A SWOT Analysis of Introducing Practical Assessments in Introductory Programming

Christine Prasad
Computing and Information Technology
Unitec New Zealand,
Auckland, New Zealand
cprasad@unitec.ac.nz

ABSTRACT
This paper reports on the introduction of practical tests as a new assessment initiative in an Introductory Computer Programming course. Firstly, a description of the background of the initiative, the rationale behind their use, and details of how they were implemented are presented. Then, based on student and staff feedback, an analysis of the initiative is discussed, based on the strengths, weaknesses, opportunities and threats faced. The feedback showed that the practical tests were generally positively received and resulted in a higher pass rate of the course and a better learning experience for students.

Keywords

1. INTRODUCTION
In mid-2003, a Division funded project was instigated to review the Teaching, Learning and Assessment (TLA) practices used within its courses. The main rationale behind this project was the need to develop student capabilities/competencies to ensure that the programmes aims, objectives and graduate profiles were met.

The key instructors on a small number of courses were selected to become part of the TLA Pilot Group that operated under the guidance and framework of a TLA Steering Committee. One of the main objectives of the project was to get instructors to think “outside the square” and apply innovations to their TLA practices.

The Introductory Programming course was selected as one of the four courses to participate in the project as a pilot group. This course is a first year, compulsory course for both Computing and Information Systems majors. Involvement in this project forced us as instructors to take a step back and take an objective look at the TLA practices we use in our respective courses.

Without going into much detail on the TLA project, this paper highlights, in a step by step manner, the main tasks that were carried out as part of my involvement in the project, and presents an evaluation of the implemented initiative based on its strengths, weaknesses, opportunities and threats.

2. IDENTIFICATION OF TLA PRACTICES
The first step of the project was to identify the TLA practices adopted in the introductory programming course that I taught during Semester 2 2003. In this course, students had four and a half hours of contact time with their lecturers during the week, of which one and a half hours was dedicated to a lecture session and three hours were dedicated to non-assessed closed laboratory practice sessions. During these practical sessions students were required to work independently on practical problems related to the concepts covered during the previous lecture session, with the instructor available for assistance. The assessment breakdown of the course during Semester 1 2003 is shown in Table 1.

3. EVALUATION OF TLA PRACTICES
The second step was to evaluate the TLA practices against the learning objectives of the course. This was done using a number of methods including class observation, discussions held...
amongst the teaching team, discussions held with students, and a student survey. Three of the major issues observed were:

3.1 Ineffective Student Feedback Mechanisms

Since the practical sessions required students to work independently, their main source of feedback were the assignments and tests. These had a 2-3 week marking turnaround time, which didn’t always provide the student’s strengths and weaknesses in time for the next assessment. The assessments were summative and effective in supporting a gradual numeric build-up towards the final grade of a student. Yet such assessments concentrate on the “final product” as opposed to the learning processes involved (Ippolito, 2004).

3.2 Plagiarism

The level of plagiarism in both the assignments was very high. Students were encouraged to work collaboratively in assignments provided they submitted their own work. This made it difficult to determine where collaboration stopped and plagiarism began.

3.3 Unrealistically High Expectations

Each assessment assessed a combination of concepts taught in the course. For example, in an assignment, students had to apply a number of concepts they had learned in class to a practical problem. Even though the problem was relatively small, it still required students to apply knowledge before most of them had even gained basic literacy in programming. Lister and Leaney (2003) connect literacy in programming to the two lowest levels of Bloom’s taxonomy, i.e. “knowledge” and “comprehension”.

When evaluating the assignments in light of Bloom’s Taxonomy of Educational Objectives (Anderson and Krathwohl, 2001), I realised that our assignments expected all students to embark on the third level of “application”, even at a very basic level, before successfully achieving the first two levels. It appeared that we were placing unrealistic demands on some of our students, particularly the under-achievers. We needed to acknowledge that these were only first year students, many of whom were forced to take this course before they could proceed with the rest of their undergraduate study.

Other issues observed were that students did not do any preparatory work prior to participating in the practical sessions and as such did not complete the required exercises; they fell behind easily when difficult concepts were introduced and/or when pressure from other courses build up; students who did not comprehend basic concepts such as control structures, variables and data types, had almost no possibility of catching up after the fourth week of class; student motiva-

<table>
<thead>
<tr>
<th>Assessment (previous)</th>
<th>Assessment (revised)</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Assignment 1 (15%)</td>
<td>Practical tests (35%)</td>
<td>Test 1 (5%) – Decisions</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Write program from given design</td>
</tr>
<tr>
<td>Mid Term Test (20%)</td>
<td></td>
<td>Test 2 (7%) – Loops</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Write program from given design</td>
</tr>
<tr>
<td>Assignment 2a (5%)</td>
<td>Assignment Part a (5%)</td>
<td>Test 3 (10%) – Functions</td>
</tr>
<tr>
<td>Assignment 2b (15%)</td>
<td>Assignment Part b(15%)</td>
<td>Read and modify code program</td>
</tr>
<tr>
<td>Final Exam (45%)</td>
<td>Final Exam (45%)</td>
<td>Test 4 (13%) – Arrays</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Design and write program</td>
</tr>
</tbody>
</table>

