FORMATIVE AND SUMMATIVE ASSESSMENT OF CLASS DIAGRAMS

Development and Evaluation of a Prototype

Narasimha Bolloju, Ken W. K. Lee
Department of Information Systems, City University of Hong Kong, Kowloon Tong, Hong Kong

Probir K. Banerjee
Department of Operations and MIS, University of Malaya, 50603 Kuala Lumpur, Malaysia

Keywords: Computer supported assessment, Assessment of diagrams, Prototype, Class diagrams.

Abstract: The computer-based assessment of student outputs in the field of software engineering is an area of interest amongst instructors and researchers. However, as previous efforts in this area have been mostly directed toward summative assessment, the formative feedback required for student learning has not received sufficient attention. This paper presents an approach for the formative and summative assessment of class diagrams used in systems development and reports the development, implementation and evaluation of a prototype assessment tool. The results from this empirical study demonstrate that the tool successfully provides formative feedback during the preparation of class diagrams, which helps to enhance student learning, and summative feedback that can be used as a proxy for manual summative assessment.

1 INTRODUCTION

Large class sizes increase the workload of instructors, thereby adding to the drudgery and the potential for inconsistency in the manual assessment of work submitted by students. To combat this problem, a range of computer-based assessment (CBA) support tools have been developed that grade student submitted work by matching it with model solutions or the correct answers. For example, one CBA tool has been developed to support interactive practice with electrical circuit diagrams and the grading of student work by comparing the submitted responses with the correct solutions (Smaill, 2005). In the field of software engineering, CBA tools such as CourseMarker (Foxley, Higgins, Hegazy, Symeonidis, & Tsintsifas, 2001a; Foxley, Higgins, Symeonidis, & Tsintsifas, 2001b) and DATsys (Tsintsifas, 2002) have been developed to assess computer programming skills. These CBA tools mostly provide summative feedback to students in terms of how well a student's response matches a model answer. The use of formative feedback to enhance learning through CBA tools was not explored in these earlier models. However, a recent study shows evidence of an attempt to generate formative feedback through a CBA tool named AutoLEP (Tiantian, Xiaohong, Peijun, Yuying, & Kuanquan, 2010). Developed for use in computer programming courses, this tool evaluates whether or not students' computer programs meets the required specification (summative assessment) and also dynamically tests the syntax and structure of the programs and provides interactive help (formative feedback) to improve students' learning experiences in programming. Thus, there is a need to explore in greater detail the potential of CBA tools to provide formative feedback to students.

Existing CBA tools also have limitations in that they are useful mostly in assessing fixed-response learning outputs where students have to choose answers from a pre-designated selection of alternatives, such as multiple-choice questions, or where the evaluation is carried out via a model solution (e.g., Ali, Shukur, & Idris, 2007; Smith, Thomas, & Waugh, 2004; Thomas, Waugh, & Smith, 2005; Thomas, Waugh, & Smith, 2009). In some problem domains and contexts, the answers may be given in a free-response format where

Bolloju N., W. K. Lee K. and K. Banerjee P..
FORMATIVE AND SUMMATIVE ASSESSMENT OF CLASS DIAGRAMS - Development and Evaluation of a Prototype.
DOI: 10.5220/0003476504020410
In *Proceedings of the 3rd International Conference on Computer Supported Education* (ATTeL-2011), pages 402-410
ISBN: 978-989-8425-50-8

multiple solutions exist for the same problem. In such situations, CBA becomes more complex because it calls for more sophisticated algorithms compared to those used in the evaluation of outputs in fixed-response formats. Class diagrams are one example of a situation where free-response format outputs may be produced. The manual assessment of such diagrams produced by students in large classes becomes a challenge because there may be multiple correct ways (free-response format) of modeling the same problem. Apart from the excessive amounts of time and effort required, the manual assessment of such work is often error-prone, resulting in inconsistent evaluation. These problems are magnified when students attempt to model complex real-life situations as the class diagrams invariably become increasingly complex and the students need support to guide them through the process. Thus, the efficacy of CBA tools in providing formative assessment during the development of students' work and summative assessment after the work is submitted in free-response outputs, such as class diagrams, is worthy of investigation.

