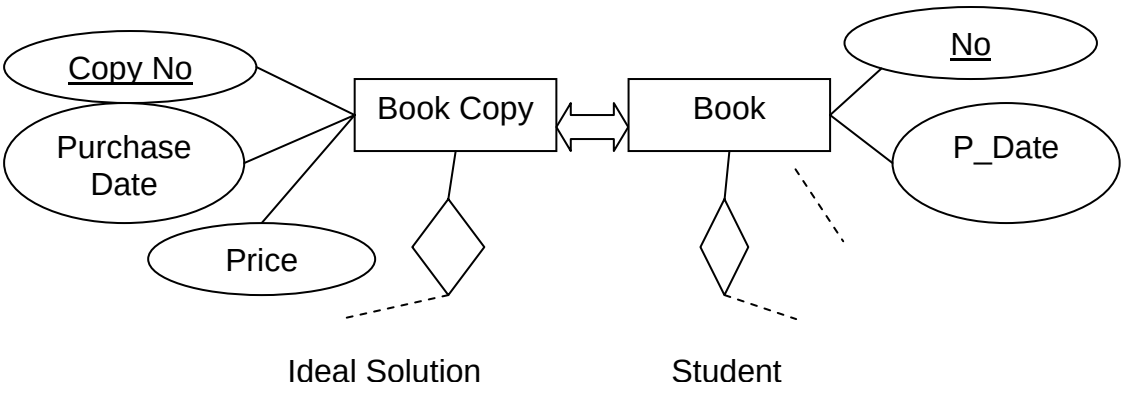


Teacher Diagram

Figure 1. Entity Name Ambiguity

The DEAP Project uses Latent Semantic Analysis (LSA) in order to determine the context of each diagram component. LSA semantically matches a word between the student text and teacher text by means of a factor analysis [4]. It relies on a large corpus of texts to build a high dimensional semantic space containing all words and texts. For instance, the word bike occurs generally in the context of handle bars, pedal, ride, etc [7]. Therefore, if a word like bicycle occurs in a similar context, the two words will be considered close the each other from a semantic point of view. The DEAP Project have recently reported that two small quite different diagrams can be regarded as equivalent [8], which is a result of using LSA. LSA doesn't work properly in the essay marking if the text size is small [7]. The DEAP Project are currently trying to overcome this problem.

In the KERMIT approach contextual meaning of an entity is given by explicitly forcing the students to highlight the related text in the scenarios. This approach simplifies finding a semantic match of the two components automatically (in figure 2). However, finding a related text to diagram components is not a straightforward task [9] and also the direct correspondence sometimes doesn't exist. The main reason is that designing a conceptual database model is an iterative process. Although the initial diagram can have a direct link to the scenario text, afterwards that initial diagram is subject to modification by applying designs rules and constraints in the domain. Although the final diagram can have implicit links to the scenario text, it is not always possible to show those links explicitly without all the intermediate steps between the initial and the final diagram (figure 3).



--- --- ---- ----
Each **book copy** has got a unique copy number and their price and purchase date is recorded.

Figure 2. Entity matching in KERMIT: Book copy and book entity are same concept

Our research suggests using not only the reference text but also the intermediate diagrams in order to define the contextual meaning of a component. However, not all intermediate diagrams are important for the context. For example, a student could initially consider the "book copy" noun phrase in figure 2 as an attribute of an existing entity in their diagram and later on they could change the attribute to an entity. It is not important to know this step to identify that component of the diagram. However, in the case that the student merges "head of department" and "lecturer" entity type to create "staff" entity type (see figure 3), knowing these intermediate diagrams is necessary to be able to match diagram components. We will call the former a direct referenced (DR) component and latter an indirect referenced (IR) component. This research proposes a tool to record the previous diagrams leading to the IR-component only.

