An Adaptation of Agent-Based Computer-Assisted Assessment into E-Learning Environment

Suraya Masrom, Abdullah Sani Abd. Rahman

Abstract— Computer programming is considered a fundamental subject in most of engineering and computer related programs in many universities. Usually, the teaching responsibility is given to a single department, which will serve the entire university. The limitation on staffing most of the time, results in the creation of big-sized classes. This in turn, will produce voluminous workload to the teaching staff, hence, rendering close monitoring of students' performance next to impossible. Relief might be possible by venturing into the realm of a Computer-Assisted Assessment (CAA). This paper describes our experience in designing and implementing a CAA System that is based on distributed and agent-oriented architectures. The system design caters for two primary objectives; to harness the power of distributed computing and to enable the integration of CAA into existing E-Learning System. We have analyzed and extracted important functionalities required for the system and describe agent's roles for each of the functionalities. We also describe the communication and accessibility aspect of the agents. We have developed an independent assessment module and performed some tests in a computer laboratory environment. We present in the last section of the paper, our findings regarding the feasibility, scalability and accuracy of the assessment module's as well as its limitations.

Keywords— Agent-based system, Computer programming, Computer-Assisted Assessment (CAA), E-learning

INTRODUCTION

THE increasing number in programming classes [1, 2] warrants the use of computer supported systems in relieving lecturers' academic tasks. Student assessment has become an important issue because of the large number of students involved. For a fundamental course such as computer programming, it has become necessary to be able to oversee students' weekly progress to make sure that every student is at par with one another. Considering the difficulties involved, Computer-Assisted Assessment (CAA), might be able to offer some help in reducing lecturers' marking chores and results management [3]. As a consequence, the number of assessment can be increased gradually and student performances can be monitored very closely.

Manuscript received June 12, 2009: Revised version received June 13, 2009. This work was supported by the Universiti Teknologi MARA, Malaysia under Grant 600-IRDC/SSP/DANA 5/3/Dsp (64/2008).

Suraya Masrom. Author is with the Universiti Teknologi MARA, Perak, MALAYSIA, Phone: 605-3742720; Fax: 605-3742261; e-mail: suray078@perak.uitm.edu.my.

Abdullah Sani Abd. Rahman. Author, is a lecturer at Universiti Teknologi PETRONAS, Bandar Seri Iskandar, Perak, MALAYSIA. Phone: 605-3687490; Fax: 605 -3656180; e-mail: sanirahman@petronas.com.my

We believe the incorporation of CAA into E-learning environment will be beneficial to both lecturers and students. Students and lecturers in an E-learning environment can be located anywhere in the globe, they don't have to synchronize their meeting and lecturer can give their notices, notes, assignment task anytime. The scope of functions in CAA for E-learning environment discussed in this paper are not just limited to the automated marking process but are also geared towards the possibility of providing active resource discovery and delivery. In this work, we rely on the agent oriented concepts and technology to create pro-active environments for the e-learning system. Agents are autonomous programs that are given specific tasks in a distributed environment. They take inputs from the environment and make decisions on what output to produce based on constraints and priorities given to them by the programmer.

I. RELATED WORKS

A. *The development of CAA*

An example of CAA can be seen in the work of [4] that introduces a scheme that analysed submissions across several criteria. The system was named as ASSYST and has capability to analyse the correctness, efficiency, complexity and style of a program. The BOSS system in [5] that is similar to ASSYST, runs on Unix operating system and is used for C programming assessment. The latest version of BOSS facilitates JAVA GUI application for the tutor's grading and assignment management. Michael [6] details a system called GAME for grading variety of programming languages by comparing the program outputs with marking scheme written in XML scripting. The system can examine program structure and correctness of the program's output. It has also been tested on a number of student programming exercises and assignment. The analysis of comparing human marking with GAME system provides encouraging results.

B. *Agent Based System*

According to [7], agents are considered one of the most important paradigms that on the one hand may improve on current methods for conceptualizing, designing and implementing software systems and on the other hand may be the solution to the legacy software integration problem. As explained in paper [8] , an agent is simply another kind of software abstraction. Compared to object oriented abstraction that describes methods and attributes of a software component, and agent is however, is an extremely high-level software abstraction. It provides a convenient and powerful way to describe a complex software entity. Rather than being defined in terms of methods and attributes, an agent is defined in terms of its behavior.

As mention in [9], the mobile agent technology can overcome some limitations of the well known client-server model regarding to the scalability as well as network delay performances issue. In agent-based concepts, the user agents are dynamically created not depending on the number of limited threshold as in the client-server concepts. In JADE [10], mobile agents and users agents can be created indefinitely from the agent repository in order entertain all incoming requests.

