# Open Research Online

The Open University's repository of research publications
and other research outputs

## Automatic assessment of sequence diagrams

### Conference or Workshop Item

How to cite:

Thomas, Pete; Smith, Neil and Waugh, Kevin (2008). Automatic assessment of sequence diagrams. In: 12th International CAA Conference: Research into e-Assessment, 8-9 Jul 2008, Loughborough University, UK.

For guidance on citations see FAQs.

Version: Accepted Manuscript

Link(s) to article on publisher's website:
http://www.materials.ac.uk/events/caa-2008.asp

oro.open.ac.uk

# AUTOMATIC ASSESSMENT OF SEQUENCE DIAGRAMS

Pete Thomas, Neil Smith and Kevin Waugh

# Automatic Assessment of Sequence Diagrams

Pete Thomas, Neil Smith, Kevin Waugh
Computing Department, Open University, UK.
p.g.thomas@open.ac.uk

## Abstract

In previous work we showed how student-produced entity-relationship diagrams (ERDs) could be automatically marked with good accuracy when compared with human markers. In this paper we report how effective the same techniques are when applied to syntactically similar UML sequence diagrams and discuss some issues that arise which did not occur with ERDs. We have found that, on a corpus of 100 student-drawn sequence diagrams, the automatic marking technique is more reliable that human markers. In addition, an analysis of this corpus revealed significant syntax errors in student-drawn sequence diagrams. We used the information obtained from the analysis to build a tool that not only detects syntax errors but also provides feedback in diagrammatic form. The tool has been extended to incorporate the automatic marker to provide a revision tool for learning how to model with sequence diagrams.

## Introduction

In previous work we showed how student-produced entity-relationship diagrams (ERDs) could be automatically marked with good accuracy when compared with human markers (Thomas et al., 2007c). In this paper we report how effective the same techniques are when applied to UML sequence diagrams (SDs) and discuss some issues that arise which did not occur with ERDs.

While there are several systems being developed for grading textual material (Burnstein et al., 2003, Haley et al., 2005) and there is a considerable literature for describing diagrams (Anderson & McCartney, 2003, Chock & Marriott, 1995, Kniverton, 1996, Marriott et al., 1998) there is very little work on grading diagrams. Tsintsfas (2002) has produced a framework for the assessment of diagram-based coursework which has fed into an ERD tool within the CourseMarker CBA system (Higgins & Bligh, 2006) and Batmaz & Hinde (2006) have investigated a semi-automatic marking system.

SDs are syntactically similar to ERDs, since both consist of boxes connected by lines. In ERDs entities, represented by boxes, are associated with one another through relationships, denoted by lines. In SDs object activations, denoted by rectangles, are associated with one another through messages, denoted by arrowed lines. Clearly the semantics of ERD relationships is very

different from the semantics of messages, but the underlying syntactic structures are quite similar. Therefore, our aim is to exploit this similarity to grade SDs in the same way that worked so well for ERDs.

Measuring the effectiveness of an automatic marker requires a substantial corpus of student-drawn diagrams for statistically reliable testing. Therefore, we collected a set of 169 hand-drawn sequence diagrams produced by students in a closed-book, invigilated examination. These diagrams had already been graded by experienced academic markers and were independently second-marked and moderated to provide a standard against which our automatic marker could be judged. However, it was clear that students had made a large number and variety of syntax errors which makes marking substantially more difficult both for experienced human markers and an automatic marker. We refer to such error-containing diagrams as imprecise diagrams (Smith et al., 2004). Typically, depending on the nature of the assessment, human markers will compensate for trivial syntax errors with a view to assessing the intended meaning of a student's answer. It is important that our automatic marker gives the same 'benefit of the doubt' when marking scripts.

In this paper we discuss two areas of investigation. First, we examine the nature of the errors made by students when drawing sequence diagrams and how that information has influenced the design of a learning tool. Second, we discuss the effectiveness of our automatic marking approach when applied to SDs. We conclude with a discussion of how the two strands of this research can be combined to provide a comprehensive revision tool to help students construct correct sequence diagrams.

