THE APPLICATION OF COMPUTER-ASSISTED ASSESSMENT IN DIAGRAM-BASED COURSEWORK

Noraida Haji Ali¹, Zarina Shukur² and Sufian Idris²
¹Computer Science Department
Faculty of Science and Technology
Universiti Malaysia Terengganu
21030 Kuala Terengganu, Terengganu
aida@kustem.edu.my

²Computer Science Department
Faculty of Information Science and Technology
Universiti Kebangsaan Malaysia
Bandar Baru Bangi, Selangor
zs@ftsm.ukm.my, sufian@ftsm.ukm.my

ABSTRACT

Learning process is not enough with just reading books or passively attend to lectures. In order for an efficient learning process to occur, students must actively doing exercises or coursework. These exercises were given by instructors to evaluate students’ level of understanding. Each coursework needs to be evaluated to ensure that the student’s solution is correct. Diagram-based coursework allows students to draw diagram as a solution of their coursework. In computer science courses, subjects such as Systems Analysis and Design that normally become the core subjects, use visual tools to describe the model of system requirements. Students draw the solution of assignment within an appropriate diagram editor based on use case modeling. There are various kinds of models in programming contexts and one of them is object-oriented model. Object-oriented modeling can be illustrated using semi-formal and formal notation. This research is to apply Computer Assisted Assessment (CAA) in object-oriented modeling, UML, and formal modeling, Z specification. In this paper, we discussed the specification of Object-Oriented Model Assessor (OOMA) which is proposed for development. The objectives of this research is to reduce the work-load of instructors by performing automatic assessment of students’ work. It could also improve students’ understanding on how to represent the system requirements in UML model as well as in Z specification model.

INTRODUCTION

Computer Assisted Assessment (CAA) is a field of learning technology to enhance the learning process. It is to be used for delivery, analysis and marking of student coursework. The automatic assessment of diagrams belongs to the free response type of assessment; the large class sizes in many universities have made automatic assessment an attractive solution to the increasing amount of work. An automatic assessment system is software that takes a submission to the coursework as an input, assesses the submission without the need for human intervention, and produces feedback as an output. The output usually includes numerical grade, textual feedback, pictures or animations. Brown et al. (1997) list three benefits of CAA:

- Reducing the work load of instructors
- Providing students with detailed formative feedback on their learning much more efficiently than is usually possible with traditional assessment.
- Bringing the new assessment culture where students closer to the computer based learning environment.

Besides that, the most valuable benefit of CAA is the ability to enhancing feedback to students and instructors, time savings, reducing work load of instructors and improving the assessment method. Feedback can be used to ensure students will be given constructive detailed and consistent feedback. This benefit can eliminate the differences in evaluation and assessment between different instructors checking the same coursework. Students can submit their answers for grading or assess whenever they wish and receive immediate
feedback on the answers. This will allow the students to submit their answers multiple times. By reviewing their old answer and the feedback given by the automatic system, the students can correct their submissions and learn from their mistakes [Venables & Haywood, (2003)].

CAA has been used in academic during the 1990’s, to assess a wider variety of subject matters such as physic, mathematic, chemistry and so on. Today, automatic assessment is used with numerous different types of exercises, starting from very simple tools to large commercialized products. Carter et al (2003) divide exercises into five categories: multiple choice questions, text answers, programming assignments, visual answer and peer assessment. Multiple choice questions are the simplest and the most common form of automatic assessment. Text answer category includes both short text answers and longer essays are very complex problems that have been studied for decades. In the computer science field, the automatic assessment of programming assignments has been a subject of research for decades. The earliest assessment systems were simple: the student program was run with several inputs, and its output was compared to the output created by instructors. These output matching methods are still used in the assessment of programming assignments. Peer assessment is not categorized as CAA since the assessment is not done by a computer.

This research focuses more to visual answer where visualization has been used extensively is theoretical computer science. It can also be used in solving coursework related to formalisms that can be shown as diagrams. It is of great importance in automatic assessment of diagrams to be able to separate the visualization (the diagram itself) from the data it represents, since typically the instructor is concerned with the correctness of the model. Through visualization exercises on different theoretical entities like semi-formal and formal languages, automata, binary tree, flowchart and logical circuits, can be made much easier for students to comprehend.

Semi-formal modeling: UML Class Diagram
The Unified Modeling Language (UML) is probably the most widely known and used notation for object-oriented analysis and design. UML consists of various graphical notations, which capture the static system structures (class diagrams), system component behaviors (state transition diagrams) and system component interactions (collaboration and sequence diagrams). UML notations can be produced with the help of CASE (Computer-aided software engineering) tools such as Rational Rose. Figure 2 shows an example of a UML class diagram for a simple process in a bank system. In this UML class diagram, there have a few notations such as classes, association relationship, and generalization and so on.

