

# APOGEE – Automated Project Grading and Instant Feedback System for Web Based Computing

Xiang Fu, Boris Peltsverger  
Georgia Southwestern State U.  
800 GSW Drive  
Americus, GA 31709  
(229) 931-2819  
{xfu,plz}@canes.gsw.edu

Kai Qian  
Southern Polytechnic State U.  
1100 South Marietta Parkway  
Marietta, GA 30060  
(678) 915-3717  
kqian@spsu.edu

Lixin Tao  
Pace University  
G320, CSIS, 861 Bedford Road,  
Pleasantville, NY 10570  
(914) 773 – 3449  
ltao@pace.edu

Jigang Liu  
MetroPolitan State University  
700 7th St E  
St. Paul, MN 55106  
(651) 793-1472  
jigang.liu@metrostate.edu

## ABSTRACT

Providing consistent, instant, and detailed feedback to students has been a big challenge in teaching Web based computing, given the complexity of project assignments and the comprehensive requirements on security, reliability, and robustness. We present a prototype automated grading system called ProtoAPOGEE for enriching students' learning experience and elevating faculty productivity. Unlike other automatic graders used in introductory programming classes, ProtoAPOGEE covers a large spectrum of system quality attributes. It is able to generate step by step play-back guidance for failed test cases, hence providing informative feedback to help students make reflective and iterative improvements in learning.

## Categories and Subject Descriptors

D.2.5 [Software Engineering]: Testing and Debugging – *testing tools*. K.3.1 [Computers and Education]: Computer Uses in Education - *Computer-Assisted Instruction (CAI)*.

## General Terms

Design, reliability, security, verification.

## Keywords

Automated grading, web application, test case generation.

## 1. INTRODUCTION

With the impact of Cyberspace reaching every corner of the globe, World Wide Web based computing has become an integral part of CS/IT curricula. For example, as recommended by ACM CC2001 [1], Web programming is one of the courses in the net-centric knowledge area for Computer Science curriculum. The IS (Information System) 2002 curriculum [9] has four technology areas, one of which is Internet systems, architecture, and development. Many universities have included one or more courses covering both the client and server technologies, e.g., J2EE and ASP.Net, for building Web based solutions to meet

needs of the business world. Such skills have proven essential for CS/IT graduates to succeed in the job market.

One major concern of teaching Web based systems is: *how do instructors provide consistent evaluation and detailed feedback to students*, given an overwhelming number of student project submissions, each of which may consist of over 10 dynamic Web forms, 100 user controls, and possibly over 2000 lines of source code? Clearly, automated grading and feedback system is one good candidate solution.

Automatic grading in teaching Web based computing is different from that of introductory programming – a faculty member has to examine not only the functional requirements (e.g., an online shopping-cart computes the discounts correctly), but also many other quality attributes such as reliability, security, and robustness (e.g., whether a Web mail system can crash if a simple string like 'drop table users is entered in its log-in textbox). To completely (but manually) inspect Web projects could take faculty tremendous amount of time. On the other hand, students expect to gain real-world project development experiences from project assignments, and hence to simply assign a grade without providing detailed feedback is clearly not acceptable.

To overcome the aforementioned challenge, we built a prototype of an automatic grading and feedback system called ProtoAPOGEE (Prototype of Automated PrOject Grading and instant fEEdback system for web computing). This paper introduces ProtoAPOGEE and outlines the development plan of its full-fledged version called APOGEE.

### 1.1 Related Work

Automated assessment of student work has been explored for decades. For example, automated grading of quizzes consisting of true/false, multiple choice, and short answer questions have been a standard feature of most instructional support system such as WebWork, WebCT, BlackBoard, and Sakai. While automatic grading of multiple choice questions is not hard technically, to evaluate programming assignment is not an easy job. Early efforts to grade programs can be dated back to 1960's where Forsythe

and Worth designed a grading system [6] that accepts programs in punched cards and grade them using a pre-set validation data. During the last decade, many automatic grading systems have been developed, e.g., Kassandra [14] on scientific computing, Ceilidh [7] on grading ML programs, Ganesh learning environment [13] which provides monitoring for discovering student problems early, and TRAKLA2 [12] on teaching data structures and algorithms. More recent efforts include LabRat [19] which grades Java program snippet, and Web-CAT [3,4] which promotes test-driven-development and allows evaluating completeness of students' test scripts.

