ISSN: 2278 – 1323
International Journal of Advanced Research in Computer Engineering & Technology
Volume 1, Issue 6, August 2012

Automatic Assessment Generation Service

Sahithi.P, Lakshmi Padmaja.D,Vishnu Murthy.G

Abstract— **Assessment is an essential element for every student in learning processes. Learning management systems (LMSs) provide support for assessment. As we are computer science students assessment is required for programming languages. Automatic assessment of programming exercises has become an important method for grading students exercises and giving feedback for them . In addition to assignments that are graded by teachers, we also support assignments that can be automatically compiled and can check for errors. This is very helpful for students so that they can check and assess themselves before submitting.**

In this paper we report about service-oriented approach for automatic assessment of programming assignments specifically taking a single programming language c#.net. There is a clear cut separation between frontends and back-ends . Assessment can be easily interfaced with different existing learning management systems (as so called frontends).We also report about different generations of automatic assessment system , their problems .

Index Terms— **automatic assessment systems, computer science education, e-assessment, service-oriented architecture.**

1. INTRODUCTION

In the last few years, the need for an automated way to assess people has increased quickly because of the growing request from either private and public structures. E-learning most often means an approach to facilitate and enhance learning by means of personal computers, CD-ROMs, audio visual aids and the Internet. This includes email, discussion forums, and collaborative software.

Automatic Assessment

E-assessment is the use of computers and computer software to evaluate skills and knowledge in a certain area. It can range from on-screen testing systems that automatically mark learners' tests, to electronic portfolios where learners work can be stored and marked.The use of computers in assessment is becoming increasingly important in modern education, providing multiple benefits to learners, teachers and those involved in the administration of assessment:

- A richer assessment experience - questions and portfolios are made clearer and more detailed through the use of text, sound and video.
- Instant feedback - e-test results can often be accessed within minutes while diagnostic information on learners' performance gives an insight into areas that can be improved upon.
- Reduced administrative burden - a simpler, more streamlined administrative process with fewer paper forms, no posting of test papers, no printing and posting candidate portfolios and no storage space needed.
- Increased flexibility - in many cases, more testing opportunities at a greater range of locations means more choice.[1]

Why It Is Important For Computer Science Students?

Instant individual feedback is a result of assessment which allows for considerable improvements in both teaching and learning. Today traditional methods of preparation and delivery of feedback proved to be unsatisfactory when the number of students is large.Programming problems and assignments are considered essential elements of software engineering and computer science education. Programming assignments can help students become familiar with the attributes of modern programming languages, become acquainted with essential tools, and to understand how the principles of software development and design can be applied. The *assessment* of these assignments places significant demands on the instructor's time and other resources. The instructor must determine the extent to which the submission has satisfied the requirements of the rubric, and possibly also verify the originality of the submission. This article details a number of attempts to automate the test-based assessment of student programming assignments for both summative and formative purposes. While the authors acknowledge that other automated approaches that may be relevant have been developed, for example peer review [Gehringer 2001] and intelligent training or tutoring systems [Sykes and Franek 2004], they are deemed outside the scope of this article, which is concerned specifically with approaches that automatically assess the success of student attempts to solve programming problems. [2]

2. RELATED WORK

The assessment, as process of measuring learning, is a problematic component of the most e-learning programs. Exercises and/or laboratory practice are essential for the learning effect, since they provide opportunities for students to solidify the knowledge acquired in lectures and to apply their theoretical knowledge to practical problems. In its traditional format, exercise groups are centered around work

on paper and a shared presentation medium, e.g., in its simplest guise, a chalkboard. However, we were dissatisfied with some aspects of this traditional way of teaching, practicing, and assessing which may be sketched as follows: Before classroom sessions . the teacher designs or chooses assignments for a weekly exercise sheet according to the state of the course, . the exercise sheet may be distributed as a printed document or made available online, e.g, as a PDF document,. students work through the exercise sheet at home. During classroom sessions . students present their solutions at the blackboard,. tutor and peers give (spontaneous) feedback,. peers take notes from the presentation,. the tutor may take notes about student's performance. As a variation, written submissions may be demanded for marking and grading by tutors. But there is always a delay between the submission and the reception of comments and/or a corrected version. For large groups of students, manual correction is labor- and time-intensive.[3]

