A System for Automatic Evaluation of ‘C’ Programs: Features and Interfaces

Amit Kumar Mandal, IIT Kharagpur, India
Chittaranjan Mandal, IIT Kharagpur, India
Chris Reade, Kingston University, UK

ABSTRACT

This article describes a model and implementation of a system for automatically testing, evaluating, grading and providing critical feedback for submitted programming assignments. Complete automation of the evaluation process, with proper attention towards monitoring student’s progress and performing a fine-grained analysis is addressed. The tool provides online support to both the instructors and students and is designed for service-oriented integration with a course management system using Web services.

Keywords: automatic evaluation; course management system; program testing; XML schema

INTRODUCTION

Systems for automatic and semi-automatic evaluation of programs have been investigated since the early days of computing with a wide variety of approaches (see the discussion in the next section). This research addresses two, relatively new, requirements for automatic program evaluation systems. The first requirement is to simplify the assessment set-up process for instructors, whilst providing more sophisticated evaluation capabilities. The second requirement is to structure an evaluation tool as a service or set of services in line with new service-oriented frameworks being proposed for use with e-learning systems (e.g., Wilson, Olivier, Jeyes, Powell, & Franklin, 2004). We have designed and implemented a system which is being used as a basis for proof of concept and experimentation around these requirements.

In this article we present both the internal working and external interface of our e-learning tool. We discuss our approach to a simple interface for advanced aspects of evaluation (flexible component testing and performance evaluation) in the context of the system. We also consider the structure of the system in terms of a set of services.

The motivation behind the production of a new system was the very large cohorts of students in almost all large educational institutions or universities across the world where the intake of undergraduates is around 600 or more students. As a part of their curriculum, at
the place where the system was developed, the students need to attend laboratories and courses and in their laboratory sessions each student has to submit about 9 to 12 assignments and take up to 3 laboratory-based tests. That amounts to nearly 10,000 submissions per semester. Even if the load is distributed among 20 instructors, each instructor is required to test almost 500 assignments. Without automation, the instructors would be busy most of time in testing and grading work at the expense of time that could be spent interacting with the students.

The evaluation tool assists instructors by automatically evaluating, marking, and providing critical feedback for programming assignments submitted by students. To benefit from this automatic evaluation technique, the instructors do have to spend more time in setting up the assignment to ensure that it is amenable to automatic evaluation, so it is important to address the ease with which this can be done.

The current system is restricted to evaluating only C programs, but the design has been kept as generic as possible so that it could be adapted for other programming languages in the future.

In the rest of this article the underlying technique for rigorous evaluation will be explained along with a discussion of the interfaces for easy assignment set-up. We address the service aspects of the design afterwards. The article is organized in the following sections: theoretical background and related work, system overview, an explanatory example, security issues, interfaces, services, and conclusions.

THEORETICAL BACKGROUND AND RELATED WORK

For automating assessment of students learning to program there is a dual problem identified in the literature. Firstly, there is a long history of recognized problems with student learning in this area. Mead Gray, Hamer, James, Sorva, and Clair (2006) provide a recent review of literature on studies of programming skills which they use to advocate a systematic, cognitive approach to curriculum design. They point out that “a coherent approach recognising the needs of novices and the cognitive demands of the discipline is still missing” (p.183). Secondly, there is the problem of providing effective feedback automatically. Anderson (2004) argues that the challenge of online learning is to provide a high quantity and quality of assessment while maintaining student interest and commitment, while Ridgway, McCusker, and Pead (2004) argue for a rethinking of curriculum, e-learning, and technology in the context of what they refer to as e-assessment (electronic technologies driving student learning assessment). Bransford, Brown, and Cocking (1999) in addressing assessment-centered learning, point out that feedback is fundamental to learning but feedback opportunities are often scarce in classrooms.

When a major goal is to enhance understanding and applicability of knowledge in problem solving (as is the case in learning to program), formative assessments that provide students with opportunities to revise and improve the quality of their thinking and understanding are essential (Bransford et al., 1999). For students to exercise cognitive skills online, a richer feedback than many existing automatic evaluation systems can provide is needed. Our system is designed with the intent to improve the quality of automated feedback used in both summative and formative assessment.

This is still a small step towards being able to assess students’ abilities to give coherent explanations, generate plans for problem solution, implement solution strategies, and monitor and adjust their activities as argued by Baxter, Elder, and Glaser (1996). We are not addressing some of the much more difficult issues associated with learner-centered online learning such as recognizing problems with fundamental models of computer/program behavior (see Ben-Ari, 1998 for a constructivist approach to such problems).

A variety of systems have been developed to address the problem of automatic and semiautomatic evaluation of programming assignments. Some of the early systems include TRY (Reek, 1989), ASSYST (Jackson & Usher, 1997), and Ceilidh (subsequently CourseMaster) (Higgins, Symeonidis, & Tsintsifas, 2001) which started...
Community College Leadership and Technology
*Technology and Its Impact on Educational Leadership: Innovation and Change* (pp. 187-201).
www.igi-global.com/chapter/community-college-leadership-technology/62920?camid=4v1a