The Marmoset Project: An Automated Snapshot, Submission, and Testing System

Jaime Spacco, William Pugh, Nat Ayewah
Department of Computer Science
A.V. Williams Building
University of Maryland
College Park, MD 20742 USA
{jspacco,pugh,ayewah}@cs.umd.edu

David Hovemeyer
Computer Science
York College of Pennsylvania
York, PA 17405 USA
david.hovemeyer@gmail.com

Abstract

Marmoset is a framework for storing and testing student submissions to programming assignments. It gives students a limited ability to release test their code using an instructor's private suite of test cases. This encourages them to start early and implement their own test cases. It also provides facilities for instructors to manage the grading process and gives researchers access to fine-grained snapshots of the student development process.

Categories and Subject Descriptors K.3.1 [**Computer Uses in Education**]: Computer-assisted instruction (CAI); K.3.2 [**Computer and Information Science Education**]: Computer Science Education; D.2.5 [**Software Engineering**]: Testing and Debugging

General Terms Measurement, Experimentation, Human Factors

Keywords Testing, project submission, code snapshots

1. Introduction

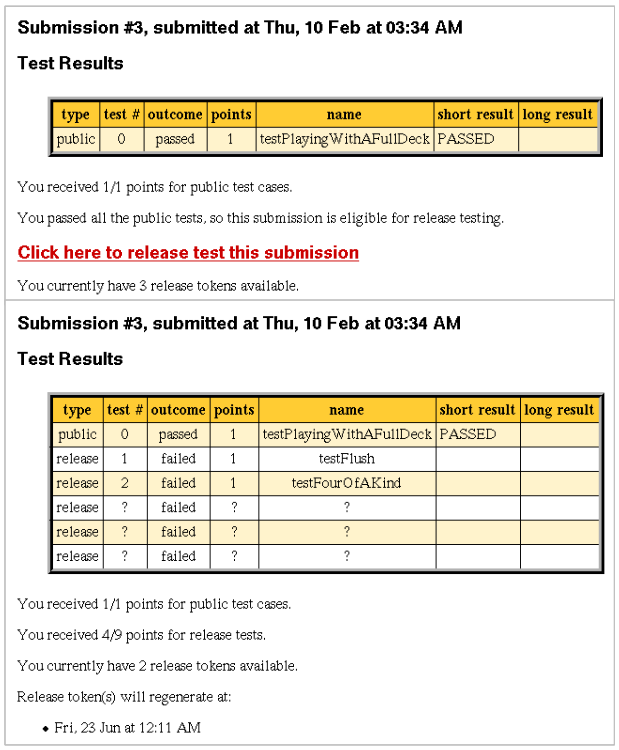
The goals of the Marmoset project are twofold: to improve the experience of students learning to program and to study how students learn to program. In meeting these goals, Marmoset provides significant technical, motivational and pedagogical advantages over traditional project submission and grading systems. In particular, it gives students and instructors feedback early in the development process and encourages students to use testing to find bugs in their programs. It also captures snapshots of student work every time they save and makes this available to researchers.

Other automated grading systems include ASSYST, Praktomat and Web-CAT. ASSYST [1] evaluates correctness, efficiency and other properties of a student submission by analyzing what is written to standard-out. It does not take advantage of unit testing systems like JUnit. Praktomat [2] offers both public and secret tests so that students have limited early information on their performance. Web-CAT [3] is a mature system that evaluates student submissions using instructor-written tests and measures the coverage of student-written tests. It does not collect the fine-grained research data collected by Marmoset.

Marmoset has been used in a number of undergraduate level programming courses at the University of Maryland with programs written in Java, C/C++, Ruby and OCaml. This paper describes the features of Marmoset and outlines how it meets the needs of its stakeholders. Marmoset is freely available to other universities. Data in our software repositories is also available to interested researchers.

2. Benefits for Students

Typically, programming assignments are tested against the instructor's secret test data after the assignment deadline, and students receive their test results days after they completed the project, when they are already starting to think about the next project. In Marmoset, students are given limited opportunities to *release test* their code. When students perform a release test, they are told the number of tests they have failed and given the names of the first two failed tests. For example, for a poker game project, students might be told that they passed 4 release tests, but failed 5, including Flush and FourOfAKind (Figure 1). The system can



Submission #3, submitted at Thu, 10 Feb at 03:34 AM

Test Results

type	test #	outcome	points	name	short result	long result
public	0	passed	1	testPlayingWithAFullDeck	PASSED	

You received 1/1 points for public test cases.

