Plagiarism : a Survival Strategy or Recipe for Disaster?

Dr. Pj Radcliffe
RMIT, Melbourne, Australia
pjr@rmit.edu.au

Dr. Heiko Rudolph
RMIT, Melbourne, Australia
heiko.rudolph@rmit.edu.au

***Abstract**: This paper outlines experiences in first and second year Engineering Computing at RMIT. It shows how an excellent subject design can have its education outcomes ruined if the scope for plagiarism is not severely limited. The reasons for the plagiarism are examined using surveys and focus groups and it is concluded that the traditional long term goals of university students are being increasingly replaced by short term goals which mitigate for plagiarism and against skills and knowledge acquisition. The proposed solution is to meet that pressure for short term goals with appropriate short term assessment strategies. The result is a "managed learning environment" where students are given clear reasons for a subject design and its purpose and then face short term regular hurdles throughout the course. A key requirement of this path is unbiased assessment along clearly defined boundaries. To this end we have developed an automatic marking tool that applies the same standard to all students in a course. The tool gives students comprehensive feedback and frees tutors and lecturers from the considerable burden of marking. The managed learning approach appears to be working well and ensures the majority of students do adequately master the skills and knowledge of Engineering Computing.*

Introduction

The Changing Face of Student Motivation

The traditional view has been that the majority of university students are at university in order to acquire skills and knowledge simply for the pursuit of knowledge in its own right, to attain a career, or at the very minimum to pass the degree title with high enough marks to be attractive to an employer. It is assumed that students must be motivated by such long term goals otherwise they would not be at university in the first place. In the full fee environment of the pre-Whitlam era that was probably a valid assumption because the norm for high school graduates was *not* to go to university but rather to enter the work force, often leaving school before year 12. The decision to go to university was often difficult, largely restricted to the professional and moneyed classes and beyond the financial reach of the majority.

Even before the later Dawkins "reforms" of the school system many students went to "Tech Schools" who's main aim was that of placing students in trade apprenticeships. As in the pre-Whitlam period many students chose a non-university pathway. Under Dawkins, High Schools and Tech Schools were amalgamated and in the majority of cases became simply High Schools. There are good arguments that this caused the trade skills shortages we see now in Australia.

The situation we face today is that the majority of students are funnelled into university ABS (2005), in fact university had become the default path for students with any academic potential. The path to University is no longer just the result of active and difficult choices that are usually related to a long term goals and motivation. The authors have had extensive discussions with large numbers of first

and second year students in the School of Electrical & Computer Engineering at RMIT including focus groups, "student at risk" interviews, and a survey of the entire second year cohort. Like Ashworth (1997) we found the value of written surveys a little suspect and better value in the focus groups and interviews. It is clear that many students come to university for other reasons such as-

- "It's expected of me by parents, teachers, and peers."
- "It was the default destination after High School."
- "It was better than being unemployed."
- "Full time menial work is boring/degrading/hard."

Focus group discussion would seem to indicate around 60% of first and second year students fall within this category. It would be interesting to test this observation with a more rigorous study.

Such reasons do not translate into an internalised, meaningful, long term motivation for going to university and so we contend that such students become motivated by short term goals. From focus groups, student at risk interviews, and surveys the common short terms goals observed in the student body include-

- University is only one (often a small) part of life.
- Make more time available for paid employment.
- 'Get a good mark' (which is not the same as skills development and knowledge acquisition).
- Minimize work, maximize leisure.

The lack of long term goals could be expected to change the way students behave and consequently the education pedagogy that is behind the design of university subjects and degree programs. Failure to alter pedagogy in the face of changing student attitudes and behaviour can be expected to result in a variety of problems in the education process and the ideal education output – a professional and highly skilled graduate.

Case Study : Engineering Computing

The consequences of short term goals can be seen in a recent restructure of first and second year Engineering Computing at the School of Electrical & Computer Engineering (SECE) at RMIT. The first two years of the 4 year degree program are common between Electrical Engineering, Electronic Engineering, Computer Systems Engineering, Network Engineering, Biomedical Engineering, and Communication Engineering. Engineering Computing together with Embedded Systems compromise one subject per semester in the first and second year. Students are first taught to problem solve with Matlab, then to program in C and C++ with an orientation toward embedded systems. The first year and second year subjects in Engineering Computing are where students get their main C and C++ experience.