Another advantage of agent-based system compared to a client-server paradigm is that an agent can be loaded with tasks and processes, which will be encapsulated to be a component or entity, and can be mobilised to another machine to execute the preprogrammed tasks in the client machine. In this way, server loading process can be released more and, thus, will reduce network transmission overhead.

Recently, the architecture of agent based has been widely adopted in many kind of system as well as in E-learning system. An example of multi-agent system used for supporting cooperative learning in the classroom can be seen in the work of Leen-Kiat *et al*.[11]. The multi-agent system was named as I-MINDS, it consists of a teacher agent that can monitor student activities and helps the teacher manage classes. A student agent, on the other hand, interacts with the teacher agent and other student agents to support cooperative learning activities behind-the-scene for a student.

An agent-based quality assurance assessment system for educational institution has been introduced in paper of [8]. The agents in the system are used to check essential requirements that educational institution clients have to meet and prepare reports for assessors. The system proves useful and helps in reducing assessment time from education expert assessors.

An Adaptive E-Learning System based on Intelligent agents or IAELS has been proposed in the paper of [12]. The intelligent agent community has been used in the system to help the learners finding out the adapting courses and learning path. The features of the system include analyzing the cause of learning inefficiency, promoting learners' learning efficiency by personalized coursed and learning paths through the information analyzed by agents. So, learners will spend less time in making teaching materials for teachers.

II. MAIN COMPONENTS IDENTIFICATION

The main components involved in a student evaluation have been identified as lecturer, student, assessment, assessment type, question, answer scheme and assessment engine. The relationship between the components is formed when the lecturer creates an assessment job. The assessment may be of different types and contains a set of selected questions. Examples of assessment type are: lab test, assignment and projects. A lecturer can create an assessment by selecting the questions from a database. Answer scheme can either be retrieved from the database or created by the lecturer along with the test data.

The assessment engine acquires the "knowledge" for evaluating students' answers from the question database. Students can write, compile and debug their program by using their own programming environment. They only need to allocate a specific folder for their answer file or source code on their computer. Pre-setting can also be done via web-based student interface. As soon as the assessment is completed students will be able to view their result directly from the e-learning system.

III. MAIN FUNCTIONAL TASKS

The next step in the analysis is identifying the functional requirements of the system. We consider the main external entity that interacts to the system as the Student and Lecturer. The main functionalities provided for the student are: undergo assessment and view results. Lecturer entity requires functionalities such as: create/schedule an assessment, modify assessment, create question and answer scheme, modify assessment and answer scheme, delete assessment and answer scheme, and view students' result. These are the required functionalities to be integrated with an existing e-learning system.

IV. AGENTS' ROLE

The system architecture is developed by transforming the functionality requirements into a set of tasks the system has to perform. Each task is then assigned to one or more agents. Our design requires three types of agents: Coordinator Agent, Personal Agent and Assessment Agent.

A. Coordinator Agent

Services related to agents creation and task distributions are under the responsibility of the Coordinator Agent. This agent is created on the main container of the JADE system and has some important parallel tasks to perform. Each of these tasks will be implemented in a separate JADE agent behavior.

The first task is to periodically get the information about the assessment scheduling from the Lecturers' Personal Agents. The Coordinator agent is programmed with receive-behavior that will enable it to receive notifications form lecturer's Personal Agents. The communication will be done via Agent Common Language (ACL) provided by JADE.

The second task of this agent is to trigger events at the exact date and time as scheduled in any assessment job. This task will be implemented using cyclic behavior, where agents check at regular intervals for new events. At the predetermined time, Assessment Agent will be created from the agent repository. Coordinator Agent will then issue instructions for automated marking to the Assessment Agent using ACL messages. Coordinator Agent will also communicate with the students' Personal Agents running at the students' machine informing that automated marking process will soon begin. The students' Personal Agents that are on-line will respond with a message requesting the migration of the Assessment Agent once it is ready to begin the automated marking process. Another receive-behavior will be added to the Coordinator Agent to receive acknowledgment from Student Personal Agent regarding the success or failure of Assessment Agent migration.

B. Personal Agent

We have made certain distinction between lecturer's Personal Agent and student's Personal Agent because both parties requires Personal agent to run different set of tasks. Both types of Personal agents reside in the client machine and all communication to the Coordinator Agent is done using ACL.

In the lecturer's computer, Lecturer Personal Agent is given the responsibilities to monitor and trigger any assessment setting done by lecturer via the E-learning system. The agent container will be created once lecturer open the applet based web page in the e-learning system. The applet will be programmed to record any activities regarding assessment managements into a log file that will being access by the Lecturer Personal Agent. A cyclic behavior will be added to the Lecturer Personal Agent to automatically access the log file every 3 minutes. When a new assessment is found in the log file a notification will be sent to the Coordinator Agent.