Table 1: Previous and Revised Assessment Breakdown

When evaluating the assignments in light of Bloom’s Taxonomy of Educational Objectives (Anderson and Krathwohl, 2001), I realised that our assignments expected all students to embark on the third level of “application”, even at a very basic level, before successfully achieving the first two levels. It appeared that we were placing unrealistic demands on some of our students, particularly the under-achievers. We needed to acknowledge that these were only first year students, many of whom were forced to take this course before they could proceed with the rest of their undergraduate study.

Other issues observed were that students did not do any preparatory work prior to participating in the practical sessions and as such did not complete the required exercises; they fell behind easily when difficult concepts were introduced and/or when pressure from other courses build up; students who did not comprehend basic concepts such as control structures, variables and data types, had almost no possibility of catching up after the fourth week of class; student motiva-
tion levels were generally low; and some students were attempting assessments even if they had not attended a single lecture or practical session all semester.

The effect of these issues was that many students did not gain basic programming skills by the end of the course. As such, failure rates were high and student confidence in the course was low.

4. CURRENT TRENDS IN ASSESSMENT

The next task was to investigate possible TLA practices that would try to solve some of the issues identified in Section 3.

The TLA project offered presentations and workshops that presented a number of innovative TLA practices from non-computing disciplines. The three most popular practices are shown below. I compared parallel innovations within introductory programming teaching from the available literature as reported briefly here:

4.1 Collaborative Learning

Incorporating an element of collaboration into teaching and assessment is popular across all disciplines since it develops skills like that potential employers look for in graduates, e.g. ability to work in teams, communication skills. Peer assessment is also used in many cases. At introductory level, pair programming, as a form of collaboration, is gaining popularity (LeJeune, 2003; Nagappan, et al., 2003).

4.2 Portfolio Assessment

Portfolio assessment is described as “a purposeful collection of student work that exhibits the student’s efforts, progress and achievements in one or more areas” (Paulson, Paulson and Meyer, 1991, p60). Maintaining a reflective journal is a form of portfolio assessment.

The key objective of portfolio assessment is to get students to reflect on what they’ve learnt in a specified period of time, and to possibly get students to think beyond the grade offered in assessments, to the actual learning that takes place.

The use of portfolios and reflective journals are slowly entering the introductory programming arena (Denton, Doran and McKinney, 2002) but has not been proven as a full substitute to the traditional forms of programming assessments. Also, at introductory programming level, it may be premature to expect students to write meaningful and deep reflections of their learning.

4.3 Problem Based Learning (PBL)

Problem based learning (PBL) is a learner centered educational method that encourages learners to take responsibility of their own learning and is based on real-world problems. One of the main advantages of PBL is that provides students with generic skills that are very useful in real-world situations like group work, problem solving and independent learning. PBL takes many forms but its philosophy stems from the principles of constructivism (Henze and Nejdl, 1998).

Those who use PBL techniques in their introductory programming courses have reported on the effectiveness of the practice (Henze and Nejdl, 1998; Kolling and Barnes, 2004; Greening, Kay and Kingston, 1996).

5. SELECTION OF TLA INITIATIVES

The next step of the project was to select suitable TLA initiatives that would solve some of the issues discussed in Section 3 and satisfy the criteria set by the TLA project, which stated that any selected initiative should:

• Not increase teacher workloads
• Motivate and improve student learning (i.e. encourage ‘deeper’ learning)
• Improve feedback to students on their learning
• Improve teacher monitoring of student learning
• Motivate and improve teacher learning
• Develop a broader range of technical and non-technical capabilities in students
• Provide ‘reasonable’ assurance of individual student achievement of learning outcomes of the course.

The current trends in assessment from Section 4 above, as lucrative as they were, would
have greatly increased teacher workload had I attempted to implement any of them. For example, to change from the traditional style of teaching to PBL requires a huge shift in pedagogy, which would require careful planning, time and effort.

5.1 COMMON ASSESSMENT THEME

The common theme that emerged from TLA practices from other disciplines as described in Section 4 was having a greater number of frequent assessments with lesser weighting as opposed to a lower number of larger weighted assessments. One of the presenters referred to this as “mouse-milking”, as it meant larger quantities of smaller assessments.

“Mouse-milking”, as presented, encouraged students to learn course content in small, manageable “chunks” as opposed to large “chunks” in batches. It also reduced the likelihood of students getting discouraged if they did not perform well in one particular assessment item, since assessment items were small and assessed a small number of concepts. It also enabled students to learn basic concepts before moving on to more complex concepts and provided continuous feedback to teachers and students.