![Figure 2. Example of UML Class Diagram for bank system](image-url)
The diagram represents most UML class constructs, namely class, and relationships. The diagram consists of two major entities in the system: Customer and Account. Each class has its own attributes and operations. UML class diagram consist the five basic relationships. There are association, association class, generalization, aggregation and composition. These relationships have their own symbol or notation and descriptions. Association is a connection between classes and it is drawn as a solid line. Association class can be attached to an association and used to add extra information on a link and it is drawn by dotted line between classes. Generalization is a relationship between a general class, called the super-class and a specific class, called the subclass. All the attributes and operations are inherited. It is drawn as solid line from the subclass to the super-class, with a hollow triangle at the super-class end of the line. Aggregation is a special case of association that indicated the relationship between the classes is some sort of "is part of". It is drawn as a line with a hollow diamond attached to one end of the line. Composition is a type of aggregation and is one with strong ownership and it is drawn as a line with a filled diamond attached to the class (Eriksson & Penker, 1998)

Formal modeling: Z Notation / Specification
The Z notation is a formal specification language used for describing and modeling the computer system. Z was originally proposed by Jean-Raymond Abrial in 1977. It was developed further at the Programming Research Group at Oxford University. Z is based on the standard mathematical notation used in axiomatic set theory; lambda calculus and first-order predicate logic. Z contains a standardized catalog known as mathematical toolkit of commonly used mathematic functions and predicates. Although Z notation uses many non-ASCII symbols, the specification includes suggestion for rendering the Z notation symbols in ASCII and in LaTeX (Jacky, 1996). The Z schema is the characteristic construct of Z. Figure 3 shows an example of a diagram for Z specification for Account class that refer to the UML class diagram in figure 2.

```
Account
acNo: A
balance: C

withdraw
| an? : A |
| amount?: C |

deposit
| an? : A |
| amount?: C |
```

Figure 3. Example of Z specification

Every system has a special state in which it starts up. In Z this state is described by a schema conventionally named by the user. The schema adds something new that we need to model the memory of system. For the Account class, it was a state scheme in Z specification and for every attribute, which are acNo and balance; it was drawn as situation variable. While for its operations, which are withdraw and deposit, it was drawn as an operation schema in Z specification. For the rest of the classes such as Customer, Transaction, CheckingAccount and so on, it will also be drawn as state scheme or operation scheme according to that class.

RESEARCH BACKGROUND
The first attempt at using computers to automate the process of assessing student work was reported in the early 1960's (Nikander, 2005). Early automatic assessment systems were built almost exclusively towards computer science related subjects, especially programming and numerical based subjects. A lot of research has been done on assessing subjects that is in the form of text based. Research that refers exclusively to the automation of assessment of student diagrams has not yet received much attention. Modeling is a process of planning a system before doing the next process, which is coding. The more complex the system is, the more important the use of good modeling techniques. By using model, it is hoped that system development can cover all customers’ needs completely and precisely. The success of a system modeling is determined by three elements, which are known as the triangle for success. These three elements are notation method, process and elements used. Models are
usually described in a visual language, which means that most of the information in the models is expressed by graphical, symbols and connection. Visual modeling means creating a graphical blueprint of your system’s architecture. From this visual, we can quickly detect problems such as inconsistencies and lack of modularity. Visual models also improve inconsistencies between requirements, designs, and implementations. Over the years, analysis and design methods and notations for system modeling have been proposed and developed. This model is used to get the prediction of cost or budget and time requirement to finish a system. In the late 1997, most of these methods have converged to a standard notation, the Unified Modeling Language (UML), which has been accepted as an industry standard by the OMG (Object Management Group, 1999). CASE tools which provide support for UML diagramming (e.g. Rational Rose, Microsoft Visio and Enterprise Architect) can benefit from the use of an automatic layout tool. Rational Rose (RR) is the most popular software used in designing software through UML (Unified Modeling Language) approach.

The purpose of CASE tools is to reduce the gap between theory and practical. However, the tools do not focus on the learning environments which need more aids to increase users’ understanding especially for students. Previous researches focus more to programming exercise assessment. Over the years, there have been dozens of different systems created, where the earliest was dated back in 1961 (Forsythe & Wirth, 1965). Examples of this type of assessment systems are created by Naur (1964) that grade the Algol programming, ASSYST that automate the assessment system for ADA program (Jackson & Usher, 1997), Celilidh, which is an assessment system for C, C++ and UNIX programming and Style++ that was used to measure programming styles for C++ program (Ala-Mutka et al., 2004). Research that refers exclusively to the automation of assessment of student diagrams however has not yet received much attention. Research by Thomas et al (2007) used the notion of patterns to successfully apply to the automated marking of students’ attempts to drawing entity-relationship (ER) diagrams. TRAKLA2 is an automatic assessment system for data structure and algorithm exercises at Helsinki University of Technology (Nikander, 2005). CourseMaster and DATsys system are two platforms built at the University of Nottingham to support diagram-based exercises. Diagram notation specifications are authored by the course developer using its own authoring tool and use its own platform for assessing UML Class diagrams, flowcharts and logical circuits automatically. Some students experiencing difficulties were helped by CourseMaster’s feedback and as a result were able to draw the correct diagram. System will then give a list of correct notations in diagram as a feedback to students. Student’s experience stated by marking status produced by the system such as ‘excellent’, ‘correct’ and ‘rotten’. In the previous system, they use their own platform to get the input of the diagram and to be assessed directly (Higgins et al., 2002).