We propose an approach of addressing the formative and summative aspects of the computer-based assessment of student's work. We also develop and validate a prototype CBA tool, named Computerized Assessor for Class Diagrams (CACD), to assist students in preparing class diagrams and instructors in grading class diagrams submitted by students for evaluation. In addition, we validate the developed tool in terms of its efficacy in providing formative and summative assessment in an example situation where students develop class diagrams for a system that manages subscriptions and editorials for a small independent software journal.

The remainder of this paper is organized as follows. In the next section, we discuss prior research on existing approaches to designing CBA tools that support software engineering education and their limitations in providing summative and formative support to users. Section 3 presents an overview and discussion of the approach used in the design of our prototype for supporting the summative and formative assessment of students' class diagrams. Sections 4 and 5 outline the implementation and validation of our prototype tool. The final section concludes the paper and makes recommendations for future research.

2 PRIOR RELATED WORK

In this section, we review prior research on approache to providing formative and summative assessment of class diagrams, entity-relationship diagrams, or similar diagrams.

2.1 Formative Assessment

Most of the existing approaches used to support the formative assessment of class diagrams or similar diagrams directly or indirectly through CBA tools can be categorized based on their functionality as generation focused, guidance focused or critique focused.

Generation focused approaches attempt to create class diagrams from textual descriptions of system requirements or through question-and-answer mechanisms. Techniques such as natural language processing and expert systems are employed in this approach to translate requirements into conceptual models (Kaindl, 2004, Overmyer, Lavoie, & Rambow, 2001, Purao, 1998). For example, Wohed (2000) discusses a computer prototype that uses a natural language processing technique to develop a conceptual object model based on user responses to a sequence of six questions. In a web-based prototype called APSARA, object-oriented designs are created from requirement descriptions written in natural language using heuristics and a patterns database (Purao, 1998). RETH, another example of such a tool, uses natural language processing to generate associations and relations between objects based on natural language definitions of classes (Kaindl, 2004). Natural language processing is also employed to extract words from a textual document to generate a corresponding object model (Overmyer et al. 2001). Because generation focused tools transform a given set of inputs to an output form, they provide little guidance or formative feedback to help users in developing object models. Thus, although the extensive automation built into the generation focused approaches make them attractive, they have drawbacks in terms of the lack of mechanisms offered for generating formative feedback.

Guidance focused approaches use practitioner-oriented recommendations to develop object models which are often in the form of guidelines for identifying classes and relationships, naming and presentation conventions, and the usage of analysis patterns and frameworks (e.g., (Bolloju, 2004), (Batra, 2005)). In some cases, this approach involves customizing reusable frameworks to suit the target environment (e.g., Hakala, Hautamaki, Koskimies, Paakki, Viljamaa, & Viljamaa, 2001, Morisio, Travassos, & Stark, 2000, Viljamaa, 2001).

Anthony and Batra (2002) describe a rule-based expert system called CODASYS that guides database designers by asking questions and restricts the search space for novice users, thus reducing errors. (Purao, Storey, & Han, 2003) present details of the development and empirical evaluation of an intelligent assistant which generates initial conceptual designs for later refinement and incorporates learning mechanisms for enhancing analysis pattern reuse. Sugumaran and Storey (2006) use domain ontologies to guide novice and experienced modelers in creating complete and consistent database designs. Thus, the guidance-focused approach has elements of formative feedback that help to develop object models which support a given set of the requirements. However, while the guidance-focused approach provides an exhaustive list of dos and don'ts, few of these suggestions have been built into the supporting tools. As a result, this approach offers limited benefits to users in terms of developing quality object models.

