A Multi-Approach Automated Grading System

Dennis Castleberry

February 19, 2014

**Abstract**

This paper discusses the design of an automated system for introductory-level computer science

coursework. An automated grader pre-processes source ﬁles, compiles them, executes them, then captures

their error and output messages into ﬁles for assessment. In our automated grading sytem, a grading

scheme ﬁle may be written to customize the grading process. Three distinct approaches are outlined for

automated grading: reﬂection, parsing, and proﬁling.

An automated grading paradigm oﬀers several advantages over traditional paper-based subjective

grading. Foremost, it is generally less time-consuming. However, it also allows for expeditious feedback

to the student, systematic corrections to systematic errors made by a faulty grading scheme, and the

ease of creating multiple, customized assignments without expense.

This paper begins with a further discussion into the advantages of the automated grading paradigm

followed by a review of literature on existing systems, then details both the design and design logic of

the current automated grading system based upon the needs of students, teachers, teaching assistants,

and industrial and academic standards. It then showcases a demonstration grading of two student

codes against their corresponding solution code and grading scheme. Finally, future work on automated

suggestion is discussed followed by a conclusion on the holistic work.

**1 Literature Review**

The concept of an automated grading system is by no means new. As early as 1973, automated grading had

been developed for COBOL programs [1]. However, like many automated systems which exist today, it only

compares the output of the student code and solution code [8, 5, 10, 7, 2, 3, 9]. Systems with this limitation

are ideally modularly language-independent to allow support for new languages to be added as required.

Aside from multi-language generalization, the auto-grading model has not seen many developments since

1973.

Perhaps the most objectionable quality of systems which rely solely on output is the all-or-nothing

character of the grading [11], in which the total assignment or sub-assignment is correct if and only if

the output exactly matches that of the solution code. With such a system, a single “oﬀ-by-one” index error

potentially yields a grade of zero. One particular needs-speciﬁc system, implemented for Java, uses reﬂection

to test the correctness of methods [6]. The Java Reﬂect API allows for methods from a student and solution

code to be invoked on user-controlled inputs and their return values, ﬁelds, etc. to be compared to determine

if the methods are algorithmically equivalent in spite of semantic diﬀerences in the student and solution main

methods. This approach is desirable for more reﬁned control of the grading.

A related goal of automated grading is to provide expeditious feedback both on laboratory exercises [7]

and programming assignments [10], for which a auto-suggestion module would ideally be integrated into the

auto-grading system and trigger in the event of an error.

A web-based assignment submission and assessment interface is a desirable component because of the

system’s ability to be used remotely, platform-independently, using intuitive controls which are familiar to

students [4, 5, 3].

**References**

[1] Norman M. Aaronson. Agsicp: An automated grading system for the instruction of cobol programming.

1973.

an automated grader. CCSC, 2007.

[3] Stephen H. Edwards and Manuel A. P´rez-Qui nones. Web-cat: Automatically grading programming

assignments. ITiCSE ’08, 2008.

[4] Xiang Fu, Bores Pelsverger, Kai Qian, and Lixin Tao. Apogee - automated project grading and instant

feedback system for web based computing. SIGCSE ’08, 2008.

[5] Olly Gotel, Christelle Scharﬀ, and Andy Wildenberg. Extending and contributing to an open source

web-based system for the assessment of programming problems. PPP3, 2007.

[6] Michael T. Helmick. Interface-based programming assignments and automatic grading of java programs.

ITiCSE, 2007.

[7] Pete Nordquist. Providing accurate and timely feedback by automatically grading student programming

labs. CCSC, 2007.

[8] Vreda Pieterse. Automated assessment of programming assignments. Conference CSERC ’13, 2013.

[9] Mark Sherman, Sarita Bassil, Derrell Lipman, Nat Tuck, and Fred Martin. Impact of auto-grading on

an introductory computing course. CCSE, 2013.

[10] Rishabh Singh, Sumit Gulwani, and Armando Solar-Lezama. Automated feedback generation for introductory programming assignments. PLDI ’13, 2013.

[11] Brad Vander Zanden and Michael W. Berry. Improving automatic code assessment. CCSC, 2013.