First year Engineering Computing in 2006 had an excellent set of project and lab activities including a report on current computing, and several assignments that culminated in a GUI based calculator. These looked interesting, exercised problem solving skills, and developed programming skills.

Second year engineering computing was revamped in 2007 and the laboratory program used an auto-tester (Radcliffe 2004) that eliminated the labour of testing and reporting on student's programs. This meant it was economically feasible to run lab tests every 2 weeks for all 160 students under proper test conditions. The results stunned staff and students, a relatively simple program had a pass rate of only 30%, 70% failed! What was wrong? Initially tutors blamed the auto-testing tool, but it proved to work correctly. Next to blame were the students, but with so many students failing that was not tenable. Proper focus groups were then held with the usual provisos-

- All participants will tell the truth.
- Blaming and flaming is banned, stick to the facts and find polite ways to state opinions.
- What is said will not effect any individual after the event (for example admissions of cheating).
- Any notes taken must not be able to identify any individual.

The twenty five involved students responded very well to this process and there was good agreement between the several focus groups held-

- In first year it was possible to plagiarise with little fear of detection.
- Short term goals such as those discussed previously caused many students to fall into the trap of plagiarism in first year and so they had not developed essential programming skills despite being warned they needed these skills for 2nd year and latest years.
- The new auto-test laboratory testing regime used the following year, left no way to hide the lack of skills for those who had not mastered the basics of programming.
- The motivation to become good at engineering programming was weak in a large proportion of the class because many intended to select degree programs with little or no programming in the following year. (This in fact was an interesting misperception as most degree programs require considerable programming in project work.)

The net result of the discovery and consultation process was an immediate overhaul of the teaching program, and considerable work for lecturers and students alike in order to ensure that by the end of semester the learning deficit was overcome.

Reflection

The root cause of the problems just outlined was the mismatch between the nature of student motivation assumed by academics and that which existed in reality. This mismatch resulted in significant plagiarism to the level where basic skills and knowledge were seriously compromised. Given that the aim of a university program is to ensure a good level of skills and knowledge what can be done?

Managed learning: working with students of less motivation is not new in teaching. There are many lessons to be learnt from standard theory and practice (Pintrich & De Groot 1990) such as -

- Provide clear reasons and motivation to master the knowledge or acquiring skills.
- Provide clear, short term goals (meet students' short term horizons with the appropriate teaching strategy).
- Revisit and remind students of key reasons to acquire skills and knowledge.
- Ensure uniform consequences for failure to perform.

How appropriate is this for a university environment? There are good arguments for saying that if students are not motivated then they should not be at university and should simply be failed from the course. It can be argued that degree program graduates must have self motivation and long term goals otherwise they are not worthy of the degree title. A counter argument is that we academics are placing unrealistic expectations on young people trying to start their adult life and earn money to pay education fees or live life as community expectations suggest. A more pragmatic answer is that all universities enrol such students and as educators we are required to do the best we can with the students we have.

In addition, societal expectations of University education have increased and the degree to which Universities rely financially on student fees may motivate some to lower their intake standards.

Accepting the pragmatic point of view requires that academics take a closer look at student motivations and redesign their subjects, effectively providing managed learning in a way not familiar to many universities.

Career focus: it was commented in the focus groups that getting a job seemed a long way away for many first and second year students and this distance negatively effected their motivation. This is an issue which can be directly addressed by academics, for example-

- Typical jobs, salaries and entry requirements can be discussed with the class.
- Job depth on popular job search websites such as: www.seek.com or www.mycareer.com can be displayed on screen.
- Examples of real world projects could be discussed if staff have such experience.

This can be done in a 2-5 minute introduction to a lecture or at any time.

Hopefully students will see that it is in their own self interest to take a longer term view and actively acquire skills and knowledge. From the authors' perspective it was noted that many students were surprised by the range of salaries available to graduates and that technical, professional and leadership skills were key in determining who obtained the better paying jobs.

The New Laboratory Structure

Both first and second year Engineering Computing courses have been reworked with a common structure in the laboratory program-

- Weekly two hour laboratory sessions.
- Odd weeks are help sessions dedicated to getting students ready for the coming tests in the even weeks.
- On even weeks the first hour is a help session and the last hour is a test under exam conditions.