## The corpus of sequence diagrams

The work reported here is based on hand-drawn sequence diagrams produced in an end-of-course invigilated examination. Among other topics, the course teaches subsets of certain UML diagrams (class, sequence, collaboration, and state diagrams). The course is a distance education course and, as a consequence, the subset of UML sequence diagrams taught and the pedagogic approach used is recorded in the printed course materials. This information is helpful in deciding whether an error is due to misunderstanding or a lack of precise teaching.

Since the diagrams were hand-drawn under examination conditions, students were under time pressure which resulted in errors that occurred because the students ran out of time (diagrams were incomplete) or diagrams were drawn too quickly (elements of the diagram were not well-formed). The latter issue is similar to that of poor writing in a conventional written examination (an issue that also occurs in diagrams in the labelling of objects and messages). In our data, there were only two incidents of ill-formed diagram elements that could not be interpreted.

The examination took place in October 2007 and provided 169 hand-drawn sequence diagrams. To-date we have analysed the first 106 diagrams in this corpus and none is totally free of syntax errors. Five of the diagrams confused collaboration and sequence diagrams, and one used a syntax that was more akin to an activity diagram. This left 100 sequence diagrams to be analysed in depth.

**Errors in sequence diagrams**

Sequence diagrams are used to show how a group of objects interact by exchanging messages. In the sequence diagrams dealt with in our course, there are five essentially different drawing elements: rectangles with text inside representing objects, rectangles representing activations (the period when an object is processing a message it has received), vertical dashed lines representing object lifelines (the passage of time is represented vertically downwards), lines with arrowheads and text representing synchronous messages, and dashed lines with an arrowhead representing a return (when control returns to an object once its message has been dealt with – an optional feature of sequence diagrams). The sequence diagram shown in Figure 1 illustrates most of these features but also contains several syntax errors. (Among other things, the drawing tool is used to capture the hand-drawn diagrams as xml files.)
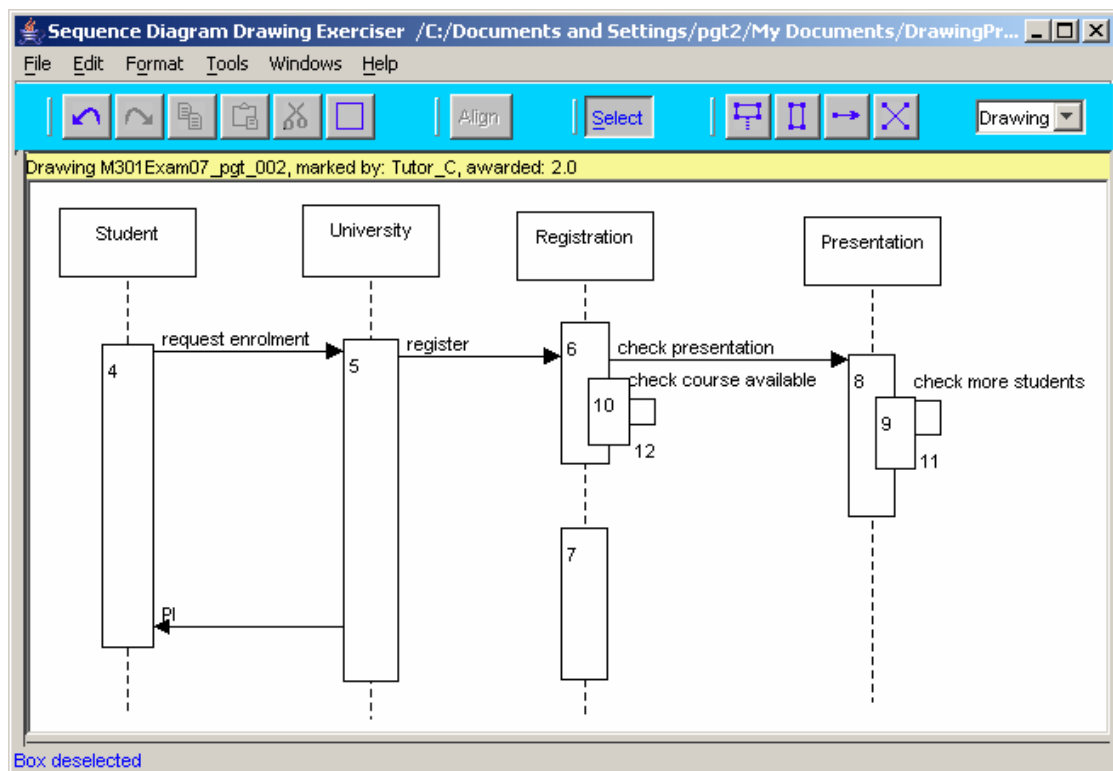