The intermediate diagrams used for the contextual meaning also represent students' reasoning process during the design. When the assessor is presented with the intermediate diagrams of each component group for

marking, they can see the process of the students' thinking that enables them to give accurate feedback to students. However an extra caution should be taken not to overwhelm the assessor with so much diagram information during marking. Later our research will investigate how best to display the diagrams to the assessor.

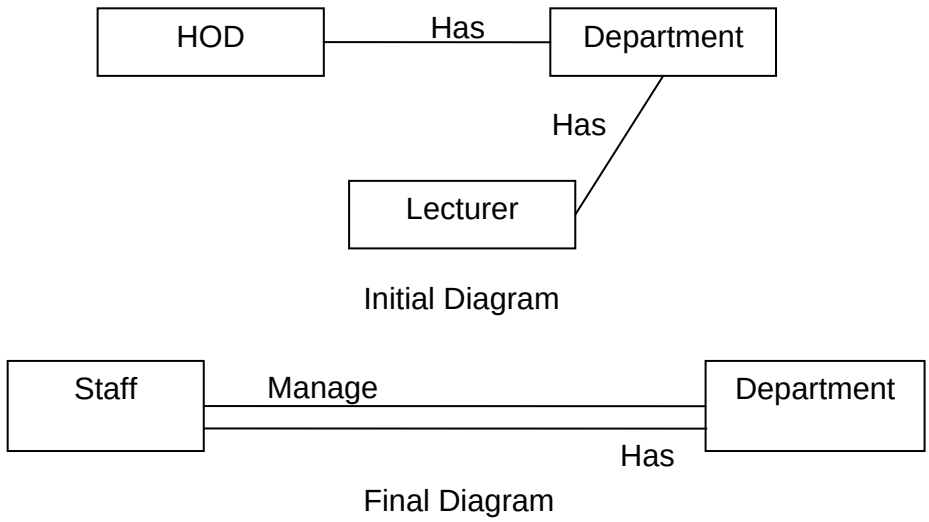


Figure 3. Conceptual Database Design is an iterative process

Merging two entities is one of the diagram modifications which results in IR-components. When the student decides to merge two existing entity types in the diagram, they could modify the diagram in various ways. For example, they might remove those entities and create a new one rearranging all the attributes and relationships of those entities or they might remove one of the entities and rename the other entity. After that, they identify the attributes and relationships of the new entity. These student actions must be interpreted to be able to identify a merging event. Even then, the interpretation may not be what the student intended. We suggest that the student needs to explicitly mention their intention during the design. This method is called self–explanation in the literature [10].

Psychological studies [12, 13] show that self-explanation (SE) is a very effective learning strategy resulting in deep knowledge. SE systems support students while they study solved examples or are asking for an explanation while solving problem. The main problem of self-explanation whilst solving the problem is the high cognitive load [9]. The proposed diagram editor is designed to reduce the cognitive load of self-explanation. The next section looks at components of this diagram editor and examines how cognitive load may be reduced.

Diagram Editor

The prototype diagram editor is based on automatic graph drawing [11]. The editor is an environment to capture student database designs. It is believed

that the student shouldn't have to draw a diagram for their design. They would simply enter the component type and name and then the tool would draw the student diagram. In this way, they can focus more on designing than drawing.

It is also believed that the automatic diagram drawing has advantages over the normal drawing tool in assessment. For example, analysis of database exam scripts reveals that students often change their diagram during the design. Moreover, some of them redraw the whole diagram when they have finalised the design. The automatic drawing could save student time during the exam in this case. Additionally, Thomas [8] found some evidence that the different orientation (shape) of identical student diagrams could be graded differently. The inconsistency of the marking can be prevented by the automatic drawing tool since it always draws the diagram in the same shape for an identical design.

The prototype editor consists of three sections; scenario text, diagram display and diagram modification sections. The scenario text section shows the scenario paragraph by paragraph so that the student considers the information in that section only. This method is called scaffolding in the self-explanation literature [12]. This section also has a feature to highlight the referenced noun phrase and sentences for the selected component. As for the diagram display section, it simply shows the automatically drawn ER-diagram of the student design. In the prototype the database diagram is not drawn or refreshed until the "Draw" button is pressed.