An individual test is administered as follows-

- Well in advance the template of specifications to be met in the test is published and examples practiced in laboratories. Students are encouraged to practice at home.
- Teamwork and peer support is assumed and not discouraged.
- In the actual test session students are issued with the same specifications template to which they must add code using a re-imaged computer with no access to the network, other computers, or other students. At the end of the test the student code is collected and sent to the lecturer.
- The lecturer runs the auto-test program that automatically and thoroughly tests the code with a large variety of test vectors and creates a detailed email response for each individual student listing which specifications passed and which failed. This is sent to each student automatically.
  The testing process also creates a marks file, statistical summary and administration reports.
- Early in the semester students may get a second chance to try the test but with a marks multiplier of 0.7 .

Unlike Wiedemeier (2002) we did not find it necessary to have an individualized tasks, merely a non-trivial task where students were tested in isolation.

This system has a number of practical advantages:

1. Marking is according to an unbiased standard, independent-of-tutor interpretation. Different marking standards are an important concern for many students which causes laboratory classes by tutors who are perceived as lenient to be swamped with students while others are comparatively deserted.
2. The marking load on the tutors and lecturer is vastly reduced leaving much more time for help and tutoring. Previously, without the auto-tester program, tutors spent most of their time marking students with little time left for actually helping students. Tutors were required to straddle two vastly different roles, that of assessor/executioner and that of helper, something which is innately difficult and especially so for beginning tutors.

The Auto Test Tool

The Auto Test tool can be used exclusively by the lecturer or be delegated to the tutors. Before the tool can be used student code must been loaded into a single directory. Next the user must choose a test vector file and hit the "run" button. The image that follows shows an auto-test session which has just completed testing student code and shows summary statistics on the class. The testing is relatively fast, for example applying 40 test vectors to 150 student programs took some 7 minutes on a Pentium 3 class machine Other features of the auto-tester include-

- A simple yet powerful test language for defining test vectors (an expected input plus the expected output).
  A "check vectors" function allows the lecturer to check that the test vector file is correct.
- The ability to quarantine student code so it cannot behave maliciously to the system.
- The ability to timeout and progressively reduce timeout periods for student programs that consistently hang up for successive tests.
- A report file that details all students and the mark they achieved.
- A report file that describes the average class mark by individual test and category of test.
- A detailed email to each student outlining tests that were passed and not passed and how the final mark was arrived at. Such individually tailored feedback would not be possible using purely human means.
- Adjustable marking for each test with may include a negative mark for a failed test.