**Figure 1. Sequence diagram drawing tool**

We examined the diagrams and identified 33 types of syntax error. These are listed in Table 1 and are given in order of frequency of occurrence (the number of diagrams containing at least one example of the error).

It is quite clear that many of the errors observed were not due to misunderstanding of sequence diagrams but occurred as the result of, for example, poor drawing skills, time pressure causing diagram elements to be missed or drawn inaccurately, and crossing out leading to unrecognisable elements or a lack of space to draw accurately.

| Message Errors | Freq (%) |
|---|---|
| Wrong arrowhead on a synchronous message | 71 |
| Incorrect syntax for object name | 65 |
| Multiple synchronous messages sent to the same activation | 46 |
| Initiating message not at the start of an activation | 39 |
| Solid line rather than dashed line used for lifeline | 37 |
| Activation finishes after the activation which initiated it finishes | 33 |
| Incomplete activation (activation shown as not finishing) | 29 |
| Missing sub-activation | 29 |
| Missing rectangle for object | 23 |
| Missing arrowhead | 21 |
| Message text associated with return arrow | 20 |
| Solid line rather than dashed line used for return | 17 |
| Arguments in formalised message not related to objects | 16 |
| Return not at end of activation | 15 |
| Creation (new) message sent to activation/lifeline not object | 12 |
| Wrong arrowhead on return | 12 |
| Sub-activation without an initiating message | 12 |
| Missing text for message | 10 |
| Missing lifeline | 7 |
| Return sent to wrong activation | 5 |
| Activation without initiating message | 5 |
| Sub-activation in incorrect place | 4 |
| Non-creation message sent to object | 4 |
| Message sent while blocked | 4 |
| Message sent before activation initiated | 2 |
| Message sent back in time | 2 |
| Message sent from sub-activation to parent | 2 |
| Undecipherable element | 2 |
| Message from sub-activation sent back in time | 1 |
| Message does not start at an activation | 1 |
| Incorrect syntax for denoting a condition | 1 |
| Activation not situated on a lifeline | 1 |
| Invalid activation | 1 |

**Table 1. Message error frequency**

Nevertheless, all errors noted in Table 1 could have arisen through misunderstanding. It is also worth noting that the ambiguous nature of UML diagrams (Morris & Spanoudakis, 2001) causes difficulties for automatic

identification of errors. For example, being able to express the text of messages both formally and informally means that when the informal method is used some errors cannot occur but this does not imply that a student understands the constructs that have not been used.

We categorised the errors in two ways. First, we associated each type of error with either one of the five drawing elements of an SD or the fundamental purpose of an SD, according to whether that type of error would occur primarily as the result of a misunderstanding of that drawing feature. This categorisation helps with the generation of appropriate feedback.

Second, we categorised the errors into three types: those that are easy to correct, those that could be corrected with reasonable certainty and those that are difficult, if not impossible, to correct because to do so would rely on knowledge (unknown) of what the student was trying to express.

Many minor syntactic errors are easily recognised and corrected. For example, a lifeline drawn using a solid instead of a dashed line makes little difference to the understanding of the diagram provided it is in the expected position relative to an object (as were all the examples in our corpus). However, a similar error in drawing a return (using a solid line rather than a dashed line) is much less easy to detect particularly if text is present because it is so similar to a normal synchronous message. Positioning relative to an activation can help to distinguish the two.

Multiple messages sent to the same activation is an error that cannot be corrected with absolute certainty but can be corrected with reasonable certainty in some circumstances. Activations that are incomplete, that is, have no distinguishable end point, cannot be corrected with great certainty.