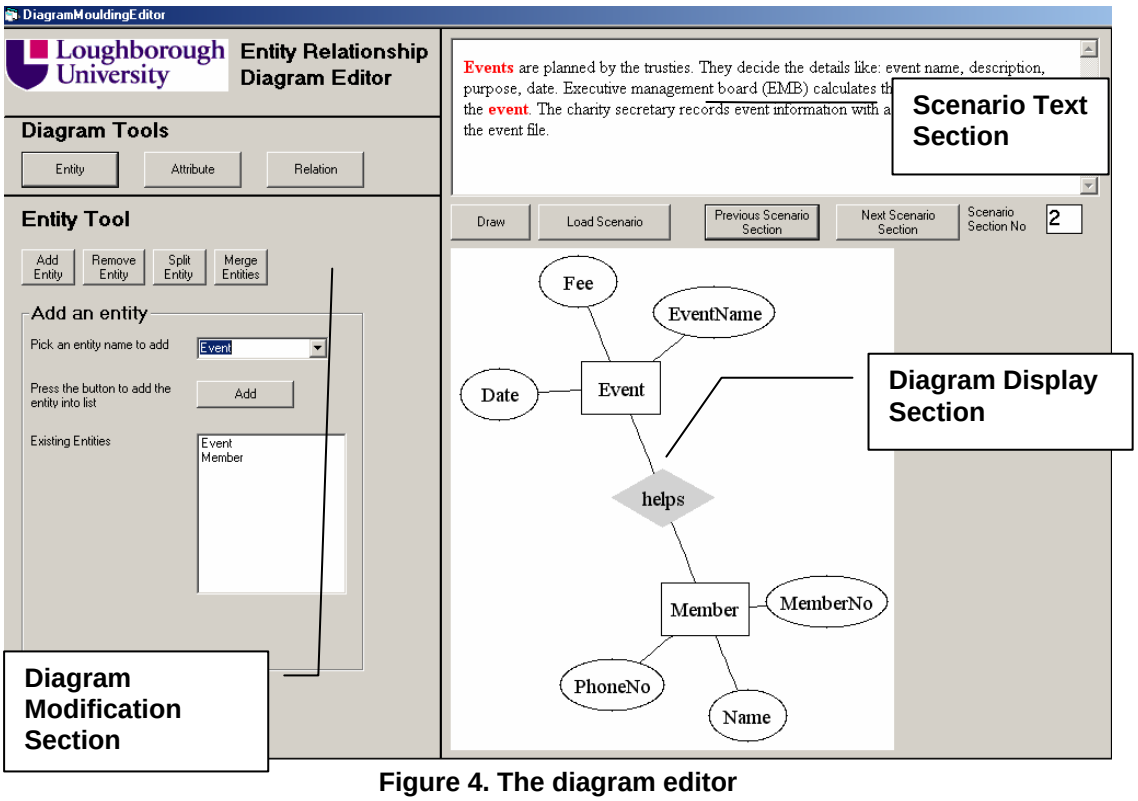


Figure 4. The diagram editor

The diagram modification section is the main part of the editor. In this section the student can add new components or modify existing ones. To create a

new entity type or attribute the student picks the component name from a list. The list has got all different noun phrases appearing in the current paragraph of scenario text. In this way, direct reference of the component is captured. Unlike KERMIT, the editor does not allow the student to name the DR-component. It is believed that the naming sometimes causes inconsistencies between student diagram and the referenced phrase. For example, the student can highlight “member” noun and name “book title” to create an entity type. KERMIT also forces the student to highlight the noun phrase in the text rather than picking it from the list. The “picking” method is suggested to reduce cognitive load without losing any educational proprieties of the assessment. However, research is needed to compare the “highlighting” with the “picking from list” methods.