One limitation of many of the aforementioned assessment tools is that they can only work with text mode programs. GUI program evaluation is not available until the recent progress in automated testing. Currently, there are initial efforts to utilize the open-source GUI testing tools in building automatic grading tools for Java programs used in introductory programming courses. For example, Feng and McAllister [5] developed a tool for automatically grading Java programs based on Jemmy library [15]. A recent granted NSF proposal [2] by Stephen Edwards is based on an open-source Java GUI testing package called "Abbot" [18].

ProtoAPOGEE is based on the open-source Web application testing tool Ruby [16], Watir [11], and WatirN [10]. It drives Web browser through Ruby, simulates user actions on the program to be tested, evaluates the submitted student project against standards set up by instructors, and generates grading report automatically. Notice that the uniqueness of ProtoAPOGEE is its niche fits into in the area of Web application development. Although there are similar technology in the Java unit testing, e.g., Feng, McAllister [5] and S. Edwards [2,3,4]. However these techniques can not be directly ported to the application area of Web programming education. From the pedagogical point of view, APOGEE concentrates on the Web computing classes at the upper division level, where advanced issues such as atomicity of database access, thread safety, reliability, robustness, and security are the major concern.

The paper is organized as follows. Section 2 introduces the general architecture of ProtoAPOGEE. Section 3 presents the project specification approach used by ProtoAPOGEE. Section 4 discusses project grading and informative feedback to students. Section 5 outlines the development plan of APOGEE, the full-fledged version of ProtoAPOGEE. Section 6 concludes.

## 2. General Structure of ProtoAPOGEE

ProtoAPOGEE is a proof-of-concept implementation of APOGEE. The tool and the demonstration video are available at <http://vlab.gsw.edu/Projects/APOGEE>.

The general architecture of ProtoAPOGEE resembles many online course management systems such as WebCT, BlackBoard, and WebWork [8]. It has a log-in interface and supports at least two types of users: instructors and students. Instructors can manage projects, e.g., creating project specification, specifying evaluation measures, and viewing grading reports. Students can view project specification, submit projects, and get feedback on-the-fly. Students can submit their projects as many times as they like until the project deadline. Once a project is submitted, the grading report and failure demos are available immediately.

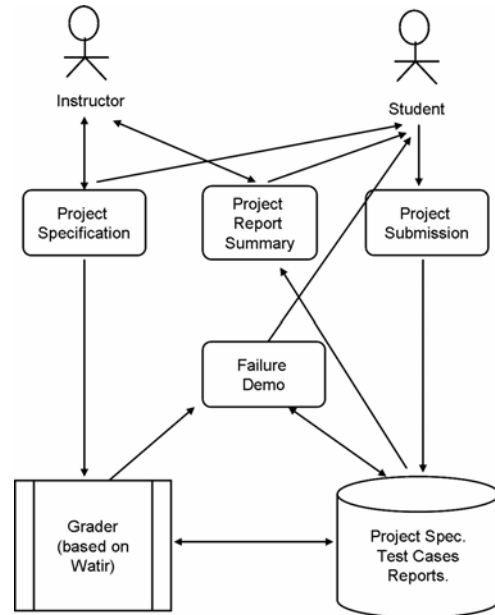


Figure 1. Architecture of ProtoAPOGEE

Figure 1 shows the general structure of ProtoAPOGEE. The prototype tool consists of three main modules:

- A project specification tool, which allows faculty member to specify the background information, requirements, grading policy, and the test cases for evaluating requirements. The dynamic web page is also visible to students, except that the test case specification scripts are hidden.
- An automatic grading module, which based on Watir [11], drives an Internet explorer browser at the background and evaluates a submitted student project using stored test scripts. At this moment, the tool relies on Microsoft Internet Information Service (IIS) server and .Net run-time. However, platform independence can be achieved by using other GUI testing tools for Web applications, such as LeUnit [21].
- A grading report viewer, which displays the itemized grading summary on all the requirements preset by instructor. The report provides step-by-step animation playback of the evaluation of each requirement, and informative textual feedback.