Overall, this strategy tended to increase motivation levels and provide a better learning experience for students. In educational theory, this is known as formative assessment. A formative assessment is one that “is utilized to monitor learning progress during instruction, provide continuous feedback to the student, identify areas for improvement, and reinforce learning” (Ippolito, 2004). There is evidence to show that strengthening the practice of formative assessment produce significant and often substantial learning gains in a class (Black and William, 1998).

5.2 The Selected Initiative

I decided, as a starting point, to attempt “mouse-milking” in my introductory programming course, i.e. have smaller, frequent assessments that would give quick feedback to students. This would be in the form of a series of practical tests. A practical test requires that students carry out a small practical programming task under test conditions during their closed laboratory session.

Formative assessments do not normally have marks or grades awarded for task, but to serve as a motivator for students, our practical tests were also summative and awarded marks to students.

The practical tests were selected because it was likely to satisfy most of the TLA project criteria (except for developing non-technical skills), and was likely to address some of the issues raised in Section 3 above.

The revised assessment breakdown now included a 35% practical test component that replaced one assignment and a mid-term paper test from the previous breakdown, as shown in Table 1.

The final exam was a closed-book test that contained a mixture of theory questions, and small design and coding questions which were heavily based on the second assignment.

5.3 Preparing Practical Tests

Practical tests aren’t a new TLA practice and are being used quite extensively in programming courses, yet the implementation involved putting careful thinking into the preparation of the tests as they were specific to the context of our institution. The following sections describe identified issues.

5.3.1 Assessment

- Each test will involve students having to write either entire programs or parts of programs.
- Each test is to assess one concept only e.g. loops, decision, functions, array, debugging etc.
- Each subsequent test will gradually increase in complexity

5.3.2 Operational and administrative issues

- Four tests will be held during the semester; each with different weightings according to the complexity of the question.
- Each test to be held in a computer lab during a practical session
- To minimise plagiarism, 3-4 versions of the same test would be produced
- Each test had to be short enough to complete in a 1.5
hour session with different times allocated according to the complexity of the question.

- Tests were to be open book but only the programming workbook/textbook is allowed.
- The tests must have a casual feel to it as opposed to the formal exam-like atmosphere.
- Students are not to communicate with each other during the test.
- Different student logins will be allocated for each test so that students cannot re-use pre-existing programs.

5.3.3 Style
- Tests should have the same look and feel as practical exercises.
- The specification will need to be simple enough to provide sufficient and unambiguous explanation of the program, yet not involve too much reading.
- A standard template had to be designed.

5.3.4 Revision
- There were to be no nasty surprises for students, they were to be given a practice test before the actual test with sufficient time to practice and ask questions. A sample solution would also be provided.
- The actual tests should be similar in style and logic to the practice test.
- A detailed marking schedule will be given with the test so that students are aware what they’re being marked on.

5.3.5 Marking
- The detailed marking schedule included with the test will be used so that the marking process is transparent and easy.
- Marked test should be returned to students within a week.

5.3.6 Feedback
- Concise comments will be written when tests are marked.
- Students would be allowed to re-sit tests if they wanted.

6. EVALUATION

Following the successful administration of the tests for a semester, its effectiveness was evaluated by personal observation, and feedback from students via a survey in which 60 students out of 118 responded (51%). The evaluation was done on the basis of the strengths, weaknesses, opportunities and threats of this new practice.

6.1 Strengths

6.1.1 Increase in Pass rate

The pass rates for the course increased from 62% (Semester 1, 2003) and 54% (Semester 2, 2003) to 75% in Semester 1, 2004 after the tests were offered, being one positive indication of the success of the implementation.

Another positive spin-off was the increased number of students who decided to proceed to the CS2 equivalent course, which is not a compulsory course.

6.1.2 Plagiarism Issues

Having different versions of the test and allocating different logins to students meant that they did not have much opportunity to cheat. At least one student favored this by commenting “there were a variety of tests so students couldn’t copy”.

6.1.3 Smooth Administration and Marking Process

The administration of the tests went very smoothly, there were no major hiccups except for one instance when the computer network showed instability.

The marking process was also very straightforward. For the easier tests, it took an average of 5 minutes to mark each test. For the longer tests, it took an average of 8 minutes to mark each test.

