Available online at www.sciencedirect.com

ScienceDirect

Computers & Education 50 (2008) 1203–1216

COMPUTERS & EDUCATION

www.elsevier.com/locate/compedu

Performance analysis of GAME: A generic automated marking environment

Michael Blumenstein [a,*], Steve Green [a], Shoshana Fogelman [b], Ann Nguyen [a], Vallipuram Muthukkumarasamy [a]

[a] *School of Information and Communication Technology, Griffith University, PMB 50 Gold Coast Mail Centre Bundall, Qld 9726, Australia*
[b] *The Griffith School of Environment, Griffith University, PMB 50 Gold Coast Mail Centre Bundall, Qld 9726, Australia*

Received 30 November 2005; received in revised form 12 November 2006; accepted 22 November 2006

Abstract

This paper describes the Generic Automated Marking Environment (GAME) and provides a detailed analysis of its performance in assessing student programming projects and exercises. GAME has been designed to automatically assess programming assignments written in a variety of languages based on the "structure" of the source code and the correctness of the program's output. Currently, the system is able to mark programs written in Java, C++ and the C language. To use the system, instructors are required to provide a simple "marking schema" for each given assessment item, which includes pertinent information such as the location of files and the model solution. In this research, GAME has been tested on a number of student programming exercises and assignments and its performance has been compared against that of a human marker. An in-depth statistical analysis of the comparison is presented, providing encouraging results and directions for employing GAME as a tool for teaching and learning.
© 2006 Elsevier Ltd. All rights reserved.

Keywords: Architectures for educational technology system; Post-secondary education; Programming and programming languages; Teaching/learning strategies

1. Introduction

Over the past few years, the development of tools for automatic assessment of computing programs has generated considerable interest. The desire to advance these learning tools has been in part prompted by the changing roles that are being ascribed to educators, as new learning paradigms are being adopted. An educator is no longer seen as a lecturer or supervisor, instead the role assumed is that of someone that can assist students in their ability to learn by providing an adequate learning environment, and very importantly by providing sufficient feedback on students' work (Malmi, Saikkonen, & Korhonen, 2002). Although sufficient student feedback has been achievable in fairly small classes, this instruction methodology has been

* Corresponding author. Fax: +61 7 5552 8066.
E-mail address: M.Blumenstein@griffith.edu.au (M. Blumenstein).

0360-1315/$ - see front matter © 2006 Elsevier Ltd. All rights reserved.
doi:10.1016/j.compedu.2006.11.006

more difficult to apply to courses involving hundred of students, as well as courses undertaken through long distance education. Hence, automatic systems are being investigated not only to address student feedback issues, but also to improve the consistency, accuracy and efficiency of marking assessment items in large computer science courses (Jackson & Usher, 1997). This could allow educators to perform more efficiently by concentrating on tasks that cannot be automated, such as helping students grasp larger concepts and ideas.

Current automated marking systems are advancing in their ability to accurately mark students' programming assessments, but there is still much research that must be undertaken to develop an accurate marker that can be used for small and large computer programs and can be accurately used over a range of programming languages and assessment items. Specifically, a single generic marking system is less costly to maintain and is more convenient to use than a number of smaller ones.

In an effort to address the challenge of marking large volumes of electronic assessment, a number of automated systems have been developed and tested (Jackson & Usher, 1997; Reek, 1989; English & Siviter, 2000; Saikkonen, Malmi, & Korhonen, 2001; Jones, 2001; English, 2004; Ghosh, Verma, & Nguyen, 2002). Jackson and Usher (1997) proposed a system called ASSYST that checks the correctness of an ADA program by analysing its output and comparing it to a correct specification using in-built tools of the UNIX operating system. Reek (1989) details a system called TRY for grading students' PASCAL computer programs by comparing the outputs of their program against a "model" output. Their system does not take into account style or design issues. Saikkonen et al. (2001) proposed a system called "Scheme-robo" for automatic assessment of scheme programs by analysing the return values of procedures and the structure of student code. English (2004) has recently proposed a system for automated assessment of GUI-based Java programs using the JEWL library. Finally, Ghosh et al. (2002) developed a preliminary system for computer-program marking assistance and plagiarism detection (C-Marker). The system compiles and executes each student program and performs a simple comparison between the program's output and a model output file.

The main drawback of each of the systems mentioned above is that they were tested on one particular programming language. Although it was stated that some of the systems could be extended to operate on programs of other programming languages, an investigation was not conducted to determine the feasibility. Another deficiency inherent in some systems is their inability to deal with a wide variety of assessment items. In some cases the criteria for marking the correctness of student programs is hard-coded and requires the system to be updated regularly. This is a substantial limitation of these systems and needs to be addressed. Finally, many of the existing systems have been tested on small programs, whereas the challenge of marking larger pieces of software has been essentially overlooked. In order to try and overcome the problems faced with current automated marking systems, GAME (Blumenstein, Green, Nguyen, & Muthukkumarasamy, 2004a) was developed in SDK 1.4.1 and has emerged building on a previous system for marking C assignments (Ghosh et al., 2002) (C-Marker). It was designed to address the limitations of the C-Marker system and other existing systems. An overview of GAME is presented in Fig. 1; it consists of four modules: structural analysis,

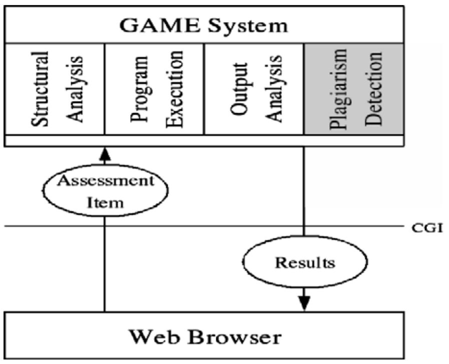

Fig. 1. Overview of GAME.