Jurnal Natural
Vol. 11, No. 2, 2011

USING A USER MODEL FOR ENHANCING ASSESSMENT AND FEEDBACK IN E-LEARNING

Rajibussalim

Jurusan Fisika, Fakultas Matematika dan Ilmu Pengetahuan Alam, Universitas Syiah Kuala Darussalam, Banda Aceh
Email: rajibussalim@fmipa.unsyiah.ac.id

Abstract. The study is about a method for enhancing the assessment in e-learning. We study summative assessment that enables teachers or examiners to know if their students have reached an appropriate level of knowledge and formative assessment that enables a student to check how well they are performing in learning. This type of assessment takes the form of a more detailed feedback such as in what aspects a student posses strengths as well as where they have weaknesses. We implement a new approach for enhancing assessment in e-learning by building a system that utilizing a user model to capture user's knowledge of a concept and to keep updating it in their user models. We implement the system by using the Multiple Choice Questions (MCQs) that were combined with a standard of test called the Question Test of Interoperability (QTI). As the result, more positive and negative feedback can be offered to e-learning users.

Keywords: assessment, user model and feedback.

I. INTRODUCTION

Assessment is one of the important component of e-learning in which it deals with a diverse range of computer supports for improving learning. A computer based assessment, where a computer provides systematic testing of a learner's knowledge, has been widely used in the last two decades. The Objective tests and Multiple Choice Questions (MCQs) were among the most widely used and well-developed assessment tools in e-learning. The other type of questions were yes-no questions, fill-in questions with a string or numeric answer, and graphical representation questions [1].

It is important that the assessements in e-learning systems are able to provide a systematic testing of a learner's knowledge. For example, an Intelligent Teaching System (ITS) aims to personalise the teaching on aspects like the learner's knowledge. In order to do this, the ITS needs to create and continuously update and refine its model of the learner's knowledge (user model). In this context, the assessment is called diagnosis. There are also many other roles for assessment in learning. These are often splitted into two categories, formative and summative assessment. A summative assessment typically takes the form of an exam in which the result is translated into a grade; typically a single number. By conducting the summative assessment, the teachers or examiners were able to evaluate if the students have reached an appropriate level of knowledge. In contrast, formative assessment is an assessment that is used for self-assessment and self-evaluation. The formative assessment allows students to check how well they are learning. This can, and should, take the form of more detailed feedback of which aspects the student have learnt as well as those they have not learnt. From this feedback, the student can plan future actions to overcome the weaknesses and build upon their strengths. In either case, the assessment serves to provide measures of the learner's knowledge [2].

There has been some important research related to the uses of MCQs for the assessment. Each study serves to provide measures of learner's knowledge for different purposes. For example Conejo used MCQs in SIETTE [1]". SIETTE allows teachers to modify and develop their MCQs on-line and ensure the consistency and the accuracy of assessments to their students. SIETTE makes use the concept of Computer Adaptive Testing (CAT) and Item Repository Theory (IRT) and combines them to build the system. The CAT was used as an iterative algorithm that started with an initial estimation of the learner's knowledge that served as preliminary knowledge of their user model. The information in this model was updated continuously by the system each time they take a test and be used to provide the test generator module with adaptive capabilities [1].

In addition, MCQs are also used in TOKA, a web-application used as Computer Assisted Assessment (CAA) for generating and automatically evaluating multilingual test exercises. It helps teachers or instructors to generate questions and to mark the tests as explained in Kerejeta [2]". Furthermore Kerejeta said: ``*It is based on multiple choice questions and provides a platform for teacher-directed assessment and self-evaluation*" and ``*TOKA provides some utilities that help users (the teacher or the students) to construct the exams*". In TOKA, subjects were represented in a tree topic that possess the information on weight and the level of difficulty. The user, when creating an exam automatically, selects the item parameters of the exam that must be fulfiled. This parameter include the number of items, topics, difficulty levels, the importance and inferred difficulty. In the next step, TOKA searches appropriate items in a database and then presented to the user. The user then can inspect and change the topic distribution of the items and the overall difficulty, importance and inferred difficulty. An essential advantage of TOKA lies on its interface that is internationalised and which can support multiple languages. Thus, it allows management of a course in more than one languages [2].

