System for Automated Assistance in Correction of Programming Exercises (SAC) ★

Benjamin Auffarth, Maite López-Sánchez [1],
Jordi Campos i Miralles, and Anna Puig

Department of Applied Math and Analysis, University of Barcelona, C/Gran Via, 585, Barcelona, 08007 Spain

Abstract

In university programming classes often hundreds of students participate having to solve each hundreds of programming assignments a situation which puts instructors to the difficult task of validating hundreds of programming assignments. We present a framework that can help instructors and students in organization and validation of program code. Our "System for Automated Assistance in Correction of Programming Exercises" (short: SAC) is a web-platform for test-driven development and automated validation. The web-platform is based on Java Server Pages technology with tomcat as servlet container, and allows teachers to specify and define program exercises and students to upload their solutions. Students can get immediate feedback on the validity of their code and both instructors and students can see statistics about each programming assignment. We explain our platform and propose how the automatic validation can be extended.

Key words: Computer aided assessment, Source code evaluation, Computer Science, Education

1 Computer aided assessment for programming courses

In programming courses students should learn to solve problems producing appropriate, compilable, working, and efficient program code. Assessment in programming classes must conform to these objectives testing the students' ability to create programs. Typically students receive a number of problems as take–home assignments or in–class activity (charrette) during a course demanding the students to

★ This research was supported by the grant "Inovació de Docencia" of the University of Barcelona, Spain.

[1] Corresponding author. E-mail address: maite@maia.ub.es

develop skills in abstraction, generating of sub–problems, finding solutions, implementation, and evaluation and testing. In performance-based assessment, programs are tested in several ways: i) do they do work at all? (execution), ii) do they work given a set of inputs (verification) iii) given a set of inputs do they output the expected results? (validation).(1)

McCracken et al. identified deficiencies in programming skills of first year computer science students. Student performance was incommensurate with instructor expectations. Additionally students failed to recognize the main source of their difficulties and tended to attribute their failure to factors other than themselves. Our experience as teachers of first year courses at the University of Barcelona confirms this finding.

McCracken et al. imply that students should receive accurate feedback that helps them become aware of their own limitations and difficulties. This stance is supported by students. In class surveys conducted by the department of Applied Mathematics and Calculus (MAIA) many of programming students noted the importance of programming exercises for the learning process, some suggesting it would be useful to have more immediate feedback. McCracken et al. further conclude that it was unfortunately rather students' abstract knowledge than their programming skills that allowed them pass programming classes. They conjecture that performance–based assessment is often compromised on for simpler testing of conceptual knowledge. According to Ala-Mutka (2), at universities, computer assistance for programming classes – if existent – seems to be chiefly limited to submission management or objective–based assessment (such as multiple choice knowledge tests).

Manually validating student source code proves to be quite burdensome on teaching assistants and may result in untimely reporting of feedback. Students complain of inconsistencies and subjectivity. From these insufficiencies we can directly draw out requirements for a system that is currently in test phase at the University of Barcelona. The System for Automatic Assistance of Code Validation (SAC) automatically executes and validates student source code and promptly reports results back to students and professors.

The rest of this paper is organized as follows: section 2 presents related work on platforms for the automatic assessment of student source code, then in section 3, we will give an overview of the SAC platform and its components, briefly states issues of implementation, we mention extensions by way of plugins as means for evaluation of source code with the objective of performance–based assessment in mind . Section 4 lines out conclusions and future work.

2 Related work

While there is a long history in our field of automated source code validation systems at various institutions, there is relatively little sharing of tools or techniques among universities. A number of significant obstacles exist to using tools at other institutions, including the following:

- Inconsistent scoring or feedback approaches.
- Focus on architectural modularity and flexibility.
- Different programming languages used by students.

Most assessment tools for programming assignments that we found follow the three steps of analytic testing, execution, verification, and validation. The most basic of validation techniques is text–based comparison (e. g. using the unix *diff* utility or matching regular expressions).

Some assignment validation tools are CourseMarker (3), BOSS (4), DOMjudge (5). CourseMarker and BOSS are very extensive programs, which have grown beyond submission platform and validation, do however require the download and installation of software on the client side. DOMjudge has been used in programming contests. All of these rely on open source.