One of the motivating aims of our work is to provide tools that can help students become familiar with formal diagrams (such as those of the UML) and help them develop solutions to design problems. Our earlier work has resulted in a revision tool for ERDs in which the automatic marker is used to analyse student attempts at solving problems and provide feedback (Thomas et al., 2007b). We do not wish to use the normal professional drawing tools which are designed to produce syntactically correct diagrams preferring instead to allow students the freedom to make mistakes and have the tool point out the mistakes and hence help the students to learn. However, we do not want a tool whose drawing primitives are so basic that it becomes laborious to produce the fundamental drawing elements of the domain (boxes and arrows). Nor do we want a tool that contains too many distractors that could confuse even good students, for example, by having menu items that are illegal but give the impression of legitimacy.

The identification and classification of errors enabled us to produce a drawing tool (see Figure 1) that would allow users to make the majority of errors reported above. These errors could then be corrected and appropriate feedback given. But we wanted a tool that would not be laborious to use: the tool should be helpful when drawing elements where errors never occurred,

but provide a certain amount of latitude in areas where errors frequently occurred. For example, in all hand-drawn diagrams in our corpus, lifelines were drawn in the correct place relative to objects, so we decided that, when the user wishes to draw an object, the tool should draw a rectangle with an associated dashed lifeline leaving the student to type the object's name. However, when drawing a line to represent a message or return, the tool should provide a menu of different arrowheads from which the student must select the one they think is appropriate.

In our current implementation, the following errors are not reproducible:

- incomplete activations
- messages starting other than at an activation or lifeline
- a message sent from a sub-activation to its parent activation
- an object represented other than by a rectangle
- a lifeline represented by a solid line
- an omitted lifeline

*Checking syntax and diagram repair*

The tool shown in Figure 1 incorporates a syntax checking facility (within the Tools menu). This facility can be disabled if required, for example if the tool is used in an exam. The syntax checker identifies errors by shading (objects and activations) and change of colour (messages) – although this mechanism may change after usability testing. A textual description of an error can be obtained by right-clicking on the erroneous element and requesting feedback as shown in Figure 2.
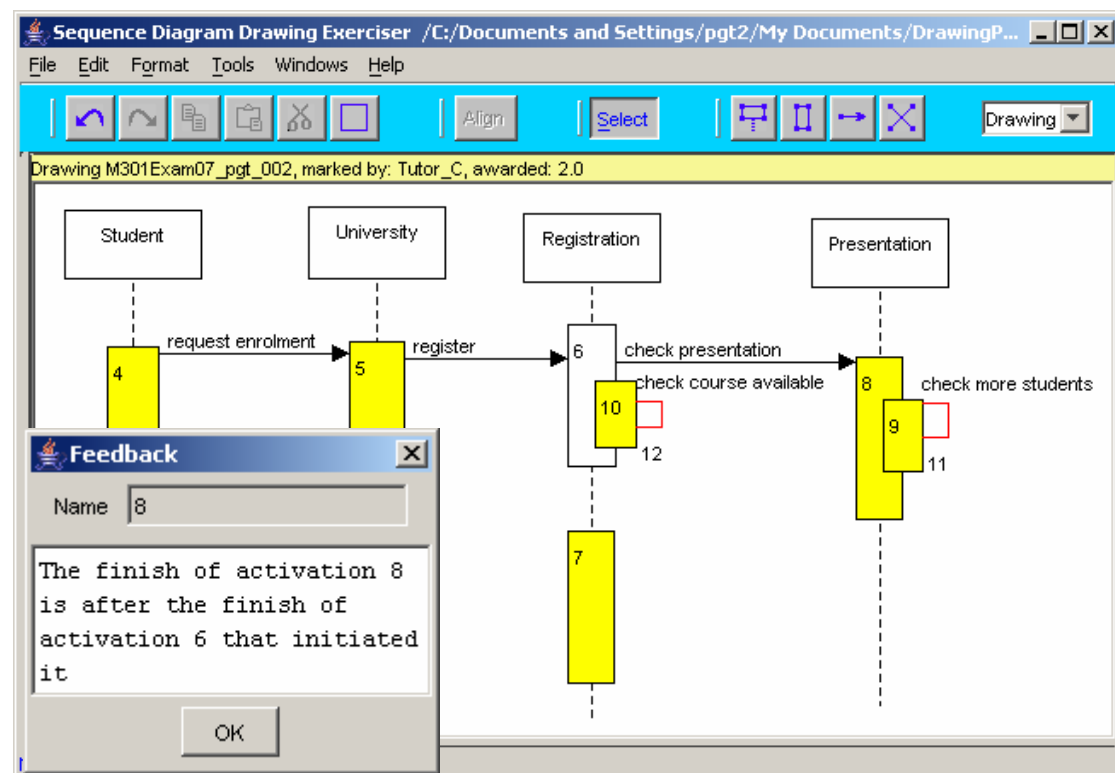