Another important implementation of MCQs in e-learning have been reported in a research of the QuizPACK [3]. The QuizPACK is a system that able to generate parameterised exercises for the C language and automatically evaluate the correctness of student answers. This research suggested an automatic generation and evaluation of MCQs code-execution exercises based on the run-time execution of parameterised code fragments rather than the standard MCQs that are generated by specifying the text of a question and a set of possible answers. For example, after the teacher provide a parameterised fragment of code to be executed and an expression within that code, the system, by using this parameterised code, continues the work to randomly generates the question parameter and presentation in a web-based quiz, accepted input from students, generate the answer and provide the score and finally records the results. It has been reported that the implementation of this research was exceptionally useful when it was used in an out-of-class assessment mode [3].

Furthermore, the MCQ was also used by ITiCSE working group as a mean to evaluate the poor performance of students in an introductory programming courses at twelve institutions from several different countries [4]. The group of researchers were conducting similar research on their students at their own institutions. The research aim was to evaluate why many students do not know how to program at the end of their programming courses. In this research, the students were asked to answer twelve MCQS with different level of difficulties in Java programming that could be easily re-written to the other programming languages. The purpose of these questions was to test the generic student's knowledge and skills in programming. As a result, it was concluded that many the first-year programming students could not program due to the lack of knowledge and skills that were prerequisite to the problem-solving in programming.

The Question and Test Interoperability

The Question and Test Interoperability (QTI) specification was defined by the IMS Global Learning Consortium to enable the exchange of questions and tests, and their correspondence result reports within the Learning Management Systems (LMS). The IMS-QTI format is among widely used and well-developed tools for online testing. It is because its high-modularity and extendability of the format that simplifies the assessment procedures in term of generation, execution, presentation, grading and archiving. Although it follows IMS specifications, the IMS-QTI able to export and import assessment from a wide range of format and maintaining a database question that can be reused in a wide variety of authoring situation [5].

II. METODOLOGY

Creating a User Model

A user model is a collection of information representing the actual learner, that is to record behaviour, learner's level of knowledge and preferences [6]. Representation of a user model can take various forms such as a simple display of user's knowledge progress measurement meter as in [7] to a more complex representation such as a hierarchical tree structure in [8]; or a conceptual graph in [9]; or textual description of knowledge and misconceptions [10].

A user model can be created by using various available um toolkit as mentioned in Kay [6]. To construct a suitable user model for assessment purposes, however, could be a tough process as reported by Self in [11], Kay in [6] and Guzman in his recent publication [12]. This is because some requirements to ensure scientific adequacy must be fulfilled by all measurement tools.

A more advanced user model enables learners to interact with his/her user model and to add some

valuable information on it as explained by Kay [13] and Bull [14]. The ability of the user to interact with his/her user model is important in order to keep information of user's behaviour, preferences and level of knowledge in the model is accurate and updated. This is also important because the updated information in the user model will be used in diagnosis assessment of the user in the actual test.

In this research, a user model that support diagnosis approach proposed by Dimitrova [15] is built. A user model is important to enable personalisation the teaching on aspect like the learner's knowledge of programming [6]. This user model is continuously updated and refined. This condition is necessary in order to be able to evaluate the user's programming knowledge as precisely as possible. By having the information updated in learner's user model, the intelligent system that perform the test can adjust the level of difficulties of MCQs provided to the learners. Then, any response given by the learner will be used as evidence that contributes to learner's knowledge in its user model. Hence, the test can closely identify an accurate level of programming knowledge the learners have.

PersonisLite

To create a user model, however, is not easy and time consuming. Therefore, the PersonisLite: a user model server that have been developed at The University of Sydney will be used. The PersonisLite user model is a simplified version of Personis software [16]. The principal idea of the personis is to allow the adaptive systems scrutable, hence the learners not only enable to see the details of the information held about them but also the way that the information is used to personalise an adaptive hypertext.