The student can also modify the diagram by changing existing components. The editor provides function buttons to apply this modification on components. For example, to split an entity into two entities, the student presses “split function” button and then fills the required fields. These buttons reduce the cognitive load of self-explanation.

Database modelling is an iterative process [9]. Students produce their design incrementally for the system. Students start the design with an initial diagram by identifying entities from noun phrases and identifying relationships from verbal expressions. Then they apply the design rules and system constraints to build their design until it satisfies all the system’s requirements. This conceptual database design methodology is supported in the editor. “Scenario scaffolding”, noun phrase list for each section and “Function buttons” are the important features of the editor forcing students to design their database model systematically

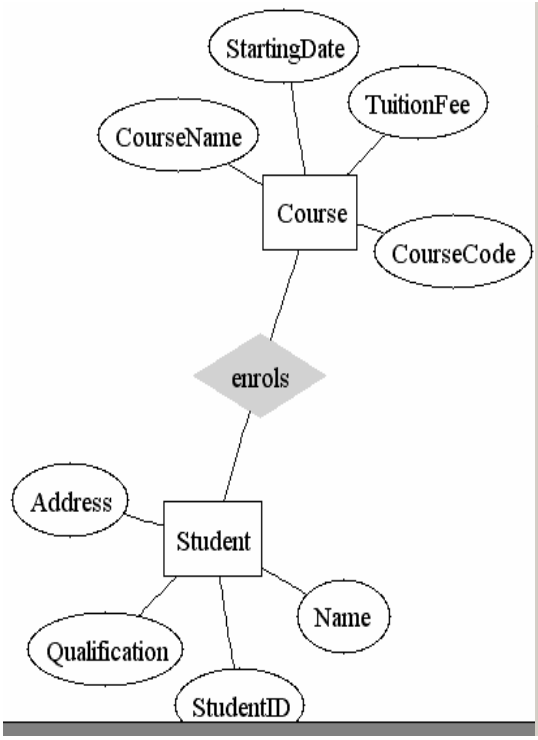
"The Blue Computer Training School *(BCTS)* provides a wide range of computer training short courses. *BCTS* 's manager gives you the following description of the business:

- The administrator records the details of any new course: course code, course name, description, level, tuition fee, and starting date.
- The details of new students are kept into the student file. The school needs to know their name, address and qualification. Each student is assigned a unique student id.
- A student may enrol on several courses. At the end of a course, the student is assessed and the grade achieved is recorded.
- Same course is offered several times a year. Students select a suitable starting date of the course during the enrolment.
- If necessary, the tuition fee of the same course is adjusted whenever it is offered. "

Figure 5. Sample Scenario Text

The scenario test in figure 5 requires using the “split” function button during design. The editor displays each bullet point of the scenario separately. The user sees the list of noun phrases which are in the current bullet point. Then they select a noun phrase to create an entity or an attribute of an entity.

Figure 6 shows an intermediate diagram of a user for this scenario. When the user considers the last two sections of the scenario, they may modify the diagram. The user needs to apply the “Split” function button for this modification (Figure 7), they then create relationships between “Course” and “Course offering” entities to reach the final ER-Diagram (figure 8).