Similar to the lecturer's case, once a student logs into the e-learning system, an applet will create and run student's Personal Agent in client's agent container. The student's Personal Agent has to maintain proper communication with the Coordinator Agent at all time. This is implemented in a receiver behavior that will notify student's Personal Agent when there is a migration of the Assessment Agent. Once the Assessment Agent has arrived, the student's Personal Agent will communicate the location of the answers as well the location of the compiler in the student's machine. This is crucial since the Assessment Agent requires access to these two components in order to carry out the assessment process. Once automated marking process is completed, the student's Personal Agent will notify the student and display the assessment result.

C. Assessment Agent

Assessment Agents will be generated from agents repository supervised by the Coordinator Agent. Assessment Agents, which are loaded with the automated marking module, are mobile agents. They will migrate from the server to any client's machine requiring their services. Assessment Agent is therefore responsible for traveling to the Student's site, cooperating with student's Personal Agent in order to access student programming file, evaluating student's program and finally sending the results and assessment information to the student database. The results of assessment will be made available to the e-learning system as soon as the assessment process is completed.

Fig. 1 illustrates the overall process of the automated marking module in the Assessment Agent. The program functional test is to verify the program output of student's program. The verification process can be done by performing similarity test, which has been illustrated in figure 2. The tested program will be accepted if passes the similarity tests. Acceptance or rejection is determined by comparing the calculated similarity value with the set threshold.

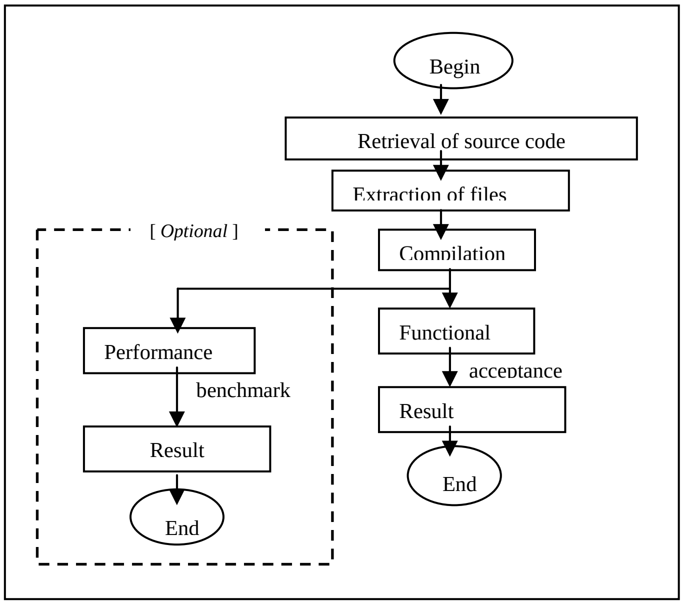


Fig. 1: System flow of automated process

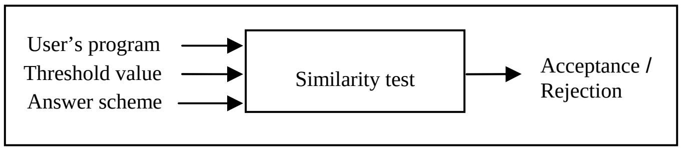


Fig. 2: Block Diagram for Similarity Test

The performance test measure is collected through the testing process shown in Fig. 3. It is, however, not always necessary to run performance tests unless the problem given to the students requires complex functions, in which case the lecturer will be interested to evaluate the efficiency of the implemented functions. It is impractical to compute the Big-O complexity [13] for each of the programs, given the limited assessment time. The performance measure is therefore the closest yardstick to gauge the elegance of the program and students' programming style. Questions requiring performance test will be flagged so that the Assessment Agent will know when to perform the elaborate assessment.

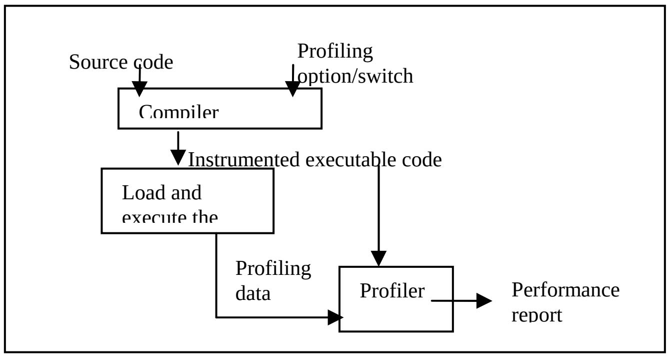


Fig. 3: Block Diagram for Performance Test

V. Student Assessment Method

The assessment process can be divided into three steps:

A. Question design and selection

The questions have to be designed by lecturers in such a way that weekly progress can be evaluated easily. Each question should be tailored for specific learning objectives/topics. To distinguish between good and average students, however there will be questions that regroup several topics at once. These questions will be stored in dedicated questions banks and will not be provided to students except during the administration of the tests. The simple reason behind this is to make sure that future students will not have already been exposed to the same questions hence rendering further experiments invalid or biased.