Using PersonisLite could save a considerable amount of time and effort in creating personalised user models that are scrutable and reliable. When the time constraint is important as in this research, the use of a readily available tool such as PersonisLite for creating a user model is very beneficial. Thus, the research could be concentrated more on adding a knowledge layer to MCQs as this thesis aims to address this issue by proposing a new approach to evaluate the effectiveness of MCQs type questions in examining the learner's knowledge of programming.

Incorporating MCQs into QTI

Ideally, an exam have to be represented in standard formats that ensure their independence of any Learning Management System (LMS), thus it can promote the durability and portability, and also reusability of individual questions in the authorisation of a new exam. The IMS-QTI format are able to address these issues and able to facilitate the production of exams on the fly by the most LMS in order to evaluate the skills of a particular learner in the context of learning process [5]. The interchange data that follows the IMS-QTI standards is enable by binding its abstract model with XML format.

Adding a New Layer to MCQs Based User Model

A new approach have been explored to enhance assessment in e-learning that is by adding a knowledge layer to a standard approach of testing. The particular testing method we explore is the Multiple Choice Question (MCQ) type of testing. This is because the MCQ is widely used in e-learning testing, as reflected in the existence of a well accepted standard, the Question Test of Interoperability (QTI) standard.

As part of this work, a new design and prototype implementation of Knowledge layer User Model (K-umCQs) has been built. This design makes use of Question and Test Interoperability (QTI) standard compliant MCQs and expands it into the extended QTI-XML MCQs format to enable information on particular area of learner's programming knowledge to be precisely captured in their user model. K-umCQ stores detailed information captured about a user into a user model. This means that every response a user gives in answering MCQs is added or updated to their user model. The information is become evidence to the learner's level of knowledge of particular domain.

Although only one possible correct answer exist for each question, there are a number possible answers presented. This means that every answer can contribute to a user model being updated and for each answer given, it is also possible to have one or more user models updated.

The procedure for updating the user model involve three classes, the QuestionComponent class, the AnswerComponent class and ModellingComponent class. The QuestionComponent class stored the question and possible answers that will be presented in K-umCQ question interface. Each of question and possible answers possess a unique identifier (question id and answer id). The possible answers are store in an ArrayList in the QuestionComponent object as AnswerComponent object.

The possible answers to each question either correct or incorrect and a list of ModellingComponent object is stored by the AnswerComponent class. ModellingComponent object are also stored in an ArrayList. Every QuestionComponent object has a list of unique AnswerComponent objects

Within a user model many components of learner knowledge can be updated. One part of user model component is updated by one ModellingComponent object within the ModellingComponent class. Each of them can have either ``positive'' or ``negative'' value updated. ModellingComponent object is also responsible for diplaying its description as a feedback to the user. Due to each of these objects relate to its correspondence part within the user model, it is possible to share references to these objects between one or more AnswerComponent objects. Each of these objects has a unique identifier thus any reference made by AnswerComponent object to these ModellingComponent objects is using this unique identifier.

To keep the consistency of the data throughout the application that using these data, all methods and variables is declared static and each QuestionComponent object is stored in a globally shared collection inside a QuestionXML class This is particularly useful to maintain QuestionComponent objects are being altered unintentionally during question authoring by the teachers.