Web-CAT proposed by Edwards and Pugh (6), is a web application with a plugin architectures that provide a variety of services for students. It is typically used on assessing the student's performance at testing his or her own code, and on generating concrete, directed feedback to help the student learn and improve its testing codes. On basic programming courses, software testing knowledge are not required.

Our proposal, named SAC, is inspired from Web-CAT as a web-based environment for submission programs to a set of a unit tests. These unit tests are defined by teachers and they validate the student submissions.

We concentrated with SAC on the core–functionalities, and emphasized open access and modularity. As for open access, first, students and instructors should be able to access respective resources, and second, the source code should be open and amenable. As for modularity, external programs should be easily plugged in. We will now describe the SAC platform.

3 System for Automated Assistance in Correction or Programming Exercises (SAC)

The System for the Automated Assistance in Correction of Programming Exercises (short “SAC“) is a web-based environment for submission of unit tests and unit

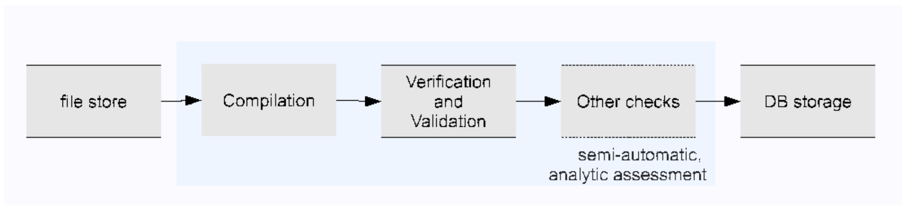

file store
Compilation
Verification and Validation
Other checks
DB storage
semi-automatic, analytic assessment

Fig. 1. SAC's workbench for computer aided performance–based semiautomatic analytic assessment

cases and for remote validation of programs. Our objective was to facilitate the correction of student exercises in university level programming classes, where often hundreds of students have to write hundreds of small programs each.

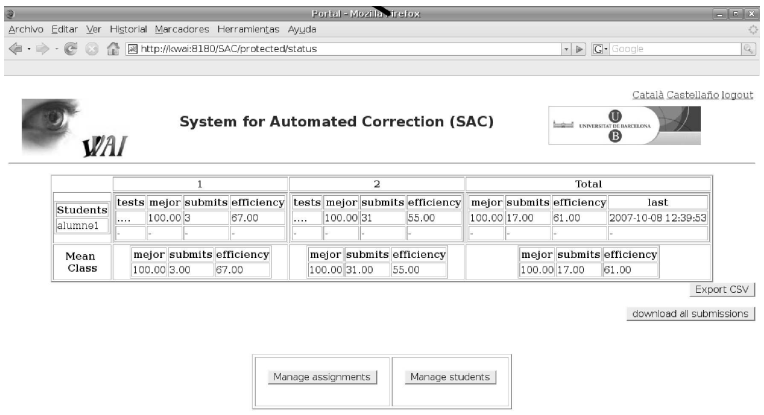
We separated submission platform and validation stages, sourcing out the compilation and testing stages to simple shell scripts, therefore extensions and adaption to different programming languages (other than Java) should be very easy. The system is very lightweight, the only requirements being Apache tomcat, the compiler of the programming language used in assignments, and very few more.

The platform has been translated into three languages (English, Spanish, and Catalan)[2] and tested with different browsers (Internet Explorer, Opera, Mozilla Firefox) under GNU/Linux and Microsoft Windows. Implementation is based on Java Server Pages (JSP (8), JDK v. 1.5) with Apache Tomcat 5.5(9) as servlet container. PostgreSQL (v. 8.1) serves as database backend. Pages are protected based on JDBC Security Realm (version 3) authentification, with separate roles as professor and student.

SAC is a web-based software tool for electronic submission of source code and validation of code and could replace the slow and inflexible bottom-neck of traditional assignment correction providing semiautomatic and analytic means of first–pass correction (cf. fig. 1) and facilitating instructor–student feedback. Using SAC, assignments and student solutions can be uploaded in HTML forms. Upon submission of student solutions, these are registered with time stamp and validation statistics are immediately delivered back to students allowing them to get a better understanding of the correctness of their work. Instructors can access performance statistics of individual students and entire classes, they can download solutions, and see them through for plagiarism check and fine-grading by eye-inspection. By these means, SAC reduces the overall workload of instructors, especially the necessary organizational effort.

