Computer Assisted Assessment for Computer Programming Course with Agent Based Architecture

S. MASROM  
Faculty of Information Technology & Quantitative Sciences  
Universiti Teknologi MARA  
Seri Iskandar, Perak  
MALAYSIA  
suray078@perak.uitm.edu.my

A. S. Abd. Rahman  
Computer and Information Science Department  
Universiti Teknologi PETRONAS  
Tronoh, Perak, MALAYSIA  
sanirahman@petronas.com.my

A.S. Shafie  
Faculty of Information Technology & Quantitative Sciences  
Universiti Teknologi MARA  
Seri Iskandar, Perak  
MALAYSIA  
anas674@perak.uitm.edu.my

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 service the entire university. The limit on staffing will most certainly result 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. This paper proposes a distributed and agent oriented architecture for a Student Assessment System. In the realm of Computer Assisted Assessment (CAA) a lot of systems in used currently are both centralized and standalone. Our design concentrates on two primary goals; to harness the power of distributed computing and to make CAA an integrated part of any existing E-Learning System. We have analyzed and extracted important functionalities required for the system, describe agent’s roles for concerning each of the functions. 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 assessment module’s performance.

Key-Words: - Agent based system, Computer programming, Computer Assisted Assessment (CAA), E-learning, Performance

1 Introduction
The increasing number of students 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) as explained in [3], might be able to offer some help in reducing lecturers’ marking chores and results management. As a consequence, the number of assessment can be increased gradually and student performance can be monitored very closely.

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.

2 Related works
2.1 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, ran on the Unix operating system and used for C programming assessment. The latest version of BOSS facilitates JAVA GUI application for the tutor 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.

2.2 Agent based system
As mention in [7], 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 [8], mobile agents and users agents can be created without limits from the agent repository so can eliminates request loses.

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

An example of multi agents system used for supporting cooperative learning in classroom can be seen in the work of [9]. The multi-agent system was named as I-MINDS, consist of a teacher agent that can monitors the student activities and helps the teacher manage class. 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. However, there is no concern in the article mention about the assessment task perform or managed by agents.

3 Methodology
3.1 Main component identification
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.

3.2 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. We require these functionalities to be integrated with any existing e-learning system.

3.3 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:

3.3.1 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 Lecturer 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, in which agent checks regularly at very small time interval. At the time determined, Assessment Agent will be created from the agent repository and Coordinator Agent will communicate with Assessment Agent via ACL message to perform automated marking task. Communication to the Student Personal Agents at the student machine from the Coordinator Agent is needed to inform that automated marking process will be executed by Assessment Agent. Then, the Student Personal Agents that are on-line will respond with a message requesting the migration of the evaluation agent once it 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.

3.3.1 Personal agent
We have divided Personal agent into Lecturer Personal agent and Student Personal agent because both components require the 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 responsibility 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 for the e-learning system. The applet also will be programmed to record any activities regarding assessment management into a log file that will be accessible 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.

The Student Personal Agent has to ensure a proper communication with the Coordinator Agent. As in the lecturer machine, once a student login to the e-learning, an applet will run Student Personal Agent in client agent container. The first task of the Student Personal Agent is to assure a proper communication between both the student and the Coordinator Agent. This is implemented in a receiver behavior that will notify Student Personal Agent when there will be a migration of the Assessment Agent. Once the Assessment Agent has arrived, the Student 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 sets of files in order to carry out the assessment process. Once automated marking process is completed, the Student Personal Agent can inform students about the results.

3.3.3 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 Personal Agent in order to access student programming file, evaluating student’s program and finally send 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.

Figure 1 illustrate 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. We determine the acceptance by comparing the similarity value with the set threshold.

![System flow of automated process](image1)

**Fig 1: System flow of automated process**

![Block Diagram for Similarity Test](image2)

**Fig 2: Block Diagram for Similarity Test**

The performance test measure is collected through the testing process shown in figure 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[10] for each of the programs given the limited assessment time. The performance measure is therefore the closest yardstick to gauge the elegance and programming style of the contestants (or students). Some questions have been set to must go through the performance test so that it will be perform by evaluation agents.

![Performance measurement process](image3)
Fig 3: Block Diagram for Performance Test

3.4 Overall system architecture

Overall architecture of the system has been drawn in figure 4. Lecturer can select course, adding assessment/answer scheme and make announcement from the web-based e-learning system as can be seen in figure 5. Students can view their results of assessment from the web-based e-learning system. Instead of accessing questions from e-learning, special for a lab test session, lecturer can project the test question on LCD projector screen in computer lab. Coordinator Agent waits for assessment notification from the Lecturer Personal Agents. Coordinator agent will create new Assessment Agent at the set time and notify Student Personal Agent about the migration. Once ready, Student Personal Agent acknowledge Coordinator Agent and the process of migration and automated marking will be done at student computer. After completing the process, Assessment Agent then move back to Coordinator Agent and pass the results to student DB. The latest results will be display on the e-learning system and ready to be viewed by the lecturer and student. The dotted arrow in the figure 4 represents the Assessment Agent migration.

Fig 4: Overall System Architecture

Fig 5: GUI for lecturer assessment setting

4. Results and discussion

We were interested to know how the automated marking module will perform in an actual teaching environment. We have therefore selected 10 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. Below are the details of the setup that have been used in the performance test.

- Computer: Intel Core2 Duo 2GHz
- Memory: 3 Giga Bytes
- Compiler: GCC v3.2.3
- Profiler: gprof v2.13.90
- Similarity function: sim_text
- Number of students: 30
- Number of programs: 190
- Number of program lines: 6494
- Average number of lines per program: 34

<table>
<thead>
<tr>
<th>Table 1: Results from performance test</th>
</tr>
</thead>
<tbody>
<tr>
<td>Compilation time</td>
</tr>
<tr>
<td>Compilation time (instrumented)</td>
</tr>
<tr>
<td>Loading, Execution and report generation</td>
</tr>
<tr>
<td>Loading, Execution and report generation (instrumented)</td>
</tr>
<tr>
<td>Similarity checking</td>
</tr>
<tr>
<td>Total processing time</td>
</tr>
<tr>
<td>Total processing time (instrumented)</td>
</tr>
<tr>
<td>Average processing time / program</td>
</tr>
<tr>
<td>Average processing time / program (instrumented)</td>
</tr>
</tbody>
</table>
The test conducted has shown as in Table 1 that any instructor would require typically less than four minutes to complete the assessment process given a class of thirty students. Even if the number increases to forty students, one would require only about six minutes to complete. 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; therefore it does not affect the performance of the assessment system.

5 Conclusion
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 thus unable to measure aspects such as 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.

References: