TOWARDS THE DEVELOPMENT OF AN AUTOMATIC FEEDBACK SYSTEM FOR PROGRAMMING ASSIGNMENT

Abimbola, B.L
Department of Computer Science
Tai Solarin University of Education,
Ijagun, Ijebu Ode, Nigeria
bolaikotun@yahoo.com

ABSTRACT
Programming problems and assignment are considered essential elements of software engineering and computer science education. We propose a framework with which student programming assignments can receive automatic feedback on the semantics of their program codes. The proposed system increases the interest to study and understand the concept of the programming subjects. The objective is to assist teachers to promote programming as a subject and increase, increase student's performance while improving the quality of content delivered in computer programing courses.

Keywords: Programming, students, framework, feedback, understanding and assignments.

1. INTRODUCTION

Computer Programming (often shortened to programming, scripting, or coding is the process of designing, writing, debugging, and maintain the source code of computer programs. The process of writing source code often requires expertise in many different subjects, including knowledge of the application domain, specialized algorithm and formal logic. Programming is a fundamental skill that all computer science students are required to learn. Several non-Computer Science disciplines may require students to take computer programming courses . [1][2]. Examples includes information systems, Educational technology, Engineering, and Business management. These curricula typically are designed to provide the enrollee's with exposure to the application of computer programming, development of problem-solving skills, and possibly the background in a language that can be used for further study in research, analysis, or data structure design.

We live in an information age where virtually all of man's life processes and phenomena are now being customized to incline towards meeting up with the fast pace of events and trends.

Learning is one of such processes or phenomena while computer technology is the trend rapidly invading and customizing this core area (learning) needed by humanity to co-exist. Education is giving systematic instruction which leads to the development of a character or mental powers. Education of today seems to be based on five criteria: teaching style, limited domain, feedback, student interaction and help style.

Programming is related to several fields of technology, and many university students are studying the basics of it. Unfortunately, they often face difficulties already on the basic courses. The tutoring system will be responsible for correctly defining terms to the student and helping students to solve programming assignment and get automatic feedback on the semantics and syntax of the programs [7].

2. PROBLEM STATEMENT

As stated in ACM Computing Curricula 2001, programming- involved courses are regarded as the basis of most of the computer science studies. In other words, possession of good programming skill is necessary to secure the learning outcomes in this field. Both learning a programming language and giving a programming language course can be tedious tasks. A full programming language is usually a complex subject, so concentrating on some basic aspects first is necessary. Learning a programming language is that the student may get quick rewards, namely by seeing one's own program actually being executed by a machine and getting the desired effects upon its execution. [4][5]However, even writing a simple program and running it is often not so simple for beginners. In distance learning and education, additional difficulties arise. Direct interaction between students and tutors not possible. While communication via phone, e-mail, or newsgroups helps, there is still need for more direct help in problem solving situations like programming.

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 essentials tools and to understand how the principles of software development and design can be applied. Assessing programming assignments is a difficult and time-consuming task, and an educator's time may be more effectively spent giving guidance to students and explain concepts that they find difficult to grasp [3]

3. RELATED WORKS

In this section, related work to programming will be discussed from the areas of automated programming tutor, automated grading and submission of programming assignment and automated error detection.

3.1 Programming Tutors

There has been a lot of work done in the AI community for building automated tutors for helping novice. Programmers learn programming by providing feedback about semantic errors. They are so numerous but some are highlighted as follows:

LAURA: converts teacher's and student's program into a graph based representation and compares them heuristically by applying program transformations while reporting mismatches as potential bugs.

TALUS: matches a student's attempt with a collection of teacher's algorithms. It first tries to recognize the algorithm used and then tentatively replaces the top-level expressions in the student's attempt with the recognized algorithm for generating correction feedback. The problem with this approach is that the enumeration of all possible algorithms (with its variants) for covering all corrections is very large and tedious on part of the teacher.

LISP Tutor: It creates a model of the student goals and updates it dynamically as the student makes edits. The drawback of this approach is that it forces students to write code in a certain pre-defined structure and limits their freedom [10] .

MENO-II parses student programs into a deep syntax tree whose nodes are annotated with plan tags. This annotated tree is then matched with the plans obtained from teacher's solution.

PROUST on the other hand, uses a knowledge base of goals and their corresponding plans for implementing them for each programming problem. It first tries to find correspondence of these plans in the student's code and then performs matching to find discrepancies.

CHIRON is it's improved version in which the goals and plans in the knowledge base are organized in a hierarchical manner based on their generality and uses machine learning techniques for plan identification in the student code. These approaches require teacher to provide all possible plans a student can use to solve the goals of a given problem and do not perform well if the student's attempt uses a plan not present in the knowledge base

3.2 Automated Grading Of Programming Assignment

Assessment provides the teacher with a feedback channel that shows how learning goals are met. It also ensures for an outside observer that students achieve those learning goals. Research in the context of automatic programming assessment has a long history. It has been of interest to computer science educators started from 1960s and has continued to gain vast attention till present. Its core aims are mainly to promote an automated tool to reduce the workload of human teachers, to improve consistency of marking assessment items and to include thorough testing of students' programming exercises [11][12].

The survey by Douce et al. presents a nice overview of the systems developed for automated grading of programming assignments over the last forty years. Based on the age of these systems, they classify them into three generations. The first generation systems graded programs by comparing the stored data with the data obtained from program execution, and kept track of running times and grade books. The second generation systems also checked for programming styles such as modularity, complexity, and efficiency in addition to checking for correctness. The third generation tools such as RoboProf combine web technology with more sophisticated testing approaches. All of these approaches are a form of test cases based grading approach and can produce feedback in terms of failing test inputs [10].