GENERATIONS OF ASSESSMENT SYSTEMS

First Generation – Early Assessment Systems

Automated assessment systems seem to have existed for as long as educators have asked students to build their own software. The earliest example of automated testing of programming assignments may be found in Hollingsworth [1960]. Rather than using compilers and text editors, students submitted programs written in assembly language on punched cards. A grader program was run against a student program and two different results were returned, either "wrong answer" or "program complete". The advantages of this system did not only consist in a good use of tutor resources -- a key advantages was also the efficient use of computing resources, which allowed a greater number of students to learn programming. As programming systems evolved, so did assessment systems. Forsythe and Wirth, along with Naur, present a grader system that examines programs written in Algol [Naur 1964; Forsythe and Wirth 1965]. The system, for an introductory programming and numerical analysis course, was used intermittently (since 1961) at Stanford University.

The system operates by using a "grader" program to test submitted programs. The three functions of the grading programs are to supply test data, keep track of running time, and maintain a "grade book". Several new ideas were introduced in Hext and Winnings [1969]. To implement their automatic assessment system, modifications to both compilers and operating systems were necessary. Program testing was done by comparing stored test data to data obtained by executing the students' assignments. Following execution of program testing, a report was generated, including detailed test results. Interestingly, the authors say that it may be possible to use the results to check for cheating (based on the size of the program and time spent on execution); later system designers have often addressed the issue of assignment plagiarism.

Second Generation – Tool-Oriented Systems

To develop and use first-generation testing mechanisms successfully demanded a great deal of expertise. A second-generation system, however, can loosely be labeled as "toolbased". First, assessment systems are developed using pre-existing tool sets and utilities supplied with the operating system or programming environment. Second, testing engines and systems are often used and activated in the form of command-line or GUI programming tools. An example of a second-generation assessment tool can be seen in the work of Isaacson and Scott [1989]. The authors state that assessing programming assignments involves two activities: checking the program to see that it operates correctly and checking the program to see that the programming style has been applied sensibly. Like the earlier assessment systems (and all systems since), the focus of the methodology is the correct functioning of the program submitted. In the same year a different system was introduced by Reek [1989]. Rather than behaving as a tutor-oriented tool, the TRY system introduced automated testing to the student. TRY allows students to test their programs using a tester program. When the tester program is executed, the student is presented with a set of results and the test attempt is recorded. Like other systems of that period, testing is performed by a simple character-by-character comparison of results generated against expected ones. regarding the didactic implications of using an automated testing system. Instead of allowing the student an unlimited number of tries, his system limits students to a set number of submissions per assignment. Thus students are forced to think about the operations of their programs more thoroughly before submitting them for evaluation, rather than relying on the computer to act as a formal functionality testing mechanism. The Kassandra project [von Matt 1994] represents an interesting development, since it facilitates testing programs written in Matlab and Maple (mathematical languages), as well as in the more traditional Oberon, a successor to Modula-2. The correctness of the submissions is again determined by comparing output data to stored test data, which is created by the tutor. An innovation is introduced by letting assignment submissions to be made to a different process via internet socket technology, allowing a degree of isolation between the operation of the assignment code and the test management system. The ASSYST system developed by Jackson and Usher [1997] introduces a scheme that analyzes submissions across a number of criteria. The BOSS system originated at the University of Warwick in the UK. Its initial specification was similar to that of ASSYST [Joy and Luck 1998; Luck and Joy 1999]: they both ran on the Unix operating system and assessed programs written in C. The first version of BOSS was comprised of a suite of "easy-to-use" command line programs. A student could use one program to perform a test to determine whether his or her assignment was correct and another to submit the program to a secure location which could then be reviewed by a tutor. One of the most notable developments of the mid-eighties was the Ceilidh system, engendered at Nottingham University [Higgins et al. 2003], where it was used for thirteen years before being superseded by CourseMarker, which is described in the following section. While the majority of the assessment systems examine software written in C and Java, there are some that assess code written in more exotic languages. Scheme-Robo, described by Saikkonen et

al. [2001], automatically assesses programs written in Scheme, a functional language. One of the differences between this project and the others is that Automatic Test-Based Assessment of Programming: the testing service is via an e-mail interface, originally developed for the *algorithm* assessment system, TRAKLA. These second-generation systems have continued to evolve and develop. The Scheme-Robo project has been supplemented by a graphical user interface and an algorithm-animation component [Korhonen and Malmi 2000]. Some second-generation systems have evolved into third-generation web-oriented ones, which is the focus of the next section.