6.1.4 Student Motivation and Learning

From observation, it appeared that students were working hard towards achieving an understanding of the test material from the sample tests. The tests also pushed the students to learn the basic concepts (i.e. gain literacy in programming), before moving on to more complex concepts. They were also exposed to typical design patterns using loops and decisions, which they would later use in their assignment. This mastery of basic concepts appeared to motivate them to learn the more complex concepts. Student comments that enforced this observation were:
“It forced me to learn the knowledge in time; otherwise I might not review them at all”

“They make you keep learning so I don’t slack off because I don’t have any tests”

“It can increase the practical programming skills and boost confidence”

“Can learn new things which is the main reason I will work hard and revise before the test”

6.1.5 Student Satisfaction

Students were generally satisfied with the testing process as demonstrated by their responses to the question “What did you like the most about the practical tests?”, which were:

“They only concentrated on specific areas like functions. We didn’t have to remember everything and put it all into the program”

“Open book, so you only worry about the logical thinking, nothing else”

“The fact that my brains got used to thinking in a logical way”

“The problems looked like real problems”

“Fair marking”

“Easy to follow”

“They were practical”

“Get to practice our programming skills”

Students were also happy with the fact that the tests were open-book, short, and not very difficult. In addition, 71% of the students who responded felt that four tests were sufficient, while 61% felt that the time allocated for them to do each test was sufficient. 68% of respondents felt that the instructions given on the test was clear and 73% felt that the test specifications were sufficiently logical to follow.

6.1.6 Sufficient Feedback

As an instructor, I found myself more aware of individual student capabilities. It also gave me an idea of students’ attitudes towards the course. In turn, I offered one-on-one sessions to help those students who requested assistance.

It also appeared that students were benefiting from the feedback they received. This was shown by improvement in certain areas in subsequent tests, for example, code layout, documentation, variable naming. Some positive student comments relating to getting feedback were:

“I can test myself as well as the examiner does”

“Theory was put into practice and feedback was given on your work”

“Help use to review the knowledge during our study”

6.1.7 Success of Sample Test

Students were very satisfied with the sample/practice tests. Some of the comments they made on the sample tests in their feedback questionnaire were:

“It gave me a good idea of what was expected in the tests”

“I have learned more from the sample tests”

“Good stuff to get ready to the real test”

“Gave us a picture of what things will be tested in the exam”

6.2 Weaknesses

Despite student satisfaction, the problem of students’ not preparing for practical classes still remained.

Many students, who appeared to have gained basic literacy from doing the practical tests, could not apply the same concepts to slightly more complex problems such as the one presented in their assignment. We need to bear in mind, though, that this is an introductory course and many students are not likely to continue with any other programming course.

A few students pointed out that the things they liked the least about the test were:

“Tests were too specific, only had to do one thing”

“Failing – not getting the program to run”

Perhaps the biggest weakness of this method lied with those students who were not interested in the formative learning process but the marks that were awarded for it. These were a small number of students who did not bother to collect their tests but only checked their marks on the online system.

Another weakness was the initial preparation that went with the implementation of the tests. As such, it did increase teacher workload, at least during the preparatory stage.
6.3 Opportunities

Introducing formative assessments combined with a summative component opens up many opportunities for teaching practical courses like programming. For example, using an online testing system will enable students to get instant feedback on their tests and lower teacher workloads.

Other opportunities include aligning teaching towards practice by minimizing formal lecture presentations and maximizing discussions with students and practical sessions, and decreasing the final exam component or totally substituting the final exams with more practical tests.

Opportunities also exist for incorporating new and other innovative TLA practices as described in Section 4. These, however, will require more planning and preparation, but the lessons learnt from implementing this initiative will assist in implementing more complex ones.

6.4 Threats

The greatest threat of this method was its potential to inhibit creativity in students since the tests were very specific to particular concepts. However, this issue was likely to largely affect the over-achieving students and they were given the opportunity to show their creativity in the larger assignment. A small number of student comments reflected this threat:

"The tests were too easy. If you look at past tests, you know what the test is going to be like. You learn that but you don’t really get to ‘play’ around with the programming language”.

“We cannot use our creativity. You don’t need to find the solution, just to translate to code a given solution”.

The tests were obviously effective for assessing programming skills, but not as effective for assessing problem solving skills. As such, they were beneficial for the underachieving students but not necessarily for the overachievers.

Such problems of assessment are typical at introductory programming level, where the mix of students is diverse (Nagappan et al., 2003). A possible solution to this problem is to use the criterion-referenced assessment method (Lister and Leaney, 2003) and offer extra assessment opportunities for the overachieving students.

7. CONCLUSION AND FURTHER WORK

Being involved in a project such as the TLA project provided an ideal opportunity to take an objective look at the teaching, learning and assessment practices of the introductory programming course. Although a small initiative was implemented at this stage, the effects of it were significant in solving some of the issues that were formerly apparent in the course. As demonstrated by the evaluation, students have expressed a high level of satisfaction with the tests and instructors found it very useful. There is, however, room for improvement in the current implementation of practical tests, particularly in the area of providing the stronger students with more challenge and introducing other forms of teaching, learning and assessment practices that go beyond developing only technical skills in students.

REFERENCES