The critique focused approach is aimed at providing advice either during or after the development of a conceptual model. For example, an ArgoUML implementation of this approach incorporates a set of critiquing features that aim to address the cognitive needs of software designers by using agents that continuously check an existing model for potential problems and advise on areas of improvement (Robbins & Redmiles, 1998; 2000). Another of these approaches, which employs domain ontologies for supporting database design, is helpful in suggesting new entities and relationships and in validating the data model (Sugumaran & Storey, 2002). Thus the critique focused approaches exhibit certain elements of formative feedback that help in the preparation of outputs. However, as with the guidance focused approaches, the critique focused approaches mostly provide broad sets of recommendations in the form of lengthy checklists that are of limited use in the development of class diagrams.

In summary, the support of formative assessment requires a combination of guidance and critique focused approaches that are capable of analyzing class diagrams as they are being created and providing feedback on missing and invalid elements based on given problem specifications.

2.2 Summative Assessment

In the area of summative assessment, some CBA tools have been developed to assess computer programming skills (Charman & Elmes, 1998) (Rawles, Joy, & Evans, 2002); (Symeonidis, 2006). One such CBA tool, named the CourseMarker CBA system (Foxley et al. 2001a; 2001b), grades student-produced programming exercises and also has the potential to assess class diagrams through an integrated system known as DATsys (Tsintsifas, 2002). The feasibility of automated assessment of entity-relationship diagrams using an expert system with domain specific inference rules has been explored in relation to class diagrams (Smith et al., 2004). A recent study has proposed the use of a CASE tool for assessing class diagrams which uses evaluation metrics and feedback mechanisms to facilitate student learning, though there has been no attempt to build and evaluate a prototype (Ali et al., 2007).

Our review of prior research indicates that, while aspects of the formative and summative assessment of class diagrams or similar diagrams have received attention and tools have been developed that address aspects of both, no effort has been made to investigate the feasibility of incorporating both forms of assessment in a CBA tool. Elements of the summative assessment of class diagrams and some formative guidance during development are evident in certain CBA tools. However, they have not been fully developed and integrated in CASE based tools to ensure the quality of the final output. Our review also indicates that the existing CBA tools have limitations in that they provide no means of validating the outputs produced. Because the class diagram development process is often interactive and incremental, CBA tools must be able to provide interactive feedback to help students understand elements of the domain and their associations. Furthermore, variations in the students' solutions must be able to be detected in relation to the model solutions and the students' given credit for providing correct alternative solutions to the problem. In the next section we discuss our approach for developing such a CBA tool.

3 AN APPROACH FOR CLASS DIAGRAM ASSESSMENT

We propose a two-phased approach to support the formative and summative assessment of class diagrams. The first phase uses a knowledge based or expert system component to support formative assessment of class diagrams targeted at providing students with feedback as they prepare their solutions. This interactive system component analyzes the current version of a class diagram and offers recommendations for improvements when the

student calls for such an assessment. The underlying knowledge based system for this component validates the classes, attributes and relationships presented in the class diagram against requirements that are specified in use case descriptions, and points out missing and invalid elements by making use of the system specifications. Thus, this component guides the student towards achieving a complete solution to the problem. First, the knowledge based component returns a list of recommendations that may help in improving the current version of a class diagram. The student can then repeatedly ask for feedback on the revised versions, as the tool does not restrict the number of times that feedback can be called for. When a student is satisfied with his/her class diagram, he/she submits it for summative assessment. This formative assessment component thus helps in improving the quality of class diagrams in terms of elements that are missing (completeness) and elements that are not required or invalid (invalidity) according to a given set of requirements.

The second component provides the functionality for the summative assessment of the submitted class diagram(s). It includes two components: a) an automated component that assesses the final submitted class diagram and assigns a grade after comparing it to a model or expected solution, and b) an interactive component that lists unmatched elements to enable the instructor to determine possible alternative, but correct representations of the expected solution and to manually revise the grades initially given by the tool. Manual refinement of the initial assessment can be done by the instructor whereby credit may be given for elements that the tool identified as unmatched but which the instructor determines to be a correct alternative representation of the problem. Both assessment stages are integrated into a single system so that the grader can initiate an automated assessment and review the partial result in the same environment.