You passed all the public tests, so this submission is eligible for release testing.

Click here to release test this submission

You currently have 3 release tokens available.

Submission #3, submitted at Thu, 10 Feb at 03:34 AM

Test Results

type	test #	outcome	points	name	short result	long result
public	0	passed	1	testPlayingWithAFullDeck	PASSED	
release	1	failed	1	testFlush		
release	2	failed	1	testFourOfAKind		
release	?	failed	?	?		
release	?	failed	?	?		
release	?	failed	?	?		

You received 1/1 points for public test cases.

You received 4/9 points for release tests.

You currently have 2 release tokens available.

Release token(s) will regenerate at:

- Fri, 23 Jun at 12:11 AM

Figure 1. A student has 3 tokens to release test each submission. After using a token, the student sees the number of release tests failed and the names of the first 2.

	Exception	Projects
1	NullPointer	1,172
2	ClassCast	512
3	IndexOutOfBounds	400
4	ArrayIndexOutOfBounds	359
5	StackOverflow	281
6	NoSuchElement	238
7	IllegalState	228
8	IllegalArgument	195
9	StringIndexOutOfBounds	195

Figure 2. The most frequent exceptions in student submissions.

also be configured to provide more information such as a stack trace or line numbers.

Performing a release test consumes a release token; each student is typically given 2 or 3 release tokens for each project. Each release token regenerates 24 hours after use. Limiting the students' access to release tests in this way encourages them to write their own test cases and reason about why their project might be failing or should be correct. It also encourages them to start work on the project early (so they have more opportunities for release testing), discourages them from programming by "trial-and-error", and encourages them to seek feedback on whether they are on the right track.

Students also optionally receive additional information about bugs found by the FindBugs static analyzer and the code coverage of student-written tests. Marmoset works with multiple programming languages, although some features, such as integrated code coverage analysis, are currently only implemented for Java.

3. Benefits for Instructors

Instructors can view live summaries of student progress as students work on the project. Days before the assignment is due instructors can learn which tests cases students are having the most trouble with, which might indicate a topic that needs to be discussed in lecture, an ambiguous specification, or perhaps a faulty test case.

Marmoset provides facilities to help instructors modify a project in response to any problems that may arise. Instructors can update the test suite to change faulty tests. Experience at the University of Maryland has shown that over a third of test suites change after the project has started. Instructors can also retest submissions to look for inconsistent results that can occur in multithreaded or hashcode dependent applications. Marmoset does some of these retests in the background to look for these inconsistencies or in response to a system failure.

Infinite Recursive Loops:

```
// Construct a new Web Spider
public WebSpider() {
    WebSpider w = new WebSpider();
}
```

Statically doomed casts:

```
public Server makeServer(URL url){
    Object o = (Object)url;
    return (Server)o;
}
```

Figure 3. Examples of bugs found in student code.

4. Benefits for Researchers

Marmoset non-intrusively gives researchers access to the software development process of novice programmers. Marmoset can store a code snapshot every time a student saves his or her work, thereby preserving information about the status of student code between submissions and before the final submission. It does this by using a plugin for the Eclipse IDE to capture and transmit snapshots to a CVS repository. Student information is later removed from these snapshots to make them anonymous.

Marmoset has been used at the University of Maryland since the fall of 2004. It has accumulated over 200,000 source snapshots and 2 million test runs. The snapshots have revealed some interesting trends about student programming. Figure 2 shows the most frequent exceptions. It shows that stack overflow exceptions occur quite often. This is surprising given the simple nature of student programs at the introductory level. Further exploration of the faulty code indicates that many of these overflow exceptions are due to an infinite recursive loop similar to the one found in Figure 3.

One way to prevent this error in the future is to warn students of this bug using the FindBugs static analyzer. Following this observation in Marmoset, FindBugs was updated to report infinite recursive loops as errors. An interesting side note is that this bug was later found by FindBugs in popular open source code.

5. References

[1] Jackson, D., and Usher, U. Grading student programs using ASSYST. In *Proceedings of the twenty-eighth SIGCSE Technical Symposium on Computer Science Education,* pages 335–339. ACM Press, 1997.

[2] Zeller, A. Making students read and review code. In *Proceedings of the 5th Annual Conference on Innovation and Technology in Computer Science Education,* pages 89–92, New York, NY. ACM Press, 2000.

[3] Edwards, S. Rethinking computer science education from a test-first perspective. In *Proceedings of the 18th Annual Conference on Object-oriented Programming, Systems, Languages and Applications,* pages 148–155, New York, NY. ACM Press, 2003.