The design and the implementation of MUML

Hazim RAWASHDEH
Tafila Technical University
Tafila, Jordan
hazim@ttu.edu.jo

Abdullah Moh ZIN
Universiti Kebangsaan Malaysia
Bangi, Malaysia
amz@ftsm.ukm.my

Sufian IDRISS
Universiti Kebangsaan Malaysia
Bangi, Malaysia
sufian@ftsm.ukm.my

ABSTRACT

This study describes the design of an automatic grading system for assessing graph-based assignments. Graph-based concept is taught in several computing courses such as Database, System Analysis and Software Engineering. In this study we take two necessary concepts into concern when assessing the answers; the model syntax and the model semantics. The syntax of the answers model or design can be achieved by using a free-syntax error modeling CASE tool such as Rational Rose or MagicDraw. On the other hand, the semantic of the model can be checked by mapping the model into a formal language like SMV. Therefore, the model properties can be checked against the requirements. In this study, the MagicDraw extremely is used for drawing the model and at the same time SMV is used for checking the model.

***Keyword**--: UML, SMV model checker, Automatic evaluation, diagram-based evaluation.*

INTRODUCTION

There is a great deal of literature on how to represent and recognize diagrams and a well developed theory of visual languages[1]. In spite of that most of automated examination evaluations systems are text-based tools where the questions are in a multiple choice format. Many of diagram-based evaluations tools were produced with various techniques and methods. For UML models so many of researchers are interesting and welling to have their grading tools in this language, but the work that has been done focus only on the

syntax check of the model that might be done by choosing the right modeling tool such as MagicDraw.

The common marking tools used to grade answers, are mostly a multiple choice based systems. The developers of these tools use several programming language to design the exam, but as most of them are web based exams, they use in that web pages design the scripting languages such as java script or VB script to enrich these tools. Instantly Ng[2] has showed up a Javascript approach to create a multiple self-marking engine. This engine was developed with the following features:

The questions are to be randomly selected from a pool of questions prepared. If the student is presented with a question that he/she does not wish to attempt, it should be possible to skip that particular question
The prepared questions are contained within a selection of HTML web pages
A student responds to the question by a simple selection of choices, from A to D, using the computer's mouse
The program compares the student's response with an attempted answer to the question and indicates whether it is right or wrong
If the response given is wrong, the student should be able to reselect until he finally gets it right
The engine does not grade the student's attempts as it is meant to be used as a tool by the student to gauge his/her understanding of the topics taught

The automatic grading of answers presented in a textual form has gained much attention over the years, as there are several systems being developed for marking textual material automatically as investigated in Burnstein et al.[3] and Haley et al.[4]. Most of these systems need students to type their answers into a web form using their computers at home to submit the answers passing through the Internet into a server for grading. This type of electronic examination system can give an instant feedback of the grade and can provide a textual feedback for the answers, so the results can be well accepted[5]. Constantinos Nikolou [7] has reviewed the state of the art in Automated Essay Scoring AES that evaluate written essays. Thomas [5] described the development of a marking tool that can help the marker with the marking of free-text response questions. His approach is to build certain tools to aid the instructor (marker) in the process of marking and not to automatically mark the free-text responses. Thomas [5] approach concludes with a discussion of suggested tools and features for inclusion in future development. In contrast to multiple choice and textual marking tools, there has been a little work on the creation or marking of diagrams marking of diagrammatic answers. Tsintsfas [7] has produced a framework for diagram-based evaluation of coursework which has fed into an ERD tool within the Course Marker Computer Based Evaluation (CBA) system. On the other hand Higgins and Bligh [8] as well as Batmaz and Hinde [9] have proposed a semi-automatic grading system.

A tool for Learning and automatically assessing graph-based diagrams for Entity Relation Diagrams (ERD) is produced by Pete Thomas e al. [6].

In the following sections we discuss the problem of diagram mapping and formal presentation, the approach to automatically marking diagrams and the built tools to make use of this technology for learning and evaluation. The last section discusses how we intend to take this research forward.