The overall assessment process is depicted in Figure 1 as an activity diagram with activities corresponding to the instructor, student, CACD tool, and a grader who can be different from the instructor. The instructor sets the assessment task by preparing a use case diagram and a set of use case descriptions, and provides a correct diagram corresponding to the problem. Each student prepares a draft class diagram and invokes the formative assessment component for feedback. Once satisfied with the revisions, which have been made according to the recommendations given by the tool, the student submits the final class diagram for summative assessment. The summative assessment component collects and processes all student submissions using the correct or expected class diagram, and reports the grades. A grader may review any of the class diagrams submitted and adjust the matches performed by the automated assessment subcomponent. The system then recalculates the grades for any manually adjusted diagrams.

4 SYSTEM IMPLEMENTATION

This section describes the technical details and implementation of the prototype system for supporting formative and summative assessment. The implementation of the proposed approach includes: a) the extension of an existing CASE tool for developing class diagrams, b) the identification and representation of a knowledge base for the diagnostic process, c) a mechanism for diagnosing a class diagram, and d) a facility for scoring and comparing class diagrams.

The formative assessment component is implemented through the extension of an existing UML CASE tool ArgoUML. This component performs a diagnosis of the current diagram when invoked to do so by the user. A windows-based facility in the CBA tool allows the instructor to see the recommendation in terms of missing and unmatched elements in the student's submitted work generated as a result of the initial assessment by the CBA tool. The display of feedback, presented in the form of a table, is designed to be simple to follow and yet facilitate critical thinking and analysis of how the recommendations may be incorporated in the revised version of the class diagrams.

In the typical usage of the system, the instructor is required to prepare one use case diagram and specify descriptions of the use cases present in the diagram in terms of trigger, pre- and post-conditions, and to describe the steps in the main and alternate scenarios. The instructor then releases the diagram as an ArgoUML project for the student to complete. The student is expected to proceed through the activities of reviewing the recommendations, applying any relevant recommendations to the class diagram and invoking the formative assessment function, as long as the recommendations provided by the system are applicable, before submitting his/her work to the tutor. This component uses natural language processing techniques to extract relevant information from the textual use case descriptions, domain-independent heuristics and knowledge collected from practitioner guidelines and analysis

patterns to analyze a given class diagram. The diagnosis results are tabulated as a set of recommendations for improvements in terms of missing and invalid elements.

Stage 1 – Automated assessment: This assessment component takes two inputs: a model solution as the expected class diagram and a class diagram submitted by a student.

Stage 2 – Incremental manual assessment support component: This component helps in adjusting the matchings performed by the stage 1 component. Because it is often expected that multiple class diagrams can be acceptable solutions, it is not possible to recognize every element during the first stage of automated assessment.