B. Selection of evaluation criteria

The evaluation criteria will be set into three metrics which are functionality, time to complete and program performance. Functionality is measured simply by conducting a black-box testing of the program. If the program conforms to the set requirements it is accepted otherwise it will be rejected. Time to complete is obviously the duration taken by the students to complete a functional/acceptable program. It is recorded once a student submits a program to the dedicated folder.

C. Students performance

In addition to the above evaluation criteria, there were another two derived measures have been program namely as competency and proficiency.

We define competency as a measure that link learning objectives/topics to the validity of the program that they write. A program is considered valid only if it successfully compiles runs and produces output that has a similarity value greater to the set threshold. When a program passes the validity check we record the mark in the respective student record. An example of student's competency can be illustrated in Table 1. Proficiency is defined in this context as the ability to produce the correct program within a reasonable time. In certain cases we also link proficiency with the ability to produce elegant solutions within reasonable time. Reasonable time is in turn defined by the average time-to-complete (any specific program) of the whole class or student batches. While there might be more complicated measure of elegance in programming, we limit ourselves in this work by relying on the performance report of each of the program. Table 2 shows an example of student's proficiency measurement.

Table 1: A student's competency by topics

Lab test	Topic	Student Name		
		Run (0 / 1)	Similarity (4*)	Total Marks (5)
1	Basic data types and arithmetic operators	1	4	4
2	Sequential control structure	1	2	2
3	Selection control structure	0	0	0
4	Repetition For	1	1	2
5	Repetition While	0	0	0
6	Functions	0	0	0

Table 2: A student's proficiency by topics

Lab test	Topic	Student Name			
		Run 0/1	Performance (4*)	Time to complete (5*)	Total Marks (10)
1	Basic data types and arithmetic operators	1	4	4	9
2	Sequential control structure	1	3	4	8
3	Selection control structure	0	0	0	0
4	Repetition For	1	2	2	5
5	Repetition While	0	2	0	3
6	Functions	1	1	3	5

VI. OVERALL SYSTEM ARCHITECTURE

The overall system architecture is depicted in Fig. 4. It is divided into three sections: the server sit, lecturer's site and student's site. A lecturer selects a course, add assessment/answer scheme and make announcement in the web-based e-learning system as depicted in Fig. 5. Students are supposed to check for e-learning announcements from time to time. When they see a scheduled test, which can either be done through the E-learning or in a lab environment, they are expected to present themselves at the stated venue to take the test. Coordinator Agent waits for assessment notification from lecturer's Personal Agent. Coordinator Agent will create a new Assessment Agent at the set time and notify student's Personal Agent about the migration. Once ready, student's Personal Agent acknowledges the Coordinator Agent and the process of migration and automated marking will be done at the student's computer. After completing the process, Assessment Agent then move back to Coordinator Agent and pass the results to student database. The latest results will be displayed on the e-learning system, ready to be viewed by the lecturer and student. The dotted arrow in Fig. 4 represents the Assessment Agent migration.

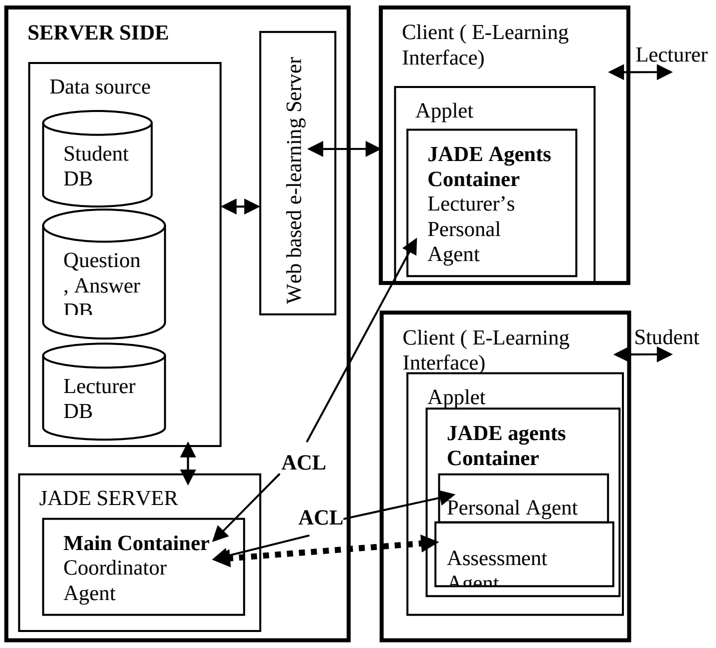
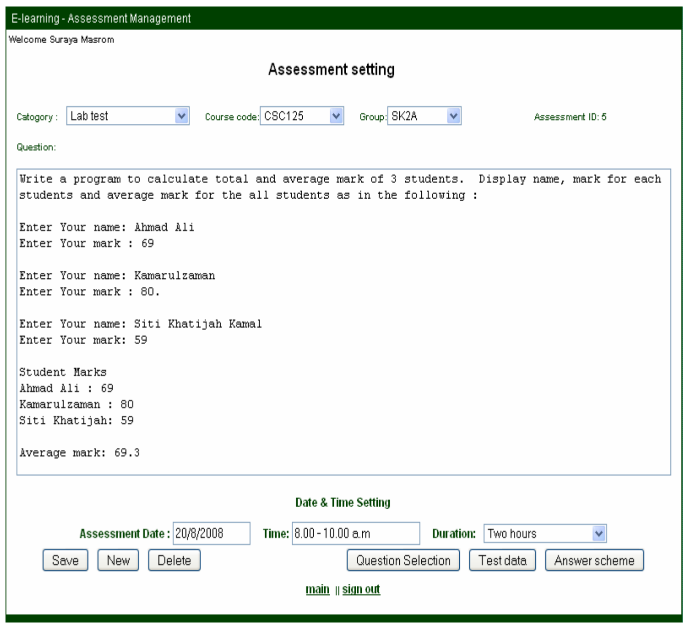