III. RESULTS AND DISSCUSSION

In K-umCQ, the *QTI-XML* standar is extended to satisfy range of learner's knowledge representation that wanted to be captured in a user model. IMS-QTI specification for K-umCQ is altered in order to fit in with the requirements of authoring and feedback mechanism standard. The modification is in the attributes and tags. Due to these tags are also related to a user model and update information on it, some extra information is embedded into different tags so that it is sufficient information to be captured and stored in a user model. The tag that hold important information about the user model definition file is defined as <qticomments>. This important file is required to build a complete user model. The opening and the closing tag of an QTI-XML file is defined by a pair of <questestinterop> tag as illustrated in the Figure 1. The <questestinterop> tag is the element of the outermost container for the QTI content i.e the container of an Assessment, a Section and Item and only defined once for each QTI-XML instance file.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE questestinterop SYSTEM "http://www.it.usyd.edu.au/~raji0050/rssmcqfeed/ims_qtiasiv1p2p1.dtd">
<questestinterop>
  <qticomment>http://www.it.usyd.edu.au/~raji/rssmcqfeed/test_question.xml</qticomment>
    Question Definition
</questestinterop>
```

Figure 1. The *QTI-XML* files representation

In the IMS-QTI format, a question is identified as an item and it is representated in a pair of triangle bracket e.g <item>. This is called an item block. Inside this item block, a number of subsection are defined. The defined subsections include the question itself, a number of possible answers, answers processing and feedback. However, those subsections can be classified into three general classifications namely the presentation section (<presentation>), the processing section (<resprocessing>) and the feedback section (zero or more <itemfeedback>). This classification follows the IMS-QTI standard specification version 1.2.1.

For our work, however, this standard need to be extended to satisfy range of learner's knowledge representation that wanted to be captured in a user mode for example to provide better feedback to the learner. We consider that the term of ``*to provide better feedback*'' as ``*to provide more detailed feedback*'' according to the context and components that the learner have the weaknesses or strength. This is opposite to the typical available ITS systems that can only give a correct or incorrect feedback without further explanation in what aspect the learner knowledge of certain programming knowledge is weak or strong.

The extensions are on the addition of the new tags responsible for capturing more information about the users and thus enable sytem to generate feedback to the user based on the answer they give, either correct or incorrect responses. For a correct response a user gives, the QTI-XML subsection file will generate one or more positive feedback according to the context and component the question want to captured. This feedback is presented to the user in feedback user-interface windows and also automatically stored into learner's user model as a positive evidence to the corresponding context and component of particular domain. In opposite, a response to an incorrect answer will generate one or more negative feedback in feedback user-interface windows and will be stored in the user model.

K-umCQ can satisfy this requirement. It can provide the positive or negative feedback to the learner and automatically update the learner's user

model according to the context and component of the answer provided. The positive and negative feedback stored in student's user models could be projected and evaluated as illustrated in Figure 2.

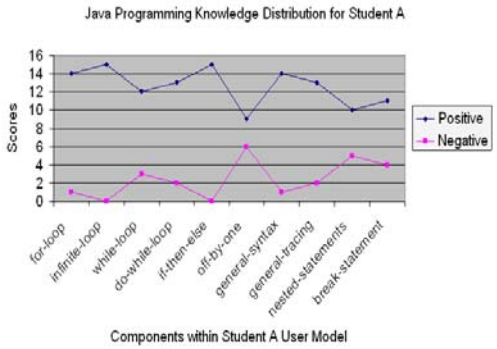

Figure 2. Example of saved attributes in a user model

As we can see from Figure 2, the student A's user model stored both positive and negative feedback for each programming topic. There was a lot more positive feedback recorded than the negative one. The feedback was projected in score. From the figure we can learn that student A received on average 14 feedback for the infinite-loop, if-then-else, and general syntax category and below 14 feedback for the other category. This is the opposite for the stored negative feedback in which the three categories mentioned before received only a few negative feedback (below 2). By knowing this figure, the teacher could direct student A to aware and improve his/her programming skill in the area that received more negative feedback that the positive ones, for example in the off-by-one topic as shown in Figure 2.