Our proposed automatic assessment system can get the input from existing authoring tools to draw diagram, such as Rational Rose for draw UML class diagram and Z/EVES to draw Z-specification. The assessment process is divided into several modules where each module process is inter-related with each others. Assessment processes are carried out in order to verify that the input diagram is error free. Apart from that, it will also generate list of feedback to be used as guidance to the students and hopefully it helps to improve their understanding in subject matter. The automatic assessment process done by the system is accomplished by comparing the students’ diagram with the solution diagram. Prior to implementing the comparison process, we use notation extraction technique to derived each notation in the diagram and keep it as data in appropriate tables. Our proposed system prototype is not a marking system nor could it produce any grade for student’s answer. Our main objective is to enhance student’s understanding and give opportunity to student to learn from their mistakes. Students are allowed to resubmit their answer for the coursework according to the feedback that was generated from the system prototype. This process will iterate until their answer meets the solution answer by instructor using notation matching technique.

ARCHITECTURE OF OBJECT-ORIENTED MODEL ASSESSOR (OOMA)
The objective of architecture design is to identifying the process or subsystem and to show the control framework and communication within process or subsystem. By defining an appropriate architecture, it makes it easier to navigate the system, to find the location of a specific function and to map to the actual code easier. The architecture of Object-Oriented Model Assessor (OOMA) was shown in figure 1 below.
Generally, OOMA consists of two modules: a module for instructors and a module for students.

**Instructor Module**
This module requires inputs from instructor. Instructor can input UML class diagram as the model solution for given coursework problem from Rational Rose tool. Also instructor can input model solution for Z specification diagram for same coursework problem from Z/EVES tool.

**Student Module**
This module requires two inputs from the students; UML class diagram and Z specification. In our research, UML class diagram will be drawn by using Rational Rose and Z specification will be drawn by Z/EVES. The system will check the diagram based on three aspects;

- **Structure Analysis**
  This process will check the structure for UML class diagram and structure for Z specification as well. For example, UML class diagram must be having enough class according to system requirement. Each class must be having enough attributes and operations to ensure the system requirements specification is well designed. In Z specification, it must have enough schemes to ensure the all specification of the system is well modeled.

- **Verification Process**
  This process will check whether the inputs of the UML class diagram and Z specification are accurate. This process will compare students’ input with answer schemas by instructors.

- **Language Checking**
  The use of words within any diagrams that represent a software model is very important. The models can be misunderstood if the words used are not relevant. This process evaluates the naming of classes, attributes and operations for semi-formal modeling. The OOMA uses an establish dictionary to evaluate the suitability of words used in the diagram.

The above processes are carried out in order to verify that the input diagram is error free. The process of assessment is to compare students’ diagram with a solution model diagram. The solution model diagram is provided by the instructor. As a result, a list of feedback will be instantly generated from each process above that can be used as guidance for students to recognize their mistakes and to improve their understanding in the subject matter. For this purpose, each process will give feedback to student after the assessment process has been implemented. Each process will be implemented in sequence. This sequence process is vital.
in order to make sure the input of the students’ diagram match with those of instructors. The automatic checking process done by the system is accomplished by comparing the students’ diagram with the solution diagram, which was derived from instructor module. This system allows students to resubmit their answers to the coursework until their model is well design according to solution model by instructor.

EXAMPLE OF DIAGRAM ASSESSMENT
To indicate the implementation of assessment process for diagram-based coursework, we show an example of input and system output for semi-formal Modeling: UML Class Diagram. Figure 2 and figure 3 show the example of student’s input for UML Class Diagram and solution model for same UML Class diagram respectively.

![Figure 2. UML Class Diagram: student’s input](image)

![Figure 3. UML Class Diagram: solution model / instructor’s input](image)
In figure 2 and figure 3, we can see the difference between them. Through OOMA system prototype, this diagram will be assessed by a few steps of processes. In the first process, structure analysis, the quantity of class, attribute and operation for each class will be assessed. Referring to the above example of student's answer and instructor's input, the possible feedback or output is shown in table 1 below.