Third Generation – Web-Oriented Systems

While the second-generation tools can often be characterized by command-line interfaces and the manual operation of scripts, third-generation assessment systems make use of developments in web technology and adopt increasingly sophisticated testing approaches.
CourseMarker, developed at Nottingham University, builds on Ceilidh: it supports four types of users: students, tutors, teachers, and developers/ administrators and contains a number of content management components. One of the most interesting developments is the introduction of a "diagram" assessment system and its support for an impressive variety of languages including Prolog, SQL, and FORTRAN. Assignment assessment is done by an automated mechanism that analyzes the program across a number of criteria, with the emphasis on exploring program design. The design criteria are program format (typographic, lexical structure, and presence of particular features), program operation in response to test data (dynamic operation), program complexity and execution efficiency (time spent on execution). The marking actions are defined in a mark-action file that is constructed by the tutor. CourseMarker provides a wealth of course, student, and assessment administration facilities. Using administration tools the tutor can add and delete users, edit course documents, create and install new courses, and assign permissions to users. The tutor is also given wizards that allow exercises to be constructed, and for the creation of new exercises from existing ones. There is also provision for reporting, including information on cohort statistics and access to data logged as a part of the assessment submission process. Further developments include the implementation of multiple-choice questions and the conversion of all existing Ceilidh projects to the CourseMarker form. The BOSS system has also continued to develop. As well as providing a set of graphical user interfaces to students and tutors, the latest development also includes a web server component. This allows the tutor to review submissions using a traditional web-browser. BOSS system also introduces the notion of plagiarism detection. BOSS is currently used in two different forms: an open-source version that can be installed onto a system utilizing the Java platform, and an implementation that is specific to Warwick University. The latter department-specific version is integrated with the university information services.

Daly and his colleagues have developed a Java-oriented assessment system called RoboProf, deployed in an honors degree program at Dublin City University [Daly 1999; Daly and Waldron 2004]. The system presents programming problems within a web browser and the student is asked to type a program into a text box. When complete, the assignments are submitted, compiled, and results returned. If the program is valid, the student is invited to move onto the next programming activity. The system is comprised of 39 problems, divided among the areas of variables, control flow, arrays, and strings. RoboProf has a number of particularly attractive features: The notion of levels gives the student a view of progress; the automatic programming assessment element is integrated with multiple-choice assessment, which complements individual programming tasks. The multiple-choice questions are generated randomly, thereby reducing a student's potential to copy the results submitted by others. It is interesting to note that Daly has also contributed to developments in plagiarism detection [Daly and Horgan 2005]. The Automated System for the Assessment of Programming (ASAP), from Kingston University, is another development that has much in common with earlier ones. ASAP focuses primarily on the Java teaching language, which is used at Kingston University. Testing is carried out using a combination of IO stream analysis and by testing individual member functions.
The initial intention was to develop a submission system through a university-wide virtual learning environment. The submission system would then be tied to a proprietary "grade book" mechanism that allows tutors to view the students' submissions. Hence, a marking service was constructed that could be accessed by other software components through a web-service, and this approach informed other, subsequent, design decisions. As in other projects, plagiarism was considered a potential classroom issue. The ASAP project worked with a partner at another university to implement a program interface to an internationally recognised Java plagiarism-detection engine, called JPLAG. Additional components such as randomized fixed-response questions [Daly 1999] were also made available through a separate web service interface.[4]

3. PROPOSED SYSTEM

Proposed system provides approach focuses on a clear separation of all aspects regarding the management of learners, assignments, and submissions from the actual testing of programming assignments. To achieve this goal, we employ a service-oriented architecture (SOA).