Fig. 4 : Overall system architecture


E-learning - Assessment Management
Welcome Suraya Masrom

Assessment setting

Catogory : Lab test Course code: CSC125 Group: SK2A Assessment ID: 5

Question:

```
Write a program to calculate total and average mark of 3 students.  Display name, mark for each
students and average mark for the all students as in the following :

Enter Your name: Ahmad Ali
Enter Your mark : 69

Enter Your name: Kamarulzaman
Enter Your mark : 80.

Enter Your name: Siti Khatijah Kamal
Enter Your mark: 59

Student Marks
Ahmad Ali : 69
Kamarulzaman : 80
Siti Khatijah: 59

Average mark: 69.3
```

Date & Time Setting

Assessment Date : 20/8/2008 Time: 8.00 - 10.00 a.m Duration: Two hours

Fig. 5: GUI for lecturer assessment setting

VII. Evaluation

We have developed an independent assessment module and performed some tests in a computer laboratory environment. Two evaluations test has been done, one involved a survey on the implementation of the system in order to gain students' perception and the next one was to test the performance of the automated making module in term of the processing time.

A. Students' perception

Thirty students were randomly divided into ten groups of programming teams. They were given ten questions of medium difficulty and asked to answer as many as they can within a period of two hours. They were given the report generated by the system at the end of the lab session and were asked to look at their answers and compare the results obtained with the other teams. Finally each one of them was given a questionnaire as in Table 3. Five criteria will be measured from the questionnaire namely as enjoyable, increased motivation, fairness mark, identify weaknesses and overall experience.

Table 3: Questionnaire given for student survey

Tick your selected value for each question 1 to 6.					
Question	N	1	2	3	4
1. Do you find the system enjoyable to use in a learning environment?					
2. Rate how the system increases your motivation in learning computer programming?					
3. Do you find the marks given by the system fair?					
4. Rate how the system helps you identify your weaknesses in programming.					
5. Rate your overall experience					

N="No answer", 1="Poor", 2="Below average", 3="Good" and 4="Excellent"

B. Performance test

We were interested to know how the automated marking module will perform in an actual teaching environment. We have therefore selected ten programming questions of average difficulty. The instruction given to our students was to answer correctly as many as they can, as fast as they can. Table 4 list the details of the setup that have been used in the performance test.

Table 4: Hardware and software specifications for the performance test

Hardware	**Specification**
Computer	Intel Core2 Duo 2GHz
Memory	3 Giga Bytes
Software	**Specification**
Compiler	GCC v3.2.3
Profiler	gprof v2.13.90
Similarity function	sim_text v2.6

Number of programs	190
Number of program lines	6469
Average number of lines per program	34

VIII. Results and discussion

A. Students' perception

Results from the survey can be seen in Table 5. The survey indicates that 86.6% of the student found the experience enjoyable. One student did not like the experience at all while two other students did not answer. 96.6% of respondents conclude that the system helps increase their level of motivation in learning computer programming. All except one student agreed the marks awarded by the system were fair. The result also shows that 100% of the respondents think the system could help them pinpoint their weaknesses in computer programming. All except three students find the exercise helps them increase their teamwork. Finally 86.6% of the respondents agree the overall experience was good. Please refer to APPENDIX part that illustrates student's perception on the automated marking module based on the five criteria given before.