CONCLUSION

The primary goal of this project was to develop a prototype of the system that enables the information given by the learner when answering MCQs be effectively captured and stored as evidence in the learner's user model. Thus the evidence can be used as valuable feedback to improve learner's knowledge in particular area of programming. In order to achieve this goal, K-umCQ model was developed. The K-umCQ is a prototype and a novel design for enhancing assessment in e-learning by adding a knowledge layer to a standards MCQs. The system makes use of Questions and Test Interoperability (QTI) standards compliant MCQs and extended it to enable information on particular areas of a learner's programming knowledge to be precisely captured and stored in the user model. As a result, a more detailed feedback can be offered to the user of e-learning systems.

REFERRENCES

1. R. Conejo, E. Guzman, E. Millan, M. Trella, J.L. Perez-De-La-Cruz, and A. Rios, "SIETTE: A web-based tool for adaptive testing," *International Journal of Artificial Intelligent in Education*, vol. 14, 2004, pp. 1-33.

2. M. Kerejeta, A. Arruarte, and J.A. Elorriaga, "TOKA: A Computer Assisted Assessment Tool Integrated in a Real Use Context," *ICALT '05: Proceedings of the Fifth IEEE International Conference on Advanced Learning Technologies*, Washington, DC, USA: IEEE Computer Society, 2005, pp. 848-852.

3. P. Brusilovsky and S. Sosnovsky, "Individualized exercises for self-assessment of programming knowledge: An evaluation of QuizPACK," *Journal of Educational Resource Computing.*, vol. 5, 2005, p. article 6.

4. R. Lister, E.S. Adams, S. Fitzgerald, W. Fone, J. Hamer, M. Lindholm, R. McCartney, J.E. Moström, K. Sanders, O. Seppälä, B. Simon, and L. Thomas, "A multi-national study of reading and tracing skills in novice programmers," *ITiCSE-WGR '04: Working group reports from ITiCSE on Innovation and technology in computer science education*, New York, NY, USA: ACM Press, 2004, pp. 119-150.

5. I. Martínez-Ortiz, P. Moreno-Ger, J.L. Sierra, and B. Fernández-Manjón, "<e-QTI>: A Reusable Assessment Engine.," *ICWL*, 2006, pp. 134-145.

6. J. Kay and R.J. Kummerfeld, "An Individualised Course for the C Programming Language," *Proceeding of The Second International World Wide Web Conference*, Chicago, USA: 1994.

7. G. Weber and P. Brusilovsky, "ELM-ART: An adaptive versatile system for web-based instruction," *International Journal of AI in Education*, vol. 12, 2001, pp. 351-384.

8. J. Kay, "Learner Know Thyself: Student models to give learner control and responsibility," *International Conference on Computer in Education*, AACE, 1997, pp. 17-24.

9. V. Dimitrova, "Style-OLM: Interactive Open Learner Modelling," *International Journal of AI in Education*, vol. 13, 2003, pp. 35-78.

10. S. Bull and H. Pain, "Did I say what I think I said, and do you agree with me? Inspecting and Questioning the student model," *Proceeding of AI in Education*, AACE, 1995, pp. 501-508.

11. J. Self, "Formal approaches to student modelling," *G.I. McCalla and J. Greer (eds.), TR 92, AI in Education, Student Modelling: the key to individualized knowledge-based instruction*, 1994, pp. 295-352.

12. E. Guzman, R. Conejo, and J.L. Perez-De-La-Cruz, "Adaptive testing for hierarchical student models," *User Modeling and User-Adapted Interaction*, vol. 17, 2007, pp. 119-157.

13. J. Kay and B. Kummerfeld, "User Models for Customized Hypertext," *Intelligent Hypertext: Advanced Techniques for the World Wide Web*, London, UK: Springer-Verlag, 1997, pp. 47-69.

14. S. Bull and J. Kay, "A Framework for Designing and Analysing Open Learner Modelling," *Proceeding of AIED 05 Workshop on Learner Modelling for Reflection: the 12th International Conference on Artificial Intelligence in Education*, Amsterdam, Netherlands: AIED, 2005, pp. 81-90.

15. V. Dimitrova, J.A. Self, and P. Brna, "Maintaining a Jointly Constructed Student Model," *The 9th International Conference of Artificial Intelligence: Methodology, Systems, and Applications*, Heidelberg, Berlin: Springer, 2000, pp. 221-231.

16. J. Kay, B. Kummerfeld, and P. Lauder, "Personis: A Server for User Models," *AH '02: Proceedings of the Second International Conference on Adaptive Hypermedia and Adaptive Web-Based Systems*, London, UK: Springer-Verlag, 2002, pp. 203-212.