In the following, we describe these three major modules one by one.

## 3. Project Specification

Project specification actually consists of two components: (1) functional requirements and quality attributes, and (2) GUI naming convention. We discuss component (1) first.

Generally, all requirements are classified into several categories, e.g., functional requirements, robustness, security, etc. Each category has a number of requirements. For example, the "robustness" category may have requirements on input validation,

data persistence against system crash, etc. Each requirement can be either inspected by ProtoAPOGEE automatically, or by instructor manually, or using a hybrid approach. Each requirement is evaluated by one or more test cases, with each of which, a number of points are associated. Then performance of a student project on each requirement is the sum of the points from all passed test cases. The functional and non-functional requirements are all specified using Ruby Watir[11] scripts. At this moment, instructors need to know the Ruby [16] programming language to write a script. We expect to improve the convenience of the tool by introducing visual programming authoring tools later.

### 3.1 Case Study: Free-Lance Bulletin

Consider a simple Web application project as shown in Figure 2. Students are required to complete one simple Web based bulletin system called Freelance Bulletin. The project consists of two pages: one page for listing all existing posts in the system and the other page allows any visitor to create a new post or update an existing post. Freelance Bulletin is not required to have any user access control. As shown at the bottom of Figure 2, the tree view on the left hand side displays the requirements of the project. The project has five categories of requirements, e.g., functional requirements, security, etc. Each category has one or more requirements, each of which is supported by one or more test cases. For example, in the “Security” category, there is one requirement on the defense against SQL injection vulnerability. The requirement is examined by two test cases. On the right bottom of Figure 2, the detailed setting of Test Case 1 is presented.

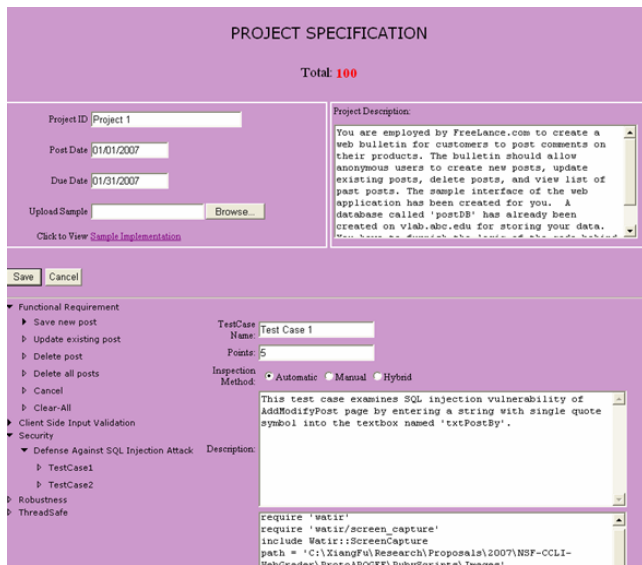


Figure 2. Project Specification

### 3.2 Challenge of Specifying GUI

Each test case is specified by a Ruby script. In the script, an instructor can orchestrate the sequence of testing actions, e.g., to type a string in a textbox, move mouse over a certain <div> region for a moment, to click a radio button, etc. Watir provides very powerful system functions for instructors to extract

information, e.g., using regular expression matching, from the HTML file in browser. Hence, the output HTML generated by student project can be examined by comparing it with expected result. In addition, Watir provides the capability for taking screenshots of browser contents. This allows ProtoAPOGEE to generate step-by-step play-back information.

Notice that, however, the effectiveness of a Ruby testing script severely relies on the assumption that all HTML controls in a student project use names as exactly expected by the script. Currently, we provide the GUI project “shell” (without detailed logic implementation) to students so that the generated HTML will meet our naming convention. Later, we expect to derive project GUI specification automatically from sample “shell” projects.