Table 5: Survey results

	Number of responses				
	No answer	Poor	Below average	Good	Excellent
Q1	2	1	1	18	8
Q2	0	0	1	19	10
Q3	0	2	2	15	11
Q4	0	0	0	12	18
Q4	0	0	4	19	7

B. Performance test

The test results (see Table 6) show that an instructor would require typically less than four minutes for completing the assessment process, given a class of thirty students. Even if the number increases to forty students, one would require only about six minutes. This is of course a big leap compared with the old way of manual marking.

We have not considered the delay taken for the transfer of the programs to the server. The reason is, it happens before the assessment process and, therefore, it does not affect the performance of the assessment system.

Table 6: Results from performance test

Compilation time	84.00 sec
Compilation time (instrumented)	85.00 sec
Loading, Execution and report generation	25.65 sec
Loading, Execution and report generation (instrumented)	30.40 sec
Similarity checking	6.36 sec
Total processing time	116.01 sec
Total processing time (instrumented)	121.76 sec
Average processing time / program	0.61 sec
Average processing time / program (instrumented)	0.64 sec

IX. CONCLUSION AND FUTURE WORKS

We have shown from the design, the feasibility of a simple and lightweight of the agent-based architecture. We have also demonstrated, from the experiment done for automated marking module that the assessment process does not require a lot of computing power and takes only minutes to complete.

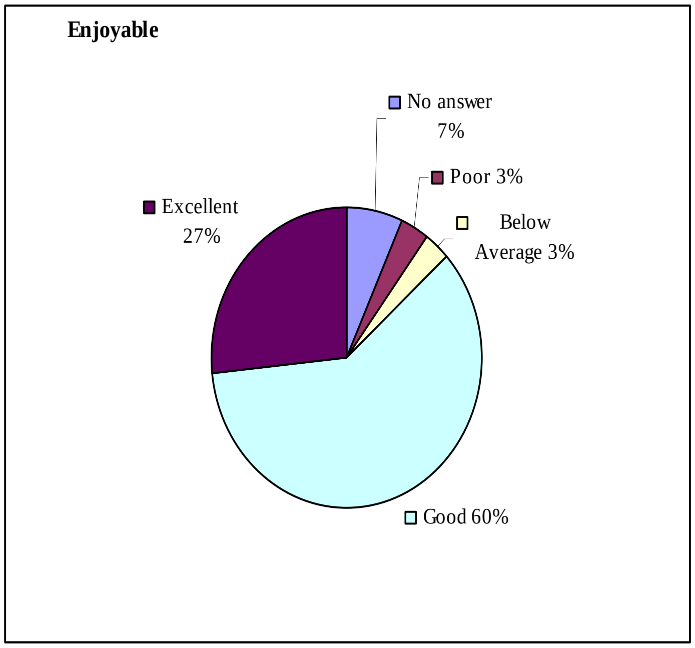
Through the experience, we must also acknowledge that the assessment module is only capable of assessing the functional aspect of the program given and the verification method employed here is a black-box test. We are unable to measure the elegance of the program, except when it is tied to its performance. We are also not able to test programs behaviour when it comes to handling illegal input and exceptions.

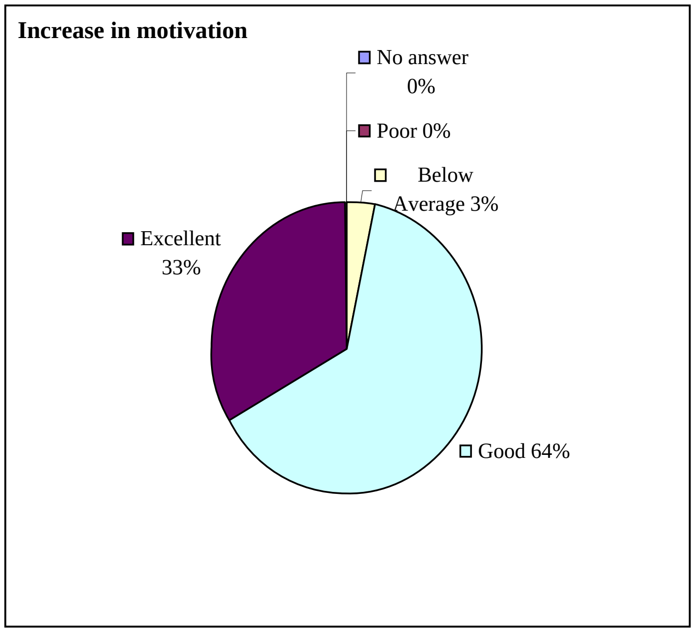
Future work can be directed towards improving the similarity checking function, to allow more flexibility when comparing outputs of the programs with the answer scheme provided by the instructors. A better similarity checking function can also be used in detecting plagiarism in the computer programs submitted to the system.