UML FORMAL PRESENTATION

We consider that the behavior of systems can be specified within a single statechart diagram. A requirement to formally verify behavioral properties is to map the statechart diagram into a formal semantic model. Definitely, the semantic model should satisfy the UML semantics of statecharts. A behavior of a single object or state a single statechart to be model checked needs indeed an interpretation of the statecharts model as a Kripke model. It is the responsibility of this research methodology to build a semantic model, required by the model checker CaSMV, for the system to be used later as an infrastructure for the proposed marking tool. It is the starting point to build the architecture of easy-to-use verification tool (Fig. 1) and the improved marking tool that assists in the process of formal verification and on the other hand to assess the student model in order to grade his/her level (Fig. 2).

The main objective of this research is to develop a method and criteria for grading the semantic of UML models. To achieve this objective a suitable platform should be used by the students to model the desired systems. MagicDraw case tool can be used or any other tool that support syntax check of UML models.

In this research, we attempt to model check the statecharts of UML using SMV model checker. For this purpose, we need a suitable algorithm to transfer UML statecharts into SMV codes. Figure 1 states the architecture of model checking UML statechart diagrams for this research. Not only class diagram, statechart and collaboration diagrams in a UML model are included, as we need only these diagrams for the model checking purpose.

Further more, the transformation algorithm process is applied to the said diagrams and we get SMV modules for each class in the UML model. Class diagrams confirm associations between classes and make the connection of all relative SMV modules connected. Collaboration diagrams are responsible of the information of message sequences to figure the specification of the model to be checked and to determine how the main SMV module would looks like. Once translation process is completed, the SMV model checker is applied to check the specifications in this formal language with a concrete mathematical basis which means that it is possible to check whether the system complies with definite desirable properties. As the more increasing complexity of today's software systems that perhaps requires to develop a new verification methods and tools, to carry the verification process either in an automatic or semi automatic manner. Note that the model checker will be run not only on the message sequences specifications but also on requirements specification given by the student or the teacher who design the model for desired properties. Later on when SMV generates a negative counter example the process might be applied to UML model or to a given specifications.

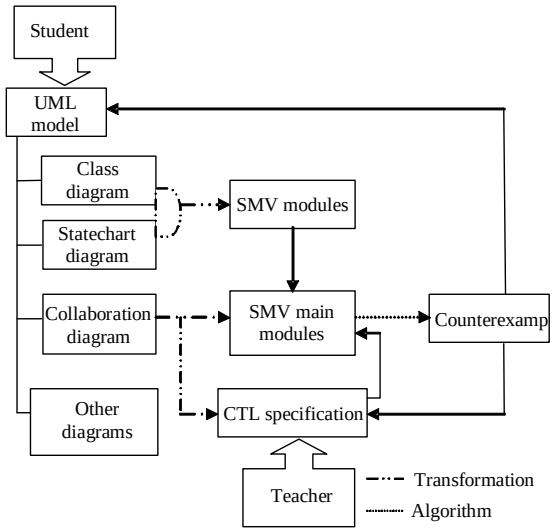

Figure 1: architecture of model checking UML statechart diagrams.

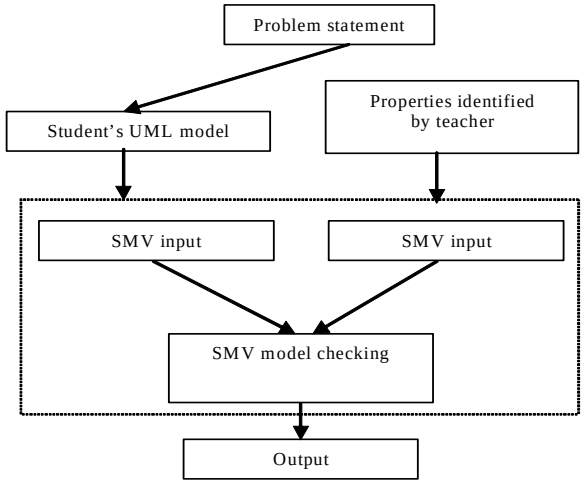

Figure 2: graphical representation of the verification tool

To construct the transformation algorithm, we adopt the STP-approach because it quite translates STATEMATE statecharts into SMV successfully. Furthermore STP approach is suitable for translating UML statecharts as the structures of UML and STATEMATE statecharts are very similar. In our study we are going to adopt a specific parts of STP algorithm that suits the transformation rules of our tool.