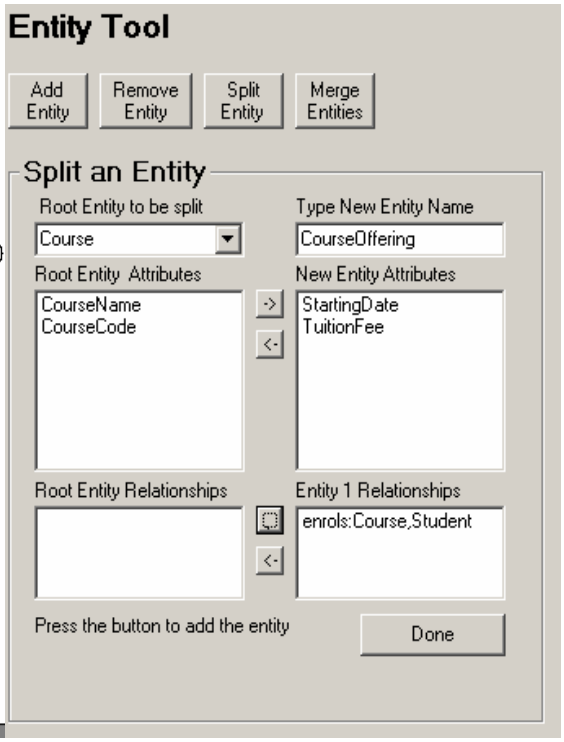


Figure 6. The intermediate diagram **Figure 7. "Split" function button box**

The tool is designed to have function buttons for diagram modifications which result in IR-components. However, function buttons for other kinds of modification can be also created. For example, changing an attribute to an entity type can be done by using function buttons. In this way, eventually, the reasoning processes of students can be gained as well as their final diagram. The examiner is able to understand student behaviour better and give more

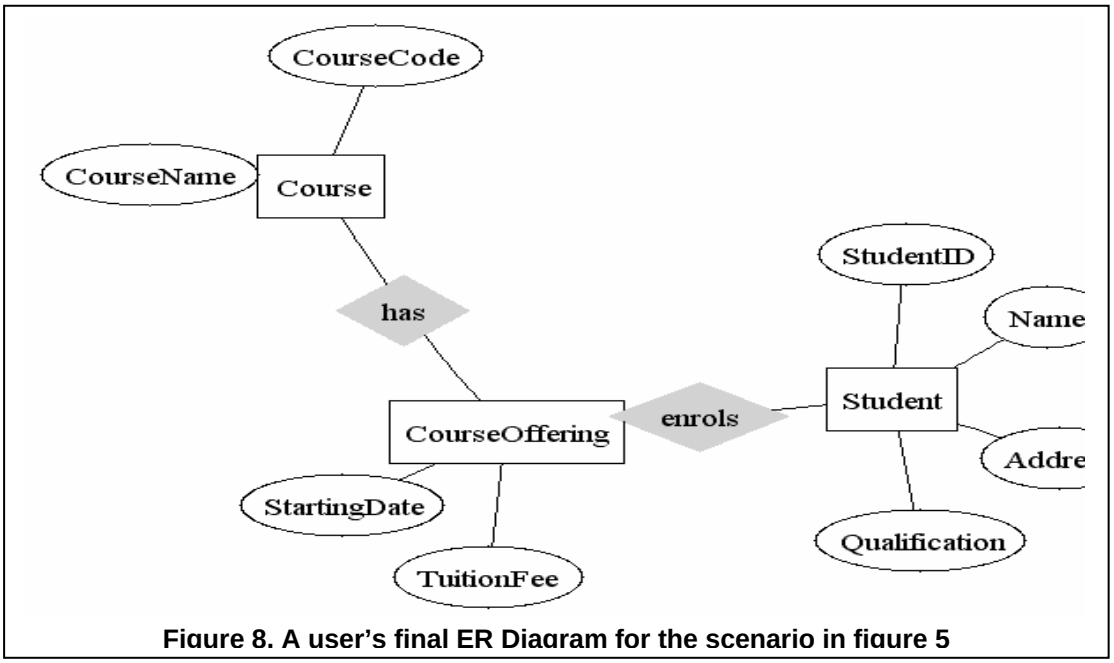


Figure 8. A user’s final ER Diagram for the scenario in figure 5

detailed feedback. On the other hand, there should not be too many function buttons since it increases the cognitive stress.

The usability of the diagram editor depends on the way the scenario text is written. If the scenario text is written in such a way that all the diagram components of the teacher's ideal solution are explicitly mentioned, then function buttons will not be needed. On the other hand, scenario text can be written in such a way that the student has to use function buttons to express their design or using the function button makes the design easier.