Recently, some online assignment systems have been designed to support students and teachers in a conventional coursework activity. For instance, some systems provide assistance for teachers and students to manage the process of the conventional coursework activities, such as automatic assignment submission, assessment, and feedback [4][5][8][12]. The systems can help teachers manage the process of an assignment, and so reduce teachers' workloads. However, they do not provide support for the assigning of appropriate exercises for each student or for students 'completion of these assigned exercises in the coursework activity.

Some systems provide personal tutoring that assigns adaptive questions for students and then guides students of varied abilities to correct their own assignment errors [13][9]. These systems usually are applied in the Computerized Adaptive Test (CAT) domain to select the most appropriate questions based on Item Response Theory (IRT) [15][16]. However, in order to achieve reliable results, these systems require substantial interaction between a user and the system. There is need to provide automated systems that will generate tailored feedback about the changes required in the students submission to make it correct.

3.3 Automated error detection

A lot of research has been done in the past decades to automate detection of errors in programs, be it software or hardware . Automated debugging techniques like Delta Debugging and Quickplain aim to simplify a failing test case that still exhibits the same failure . Static and dynamic analysis are two approaches to automated debugging. Dynamic analysis require program execution on specific examples while static examine program source code rather execution traces. The dynamic analysis approach can rapidly locate bugs in procedures with minimal program analysis. However, errors in programiimg style can be difficult to detect and some, such as unreachable code, cannot be detected at all while static analysis can detect errors that are difficult or impossible to detect with dynamic analysis. But a more thorough program analysis is required, and partial or failed program analyses can result in a number of undetected bugs and erroneous bug reports(false alarms).Two different approaches to static analysis are plan-based program analysis and program verification.

Plan-based program analysis are form-based; this means that they look for surface structural forms, such as code templates, in student programs.
In program verification, student's program is compared to a task specification. A proof of correctness is constructed. Failures in the proof are interpreted as indicating errors in the student's program.
In conclusion ,once an error is detected, the hard work only begins: the error has to be located and corrected. This is usually done manually, which is time –consuming, costly, frustrating, and increases time-to-market. More and better automation in these steps is needed

4 RESEARCH AGENDA

We set to develop a system that provides support for assignments and assessment of exercises for programming courses. The system will be an automated system which will provide students with precise feedback about what they did wrong and how to correct their mistakes. The system will be test run with courses in the open university system and their benefit for programming courses in distance learning will be evaluated.

4.1 Research Method

We intend to implement a knowledge-based systems(KBS) in this research work. The four main component of KBS are knowledge base, an inference engine, a knowledge engineering tool, and a specific user interface. The KBS will includes all information about the assignment of programming course that may prove helpful to manage the knowledge based systems of the programming course.

4.2 Expected Results

The system to be design will be web-based interfaces for activities occurring in the assignment process such as provides access to the tasks to be solved by the students, the lecturer sets an assignment and define s the plan, a student solving it and a corrector correcting and grading the submitted solution. However, at the end of the research the designed and developed system will modify student assignment and correct their solution before eventually submitting them for assessment.

Works Cited

1. A.Robins,J.Rountree,and N.Rountree.Learning and Teaching programming:A review and discussion.Computer Science Education,13(2):137-172,2003.
2. A. Zeller and R. Hildebrandt. Simplifying and isolating failure inducing input. IEEE Transactions on Software Engineering, 28:183–200, 2002.
3. C.Douce, D.Livingstone,and J.Orwell. Automatic Test-based assessment of programming: A review.J.Educ.Resour.Comput.,5(3),Sept.2005
4. Collis, B., De Boer, W., & Slotman, K. (2001). Feedback for web-based assignments. *Journal of Computer Assisted Learning, 17* (3), 306-313.
5. Dawson-Howe, K. M. (1996). Automatic submission and administration of programming assignments. *ACM SIGCSE Bulletin 28* (2), 40-42
6. E. Soloway and J.Spohrer.Studying the Novice Programmer.Lawrence Erlbaum Associates,Hillsdale,New Jersey,1989.
7. Intelligent Tutoring Systems.Morgan Kaufmann,San Mateo,California,1988.
8. Lee, F. L., & Heyworth, R. M. (2000). Electronic Homework. *Journal of Educational Computing Research, 22* (2),171-186.
9. Murray, T., & Arroyo, I. (2002). Toward Measuring and Maintaining the Zone of Proximal Development in Adaptive Instructional Systems. In S. A. Cerri, G. Gouarderes & F. Paraguacu (Eds.), *Proceedings of 6thInternational Conference on Intelligent Tutoring Systems (ITS 2002)*, Heidelberg: Springer, 524-533.
10. R.G.Farell,J.R Anderson ,and B.J Reiser. An intervative computer-based tutor for lisp
11. R.S.Rist. Plans in programming: Definition ,demonstration, and development. In E.Soloway and S.Iyengar,editors ,Empirical studies of Programmers, pages 28-47 Ablex Publishing Corp., Norwood,New Jesrsey 1986
12. Saikkonen, R., Malmi, L., & Korhonen, A. (2001). Fully Automatic Assessment of Programming Exercises. *ACM SIGCSE Bulletin, 33* (3), 133-136.
13. Syang, A., & Dale, N. B. (1993). Computerized adaptive testing in computer science: assessing student programming abilities. *ACM SIGCSE Bulletin, 25* (1), 53-56.
14. The Joint Task Force: Computing Curricula 2001, Computer Science, ACM, 2001.
15. U. Junker. QUICKXPLAIN: preferred explanations and relaxations for over-constrained problems. In AAAI, 2004.
16. W.R.Murray,Automatic Program Debugging for Intelligent tutoring systems.Computational Intelligence ,3:1-16,1987