**Figure 2. Sequence diagram drawing tool showing errors**

Having identified errors, the tool can be asked to repair the errors (see Figure 3). Not all errors are repaired – only those that can be dealt with with reasonable certainty. Simple errors, by definition, are those that can be repaired without changing the essential meaning of the diagram. Other errors will be repaired on the basis of the most likely error – there are some patterns which occur frequently and have highly likely causes (see Figure 4).
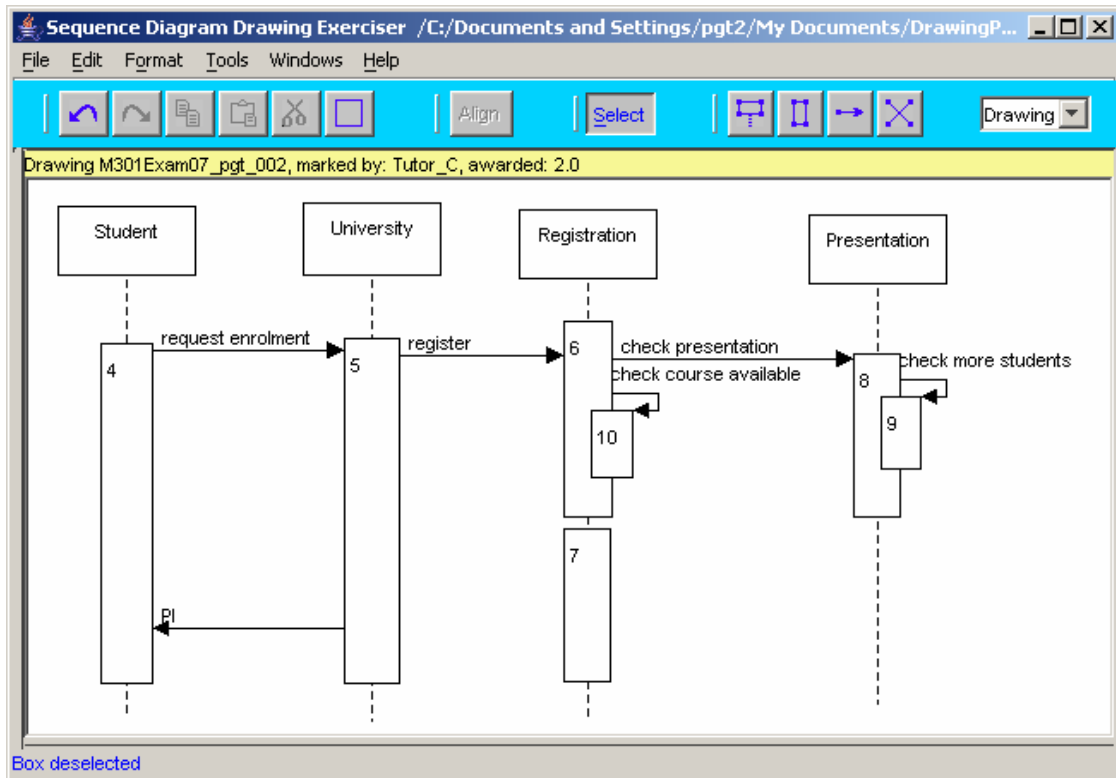