Another challenge concerning the naming convention is caused by the server control technology. As we do not restrict the computing platform based on which students build their projects. Students are free to choose platforms such as J2EE and ASP.Net. Many platforms provide server controls, whose corresponding HTML code will be generated at server side. Even the same server control name might be mapped to different names under different computing environment. This requires us to improve the grading module so that it does not use strict string matching for locating HTML controls during testing.

## 4. Instant Feedback

One difficulty of grading student projects, unlike grading true/false and multiple choice questions, is to provide detailed feedback to students. For example, consider the scenario of describing the process of crashing a Web application, which is only possible given specific boundary value input. A faculty member has to write a long textual description (e.g., which pages should a tester go through, which buttons should be clicked, and which values should be entered into which textbox, etc.). Such comments are neither straight-forward nor clear. ProtoAPOGEE can automatically generate an animation demo for each failed test case, showing step by step how a student submission fails a requirement. Further textual hint (prepared by faculty or automatically generated by ProtoAPOGEE) will be shown to students. Similar to WebWork, ProtoAPOGEE allows students to pre-submit their projects multiple times before project submission deadline, and hence students will learn from failures and iteratively improve their design.

ProtoAPOGEE provides feedback information at two levels: (1) a summary report, and (2) detailed feedback for each requirement and the corresponding test cases. A summary report displays the summary information of the project, e.g., grade received by the student. Then the analysis of each category of requirements is presented in details. It is supported by two major sources of information: the detailed analysis of each failed test case, and the corresponding hints and guidance information. Clicking on any of the links in the cell of “Failed Test Cases” brings students to another page showing the details of the failed test case. Clicking any of the Comment links displays the corresponding hint page.

### 4.1 Case Study: SIA Vulnerability

Figure 3 shows one example of the demo page for failed tested cases. This demo page is generated for one of the test cases

associated with the “Defense against SQL injection attack” requirement in Figure 2.

