A revision tool for teaching and learning sequence diagrams

Pete Thomas
Centre for Research in Computing
Open University
Milton Keynes, UK
p.g.thomas@open.ac.uk

Kevin Waugh
Centre for Research in Computing
Open University
Milton Keynes, UK
k.g.waugh@open.ac.uk

Neil Smith
Centre for Research in Computing
Open University
Milton Keynes, UK
n.smith@open.ac.uk

Abstract: This paper describes a software tool that has been developed to help students learn to produce UML sequence diagrams (SDs) in the context of object modelling. The tool, called the SD-Exerciser, developed from an earlier tool for learning about entity-relationship diagrams, presents a student with a series of questions and invites the student to produce SDs that represent the interactions for the use cases embedded in the questions. The SD-Exerciser marks each attempt at a diagram and provides several feedback perspectives on the student's answer. A new development is the ability to check a sequence diagram for syntax errors some of which the tool will attempt to repair. The SD-Exerciser incorporates an automatic marking algorithm based on the successful ERD marker and experiments to date show that it continues to be effective in the new situation.

A sequence diagram (SD) is a graphical notation used to describe how groups of objects collaborate in some behaviour (Stevens & Pooley, 2000). They are one of two types of interaction diagrams defined in the Unified Modeling Language (UML) (Pilone, 2005). When teaching object-oriented analysis and design, it is quite common to start with class diagrams to show the relationships between objects and then move to sequence diagrams to represent the messages that flow between objects when describing a use case.

Our interest in sequence diagrams stems from our work on understanding diagrams [Smith et al. 2004]), in particular the automatic marking (grading) of assignment and examination questions; an area of increasing interest [Batmaz & Hinde 2006, Higgins & Bligh 2006, Hoggarth & Lockyer 1998]. Our investigations into machine interpretation of diagrams have resulted in a framework which has been exploited in a tool to automatically mark student answers to assignment and examination questions involving entity-relationship diagrams (ERDs) [Thomas et al. 2007a,b]. The trials of the automatic marker have provided evidence that our marking algorithm performs well. We have exploited its capability in a tool, which we have named the *Exerciser*, for practising drawing ERDs in a data modelling context [Waugh et al. 2007]. The Exerciser contains a number of data modelling questions, presented as scenarios, about which students attempt to draw representative ERDs. The tool can mark the diagrams and provide feedback on the accuracy of the diagrams. The tool is intended primarily as a practice and revision tool, to be used following instruction in data modelling techniques. The usefulness and usability of the Exerciser was reported to EDMEDIA in [Thomas et al. 2007a].

ERDs are a type of graph-based diagrams with a relatively simple structure. Therefore, we wanted to see whether our approach to diagram understanding would extend to more complex diagrams - but still graph-based. In terms of

complexity, UML class diagrams are no more difficult to deal with than ERDs (and it would be straightforward to amend the Exerciser to deal with class diagrams). Sequence diagrams, however, have a richer structure and pose some new problems. Therefore, we embarked upon an investigation of how well our approach to the automatic marking of graph-based diagrams would cope with the added complexity of SDs.

In this paper we describe how we have developed the Exerciser revision tool to cope with SDs, the new challenges it revealed and how it has led to a more comprehensive teaching and learning strategy.

This paper is structured as follows. The following section describes the *SD-Exerciser's* functionality and the feedback it provides. The third section briefly describes the automatic marking algorithm and its effectiveness. The final section summarises our experiences to date and outlines the work we intend to pursue in the future.

The SD-Exerciser Revision Tool

The SD-Exerciser (see Figure 1) has the same three distinct areas as its predecessor: a question (also known as a scenario) in the left-hand pane, an SD diagram drawn by the student in the top-right-hand pane, and a model solution in the bottom-right-hand pane. The model solution is not visible when the student is constructing their diagram. The tool compares the student's diagram with the model solution using a marking algorithm and provides various levels of feedback on how well the student's diagram matches the model solution.