**Figure 3. Sequence diagram drawing tool showing errors repaired**



(a)                                    (b)
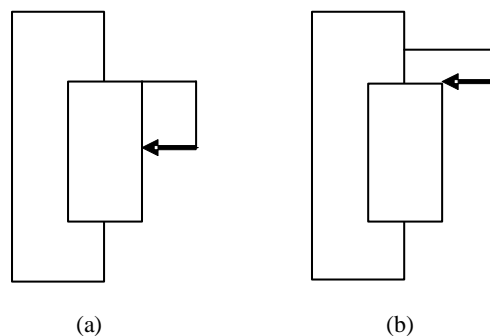
**Figure 4. An incorrect diagram fragment (a) and its repair (b)**

If a repaired diagram is re-checked, errors may still be detected, but these are the irreparable complex errors which the user should attempt to fix prior to submitting the diagram to the automatic marker. Of course, the user may not be able to fix the errors and can submit the syntactically incorrect diagram to the automatic marker which has to operate in the face of such errors.

**Automatic marking of sequence diagrams**

We mark diagrams by comparing a student's attempt with a gold standard – a specimen solution (there could be more than one acceptable solution) – to produce a set of similarity measures for certain diagram features known as minimal meaningful units (MMUs) (Smith et al., 2004) to which the mark scheme used by the human markers is applied. We then compare the results of the automatic marker to the moderated human marks.

The similarity measures are modified by certain weights and thresholds which have to be determined for the data set under test. The weights represent the significance, for marking purposes, of diagram features in a particular domain and are set by the user. Thresholds are determined by experiment and represent limiting values; for example, two objects will be considered the same if the measure of their similarity is above a certain threshold. Following the approach adopted for ERDs (Thomas et al., 2007c), we determine suitable values for the thresholds using a *training set* of diagrams; in this case the first 30 diagrams in our corpus. The trained marker is then applied to the remaining diagrams (the *test set* consisting of 70 diagrams) and the results reported for this test set.

Table 2 shows the differences between the moderated human marks and the auto marks for the test set. Most of the time, the human markers marked to the nearest whole number (as did the automatic marker). The maximum mark was 8.

| Difference | 0 | 0.5 | 1 | >1 |
|---|---|---|---|---|
| Number | 47 | 2 | 21 | 0 |
| % | 67.14 | 2.86 | 30.0 | 0 |

**Table 2. Difference between human and auto marks**

The mean difference for the 70 diagrams was 0.34 with a standard deviation of 0.468.

A good approach to comparing two markers is to use inter-rater reliability and, in particular, Gwet's AC1 statistic (Gwet, 2001). We also present Fleiss's generalised kappa measure (Fleiss, 1971) to enable easier comparison with other marking approaches. We consider Gwet's measure to be superior, as it more accurately accounts for chance agreement between markers (see Gwet (2001) for details). Critical values for these measures are, for two raters, around 0.15 for both AC1 and kappa: agreement measures above these values allow us to reject, with over 99% confidence, the null hypothesis that the marks are being allocated randomly. The results are shown in Table 3 where we have compared the original (unmoderated) mark and the moderated human mark with the automatically generated mark, where the number of categories was 9 (including zero).

| Raters (N=70, 8 point scale) | AC1 | Kappa |
|---|---|---|
| Unmoderated v Auto Mark | 0.1916 | 0.0063 |
| Moderated v Auto Mark | 0.7625 | 0.6371 |

**Table 2. Inter-rater reliability measures**

The results in Table 3 confirm the good correspondence between the automatically generated mark and the moderated mark and illustrate the poor performance of the human markers and the need for moderation.

As an aside, the original, unmoderated marks were produced by three human markers who each marked approximately one-third of the diagrams. We have analysed their individual performance using the AC1 statistic and found considerable differences. At this stage of our investigations we believe that the differences are due to different interpretations of the marking scheme. Fortunately, our University has robust procedures for identifying and dealing with differences in the performance of markers (essential when there can be many markers for a single examination).