Service Oriented Approach

An SOA is a framework for the integration of (business) processes as secure, standardized components so-called services[5]—that can be reused and combined to meet varying requirements . A service is a software component whose functionality is offered platform independently through an interface over the network. Service orientation requires loose coupling of services, which communicate with their corresponding consumers by passing data in a well-defined, shared format, or by coordinating an activity

between two or more services. The actual testing of programming assignments is highly dependent on the kind of test method, programming language, or other formal notation involved. For example, programs can be evaluated by using static and/or dynamic tests. For the latter, the output of a program can be compared to that of a model solution, or the assignment can be tested for properties which must be fulfilled by correct programs. Hence, all aspects regarding the exact testing should be encapsulated and implemented in self contained services—we call them back-ends. Back-ends are functional building blocks of our SOA, which provide test and assessment facilities over standard Internet protocols independent of platforms and programming languages. Teaching and learning are core business processes within educational institutions. These processes are typically supported by an LMS. Following the above-mentioned idea of separating all concerns related to managing from testing and assessment, learning management systems play the role of service consumers in our SOA approach. In the following, we will use the term frontend for the LMS employed. Common functions of a frontend are, for instance, storage of assignments and solutions, proper treatment of submission periods and resubmissions, communication of results to students, or statistics for individual students and whole cohorts. For automatic testing purposes, frontends access the functionality provided by the back ends. To enable uniform access to the back ends and a preferably loose coupling of frontends and back ends (avoiding too many point-to-point connections), we introduced a third component, the so-called spooler. Similar to a printer spooler, it manages a submission queue, as well as an variety of back ends, and provides the functions add new submissions for testing, get results from tests performed by a backend, show status information (e.g., available back ends and number of submissions in queue), add or remove back ends, get required input fields for testing with a certain backend, get available test method options. In this manner, the spooler plays at first the role of a service broker in our SOA, but it is also a service provider for different frontends, as well as a service consumer, since it uses varying back ends . Implementing the spooler and back ends in a service based way results in a high degree of interoperability and flexibility and also offers the option to combine a multitude of frontends and back ends. It also ensures the integration of any backend—even in heterogeneous system environments. The encapsulated testing functionality in the back ends can be reused and extended.

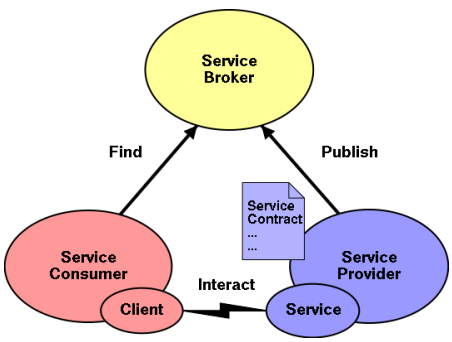

Fig-1

N-Tier Architecture

Simply stated, an n-tier application helps us distribute the overall functionality into various tiers or layers.
Each layer can be developed independently of the other provided that it adheres to the standards and communicates with the other layers as per the specifications.[7]

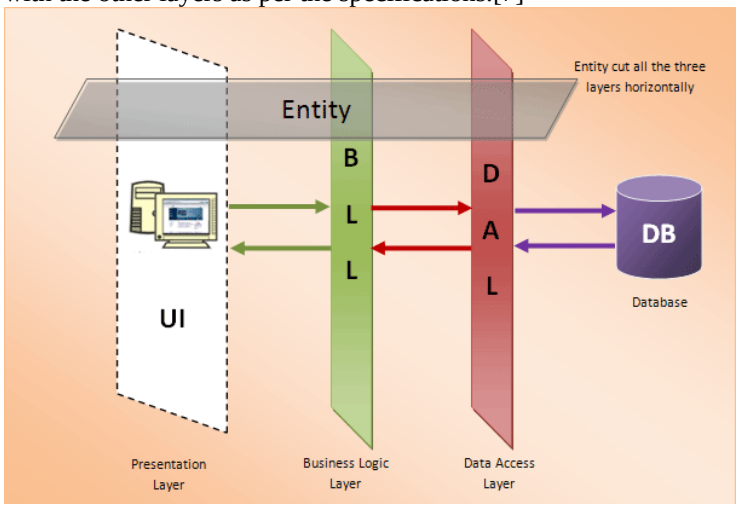

Fig-2

N-Tier Architecture

This is the one of the biggest advantages of the n-tier application. Each layer can potentially treat the other layer as a 'Block-Box'. In other words, each layer does not care how other layer processes the data as long as it sends the right data in a correct format.