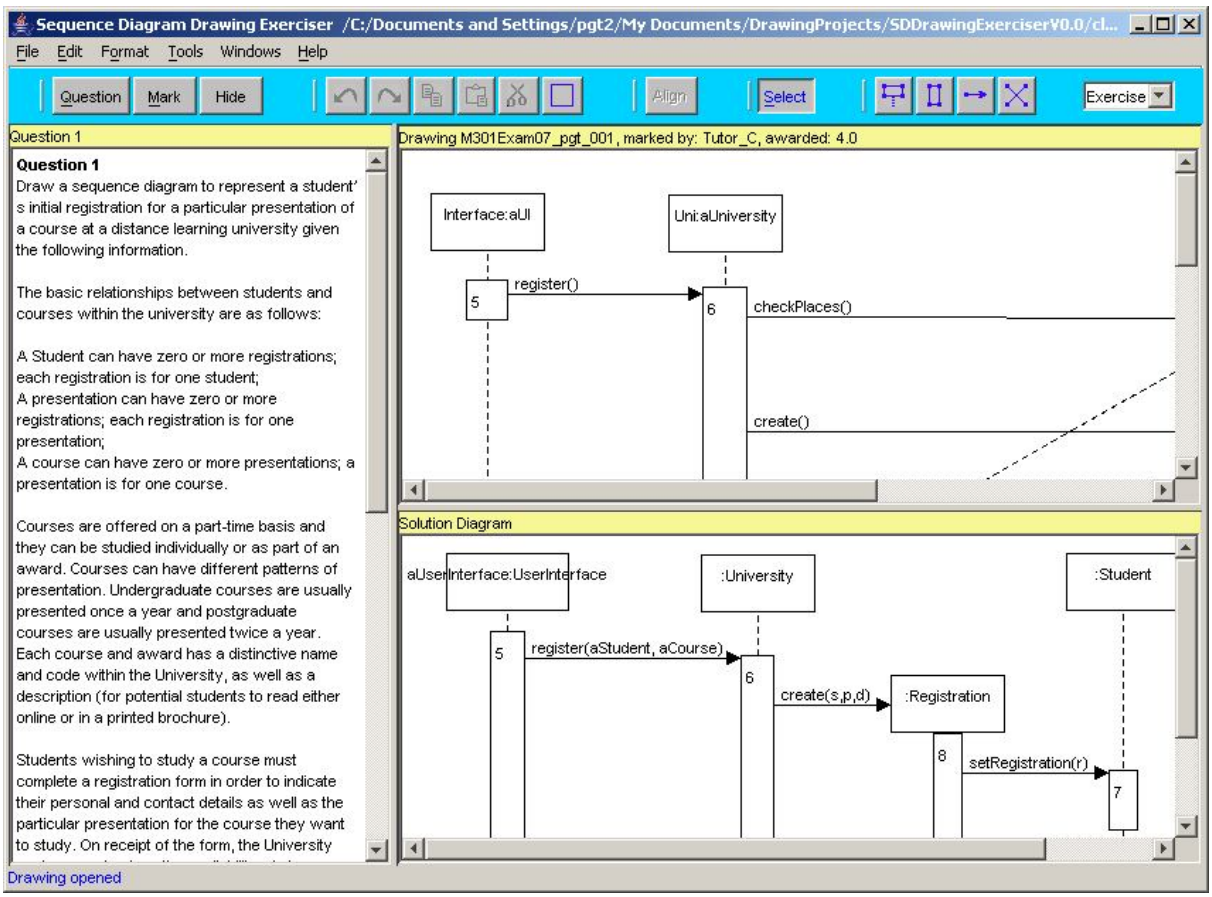

Figure 1: The SD-Exerciser revision tool

The only visible difference between the SD-Exerciser and the original *Exerciser* are the drawing tools (seen on the right-hand end of the tool bar in Figure 1) specialised for sequence diagrams. Having drawn a diagram in response to the given question, the student clicks on the 'Mark' button and receives the feedback shown in Figure 2.

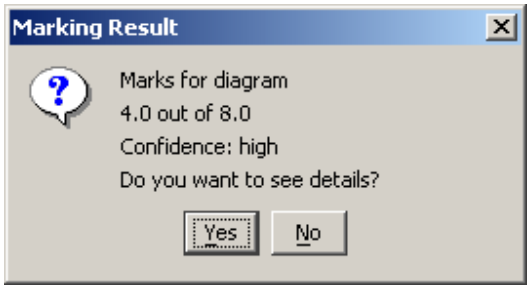

Figure 2: The result of marking a diagram

The initial feedback is simply a grade that indicates the degree of correctness of the student's diagram as a whole and an estimate of how accurate the automatic marker is in its assessment. The automatic marker measures the similarities of certain kinds of elements within the student diagram and the model solution that we refer to as meaningful units (MUs), and estimates how well it believes it has computed these similarities. The student can then request details of what the automatic marker believes are the correct and incorrect elements within their diagram. This information is provided graphically as illustrated in Figure 3.