SAC can help students and instructors. Students have direct feedback whether their solutions to programming exercises was correct and where they failed, which can help them in the development of their code. They see output of the Java compiler

[2] Using the ResourceBundle Class (7)



	1				2				Total			
Students	tests	mejor	submits	efficiency	tests	mejor	submits	efficiency	mejor	submits	efficiency	last
alumne1	....	100.00	3	67.00	....	100.00	31	55.00	100.00	17.00	61.00	2007-10-08 12:39:53
	-	-	-	-	-	-	-	-	-	-	-	-
Mean Class		mejor	submits	efficiency		mejor	submits	efficiency	mejor	submits	efficiency	
		100.00	3.00	67.00		100.00	31.00	55.00	100.00	17.00	61.00	

Fig. 2. Professor Status Screen in Firefox

and of the execution of their test. They also have statistics available including currently for each student and assignments correctness in all test runs, percentage of correct tests on the best submission, number of submissions, and ratio of correct tests over the total number of tests in all submissions.

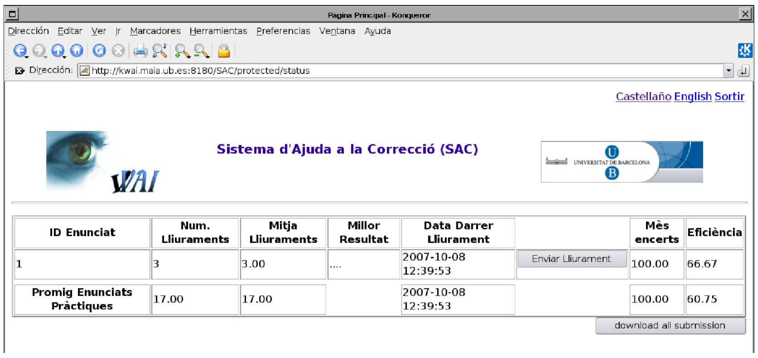
As for teaching staff, this information is available for the whole group of students. Instructors can export this information as comma separated values (10) in order to compute different statistics using spreadsheet applications such as Microsoft Excel, OpenOfficeȯrg Calc, or gnumeric (cf. fig. 2).

Students see for each of these assignments an id of the exercise, the number of submissions the student has already made to an assignment, the mean number of submissions within the group, the result string and percentage of correct tests of the best submission, the percentage of correct tests over all submissions to an assignment, and the date of the last submission. They also can find summary information. Students are prevented from late submissions and the number of submissions to an exercise can be restricted (cf. fig. 3).

On submission of an exercise, SAC will save the files to an archive, compile it with the test cases, and execute the test in a unix *chroot* sandbox for validation. Results are immediately displayed.

In our approach of automatic validation, minimum requirements would be that the code compiles and executes without failures for the whole set of allowed inputs. JUnit test cases should be designed with rubrics in mind, so standardized testing and relatively direct inference of grades is possible. Correctness of outputs on given inputs is a major requirement. An important and readily available point is the execution time, which can be limited or – alternatively – could be factored in as efficiency measure. The JUnit framework allows testing of individual functions, so different implementations of I/O access should not distort these measurements.

JUnit is a framework for validating individual units of source code in the Java


Pàgina Principal - Konqueror
Dirección Editar Ver Ir Marcadores Herramientas Preferencias Ventana Ayuda
Dirección: http://kwai.maia.ub.es:8180/SAC/protected/status
Castellaño English Sortir
WAI
UNIVERSITAT DE BARCELONA
Enviar Lliurament
download all submission

Sistema d'Ajuda a la Correcció (SAC)

ID Enunciat	Num. Lliuraments	Mitja Lliuraments	Millor Resultat	Data Darrer Lliurament		Mès encerts	Eficiència
1	3	3.00	....	2007-10-08 12:39:53	Enviar Lliurament	100.00	66.67
Promig Enunciats Pràctiques	17.00	17.00		2007-10-08 12:39:53		100.00	60.75

Fig. 3. Student Status Screen in Konqueror

programming language. Test cases specify the requirements of software units (in Java: methods). In testing units in isolation the requirements of each unit can be independently verified. This promotes functionality of all methods and clearness of source code interfaces (API). Examples and more detailed explanations on design of tests are available on our documentation website `http://kwai.maia.ub.es:8180/SAC/manual.html`.