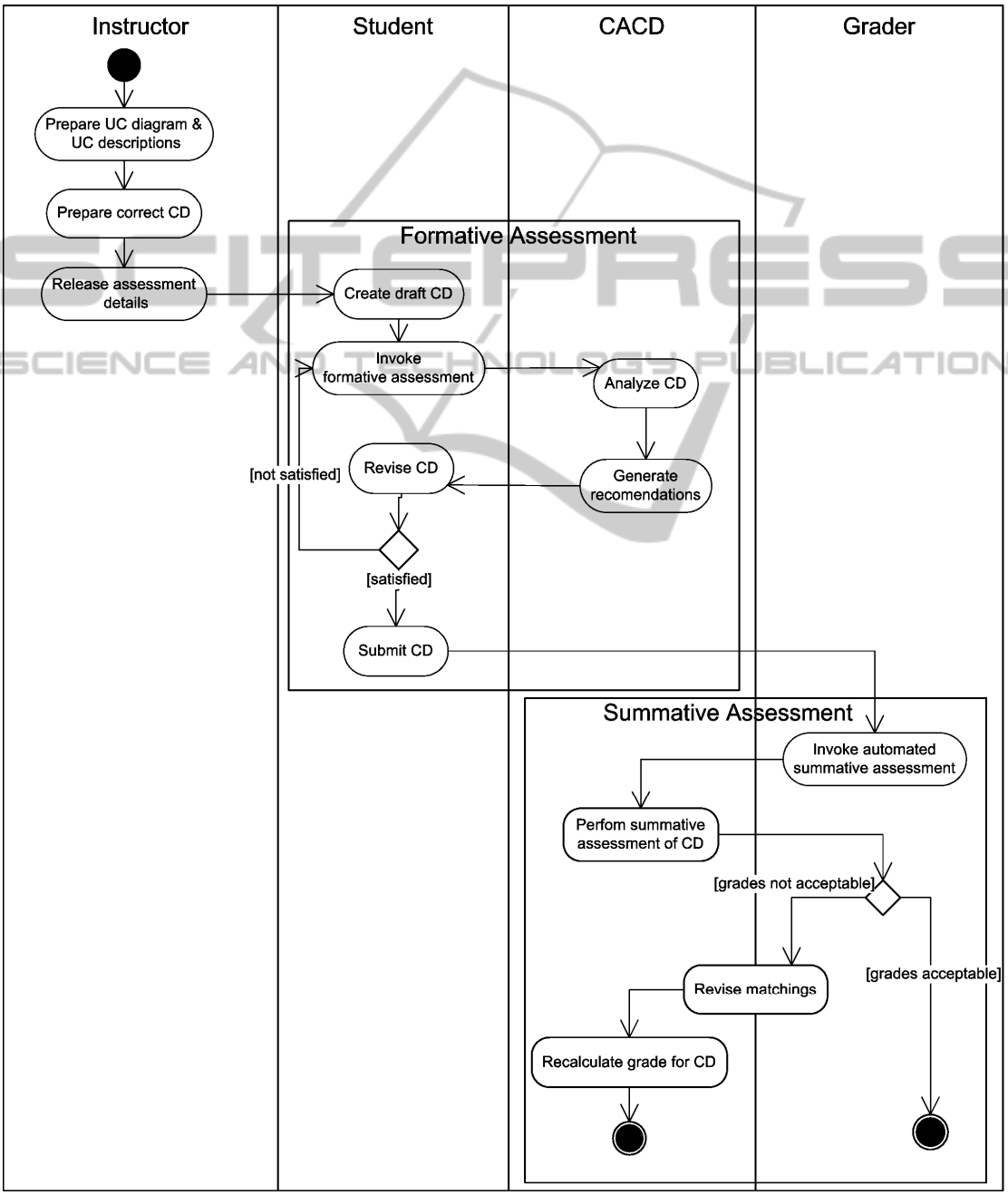

Figure 1: Integrated formative and summative assessment process.

5 RESULTS AND DISCUSSION

The prototype system, CACD, was used to assess several class diagrams produced by undergraduate students enrolled in a systems analysis and design course at a major metropolitan university in Hong Kong. During a two-hour lab session, the students created class diagrams pertaining to a journal publisher case study. The students were also provided with a use case diagram containing six use cases and associated descriptions to supplement the case study. This exercise took place after the students had been introduced to object-oriented concepts and had a chance to practice class diagramming skills using the ArgoUMl tool in a two hour lab session.

During the first half of the lab session for the case study, the students created a draft class diagram, which they submitted by uploading into a Blackboard assignment folder. During the second half of the lab session, they utilized the formative assessment component to receive feedback, modified their work based on the feedback provided and then uploaded the revised or final version of their class diagram.

To demonstrate the utility of the prototype system, we present results obtained from a detailed analysis of 41 pairs of class diagrams submitted by the students. As part of this analysis, each class diagram was compared with the model solution by a student helper after a pilot assessment of a small set of diagrams by a research assistant and the student helper. This process was supported by the incremental manual assessment component that first automatically matched elements in the student solutions with the expected solutions. The assessor then attempted to match unmatched elements while looking for alternative representations.