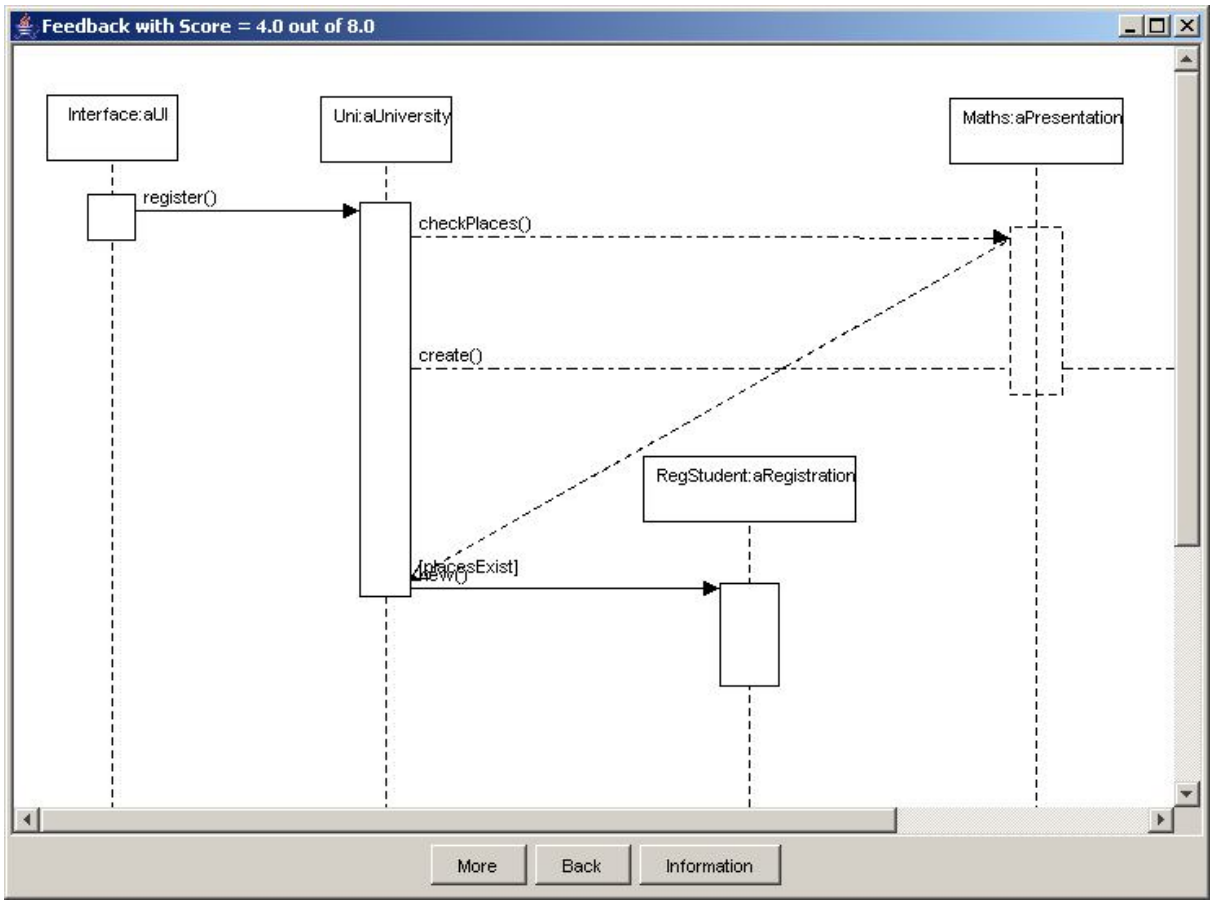

Figure 3: Feedback from the SD-Exerciser

Those parts of a student's diagram that are deemed incorrect are drawn with alternating dot and dashed lines; correct elements are drawn with solid or dashed lines (according to the UML standards). A textual summary of the results of this processing can be obtained by clicking on the 'Information' button.

