Laboratory Reports, Reflective Essays, and the Contributing Student Approach

John Hamer
University of Auckland
j.hamer@cs.auckland.ac.nz
https://www.se.auckland.ac.nz/wiki/index.php/SE250

OVERVIEW
Requesting reflective essays instead of technical reports has led to students observing more, writing more, developing their own personal writing style, and becoming more self-aware and self-critical. The essays complement a course taught using Collis' *Contributing Student Approach* by providing a rich source of material for use in subsequent learning resources such as course notes and self-evaluation quizzes. They also provide timely feedback, both for the students and for the lecturer, and help foster a collaborative, supportive learning environment.

Keywords
Laboratory report, reflective essay, contributing student approach.

INFORMATION ABOUT THE CLASS
The context of this case study is an introductory BE(Software Engineering) course on Data Structures and Algorithms. The course is now in its seventh year, and typically has an enrolment between 30 and 60 students (currently 35). The BE is a largely prescribed degree, and the course is compulsory for all SE students. The institution is a research-led University in New Zealand, and most University courses follow a traditional lecture model. Entry standards for Engineering are relatively high, and the students are generally capable and motivated. All students entering the Software Engineering specialty are computer literate, albeit with varying confidence in programming.

The class is taught using elements of the "contributing student" approach of Betty Collis (Collis 2005). Instead of traditional coursework assignments, students are asked to prepare learning resources on course topics to share with other students. Contributions can include notes, worked examples, software visualizations, self-assessment quizzes, annotated reading lists, etc.

Assessment is dominated by a test and final examination (together 75%). Laboratory work contributes 10%, and the remaining 15% is awarded for student contributions. Lectures (renamed "class meetings") follow a student-driven agenda, and include a mixture of lectured material and (increasingly, as the course progresses) student presentations. A "wiki wiki web" (Wikipedia, 2006) is used as a collaboration tool, for students to upload and share their contributions, lab reports, peer-reviews and other course-related material.

DESCRIPTION OF THE CASE
I have reported on many of these course elements elsewhere (Hamer, 2006a,b), and would like to use this case study to discuss a recent development with the course laboratories. Laboratories are held each week, in a two-hour session. An instruction sheet is provided at the start of the lab, and students may choose to work individually or in small groups on the practical activities. The final (half-) hour is spent writing a *reflective essay* and posting it to the course wiki. Once the lab is complete, students are instructed to read a sample of the essays and comment on similarities and differences from their own experience. After each lab, a group comprising 3 or 4 students is asked to read all of the essays and feedback, and write a paper describing the expected results for the lab together with any common errors and problems. The paper is also posted to the wiki, where it becomes the basis for a variety of student contributed learning resources. For example, the common errors provide a fertile source of inspiration for quiz questions.

A small number of marks (1–2%) are awarded for each lab, and take into account both the essay and the reviews written by the student. The paper written by the group is assessed by the lecturer, with each member of the group sharing the same mark.

In introducing the reflective essay, the lab instructions read:

> *Your essay should describe your approach to writing this program, identify which parts were easy and which were hard, note any problems you encountered, anything unexpected or interesting you observed, decisions you would stand by, decisions you regret, mention any help you sought (say when and why), things you would do differently next time, things you would do the same, any lessons you learnt, things you need to know more about, and the value you see in writing this reflective essay.*

The essays ranged from half a page to around two pages in length. They are invariably written in the first-person, and drafts are neither expected nor observed.

RATIONALE IN TERMS OF EDUCATIONAL IDEAS

In most institutions, assessment is an "individual and competitive" process (Boud at.al, 1999). Even with the growing shift to criterion-referencing, norm-referencing still dominates. When individualistic views of assessment are dominant, collaboration can be seen as cheating. Assessment is also the principal mechanism for staff to exercise power and control over students. This has the effect of circumscribing learning to the outcomes (unilaterally) defined as legitimate by staff. Students learn first to distrust their own judgements, and then act as agents to constrain themselves. Boud el.al (ibid), citing the work of Marton et al. (1997), argue that "inappropriate forms of assessment appear to encourage students to take a surface approach to learning… conforming to the narrowest interpretations of assessment tasks and working to beat the system rather than engage in meaningful learning."