After automatic validation teaching assistants check the code by visual inspection and mark the assignments.

3.1 *Extensions to testing functionality*

At the moment testing is restricted to functional correctness by means of JUnit tests, however the difficulty is in designing measurements that are relevant for program quality and learning programming. Assessment based solely on correct syntax, correctness of solution, and satisfaction of specification may fall short of uncovering finer qualities in the code. Additionally programs should be checked for other criteria here subsumingly referred to by “style“. This includes more concrete criteria such as conformism to coding standards, naming conventions, indentation and more subtle criteria such as design choices, and documentation. As for further assessors, there exist many projects for testing of source code on the internet the code of which is freely available under open licenses. We propose that by way of plugins, as lined out later in this section, assignments could be tested for syntax, programming style, plagiarism, among other things.

We will now talk about extraction of characteristics from the code for the purpose of quantitative source code evaluation. We will focus exclusively on the assessment of the students’ ability to write good code. There are many metrics available to measure the quality of source code, regarding complexity, redundant code, code duplication, dependencies, cycles, test coverage, and performance. While it is impractical – due to space constraints – to cover all the vast number of freely available tools, we will point to some of them.

A dedicated site to tools for improvement of code quality in Java is Java Power Tools (11). In fig. 4 you see a poll conducted on the Java Power Tools Website (11) with 184 casted votes rating the usefulness of tools between 0 and 5 for improving source code quality in the Java programming language.

For automatic validation and analysis of source code there are many resources available on–line, e. g. (12; 13). For evaluation of performance there are (14; 15; 16). For tests of graphical user interfaces, there are dogtail (17),XRadar (18), and JEWL (19) and many more. As for plagiarism detection, literature offers many approaches, e. g. (20). Checkstyle (21) check whether code conforms to coding standards (eġthe Sun Java Coding Conventions (22)).

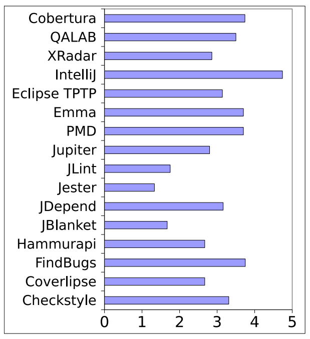


Fig. 4. Survey: Which tools do you use to improve your code quality?

4 Conclusions

We explained the System for Automated Assistance in Correction of Programming Exercises developed by the authors of this article located in the Department of Applied Mathematics and Calculus in the University of Barcelona. The ultimate goal of the project is to be able to easily create, deliver, and rapidly evaluate student programming assignments in a consistent manner. We described how this can be currently done with our system and discussed how extensions might add to students' learning experience.

Incorporation of new tools reduces complexity and can make cognitively difficult tasks routinely possible. Use of SAC in teaching practice should not be inspired by technology but by educationally sound concepts. As we hope to have shown, designs of assignments and assessment settings could profit from the versatile ready-to use off-the-shelf testing software. We consider it important to explain openly the kind of validation and evaluation that is done, so students may learn testing their code themselves, which – experience shows – they rarely do before submission (cf. (23)). In some courses we even require students to submit test data along with their programming solutions and we assess the quality of the test data. Importantly, in our system, automatic testing is always followed by human assessment, which includes individual comments and advice to students. Current plans are to integrate SAC into the e-learning platform moodle (24) thereby enhancing its user functionality and interoperability.

References

[1] M. McCracken, V. Almstrum, D. Diaz, M. Guzdial, D. Hagan, Y. B.-D. Kolikant, C. Laxer, L. Thomas, I. Utting, and T. Wilusz, "A multi-national, multi-institutional study of assessment of programming skills of first-year cs students," *SIGCSE Bull.*, vol. 33, no. 4, pp. 125–180, 2001.

[2] K. Ala-Mutka, "A Survey of Automated Assessment Approaches for Programming Assignments," *Computer Science Education*, vol. 15, pp. 83–102, June 2005.