Students can revise their diagrams and resubmit them for marking as many times as they like. Alternatively, they can reveal the specimen solution and then directly interrogate it. This is achieved by selecting an element of the solution diagram. The tool will then highlight those parts of the question which give rise to the selected diagram element.

The greater complexity of sequence diagrams when compared with ERDs leads to a significant issue when providing useful feedback to students. We have always taken the view that our teaching and learning tools should not overly constrain a student's ability to draw whatever they wish. That is, we do not want the drawing tool to be 'helpful' by constraining what can be drawn – we want students to make mistakes and then let the tool provide suitable feedback. When we examined the SDs drawn by hand in an examination, we found that almost all diagrams (over 150 of them) contained syntax errors (though some were trivial such as using solid lines for the vertical lifelines when the UML requires dashed lines). Some syntax errors were so bizarre that they revealed significant misunderstanding and provide a valuable opportunity for teaching. We have exploited this opportunity as illustrated in Figure 4.

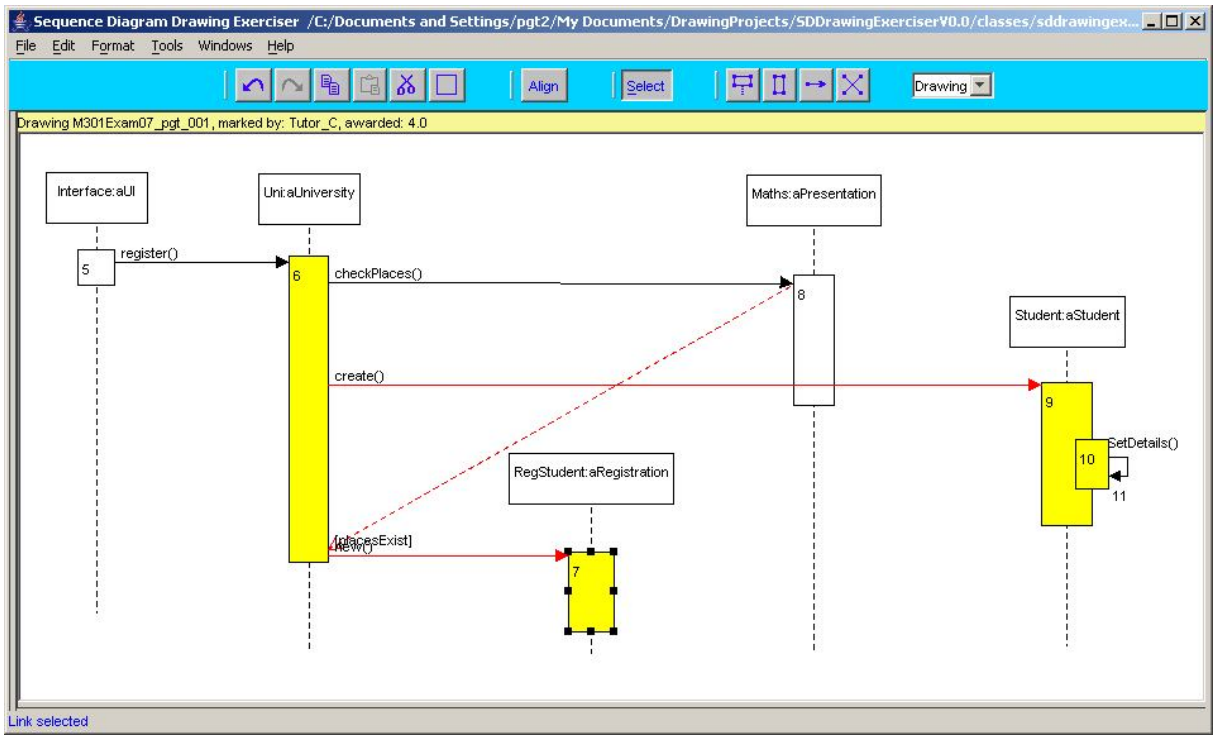

Figure 4: Feedback from the SD-Exerciser

Under the 'Tools' menu there is a tool that checks the syntax of a diagram and presents its findings in diagram form as shown in Figure 4 where incorrect items are highlighted. Clicking on an incorrect item, such as the activation labelled '7', provides some textual feedback as to the nature of the error (see Figure 5).