The contributing student approach forms the basis of a learning environment that is inherently collegial and supportive, rather than individual and competitive. Collaboration is an explicit requirement of the course. The reflective essays take this further, by **legitimizing** a student's personal learning experience. Students are expected to share their own observations, understandings, confusions, successes and failures. As a consequence, they are much more willing to record unexpected or interesting observations, the very fuel of scientific discovery.

The reflective essays are **purposeful**; the writing task is not undertaken to satisfy the whims of the lecturer, but to provide raw material for the lab maintainers' report and quiz writers. Students certainly **write more**, whether because of this, or perhaps because they are not distracted trying to conform to a prescribed report format.

They are also quick to develop an **individual** writing style, and many of the essays exhibit quirky layouts and other personalisations.

The reflective essays encourage **self-assessment**, and provide **feedback** in several forms. The immediate peer feedback is mostly non-critical observations on similar and different experiences of another student. The laboratory context adds an **immediacy** to the activity. There is no time to write and polish drafts, and so the essays are a **low-stakes** exercise. Feedback is also timely, occurring within a day or two after the lab. More detailed, generic feedback is generated within a week by the lab maintainers, in the form of a collective summary taken from all the lab reports. As a lecturer, I receive feedback about the problems students are having, and can use this to fine-tune the pace of the course.

Despite the many opportunities for generic skills development, **content knowledge** remains the primary focus of the practical lab activity.

Overall, the approach moves some way to achieving a balance between the Acquisition and Participation learning metaphors articulated by Sfard (1998).

EVALUATION

The reflective essay proved to be a surprisingly popular form with this group of students; I was concerned that it would meet with resistance as an unfamiliar and personal format. Instead, having requested a reflective essay in the first lab, almost all students adopted a reflective style in a subsequent lab that asked for a more formal technical report.

I found the resulting reports more enjoyable to read than previous years, with students much more inclined to mention unexpected observations and results:

> *The strange thing was that CRC performed badly on the randomness tests from TEST 1 which seems a little odd. It is possible though that CRC may not be as random but still evenly distribute the results. This should be more thoroughly tested... Possibly by the lab maintainers?*
>
> *I understand why it came out as such, because i only made one reference for a,b,c,x,y,z they all pointed to the same one, which i did not expect to happen.*
>
> *Clearly, my prediction was wrong, since the ? was supposed to be a ternary node. I was confused with the brackets ?(>(+(a b) c) *(z +(y b)) and thought that they should be under the * branch. The graph was much simpler than I thought.*

Students were often brutally honest in admitting the limits of their understanding:

> *I chose not to record the Monte Carlo Pi Value or Serial Correlation Coefficient as i do not understand what they are*
>
> *I couldn't do the last task because I ran out of lab-time and also cause none of it made sense to me.*
>
> *Output: Unexpected token: '+' at "+ 3)" . I have no idea what this means.*

The informal nature of the report encouraged student to include personal comments and mention other issues in their lives:

> *Ick, ignore the above message, I ran out of time, and 251 test tomorrooooow *panics and abandons rest of lab report**

In asking students what value they saw in the essays, several themes emerged. Meta-cognitive aspects of the exercise are clearly evident:

> *This essay is a very useful resource to have and to learn from, it helps you to explain why you decided to do something a particular way and also allows you to justify your choice and methods. Looking at how others did this task is a great insight into how it should be done or how it can be done better. The actual writing of the essay also makes you think more about what you have done, as you need to be able to explain it to someone else and this requires you to understand what you are doing.*

Sentiments valuing exposure to a variety of approaches and solutions were common:

> *This essay was suprprisingly a good idea as it increased my awareness of the nature of Software programming. As in stepping away from the coding for a moment to show me the big picture of how problems can be or should be approached. That it helps to think outside the box in terms of the overall approach and how there may be multiple methods of approach, and how different people will actually take different approaches to solving problems. I have also learned for this session the discussing the problem with peers is*

an effective way of helping to solve the problem as well as improving your skills and way of thinking and also finding new ideas.