As shown in Fig. 2 the tool is performing, without interference on the user's part, a complete, automatic mapping of the behavior specified in UML into an SMV specification, focusing on reactive systems in which the active behavior of the classes is represented through state diagrams and activity diagrams to be used to reflect the behavior of class operations. XMI (OMG[10]) (XML Metadata Interchange) is used as the input format, so making it independent of the tool used for the system specification. Therefore the input format to our tool is an XML representation of the student model and for the proprieties we created XML format to support interoperability amongst front-end tools. To achieve this we create these models using a UML editor called MagicDraw with a plug-in that outputs the model in the XML format.

In contrast, the tool has a versatile patterns assistant that guides the teacher in how to insert properties so they will be verified later using temporal logic. The verification process is carried out as said in the previous section in such a way that the teacher needs no knowledge of either formal languages or temporal logic to be able to take advantage of its potential. Additionally, there is no need to know about the internal composition of variables and modules for verification process.

In parallel, a wizard of TUSAG helps the teacher to write properties to be verified using LTL (Linear Temporal Logic), in addition if the property is not satisfied, the tool illustrates a counterexample trace. In our tool, verification process is carried out using the SMV as discussed. With this tool, it is possible to make a complete automatic verification. i.e., given a property, a positive or negative respond is always obtained by the combined SMV and then the property should be expressed in a temporal logic presentation, with Computation Tree Logic (CTL) or Linear Temporal Logic (LTL). Writing a property is not as easy as wish for since the teacher should be expert in CTL or LTL and should be having an advanced knowledge in formal methods, logics and the type of specification acquired from the system. In fact our tool overcomes this problem as it has an assistant that direct the teacher through out writing the properties until the property to be verified is obtained following the suitable syntax.

PATTERN CLASSIFICATION

The starting point of this study was the pattern classification proposed by Dwyer et al where each property has been specified in LTL and it is established a first classification between occurrence and order patterns since most of the properties to be verified, fit in with one of these two categories[11]. In fact occurrence patterns describe the occurrence or absence of an event (state)/sequence of events or signal during the system computation. These patterns include absence (never) which expresses the ability of a piece of a system's execution from certain events or states. Universality (always) describes a piece of a system's execution that only contains events (states) where a desired property is achieved. Existence (sometimes) expresses the occurrence of a certain event (state) in a system's execution and the last type of occurrence patterns is bounded existence which describes the appearing of a specified number of events or states in a system's execution.

Order patterns express the ordering of several events (states). They include precedence as (s precedes p) that expresses the relationship between a pair of events (states) where the occurrence of the second event depends mainly on the occurrence of the first one. The second type of order patterns is response (s responds to p) that describes a cause-effect relationship between a pair of events. The third type is the combinations of both chain precedence and chain response, where chain precedence (s and t precede p or p precedes s and t) describes the precedence relation between two sequences of events and chain response (s and t respond to p or p responds to s and t) that expresses the response relation between two sequences of events, where the first sequence is considered as the chain stimulus and the second is considered as the chain response. And finally the constrain chain (s and t without z respond to p).

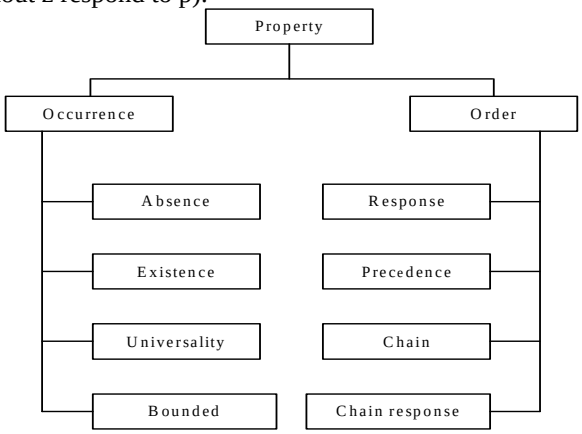

Figure 3: the pattern classification

The classification of the above patterns can be shown in Fig. 3.