If required, the tool will also make an attempt at correcting some of the syntax errors (it does not have enough information to attempt to correct all errors as some depend upon what the student was trying to express which is, of course, unknown to the tool). Nevertheless, the student can embark on an iterative process of checking their diagram, examining the repairs suggested by the tool, and amending their diagram. Figure 6 shows the repaired version of the diagram in Figure 4. Once satisfied with their diagram, the student can submit it to the tool for marking and further feedback.

Figures 4 and 6 also illustrate that the Exerciser can be used in 'Drawing' mode which provides a larger canvas for drawing diagrams.

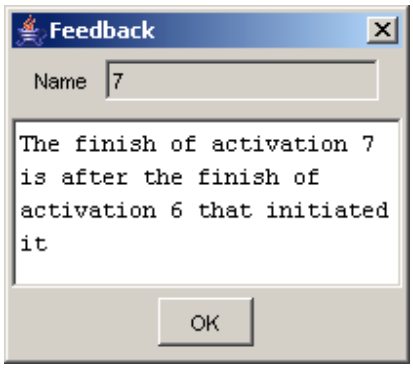

Figure 5: Feedback from the SD-Exerciser

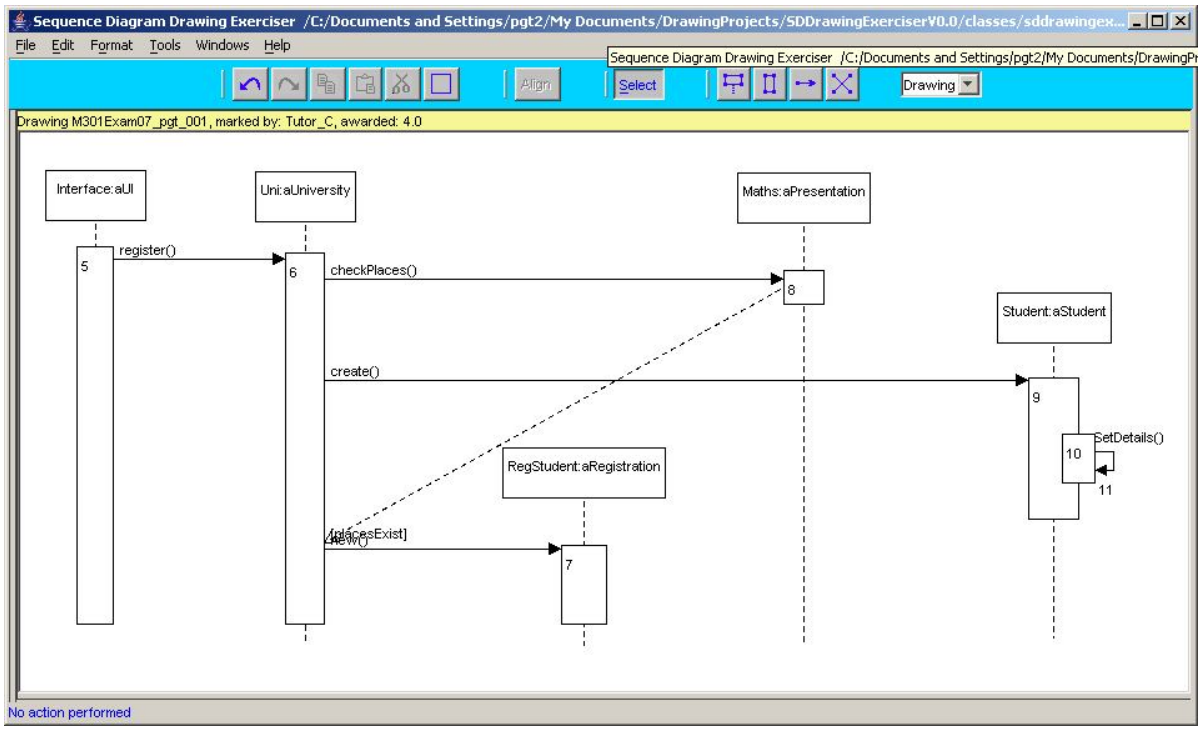

Figure 6: The diagram in Figure 4 shown repaired

As with the original Exerciser, the new tool requires a considerable amount of data to work with: the text of a question, the model answer diagram, a marking scheme and feedback texts. These data are captured using a modified version of our *Tutor Tool*, described in [Thomas et al., 2007].