The implicit archiving of material was also seen as important:

I see that writing this essay will be very valuable for me in the future to see how much I have improved over the year.

REFERENCES

Boud, D., Cohen, R. & Sampson, J. (1999), 'Peer learning and assessment', Assessment and Evaluation in Higher Education 24(4), 413–426.

Collis, B. (2005) The contributing student: A blend of pedagogy and technology. In *EDUCAUSE Australasia*, Auckland, New Zealand, April.

Hamer, J. (2006a) Some experiences with the "contributing student approach". In *ITiCSE'06 Eleventh Conference on Innovation and Technology in Computer Science Education*, pages 68–72, Bologna, Italy, June 2006. ACM Press.

Hamer, J. (2006b) Contribution-based pedagogies in engineering education. In *AAEE'06 17th Annual Conference of the Australasian Association for Engineering Education*, Auckland, New Zealand, December 2006.

Hamer, J. (2007). Peer assessment using Aropä. In Mann, S. and Simon, editors, *ACE'07 Ninth Australasian Computing Education Conference*, 43–54, Ballarat, Victoria, Australia. Australian Computer Society.

Marton, F., Hounsell, D. & Entwistle, N., eds (1997), *The Experience of Learning: Implications for Teaching and Studying in Higher Education* (2nd edn), Edinburgh: Scottish Academic Press.

Sfard, A. (1998). On two metaphors for learning and the dangers of choosing just one. *Educational Researcher*, 27(2):4–13.

Wikipedia (2006). Wiki. http://en.wikipedia.org/wiki/Wiki/. Accessed 29 September 2006.

APPENDIX I
This is a typical reflective essay, taken from around mid-way through the course.

Introduction

Welcome! This is Soda's 7th lab report.

In this lab we had fun playing around with binary trees, to gain a better understanding of how binary trees worked, particularly rotations, and how to use them to balance out skewed trees.

The Lab

Getting Started

To start off this lab, I copied the code into Visual Studio and tried to compile. However, it appeared that this was another one of those things that couldn't be done in Visual Studio. Grr. This meant that I would have to try using cygwin, I spent a bit of time trying to remember how to navigate around and compile, but once this was done, everything went fine.

I had a bit of a browse around the code, some of it was familiar, while some bits looked confusing, and I decided to revisit at a later time, since understanding the implementation details is not really fundamental to completing this lab, though the knowledge may be helpful later on.

Inserting Elements Into Tree

Seemed simple enough to understand how to insert them using the commands provided. I created a random tree with several elements and I felt comfortable navigating around this tree, so it was time to move on and answer the following question:

What insertion order results in a perfectly balanced tree?

Just for fun, I inserted 7 elements in the following order:

```
a, b, c, d, e, f, g
```

It produced a lovely linked list looking structure, which is not what one would probably call balanced.

Obviously starting in the middle is a better option, so I tried the following order:

```
d, f, b, a, c, e, g
```

Which produced a nice, perfectly balanced tree.

Do any other orders also result in a perfectly balanced tree?

Maybe?