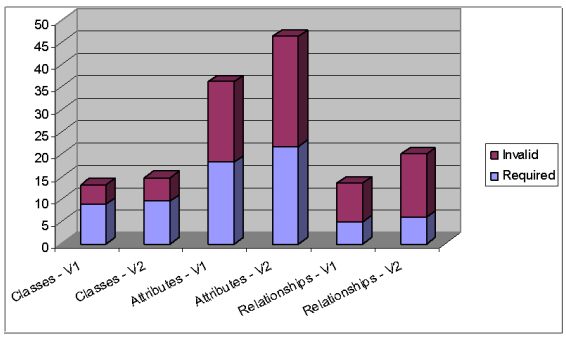

Figure 2: Average numbers of required and invalid elements in the class diagram pairs.

5.1 Effect of Formative Feedback

Figure 2 depicts the average numbers of required and invalid elements found in the 44 class diagram pairs. The values for the required elements indicate the number of elements in the student solutions that match with the expected solutions while accounting for alternative representations, such as two classes in a student's solution mapping to one class in a student's solution and differently named or designed classes or attributes. The values for invalid elements indicate the number of elements in a student's solution that are not valid (i.e., out of scope elements) with respect to the expected solution. All the increases in the numbers of elements in the second version compared to the first version were found to be significant ($p < 0.01$).

The expected solution is required to contain 14 classes, 40 attributes and 13 relationships. Comparing these numbers with the average numbers found in the student solutions we conclude that the second version of student solutions, on average, contained only about half the required elements. Contrary to our expectations, the numbers of invalid elements identified also increased in the assessment of the second version. Further analysis (discussed below) was conducted to investigate the reasons for this anomaly.

5.2 Quality of the Feedback Provided

To investigate the quality of the feedback provided by the formative assessment component, further analysis of the feedback provided was performed by two student helpers who were given training on the expected solution and its possible variations. A research assistant and the student helpers studied each feedback item and categorized it as either 'useful', 'not useful' or 'may be'. They also identified which feedback had actually been incorporated into the revised version. We accounted for the subjectivity in assigning feedback items to different categories only after obtaining a satisfactory inter-rater reliability with a small set of class diagrams as part of the training process.

Table 1 lists the different types of feedback items provided and the usefulness of those items for the current version of class diagram. Based on these results, we found that much of the feedback provided was fairly useful, except for missing relationships, and that a good percentage of feedback items were assimilated and incorporated by the students except for invalid classes and attributes.

Table 1: Quality of feedback provided and incorporated.

	% feedback items per class diagram pair	% feedback items incorporated from those suggested	% feedback items considered useful	% feedback items considered not useful[a]	% feedback items considered useful & incorporated	% feedback items considered NOT useful & incorporated
Missing classes	2.50	31.82	65.45	34.55	48.61	7.89
Missing attributes	1.00	31.82	70.45	29.55	45.16	7.69
Missing relationships	17.84	0.51	2.04	97.96	25.00	6.37
Invalid classes	6.23	3.65	40.15	59.85	9.09	1.22
Invalid attributes	23.25	4.11	40.86	49.76	10.05	3.54
Invalid relationships	3.07	26.67	82.96	17.04	32.14	26.09

[a] values in this column exclude the recommendations that fall in the "may be" category.

5.3 Efficiency of the Formative and Summative Assessment Processes

The prototype system was found to provide quick formative assessment support directly through the ArgoUML used for developing the class diagram. On average, the tool took around 5 seconds to analyze a draft class diagram and generate feedback items. For the summative assessment, Stage 1 (automated assessment) was completed in a fraction of a second for each class diagram and Stage 2 (manual incremental assessment) took around 4.4 minutes (std 2.32), compared to an average of 14.07 minutes (std 2.89) required to complete a manual assessment without any tool support. The correlation between the fully automated and manual incremental assessments was also high (required elements 0.337, $p < 0.05$ and invalid elements 0.484, $p < 0.01$), pointing towards the efficiency of the assessment process.

