Extending a Marking Tool with Simple Support for Testing

Guy Tremblay, Louise Laforest and Aziz Salah
Dept. d'informatique, UQAM
C.P. 8888, Succ. Centre-Ville
Montreal, QC, Canada, H3C 3P8
{tremblay.guy,laforest.louise,salah.aziz}@uqam.ca

ABSTRACT

Oto is a customizable and *extensible* marking tool which aims at providing *timely* feedback to students. Based on simple test cases description formats, Oto also includes operations that help students easily test—even "mark"—their own programs.

Categories and Subject Descriptors

K.3.1 [**Computer Uses in Education**]: Computer-assisted instruction

General Terms: Languages

Keywords: Educational Software, Automated Marking, Testing

1. INTRODUCTION

Various tools providing support for marking programs are available [2], including Oto [4], a *customizable* and *extensible* marking tool which can provide *feedback* to students even *before* the final submission.

Oto has recently been improved to simplify testing, by instructors and students, using simple test cases description formats, each targeting a specific type of assignment.

2. COURSE LABORATORIES

Our introductory programming course, in Java, uses an "imperative-first implementation" [3] and includes weekly (closed) laboratories. The laboratories progress as follows:

1. Filter programs—perform textual I/O on standard streams.
2. Programs providing method "libraries"—i.e., classes defining public and static methods (procedural abstraction).
3. Simple object classes (procedural and data abstraction).

Defining large sets of tests cases, whether they are textual (filter programs) or JUnit [1] (static or instance methods) tests, can be long and tedious. Moreover, defining JUnit tests is beyond the scope of novice students' knowledge.

ITiCSE'07, June 23–27, 2007, Dundee, Scotland, United Kingdom.
ACM 978-1-59593-610-3/07/0006.

3. TEST CASES DESCRIPTIONS

To simplify the specification of test cases, three different formats have been defined:

- *Tests for filter programs*: Described by data to be read on standard input and result expected on standard output.
- *Tests for class methods*: Described by method calls followed by expected (scalar or array) results. Example:

```
int minimum( int, int, int )
===
minimum( 0, 1, 2 )
---
0
===
minimum( 3, 2, 3 )
---
2
```

- *Tests for simple classes*: Described by sequences of method calls followed by expected results.

The first format generates text files (then processed by a script) where the obtained and expected results are compared using `diff`. The other two formats generate JUnit test cases [1], including signature checking test cases (implemented using reflection).

4. CONCLUSION

Using the above test cases description formats, Oto now provides a number of high-level commands, both for instructors and students, dedicated to testing. These commands greatly simplify the instructors' task of defining test cases. More importantly, they allow students to test their own programs, without any knowledge of JUnit or testing scripts. Thus, students can be introduced early to testing (first semester).

5. REFERENCES

[1] K. Beck and E. Gamma. Test infected: Programmers love writing tests. *Java Report*, 3(7):37–50, 1998.
[2] D. Douce, D. Livingstone, and J. Orwell. Automatic test-based assessment of programming: A review. *Journal on Educational Resources in Computing*, 5(3), Article No. 4, Sept. 2005.
[3] IEEE and ACM. Computing curricula 2001. Technical report, IEEE and ACM, 2001.
[4] G. Tremblay, F. Guérin, and A. Pons. A generic and extensible tool for marking programming assignments. In *IASTED Intl Conf. on Educ. and Tech.*, pages 55–60, 2005.