But this isn't really a good enough answer, so I think perhaps I should experiment more with insertion orders and have a look.
Starting with anything other than the middle letter didn't really help, and resulted in unbalanced trees, but it seems that swapping around 'f' and 'b', and swapping 'c' and 'e' with each other didn't affect the balancing. The following ordering also produced a balanced tree:

```
d, b, f, a, e, c, g
```

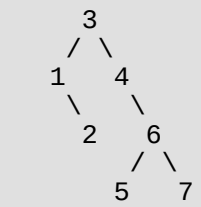
Rotation fun

Skewing and Performing Left Rotation on Tree

The aim of this part of the lab seemed to be to confirm that performing a left rotation on a tree would not alter the ordering of the tree, and thus ensure that the tree retained its 'tree' properties. I switched back to using numbers, since I felt more comfortable with them than letters =X. (I don't remember my alphabet! T_T How sad is that?)
To do this, I purposely inserted elements (1-7) in an order so that the tree would be skewed:

```
(3, 1, 4, 2, 6, 5, 7)
```

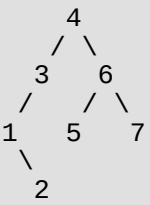
Which produced the following tree:



Using 'print', I obtained the following ordering for the above tree:

```
Tree[1,2,*3*,4,5,6,7]
```

I then performed a left rotation on the root by using the 'rotate left' command on the root when it was selected.
The resulting tree was as follows:



Using 'print', the ordering appeared to be the same as above, which is good.
I then proceeded to muck around with the left and right rotations and observed how they affected the shape of the tree.

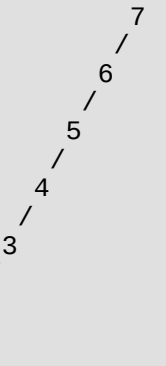
Re-shaping a right-skewed tree into a left-skewed tree

Using the 'skew' command, I obtained the lovely linked list looking thing that I obtained with my first go at inserting elements into the tree, which was right skewed.
Now it is time to see if it can be re-shaped to be left-skewed.

After a bit of experimenting (I initially started by going to element #4, but it didn't turn out as I wanted), I decided that going to the root and rotating left was the best way to get the tree completely left skewed.
These are the following rotation/movement commands I used:

```
'skew'
'rotate left'
'root'
'rotate left'
'root'
'rotate left'
 ...
 etc
```

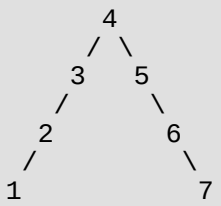
I performed a total of 6 rotations, and obtained the following tree as a result:



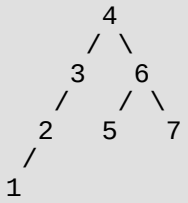
Re-shaping a right-skewed tree into a balanced tree

I obtained a right-skewed tree using the same method as I did for the previous task.
The next task was to use rotations and movement commands to try and make this tree balanced.

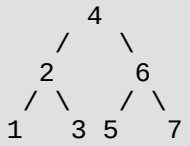
I started by shifting half of the tree to the left of number 4, by using the method above (selecting the root and rotating left, until 4 was the root of the tree). This produced the following tree:



This is starting to look more balanced at least, now it looks like some of the leaves along the right branches need some left branches. To do this, I selected element 5 ('root' and then 'right') and rotated it to the left ('rotate left'), now the tree looks like this:



Now its just a matter of doing the same to the left branch. This was simple, 'root', 'left' to select the third element, and 'rotate right' to rotate this to the right, producing the final desired outcome:



Adding two larger elements to a balanced tree and rebalancing

For this task, I added the number '8', then the number '9' to the tree above, so they were both to the right of the number '7'.

Tbc...probably after lunch...or dinner...or sleep. But it should be there by the end of tomorrow at least ^_^.

Ick, ignore the above message, I ran out of time, and 251 test tomorrooooow *panics and abandons rest of lab report*

APPENDIX II
Typical feedback (for another report):

> *Ya I agree with you - atlast Cygwin did work!! Looking at your code - everything was pretty much the same as mine. Few differences - In Task 2 you made the tree together in one statement which must have taken a while. I split it up in 4 statements. Also in the last task, I checked for the tree being null before actually calling the switch statement on it. I also faced the same problems in writing the code for the last task. So, good to know that I wasn't alone.*