The overall support for summative assessment was found to be more efficient than the manual alternative and reasonably complete. Analysis of the system-generated grades indicates that the automated assessment results are acceptable for rough grading purposes. The incremental manual assessment was also found to be quite efficient and uniform compared to the manual alternative. As a result, student helpers or graduate assistants may be able to perform this grading exercise with high levels of accuracy and consistency.

In summary, the prototype system was found to be efficient with regard to the time taken to match the initial class diagram with the model solution and to automatically assign a grade. The prototype therefore has the potential to greatly reduce the workload of instructors. The consistency of evaluation, which was verified manually, also ensures a high degree of uniformity of assessment across the student population.

6 CONCLUSIONS

We have presented an approach for the formative and summative assessment of class diagrams and outlined the details of the development and evaluation of a prototype system to support the proposed approach. Instructors are capable of benefiting from a reduced workload as a result of the automated assessment carried out by the tool, which also ensures that the students get due credit for alternative correct representations that may differ from model solutions. Students are capable of benefiting from the feedback provided by the prototype system to enhance their class diagramming skills. We believe this research makes an important contribution to the field of automated assessment of student-produced free-response outputs such as class diagrams. To our knowledge, our prototype is the first of its kind in this domain to provide integrated support for formative and summative assessment. Thus, our research makes an important contribution to the body of knowledge in the field of computer based support for learning and assessment.

A limitation of this research is that our empirical analysis only provided an indirect measure of the contribution the formative assessment component made to student learning. More direct measures,

such as estimating student learning through a follow-up test, could validate our claim for efficacy of the formative assessment component included in the prototype system. Another limitation, that is attributable directly to the prototype system, is that the students did not fully assimilate and use the recommendations generated by the system in their revised versions. Future versions of the prototype may need to be developed that address these observed limitations. On the other hand, the concept used in this research may be used to develop CBA tools that address a wide range of learning and assessment activities.

ACKNOWLEDGEMENTS

The work described in this paper was substantially supported by a grant from City University of Hong Kong (Project No. 7008016).

REFERENCES

Ali, N. H., Shukur, Z., & Idris, S., 2007. A design of an assessment system for UML class diagram (pp. 539-546). Presented at *the Fifth International Conference on Computational Sceince and Applications*, Kuala Lumpur, Malaysia: IEEE Computer Society.

Antony, S. R., & Batra, D., 2002. CODASYS: a consulting tool for novice database designers. *SIGMIS Database*, 33(3), 54-68.

Batra, D., 2005. Conceptual Data Modeling Patterns: Representation and Validation. *Journal of Database Management*, 16(2), 84-106.

Bolloju, N., 2004. Improving the quality of business object models using collaboration patterns. *Communications of the ACM*, 47(7), 81-86.

Charman, D., & Elmes, A., 1998. *Computer Based Assessment (Volume 2):* Case studies in Science and Computing. Plymouth, UK: SEED Publications.

Foxley, E., Higgins, C., Hegazy, T., Symeonidis, P., & Tsintsifas, A., 2001a. The CourseMaster CBA System: improvements over Ceilidh. *In Proc. of Intl. Computer Assisted Assessment Conference*.

Foxley, E., Higgins, C., Symeonidis, P., & Tsintsifas, A., 2001b. The CourseMaster automated assessment system - a next generation Ceilidh. *In Computer Assisted Assessment Workshop*. Warwick, UK.

Hakala, M., Hautamaki, J., Koskimies, K., Paakki, J., Viljamaa, A., & Viljamaa, J., 2001. Annotating Reusable Software Architectures with Specialization Patterns. In *Proceedings of the Working IEEE/IFIP Conference on Software Architecture* (p. 171). Washington, DC, USA: IEEE Computer Society.

Kaindl, H., 2004. Active tool support for requirements engineering through RETH. In *12th IEEE International Requirements Engineering Conference* (RE'04) (pp. 362-363). Kyoto, Japan: IEEE Computer Society.