## Conclusions and further work

The work presented here deals with the analysis and categorisation of errors in imprecise sequence diagrams and the performance of an automatic marker. The analysis has informed the design of a syntax error-checking tool. That tool not only checks and reports on syntax errors, but will also repair the majority of error types using information gleaned from the error analysis. The checking and repair of diagrams is intended to help students improve their understanding of the fundamental syntactic aspects of sequence diagrams. The semantics of a sequence diagram are dealt with in the automatic marking algorithm.

The software drawing tool does not permit all of the observed errors to be made. The choice of which errors to be prohibited has been pragmatic, but it is based on the observed behaviour across the corpus of diagrams. This will be reviewed in the light of experience with the tool. We intend to evaluate the tool with students later this year to determine its usefulness and usability.

We have begun the development of a revision tool that will add the marking capability to the check and repair tool to provide semantic feedback. This will be similar to a revision tool we have already developed for ERDs (Waugh et al., 2007). In due course, the revision tool will be augmented with wizards to help with the creation of cliché diagrams (Thomas et al., 2006). This will allow three levels of support. First, using it to check for simple syntax errors (aimed at reinforcing the basic principles of SDs), then using its semantic feedback from the automatic marker to improve modelling skills, and finally using support for faster construction of diagrams.

The tools described here have been built on the information gained from an analysis of the errors made by students. We want to place this work on a more formal basis as part of our investigations into diagram understanding. We intend to use constraint multiset grammars (CMGs) (Marriott et al., 1998) to describe SDs. This should allow us to develop a more sophisticated parser for the diagrams, in the manner of tools developed for other diagram domains (Chok & Marriott, 1995), though how to extend such a parser to accommodate the imprecise and malformed diagrams seen in our corpus remains an open question.

**References**

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

Batmaz, F. and Hinde, C.J. (2007) A Web-Based Semi-Automatic Assessment Tool for Conceptual Database Diagram. In the Proceedings of the Sixth IASTED International Conference on Web-Based Education. (Chamonix, France, March 14-17, 2007), 427-432.

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

Fleiss, J. L. (1971) "Measuring nominal scale agreement among many raters." *Psychological Bulletin* **76**(5): 378—382.

Gwet, K. (2001) *Statistical Tables for Inter-Rater Agreement*, Gaithersburg : StatAxis Publishing.

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

Marriott, K., Meyer, B. and Wittenburg, K.B. (1998) A survey of Visual Language Specification and Recognition. In *Visual Language Theory*, eds: Marriott, K and Meyer, B., Springer-Verlag, New York, 8-85, ISBN 0-378-98367-8.

Morris, S. and Spanoudakis, G. (2001) UML: An Evaluation of the Visual Syntax of the Language. In the proceedings of the 34th Hawaii International Conference on System Sciences, 2001. IEEE. Hawaii.

Shermis, M.D, Burstein, J.C. (2003) (eds.) Automated Essay Scoring: a cross-disciplinary approach. Lawrence Erlbaum Associates, Mahwah, NJ, USA. ISBN 0-8058-3973-9.

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

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

Thomas, P.G., Waugh, K., Smith, N. (2006) Using Patterns in the Automatic Marking of ER-Diagrams. *Proceedings of the 11<sup>th</sup> Annual Conference on Innovation and Technology in Computer Science Education* (ITiCSE, June 26-28, Bologna, Italy, 2006), 403-413.

Thomas, P.G., Waugh, K., and Smith, N. (2007a) Computer Assisted Assessment of Diagrams. In Proceedings of ITiCSE, Dundee, Scotland, 25-29 June 2007.

Thomas, P.G., Waugh, K., and Smith, N. (2007b) Tools for supporting the teaching and learning of data modelling. In Proceedings of ED-MEDIA, Vancouver, Canada, 25-29 June, 2007.

Thomas, P.G., Waugh, K., Smith, N. (2007c) Tools for learning and automatically assessing graph-based diagrams. In Research Proceedings of ALT-C 2007, Nottingham, 4-6 September, 2007, 61-74.

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., Thomas, P.G., Smith, N. (2007) Teaching and Learning Applications Related to the Automated Interpretation of ERDs. TLAD 07, July 2007, Glasgow.