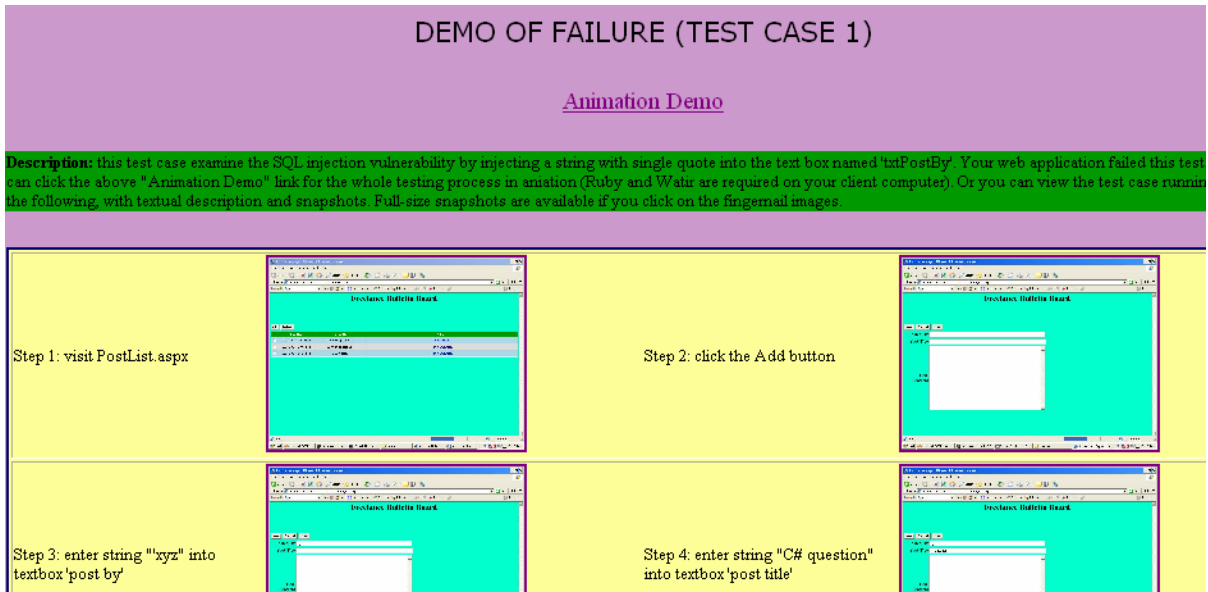


Figure 3. Demo Failure of Sample Student Project

There are two approaches for demonstrating the failed test case: (1) direct demo via animation by clicking the “Animation Demo” link, and (2) textual description of the step by step instructions on how to repeat the test case (with screenshot). By taking the direct demo approach, an instance of browser will be launched at students’ desktop, repeating and running the test scripts on the project submission. The textual description approach presents a page which shows the testing process step by step. Interested readers can visit URL <http://vlab.gsw.edu/Projects/APOGEE/APOGEEDemo.zip> for animation demo. The step-by-step textual description with snapshots is shown in Figure 3.

As shown in the figure, by entering a special string started with a single quote character we can easily crash the student implementation of Freelance Bulletin, because the project implementation did not do a very good job at input validation. In the student implementation, SQL statement is constructed dynamically by concatenating user input. When a single quote character is embedded in the user input, it destructs the original expected SQL statement structure and allows for inserting malicious SQL statements.

Without doubt, the live demo of project failure will help students to realize the defects in their programs. By fixing the problems and re-submitting project to APOGEE, students can iteratively refine their design and get much richer learning experience.

## 5. Extension of ProtoAPOGEE

Georgia Southwestern State University, Southern Polytechnic State University, Pace University, and Metropolitan State University jointly propose to expand ProtoAPOGEE into a full-fledged tool named APOGEE. The proposed APOGEE tool will contain the following new modules:

- A visual authoring tool which allows faculty members to develop project evaluation scripts conveniently. A visual representation of a test script’s control flow can be displayed for instructors who are not familiar with Ruby. New actions can be inserted into the control flow graph via visual programming by dragging and dropping visual controls. It also permits students to publish their own “open source” test scripts. It not only encourages students to adopt the test driven development methodology, but also increases fun of learning. The evaluation scripts prepared by students, after a voting process by students and approval by instructor, can be escalated to official evaluation script, which allows APOGEE to be self-adapting and self-evolving.
- A virtual laboratory environment which allows students to upload and configure their Web application projects. A virtual laboratory [17] prototype is established at Pace University using the VMWare technique. The virtual laboratory allows students to remotely deploy their Web and database projects without interfering with each other. When a project is successfully deployed, students can submit its URL in APOGEE’s Web Portal for grading. The technique to implement Virtual Lab has been very mature.

The full-fledged tool will also include the following new features.

- Recording ability: an instructor can start a testing session within the authoring tool and record the whole session. The tool will be responsible for generating the corresponding Ruby test scripts which can be further refined by the instructor. The recording function allows saving tremendous time in writing test scripts.

- Automatic generation of feedback information: during a record session, the tool automatically generates textual messages in a pre-defined format so that the textual description of the testing session can be generated and inserted into the test case demo report.
- Automatic test case generation: given an existing test case, variations of the test case can be generated by a number of well-known testing criteria such as boundary value principle. For example, given a test case which tests the “save new post” functionality of Freelance Bulletin, a number of test cases can be automatically generated by entering various boundary values into the textboxes that are accessed in the original test case. For another example, if we simultaneously start multiple instances of the test case of “save new post”, it is a test case for thread safety and database atomicity, i.e., can interleaving database actions of these multiple instances mess up with each other?

We also plan to develop companion course materials for two popular areas of Web application development: Java and ASP.Net. Each course module consists of the following: (1) background description of project; (2) formal specification of project, e.g., requirements on graphical user interface and functional requirements; (3) a collection of evaluation scripts for each requirement posted by instructor; and (4) a collection of hints for helping students improve project design when their submission fails a test script. The APOGEE tool and all course materials will be licensed using GNU public license for free distribution and derivative work.