Automatic marking

The method we use for automatically grading (marking) a diagram is based on a framework for diagram interpretation described in [Smith et al. 2004]. The approach begins by decomposing a diagram into its component parts called minimal meaningful units (MMUs) – the smallest elements of a diagram that has meaning in the domain being modelled. In SDs, the MMUs are objects, messages (there are several types of message, including synchronous and asynchronous, which are represented by arrows with text), and returns (dashed lines with arrows). The MMUs in a student-drawn diagram are compared with the MMUs in a specimen solution diagram (using a

similarity measure based on the properties of the different MMU types). A mark and the feedback are derived from a marking scheme applied to the matched MMUs. Further details can be found in [Thomas et al., 2007]. The basic scheme implemented for ERDs has been modified for SDs with the inclusion of a mechanism for comparing sequences of messages.

We have evaluated the effectiveness of the marking algorithm for SDs on 100 diagrams drawn by students in an invigilated examination. We have three sets of marks from this process. The first are the marks awarded by the exam markers (3 experts). We then reviewed these marks – an independent expert re-marked the student answers to give a second, more consistent and accurate, set of marks which we took as our 'gold-standard'. The third set of marks is that produced by the automatic marker. We used 30 of the student diagrams to train the auto marker (to determine certain thresholds), which left 70 diagrams on which to base our evaluation.

When we compared the automatically produced marks with the reviewed human marks we found that 67.14% were marked identically. The maximum mark for the examination diagram was 8 and the automatic and reviewed mark never differed by more than a single mark for all 70 diagrams. A good approach to comparing two markers is Gwet's AC1 statistic [Gwet, 2001] which, for the reviewed and auto marks, gave a value of 0.7625. Since the critical value for this statistic with N=70 and an 8-point scale, is around 0.15, this allows us to reject the null hypothesis that the marks are being allocated randomly and that we have good inter-rater reliability.

Summary and Future Work

In this paper, we have described the development of a software tool for supporting the teaching of sequence diagrams. The *SD-Exerciser* is based on a revision tool for entity-relationship diagrams that we have successfully used on database courses. The development of the SD revision tool has raised new issues, notably how to deal with syntax errors and the need to process sequences (of messages). The new tool provides additional feedback capabilities over the ERD revision tool, by highlighting syntax errors and attempting to repair those errors as a precursor to the marking stage. In essence, the tool now deals separately with syntax and semantic issues, the latter being the substance of the automatic marking algorithm.

There is an important interplay between class diagrams and sequence diagrams in the UML. The associations between objects shown in a class diagrams specify which objects can communicate directly. Messages shown on a sequence diagram should be consistent with the associations on the class diagram. Therefore, it would be extremely useful to be able to incorporate class diagrams within the questions and provide feedback on attempts to violate this rule.

The work described in this paper has shown that our basic framework for the interpretation of graph-based diagrams continues to apply in a more sophisticated domain. The automatic marker is as effective for sequence diagrams as it has been for entity-relationship diagrams. However, the single example question on which it has been tested was not sufficiently rich to fully test the sequence handling capabilities, which is work that still needs to be carried out.

The ultimate aim of the work presented here is to embed the *SD-Exerciser* into a new software engineering course. It is our intention to test the revised exerciser with students later this year. The immediate need is to produce a new set of questions (using the tutor tool) which are tailored to the new course. We also need to build up further corpora of sequence diagrams on which the automatic marker and feedback mechanisms can be tested.

References

Anderson, M., & McCartney, R. (2003). Diagram processing: Computing with Diagrams. *Artificial Intelligence* **145** (1-2), 181-226.

Batmaz, F. & Hinde, C.J. (2006). A Diagram Drawing Tool for Semi-automatic assessment of Conceptual Database Diagrams. In *Proceedings of the 10th Annual International Conference in Computer Assisted Assessment*. Loughborough University, Loughborough, UK, July 2006, 68-81.

Chok, S.S. & Marriott, K. (1995). Parsing visual languages. *In Proceedings of the Eighteenth Australian Computer Science Conference*, Australian Computer Science Communications, **17**, 90-98.

Haley, Debra Trusso, Pete Thomas, Anne De Roeck, & Marian Petre. (2005). A Research Taxonomy for Latent Semantic Analysis-Based Educational Applications. In G. Angelova, K. Bontcheva, R. Mitkov, N. Nicolov & N. Nikolov (Eds.), *International Conference on Recent Advances in Natural Language Processing'05* Borovets, Bulgaria. 575-579.