In general, our aim is to fulfill properties that can be considered as requirements. The result of this process is that teachers can speed up the initial development of requirements models by the use of the properties patterns. The inserted properties for verification purpose have been arranged to create limits for the scopes (Q and R) and to state properties order when there is more than one property (s, t o z), that is why there is no need for the teacher to know or realize the specification structure obtained in SMV. In this study the type of established properties are a particular state of a state machine, a particular state of an object activity, a generated event and an attribute value comparison.

The marking tool will generate the property in the required format automatically, that's of course regarding to the option selected by the user and the chosen pattern. Therefore the tool can start and execute SMV using the plug-in to carry out the verification process. The tool then depends on the generated counter-example when the property is not satisfied.

As shown in Fig. 2, the student needs only a knowledge of the system studied and UML language as well and the rest procedures are to be obtained automatically by the tool it self

as it will represent the model formally in SMV from XMI textual representation. In parallel, the teacher is writing the properties with the help of tool wizard that use LTL (Linear Temporal Logic). Furthermore if the given property is not satisfied, the tool can show a counterexample trace. In our study we are not going to stop on the counterexample, since we plan to analyze the result (trace) to determine how many properties are not satisfied, so based on that the tool will decide the student's mark as we depend on the number of negative traces in the SMV output file. MUML is a formative grading tool as evaluation is performed throughout a course and not at the end of it.

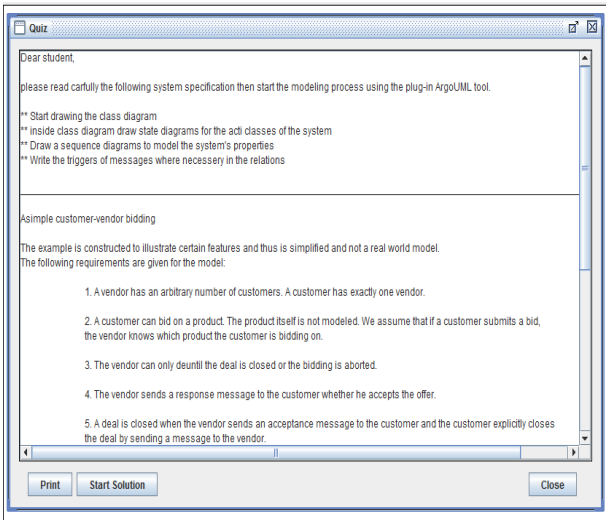

Figure 4: MUML students screen.

MUML ARCHITECTURE

This section presents the architecture of the development proposed automatic marking (grading) tool of student submitted course work. We consider the case where the exam submission consists of two parts: A design (using UML methodology) which is certainly prepared by the students. The student can read the title of the assignment displayed on MUML main screen, where this screen as shown in Fig. 4 consists of three steps explained subsequently.

The first step allows the student either to read the assignment or to print it, so in this case the student can perceive all the model specifications and constrains. If the student click the READ command then a Notepad file contains the whole system specifications is opened for him. The second step allows the student to open the MagicDraw CASE tool directly from this screen so he can draw the desired model in an independent screen. When the student finished the design he needs just to save the file as xmi (XML Metadata Interchange) and save it in the MUML folder, so MUML will export this file in order to

insert it into the integrated SMV model checker. On the other hand he or she can use the EDIT button to make any adjustment to the model (using MagicDraw) before submitting the assignment.

The third step is to choose the SUBMIT command and at this pint he the student has no control on the work as MUML imports the xmi file as an input file for the model checker. SMV will check the state space of the model and wait for the formal TLT properties of the model to be entered by the teacher. Figure 6 shows the MUML screen specified to the teacher. The teacher has many choices to do with the stored assignments. First of all he can browse the answer files of his students, whenever he chooses one, then it is the time to insert the system properties that he wants to check with the help of the plug-in patterns. These patterns appear on the right side of his screen.

The tool will accept the TLT (Linear Temporal Logic) with an assistant that leads the user through the properties writing until all of them are inserted with the appropriate syntax. So both the xmi of the student design and the xmi of the system properties are automatically mapped from the textual presentation into will start SMV formal presentation.