Morisio, M., Travassos, G. H., & Stark, M. E., 2000. Extending UML to Support Domain Analysis. In *Proceedings of the 15th IEEE international conference on Automated software engineering* (p. 321). Grenoble, France: IEEE Computer Society.

Overmyer, S. P., Lavoie, B., & Rambow, O., 2001. Conceptual modeling through linguistic analysis using LIDA. In *Proceedings of the 23rd International Conference on Software Engineering* (pp. 401-410). Toronto, Ontario, Canada: IEEE Computer Society.

Purao, S., 1998. APSARA: a tool to automate system design via intelligent pattern retrieval and synthesis. *SIGMIS Database*, 29(4), 45-57.

Purao, S., Storey, V. C., & Han, T., 2003. Improving Analysis Pattern Reuse in Conceptual Design: Augmenting Automated Processes with Supervised Learning. *Information Systems Research*, 14(3), 269-290.

Rawles, S., Joy, M., & Evans, M., 2002. *Computer-Assisted Assessment in Computer Science: Issues and Software*. University of Warwick. Retrieved from http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.19.1209.

Robbins, J. E., & Redmiles, D. F., 1998. Software architecture critics in the Argo design environment. *Knowledge-Based Systems*, 11(1), 47-60.

Robbins, J. E., & Redmiles, D. F., 2000. Cognitive Support, UML Adherence, and XMI Interchange in Argo/UML. *Information and Software Technology*, 42(2), 79-89.

Smaill, C., 2005. The implementation and evaluation of OASIS: a web-based learning and assessment tool for large classes. *Education, IEEE Transactions on*, 48(4), 658-663.

Smith, N., Thomas, P., & Waugh, K., 2004) Interpreting imprecise diagrams. *Diagrammatic Representation and Inference*, 239–241.

Sugumaran, V., & Storey, V. C., 2002. Ontologies for conceptual modeling: their creation, use, and management. *Data & Knowledge Engineering*, 42(3), 251-271.

Sugumaran, V., & Storey, V. C., 2006. The role of domain ontologies in database design: An ontology management and conceptual modeling environment. *ACM Transactions on Database Systems*, 31(3), 1064-1094.

Symeonidis, P., 2006. *Automated assessment of java programming coursework for computer science education* (Ph.D. Thesis). University of Nottingham.

Thomas, P., Waugh, K., & Smith. K., 2005. Experiments in the automatic marking of ER-diagrams. In *Proceedings of the 10th annual SIGCSE conference on Innovation and technology in computer science education* (p. 162).

Thomas, P., Waugh, K., & Smith, N., 2009. Generalised diagram revision tools with automatic marking. In *Proceedings of the 14th annual ACM SIGCSE*

conference on Innovation and technology in computer science education (pp. 318-322). Paris, France: ACM.

Tiantian, W., Xiaohong, S., Peijun, M., Yuying, W., & Kuanquan, W., 2010. Ability-training-oriented automated assessment in introductory programming course. *Computers & Education*, (In Press).

Tsintsifas, A., 2002. *A framework for the computer-based assessment of diagram-based coursework (Ph.D. Thesis). University of Nottingham*. Retrieved from http://www.cs.nott.ac.uk/~azt/research.htm

Viljamaa, A., 2001. *Pattern-Based Framework Annotation and Adaptation - A Systematic Approach*. Helsinki, Finland: Department of Computer Science, University of Helsinki.

Waugh, K. G., Thomas, P. G., & Smith, N., 2004. Toward the automated assessment of entity-relationship diagrams. In *Proceedings of the 2nd LTSN-ICS Teaching, Learning and Assessment in Databases Workshop*. Edinburgh, UK.

Wohed, P., 2000. Tool support for reuse of analysis patterns: a case study. *In Proceedings of the 19th international conference on Conceptual modeling* (pp. 196-209). Salt Lake City, Utah, USA: Springer-Verlag.