Higgins, C. A. & Bligh. B. (2006). Formative Computer Based Assessment in Diagram Based Domains. *In Proceedings of the 11th Annual Conference on Innovation and Technology in Computer Science Education* (ITiCSE 2006), June 26-28, Bologna, Italy, 98-102.

Hoggarth, G. & Lockyer, M. (1998). An Automated Student Diagram Assessment System, ACM SIGCSE Bulletin, **30**(3), 122-124.

Iizuka, K., Tanaka, J. & Shizuki, B. (2001). Describing a drawing editor by using constraint multiset grammars. *In Proceedings of the Sixth International Symposium on the Future of Software Technology* (ISFST 2001), Zhengzhou, China. November, 2001. www.iplab.is.tsukuba.ac.jp/paper/international/iizukia-isfst2001.pdf (accessed 02/06/04)

Manning, C.D. & Schutze, H. (2002). *Foundations of Statistical Natural Language Processing*. The MIT Press, Cambridge, Massachusetts. ISBN 0-262-13360-1.

Marriott, K., Meyer, B., Anderson, M., Cheng, P. & Haarslev, V. (2000). Non-standard logics for diagram interpretation. *In Proceedings of Diagrams 2000, International Conference*, Edinburgh, ISBN 3-540-67915-4-42-57.

Pilone, D. with N. Pitman (2005) UML 2.0 in a Nutshell. O'Reilly Media Inc., Sebastopol, CA, USA. ISBN 0-596-00795-7.

Smith, N, Thomas, P.G. & Waugh, K. (2004). Interpreting Imprecise Diagrams. *In Proceedings of the Third International Conference in the Theory and Application of Diagrams*. March 22-24, Cambridge, UK. Alan Blackwell, Kim Marriott, Atsushi Shimojima (Eds) Springer Lecture Notes in Computer Science, 2980**,** 239-241. ISBN 3-540-21268-X.

Stevens, Perdita with Rob Pooley (2000) Using UML, Updated Edition, Pearson Education Limited, Harlow. ISBN 0-201-64860-1.

Thomas, P.G., Waugh, K., & Smith, N. (2005). Experiments in the Automatic marking of E-R Diagrams. In *Proceedings of the 10th Annual Conference on Innovation and Technology in Computer Science Education* (ITiCSE 2005), Monte de Caparica, Portugal, 158-162.

Thomas, P.G., Waugh, K., & Smith, N. (2006). Using Patterns in the Automatic Marking of ER-Diagrams. In *Proceedings of the 11th Annual Conference on Innovation and Technology in Computer Science Education* (ITiCSE 2006), Bologna, Italy, 403-413.

Thomas, P.G., K. Waugh and N. Smith. (2007a). Tools for supporting the teaching and learning of data modelling. Proceedings of ED-MEDIA conference (ED-MEDIA, 25—29 June 2007, Vancouver, 2007) 4014—4018

Thomas, P.G., K. Waugh and N. Smith. (2007b). Computer Assisted Assessment of Diagrams. Proceedings of the 12th Annual Conference on Innovation and Technology in Computer Science Education (ITiCSE), June 25-29, Dundee, Scotland, 2007), 25—29.

Thomas, P.G., K. Waugh and N. Smith. (2007c). Learning and automatically assessing graph-based diagrams. In S. Wheeler and N. Whitton (eds). Beyond Control: learning technology for the social network generation. Research Proceedings of the 14th Association for Learning Technology Conference (ALT-C, 4-6 September, Nottingham, UK, 2007), 61-74.

Thomas, P.G, N. Smith and K.Waugh (2008) Automatically assessing graph-based diagrams. *Journal of Learning, Media and Technology*, special issue on Reframing e-Assessment. To appear.

Tsintsifas A. (2002). *A Framework for the Computer Based Assessment of Diagram-Based Coursework*, Ph.D. Thesis, Computer Science Department, University of Nottingham, UK.

Waugh, K.G, P.G. Thomas and N. Smith. (2007). Teaching and learning applications related to the automated interpretation of ERDs. Proceedings of the 5th LTSN-ICS Teaching, Learning and Assessment in Databases Workshop (TLAD , July 2007, Glasgow, UK).