1. *The Presentation Layer*:

   Also called as the client layer comprises of components that are dedicated to presenting the data to the user. For example: Windows/Web Forms and buttons, edit boxes, Text boxes, labels, grids, etc.

2. *The Business Rules Layer*:

   This layer encapsulates the Business rules or the business logic of the encapsulations. To have a separate layer for business logic is of a great advantage. This is because any changes in Business Rules can be easily handled in this layer. As long as the interface between the layers remains the same, any changes to the functionality/processing logic in this layer can be made without impacting the others. A lot of client-server apps failed to implement successfully as changing the business logic was a painful process.

3. *The Data Access Layer:*

   This layer comprises of components that help in accessing the Database. If used in the right way, this layer provides a level of abstraction for the database structures. Simply put changes made to the database, tables, etc do not affect the rest of the

application because of the Data Access layer. The different application layers send the data requests to this layer and receive the response from this layer.

4. *The Database Layer:*

This layer comprises of the Database Components such as DB Files, Tables, Views, etc. The Actual database could be created using SQL Server, Oracle, Flat files, etc. In an n-tier application, the entire application can be implemented in such a way that it is independent of the actual Database. For instance, you could change the Database Location with minimal changes to Data Access Layer. The rest of the Application should remain unaffected.

4. CONCLUSION

In this service-oriented architecture, all common aspects of managing testable assignments (e.g., submission, storage, and result reporting) are encapsulated in the frontend. Only the specifics of the testing itself (e.g., for programming tasks: which programming language with which interpreter or compiler and with which test method) are realized as self-contained services—so-called back-ends. In conjunction with the spooler, back-ends offer a flexible and portable alternative to extend a learning management system or other e-learning environments with functionality for automatic testing of programming assignments. However, back-ends do not exist for programming languages only. They have also been developed and deployed for other formal notations that are amenable for automatic testing like regular expressions, XSLT transformations, or UIMA analysis engines. There have even been experiments with automatic support for the grading of assignments in natural language.

AUTHORS PROFILE

Miss. P. Sahithi pursuing Masters in Technology in CVSR college of Engineering, Anurag Group of Institutions. I have graduated from Mahatma Gandhi Institute of Technology in the year 2010.

Mrs. D. Lakshmi Padmaja is working as an Associate Professor, Department of Information Technology in CVSR college of Engineering, Anurag Group of Institutions. She has 13 years of teaching experience. She is presently pursuing her Ph.D. in JNTU,Hyderabad and is the Head of the Information Technology Department, CVSR college of Engineering, Anurag Group of Institutions. She is the Life Member of ISTE, IEEE. She has organized and attended various workshops and conferences at National and International level.

Mr. Vishnu Murthy G received his B.E and M.Tech degrees in Computer Science and Engineering. He is having 15 years of teaching experience. He is presently pursuing his Ph.D. in JNTU, Hyderabad and is the Head of the Computer Science and Engineering Department, CVSR college of Engineering, Anurag Group of Institutions. He has organized and attended various workshops and conferences at National and International level. He has been the resource person for Institute of Electronic Governance and BITS off campus programs. He is the Life Member of ISTE, IEEE, ACM, CRSI & CSI. He had 5 publications in international journals and presented 2 papers in conferences.

REFERENCES

[1] http://www.ocr.org.uk/eassessment/

[2] http://www.cs.unibo.it/~riccucci/Paper/Paper.pdf

[3]http://www.scribd.com/doc/6947351/Past-Present-and-Future-of-EAssessmentTowards-a-Flexible-EAssessment-System

[4]http://glasslab.engr.ccny.cuny.edu/u/grossberg../Downloads/p1-douce.pdf

[5]M. Amelung, P. Forbrig, and D. Ro¨sner, “Towards Generic and Flexible Web Services for E-Assessment,” Proc. 13th Ann. Conf. Innovation and Technology in Computer Science Education (ITiCSE ’08), pp. 219-224, 2008.

[6]http://ieeexplore.ieee.org

[7]R. Saikkonen, L. Malmi, and A. Korhonen, “Fully Automatic Assessment of Programming Exercises,” Proc. Sixth Ann. Conf. Innovation and Technology in Computer Science Education (ITiCSE ’01), pp. 133-136, 2001.