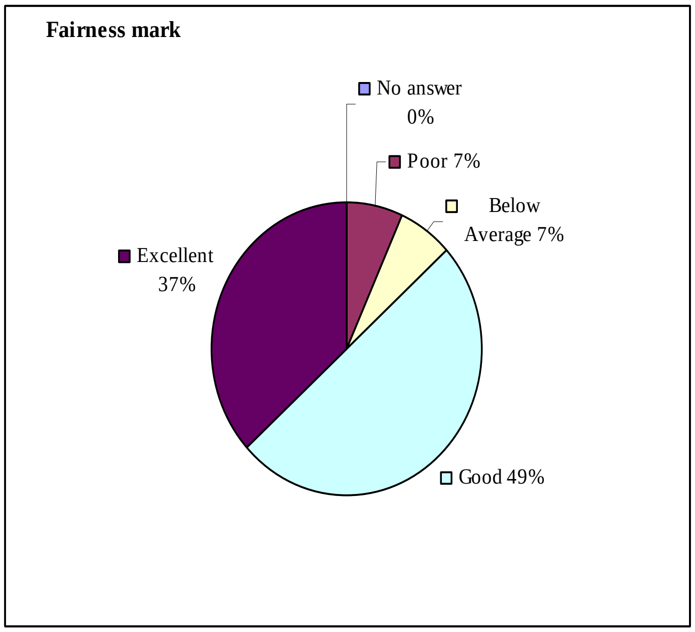
ACKNOWLEDGMENT

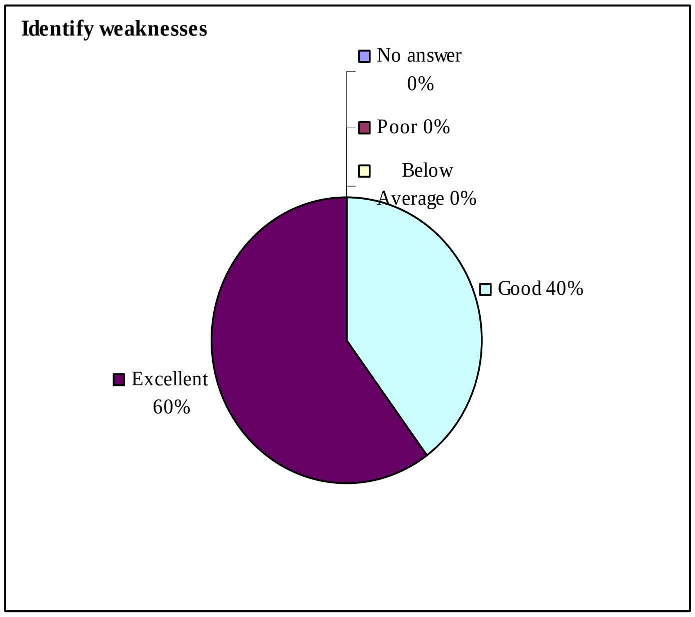
We would like to thanks the Universiti Teknologi Malaysia and Dana Kecemerlangan Akademik Panel for their trust, support and funding of this work.

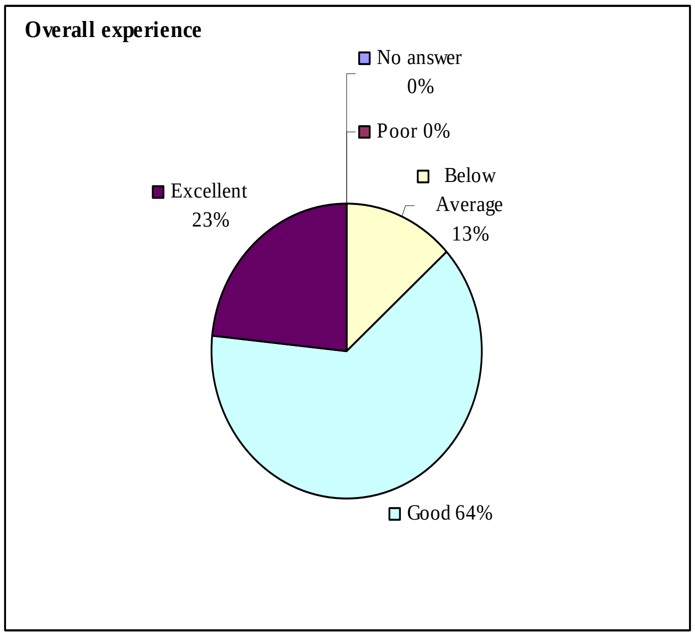
APPENDIX











REFERENCES

[1] Hussein, S., *Automatic Marking With Sakai*, In *Proceedings Of The 2008 Annual Research Conference Of The South African Institute Of Computer Scientists And Information Technologists On It Research In Developing Countries: Riding The Wave Of Technology*. 2008, ACM: Wilderness, South Africa.