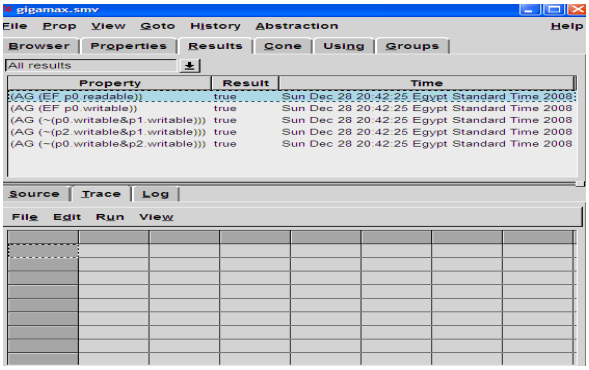

Figure 5: SMV true property verified example

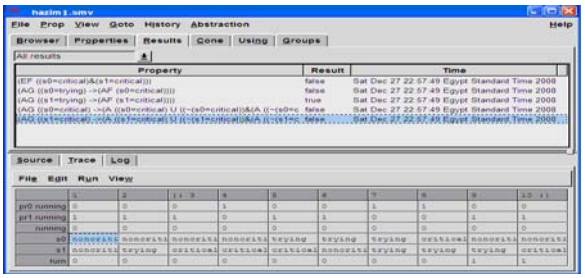

Figure 6: SMV true property verified example

After that MUML will start SMV model checker in separate window to check the properties, as if the property is not satisfied, the tool shows a counterexample trace (Fig. 5)

otherwise it will show an empty trace for the correct proprieties. In all cases MUML will deal with output file of SMV that contains the results, and has to determine the student mark depending on the number of true and false results obtained from this file. As an example Fig. 3 and 4 shows a screen shots of the verification result in SMV, while Fig. 3 shows a true result for the properties, Fig. 4 shows a false results for the given properties. In many cases the result is a mixed up of true and false results which means the student will get a mark that is between zero and ten as shown in the next example:

MUML EVALUATION AND MODEL CHECKING RESULTS.

The student's mark is to be given according to the output file generated by SMV model checker. The output file should be occurred as shown in the next example:

(EF((s0 = critical) and (s1 = critical))).....false
(AG ((s0 = trying)->(AF (s0 = critical)))).....false
(AG ((s1 = trying)->(AF(s1 = critical)))).....true
(AG ((s0 = critical)->(A ((s0 = critical) U
((~(s0 = critical)) and (A ((~(s0 = c…..……...….false
(AG ((s1 = critical)->(A ((s1 = critical) U
((~(s1 = critical)) and (A ((~(s1 = c..…….....false

In this example and according to the shown MUML will give the student two marks out of ten, hence we have four false results out of five. One more scenario of the output is to have the property not verified, which means the result of that property (properties) verification is false. In this case the student should get zero mark for the semantic part of the model (as we consider that his model is free of syntax errors) if the result is false for all properties. As discussed the tool depends on the ratio of true or false result to the total number of the verified properties.

MUML therefore is an integration of the explained tools in the previous section. These tools are the UML CASE tool whereas use MagicDraw for this purpose, TLT patterns are used for the formal presentation of the model, the symbolic model checker SMV is used to check the model semantics and finally MUML analyzer that analyze the model checker result in order to determine the student mark.

One of the major objectives of this study is to evaluate the effectiveness of the proposed method and tool MUML. This can be achieved by comparing the performance of the tool against human graders i.e., instructors of software engineering course. This study set about building a significant amount of student drawn UML diagrams. Almost 400 diagrams were gathered that drawn by the students. Figure 1 is an example solution to the created question).

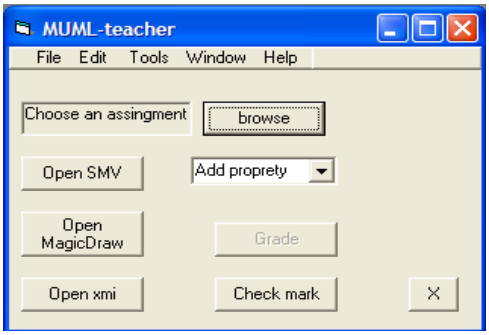

Figure 7: teacher's marking window

Table 1: the difference between the MUML and human markers.