Table 1. Example of feedback for structure analysis process

<table>
<thead>
<tr>
<th>Structure</th>
<th>Feedback</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Quantity of class is enough?</td>
<td>No</td>
<td>Modified your UML class diagram using Rational Rose</td>
</tr>
<tr>
<td>Class : Account</td>
<td>No</td>
<td>Modified your UML class diagram using Rational Rose</td>
</tr>
<tr>
<td>Quantity of attribute is enough?</td>
<td>No</td>
<td>Modified your UML class diagram using Rational Rose</td>
</tr>
</tbody>
</table>
| Class : Customer   | Yes      | -suggestion-
| Quantity of attribute is enough? | Yes      | Name
| Class : CheckBook  | Yes      | Address                                               |
| Quantity of operation is enough? | Yes      | None                                                  |
| Class : CheckBook  | Yes      | None                                                  |
| Quantity of attribute is enough? | Yes      | None                                                  |
| Class : CheckBook  | Yes      | None                                                  |
| Quantity of operation is enough? | Yes      | None                                                  |

Students can learn from their mistakes by referring to the feedback given by the system. If the feedback is a 'Yes', the system will give suggestion to ensure the name of each notation in UML class diagram is the same with the solution answer in order to implement the assessment process correctly. But if the feedback is a 'No', the student is required to modify their diagram either by adding a new class structure or changing the existed class structure and resubmit their answer to be assess again until all of the basic structure for UML class diagram is drawn correctly and matched with the solution answer. For the next process, verification process will be implemented if the first process is done successfully where the structure of class is correct. For example, the UML class diagram in figure 2 is redrawn and shows in figure 4 below.

![Figure 4. Example of resubmit UML Class Diagram by student](image-url)
In the second process, the diagram will be assessed for the relationships between classes that are very important in modeling to ensure the communication within classes is correct. Referring to figure 3, Class Account is further classified into CheckingAccount using generalization in UML (represented by triangle symbol). The CheckingAccount is associated with class checkbook. The multiplicity constrains 1..20 means that an instance of CheckingAccount maps to at least one instance of class checkbook and at most twenty instances of checkbook. The checkbook has a composition relationship with class check (represented by filled diamond symbol). An association class transaction represents a relationship between class customer and account and has its own attributes. Referring to figure 4, the example of student’s resubmit answer for the same coursework, there are errors for this relationship. Table 2 will show the possible feedback or system output according to the student’s input.

**Table 2. Example of feedback for verification process**

<table>
<thead>
<tr>
<th>Relationship</th>
<th>Feedback</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Quantity of relationship is enough?</td>
<td>Yes</td>
<td>None</td>
</tr>
<tr>
<td>List of relationship Account-----Customer is correct?</td>
<td>Yes Association</td>
<td></td>
</tr>
<tr>
<td>Transaction is correct?</td>
<td>Yes</td>
<td>Association Class</td>
</tr>
<tr>
<td>Account-----CheckingAccount is correct?</td>
<td>No</td>
<td>Modified your UML Class Diagram</td>
</tr>
<tr>
<td>CheckingAccount---CheckBook is correct?</td>
<td>Yes Association</td>
<td></td>
</tr>
<tr>
<td>CheckBook---Check is correct?</td>
<td>No</td>
<td>Modified your UML Class Diagram</td>
</tr>
</tbody>
</table>

The system will give feedback according the student’s submitted answer and give the description for all feedback to the students. Student is required to modify their diagram either by adding a new relationship or changing the existed relationship and resubmit their answer to be assessed again until all of the relationships for UML class diagram is drawn correctly and matched with the solution answer. Finally, we hope student understand how to represent the system requirement using semi-formal model, UML class diagram and learn from their mistake by instant feedback given. The same process will be implementing in Z specification diagram. It is different with its structure and relationship name.

**CONCLUSION**

In this paper, we have presented the architecture of Object-Oriented Model Assessor (OOMA). Generally, the purpose of OOMA is to evaluate the input UML class diagram and Z specification diagram where an instant feedback is generated by the system as an output. Although there are some tools developed for the same purpose, most of the tools implemented the process of mapping the Object-Oriented Model to formal specification and the mapping implementation is automatically done without explanation made to the users. It makes the tools suitable for engineers or programmers but not suitable for users in academic fields. The suggested process in this prototype are hoped to help students to have better understanding in the basic concepts of Object-oriented Model, UML and formal specification, Z and finally UML class diagram. Assessment techniques should provide a dialogue to the instructor and the student. In this research, computer or system will become as a instructor to give feedback to the student. The feedback needs to identify the problem to the student and provide guidance as to how she or he may proceed. Scores are assigned to each module and given back to students along with written comments or suggestion; so that they may learn from this feedback to ensure their diagram is accurate with the answer that was given by instructors.
REFERENCES