[2] Hidekatsu, K., Et Al., *Using An Automatic Marking System For Programming Courses*, In *Proceedings Of The 34th Annual* ACM *Acm Siguccs Conference On User Services*. 2006, ACM: Edmonton, Alberta, Canada.

[3] Janet, C., Et Al., *How Shall We Assess This?*, In *Working Group Reports From Iticse On Innovation And Technology In Computer Science Education*. 2003, ACM: Thessaloniki, Greece.

[4] David, J. And U. Michelle, *Grading Student Programs Using Assyst*, In *Proceedings Of The Twenty-Eighth Sigcse Technical Symposium On Computer Science Education*. 1997, ACM: San Jose, California, United States.

]5] Mike, J., G. Nathan, And B. Russell, *The Boss Online Submission And Assessment System*. J. Educ. Resour. Comput., 2005. 5(3): P. 2.

[6] Michael, B., Et Al., *An Experimental Analysis Of Game: A Generic Automated Marking Environment*. Sigcse Bull., 2004. 36(3): P. 67-71.

]7] Bellifemine, F., G. Caire, And D. Greenwood, *Developing Multi-Agent System With Jade*. Wiley Series In Agent Technology, Ed. M. Wooldridge. 2007: John Wiley & Sons, Ltd.

[8] Pornphol, P. *An Agent-Based Quality Assurance Assessment System*. In *The 5th WSEAS International Conference On E-Activities*. 2006. Venice, Italy: *WSEAS*.

]9] Pattie, M., *Agents That Reduce Work And Information Overload*. Commun. ACM, 1994. 37(7): P. 30-40.

[10] Belligemine, F., G. Caire, And D. Greenwood, *Developing Multi-Agent System With Jade*. Wiley Series In Agent Technology, Ed. M. Wooldridge. 2007, Liverpool University, Uk: John Wiley & Sons, Ltd. 286.

]11] Leen-Kiat, S., J. Hong, And A. Charles, *Agent-Based Cooperative Learning: A Proof-Of-Concept Experiment*, In *Proceedings Of The 35th Sigcse Technical Symposium On Computer Science Education*. 2004, ACM: Norfolk, Virginia, Usa.

[12] Chang, Y.-H., T.-Y. Lu, And R.-J. Fang. *An Adaptive E-Learning System Based On Intelligent Agents*. In *The 6th WSEAS International Conference On Applied Computer Science*. 2007. Hangzhou, China: WSEASseas.

[13] Arefin, A.S., *The Art Of Programming Contest (2nd Editon)*. Special Online Edition Ed. 2006: Gyankosh Prokashoni. 247.

Suraya Masrom.

Suraya was born in Batu Pahat Johor, MALAYSIA on 23th of June, 1974. She received her Bachelor’s Degree in Computer Science, majoring in software engineering, from Universiti Teknologi Malaysia (UTM), Skudai, Johor, Malaysia in June 1996. She later pursued her master’s degree in Computer Science at Universiti Putra Malaysia, Serdang, Selangor, Malaysia and completed her studies in 2001. A large part of her research work falls in the domain of software engineering and computer networks.

She first started her carrier in the industry when she was employed as an Associate Network Engineer by Ramgate Systems Sdn. Bhd in June 1996. Her carrier in education only started when she took up a part-time tutoring position at UTM a year later. She finally jumped into education with both feet another year later and was sent to pursue her studies. She started her career as a full time lecturer with UTM after receiving her Master’s Degree. In three years of service at UTM she managed to complete two research projects funded by the university. She was offered a position in University Teknologi MARA (UiTM), Seri Iskandar, Perak, Malaysia, in 2004, which she gladly accepted and has been lecturing Computer Science subjects there until presently. In UiTM, she has so far managed to complete three research projects funded both the unicersity and the government of Malaysia. She has also been appointed as consultant to the university’s for software system development projects.

Abdullah Sani Abd. Rahman.

Abdullah Sani was born in Pokok Sena, Kedah, Malaysia on the 18th May 1972. He received Diplome D’etudes Universitaire de Technologie (DUT) specializing in industrial systems, from the University of La Rochell (ULR) in 1995. Another year later he received the Diplome d’Etude Universitaire Professionnalise (DEUP) from the same university specializing in computer engineering. He was awarded a Master’s Degree in Computer Science, specializing in distributed domputing from the Universiti Putra Malaysia in 2003. His current research work falls in the domain of software engineering and computer networking.

He started his carrier in the industry when he was employed as a system engineer in 1996 where he was tasked to design and implement an automated cargo system for Kuala Lumpur International Airport. His carrier in education started two years later when he was offered a lecturing position in University of Kuala Lumpur, Malaysia. He divided his time there between teaching automated systems and consulting for the industry. He was later offered a position in the Universiti Teknology PETRONAS, Seri Iskandar, Perak, Malaysia in 2004 and has been teaching computer networking subjects there ever since.