[3] C. Higgins, T. Hegazy, P. Symeonidis, and A. Tsintsifas, "The coursemarker cba system: Improvements over ceilidh," *Education and Information Technologies*, vol. 8, no. 3, pp. 287–304, 2003.

[4] M. Luck and M. Joy, "A secure on-line submission system," *Software Practice and Experience*, vol. 29, no. 8, pp. 721–740, 1999.

[5] J. Eldering, T. Kinkhorst, and P. van de Werken, "Domjudge website." `http://domjudge.sourceforge.net/`, 2007. [Online; accessed 04-December-2007].

[6] S. H. Edwards and W. Pugh, "Toward a common automated grading platform," in *SIGCSE '06: Proceedings of the 37th SIGCSE technical symposium on Computer science education*, (New York, NY, USA), ACM, 2006.

[7] Sun, "java.util class resourcebundle." `http://java.sun.com/j2se/1.4.2/docs/api/java/util/ResourceBundle.html`, 2007. [Online; accessed 04-December-2007].

[8] Sun, "J2ee javaserver pages technology." `http://java.sun.com/products/jsp/`, 2007. [Online; accessed 04-December-2007].

[9] A. S. Foundation, "Apache tomcat." `http://tomcat.apache.org/`, 2007. [Online; accessed 04-December-2007].

[10] N. W. Group, "Rfc 4180 common format and mime type for comma-separated values (csv) files." `http://tools.ietf.org/html/rfc4180`, 2007. [Online; accessed 04-December-2007].

[11] J. F. Smart *et al.*, "Java power tools – poll on tools for improving code quality." `http://javapowertools.wikidot.com/code-quality`, 2007. [Online; accessed 04-December-2007].

[12] "Java-source.net – open source code analyzers in java." `http://java-source.net/open-source/code-analyzers`, 2007. [Online; accessed 04-December-2007].

[13] R. Jocham, "Java code standard checker (jcsc)." `http://jcsc.sourceforge.net/`, 2007. [Online; accessed 04-December-2007].

[14] T. Gyimothy, R. Ferenc, and I. Siket, "Empirical validation of object-oriented metrics on open source software for fault prediction," *Software Engineering, IEEE Transactions on*, vol. 31, no. 10, pp. 897–910, 2005.

[15] B. Cheang, A. Kurnia, A. Lim, and W.-C. Oon, "On automated grading of programming assignments in an academic institution," *Comput. Educ.*, vol. 41, pp. 121–131, September 2003.

[16] H. Ping Yu; Systa, T.; Muller, "Predicting fault-proneness using oo metrics.

an industrial case study," *Software Maintenance and Reengineering, 2002. Proceedings. Sixth European Conference on*, pp. 99–107, 2002.

[17] E. Rousseau, Z. Cerza, C. Lee, and D. Malcolm, "Dogtail – taking your applications for a walk." `http://people.redhat.com/zcerza/dogtail/`, 2007. [Online; accessed 04-December-2007].

[18] K. Kvam, K. JD, R. Pelisse, F. L. Droff, and A. Fleischer, "Xradar on sourceforge.net." `http://xradar.sourceforge.net/`, 2007. [Online; accessed 04-December-2007].

[19] J. English, "Automated assessment of gui programs using jewl," *SIGCSE Bull.*, vol. 36, no. 3, pp. 137–141, 2004.

[20] S. Engels, V. Lakshmanan, and M. Craig, "Plagiarism detection using feature-based neural networks," in *SIGCSE*, pp. 34–38, 2007.

[21] O. Burn, "Checkstyle." `http://checkstyle.sourceforge.net/`, 2007. [Online; accessed 16-January-2008].

[22] Sun, "Code conventions for the java programming language." `http://java.sun.com/docs/codeconv/`, 1999. [Online; accessed 16-January-2008].

[23] T. Howles, "Fostering the growth of a software quality culture," *SIGCSE Bull.*, vol. 35, no. 2, pp. 45–47, 2003.

[24] M. Dougiamas and P. Tayler, "Moodle: Using learning communities to create an open source course management system," in *World Conference on Educational Multimedia, Hypermedia and Telecommunications 2003* (D. Lassner and C. McNaught, eds.), pp. 171–178, Chesapeake, VA: AACE., 2003.