Experiment and Results

The diagram editor has two aspects. The first aspect of it is to capture contextual meaning of diagram components. This would help the examiner during marking. The second aspect is to provide an environment for the student to enter their design. Because of these aspects, the editor has a very different environment from those of traditional diagram drawing tools.

The users chosen for the experiment were people who have studied database design at university level. They were given an introduction session and shown how to use the editor on one example database scenario. The given example scenario uses one of the function buttons. Then the users are asked to design a conceptual database diagram for a similar scenario.

All the participants managed to draw the correct diagram. Although the given scenario didn't allow them to design the diagram without using the function button, none of them failed to use the editor. They all applied the required function button to modify the initial diagram during the design.

The required function button for the design expects an entity name from the user. All participants named the entity differently as expected. Different names for the same entity are not a problem for our approach since contextual information of the component is the main criteria for the entity match and this context is provided by use of the function button.

Conclusion and Further Work

The research investigated semi-automatic assessment which helps the assessor by reducing the number of diagrams to be marked. This paper proposes a new diagram editor which alters the traditional diagram drawing in order to make the assessment process suitable for semi-automation. This alteration removes the ambiguity of the contextual meaning for each component during marking. It also enables the assessor to better understand the student thinking and give accurate feedback to students. The prototype editor provides an environment in which the students can design the database model methodically and self-explain their design.

The editor was tested and initial results are very encouraging. They show that by using this editor the student design and contextual meaning of each design component can be captured without increasing the cognitive load on the student. However, further experiments are needed. Types of user and scenario are main factors which could affect the results. The users chosen could be students who are learning about conceptual database design, rather than experienced designers, and the given scenario could be written in such a way that it enforces the use of different combinations of the function buttons. Further experiments will only be done after completing the prototype editor. Currently the tool only has basic function buttons for a particular scenario type. All function buttons for different scenario types will be implemented.

The prototype has not focused on the “ease of use” aspect of the editor so the Interface needs to be made more user-friendly before the editor is used by students.

The other part of our semi–automatic assessment is the marker environment. The editor is a beneficial tool only if the contextual information of each component can be used by the marker environment to match them correctly. Therefore, implementation of this environment and experiments on it are also very important to complete the research.

Initial results with experienced database designers suggest that the tool is useful for designing database diagrams in their professional lives. This is not a current focus of the work but may become more important later on.

References

DEAP Project, Open University, LTSN-TLAD 2004 talk

Tselonis C Sargeant J McGee Wood M (2005). Diagram Matching for Human-Computer Collaborative Assessment Proceedings of the 9th CAA Conference, Loughborough University.

Suraweera, P., Mitrovic, A: KERMIT: a Constraint-based tutor for database modelling. Proc ITS'2002, LCNS 2363, 2002,377-387.

Landauer, T. K., Foltz, P.W. & Laham, D. (1998) an introduction to Latent Semantic Analysis.

Tsintsifas A., A framework for the computer-based assessment of diagram-based coursework, PhD thesis University of Nottingham UK,2002

Brett Bligh, Automatic Assessment of Diagrams, Feasibility Report, University of Nottingham UK, 2002

P. Dessus, B. Lemaire and A. Vernier, "Free Text Assessment in a Virtual Campus", Proceedings of the 3rd International Conference on Human System Learning, Europia, Paris, 2000, pp. 61-75.

P. Thomas, Comparing machine graded diagrams with human markers: some observation, Technical Report No 2004/27, DEAP Project.

Suraweera, P. An Intelligent Teaching System for database Modelling, MSc Thesis, 2001.

Chi,M.T.H. , M. Bassok, M.Lewis, P. Reinmann and R. Glaser, Self-Explanations: How students study and use examples in learning to solve problems. Cognitive Science, 1989. 15:p.145-182.

Graphviz is open source graph visualization software: http://www.graphviz.org

A, Bunt, C Conati, and K, Muldner, "Scaffolding Self-Explanation to Improve Learning In Exploratory Learning Environments", ITS 2004, LNCS 3220.