Difference	Diagrams	Percentage	Total (%)
0.00	203	68.80	68.8
0.50	66	22.30	91.1
1.00	24	8.10	99.2
1.50	1	0.33	99.6
2.00	0	0.00	99.6
2.50	1	0.33	100.0

These diagrams were actually graded by a group of human markers whose work was only to check for consistency and accuracy to be sure that these diagrams reflect the accurate diagram content. In this study a set of 100 was used in the marking algorithm development, for detecting errors and bugs as this set can be used for training.. The rest of these diagrams (295) were used to test the accuracy of the automatic marker when compared to the human markers.

Table 1 shows the results of this experimentation. The diagrams were marked out of 7 and rounded to the nearest half mark. It shows also that the results of MUML agree closely with the human markers in approximately 68.8% of all cases. However we feel that over 90% reflect the accuracy of the used automatic marker.

We consider that MUML provides the student with two elements of feed back when he/she submits any of the given diagrams for grading, first of all a grade that shows how the student's attempts matches the correct solution; second element is a copy of the student's answer, which covers the example solution showing the proprieties that either have a true or false result for the verification process.

CONCLUSION

The performed experiments that we have performed on MUML automatic marking tool have given very hopeful results. We have a method and framework for capturing, processing and mapping graph-based diagrams formally i.e. they can work in a practical learning environment. The mapping phase applied to UML diagrams' grading and compared with human markers has shown the efficiency of the tool, on the other hand we planned to enhance some aspects of its performance by letting the tool to provide a feed back of the students design drawbacks therefore it can be used as a learning tool.

As future research, more UML diagrams, both versions can be used for the verification and grading process since at the time been it is applied only on the most used diagrams such as Class Use case and state and activity diagrams as stated in the survey done by Dobing and Parsons[12].

REFERENCES

Thomas, P.G., K. Waugh and N. Smith, (2006). *Using patterns in the automatic marking of ER diagrams*. Proceedings of the 11th Annual Conference on Innovation and Technology in Computer Science Education, June 26-28, ITiCSE, Bologna, Italy, pp: 403-413.

ThesisAbstracts Retrieved from http://intranet.cs.man.ac.uk/Intranet_subweb/library/ ThesisAbstracts/MSc03/nikolou.pdf. last visited 1/2/2009.

Higgins, C.A. and B. Bligh,(2006). *Formative computer based evaluation in diagram based domains.* Proceedings of the 11th Annual Conference on Innovation and Technology in Computer Science Education, Jun. 26-28, ITiCSE, Bologna, Italy, pp: 98-102.

Batmaz, F. and C.J. Hinde,(2006). *A diagram drawing tool for semi-automatic evaluation of conceptual database diagrams.* Proceedings of the 10th Annual International Conference in Computer Assisted Evaluation, Loughborough University, Loughborough, UK., pp: 68-81.

OMG.,(2003). *XML Metadata Interchange (XMI)* v 2.0. OMG Document 2003-05-02.

Beato, M., M.B. Solorzano and C. Cuesta, (2004). *UML automatic verification tool (TABU). Proceeding of the Specification and Verification of Component Based*

Systems (SAVCBS), SIGSOFT 2004/FSE-12 12th ACM SIGSOFT Symposium on the Foundations of Software Engineering, pp: 106-109.

Dobing, B. and J. Parsons, (2008). *Dimensions of UML use: A survey of practitioners. J. Database Manage*., 19: 1-18.

Jurafsky, D. and J.H. Martin., (2000). *Speech and language processing. Proceedings of the an Introduction to Natural Language Processing, Computational linguistics and Speech Recognition*, Upper Saddle River, Prentice Hall.

Ng, T.W. (2000). *Creating a multiple-choice self-marking engine on the internet*. Int. J. Eng., 16: 50-55.

Burstein, J., M. Chodorow and C. Leacock, (2003). Criterion SM online essay evaluation*: An application for automated evaluation of student essays*. Proceedings of the 15th Annual Conference on Innovative Applications of Artificial Intelligence, Acapulco, Mexico.

Trusso, H.D., P. Thomas, A.D. Roeck and M. Petre, (2005). *A research taxonomy for latent semantic analysis-based educational applications*. Proceedings of the International Conference on Recent Advances in Natural Language, Borovets, Bulgaria, pp: 575-579.

Thomas, P.G., K. Waugh and N. Smith, (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, pp: 158-162.