## 6. Conclusion

In this paper we have presented an effective technique for providing instant and informative feedback to students in Web computing classes. We implemented the techniques in the prototype tool called ProtoAPOGEE, which demonstrates the feasibility and major technical approaches. We propose to extend our current prototype of the system into a full fledged tool. It will not only greatly increase the productivity of faculty members but also enrich the learning process of students in Web computing classes

## References

- [1] ACM IT Curriculum Committee, Computing Curricula Information Technology Volume (draft), available at [http://www.acm.org/education/curric\\_vols/IT\\_October\\_2005.pdf](http://www.acm.org/education/curric_vols/IT_October_2005.pdf), 2005.
- [2] Edwards, S. H. Education Support for Testing Graphical User Interfaces, NSF Award Abstract #0633594, 2007.
- [3] Edwards, S. H. Using Test-Driven Development in the Classroom Providing Students with Automatic, Concrete Feedback on Performance, International Conference on Education and Information Systems: Technologies and Applications (EISTA'03), pp. 421–426, 2003.
- [4] Edwards, S. H. Using Software Testing to Move Students from Trial-and- Error to Reflection-in-Action, Proceedings of the 35th SIGCSE Technical Symposium on Computer Science Education, ACM, pp. 26-30, 2004.
- [5] Feng M. Y., and McAllister, A. A Tool for Automatic GUI Grading, 36th ASEE/IEEE Frontiers in Education Conference, 2006.
- [6] Forsythe, G., and Wirth, N. Automatic grading programs, Communications of the ACM, 8(5) pp. 275-529, 1965.
- [7] Foubister, S., Michaelson, G., and Tomes, N. Automatic assessment of elementary Standard ML programs using Ceilidh, Journal of Computer Assisted Learning, Vol. 13, No. 1, pp. 99-108, March 1997
- [8] Gage, M., Pizer, A., et al., WeBWorK Online Homework Delivery System, available at <http://devel.webwork.rochester.edu/twiki/bin/view/Webwork/WebHome>, retrieved May 1, 2007.
- [9] Gorgone, J. T. , Davis, G. B., Valacich, J. S., Topi, H., Feinstein, D. L. , and Longenecker Jr., H. E. “Model Curriculum and Guidelines for Undergraduate Degree Programs in Information Systems”, available at <http://www.acm.org/education/is2002.pdf>, 2002.
- [10] jvmenen@users.sourceforge.net, “WatiN: Web Application Testing in .Net,” available at <http://watin.sourceforge.net/documentatie.html>, retrieved May 1, 2007.
- [11] Kohl, J., et al., Watir: Web Application Testing in Ruby, available at <http://wtr.rubyforge.org/>, retrieved May, 2007.
- [12] Laakso, M. J., Salakoski, T., and Korhonen, A. The Feasibility of Automatic Assessment and Feedback, In Proceedings of the IADIS International Conference on Cognition and Exploratory Learning in Digital Age, 2005.
- [13] Leal, J. P., and Moreira, N. Automatic Grading of Programming Exercises, Technical Report DCC-98-4, DCC-FC& LIACC, UP, June 1998.
- [14] Matt, U. V. Kassandra The Automatic Grading System, ACM SIGCUE Outlook, Vol 22 No. 1, pp. 26-40, 1994.
- [15] NetBeans.org. Jemmy Module Documentation,” available at <http://jemmy.netbeans.org/documentation.html>, retrieved May 1, 2007.
- [16] Thomas, D., Fowler, C. and Hunt, A. Programming Ruby: The Pragmatic Programmers' Guide,” ISBN-10: 0974514055, Pragmatic Bookshelf, 2004.
- [17] Tao, L., Qian, K., Fu, X., and Liu, J. Curriculum and Lab Renovations for Teaching Server-Based Computing, (poster paper), ACM Technical Symposium on Computer Science Education (SIGCSE 2007), Covington, Kentucky, USA, March 7-10, 2007.
- [18] Wall, T. Getting Started with the Abbot Java GUI Test Framework, available at <http://abbot.sourceforge.net/doc/overview.shtml>, retrieved May 1, 2007.
- [19] Wiley Inc., “Companion Tool of Java Concepts, 5th Edition”, information available at <http://hcd.wiley.com/WileyCDA/HigherEdTitle/productCd-0470105550,courseCd-CX0500,pageType-supplements.html>, retrieved May 1, 2007.
- [20] X. Li, LEUnit, available at <http://ieunit.sourceforge.net/>, retrieved Aug 31, 2007.