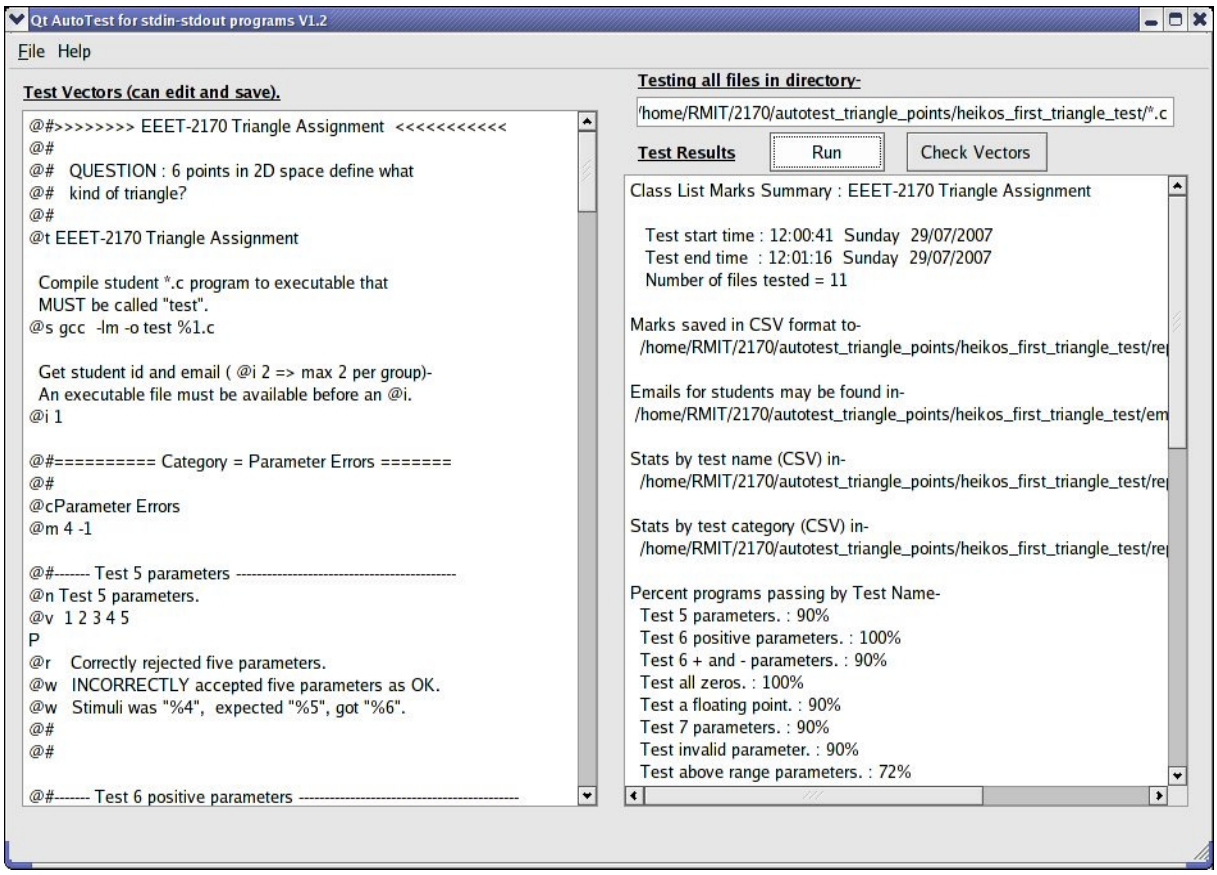

Figure 1: Auto-Tester Report on a Class Test

In Figure 1 the left hand window displays and allows editing of the test vectors. Each line starting with an "@" symbol invokes a command related to the testing. The "@v" command applies a test to the student's program and the next line holds the expected response. The large window on the right shows the results from testing a small tutorial group of 11 students. It firsts states where the individual report files are held and next gives the class statistics for each test vector.

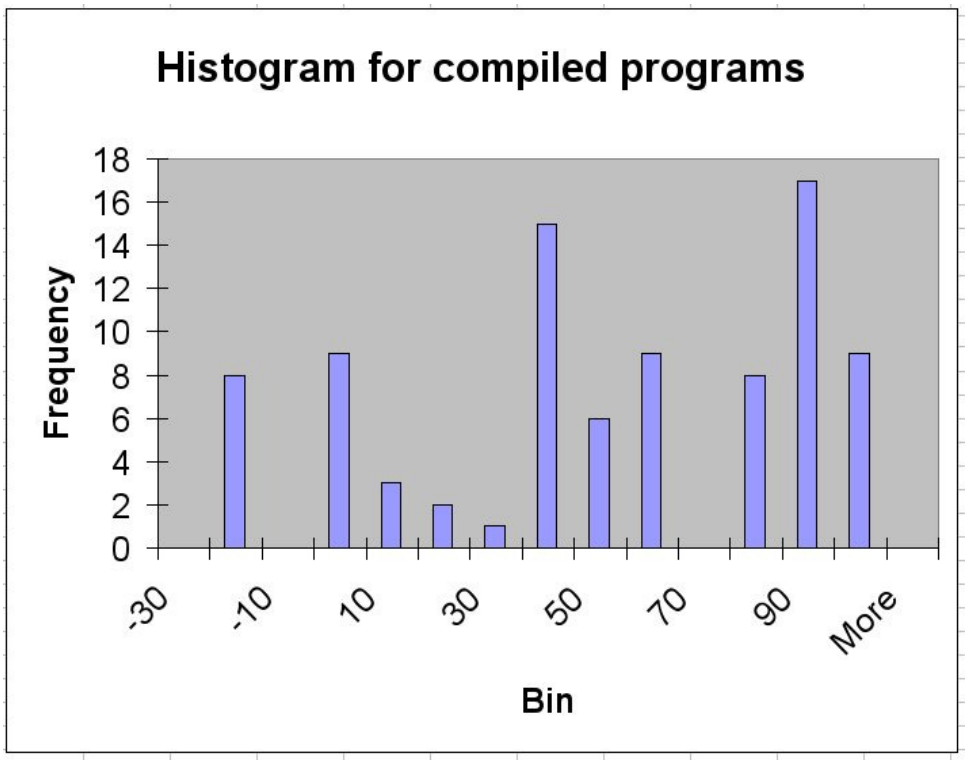

Figure 2: Histogram of First Project

Figure 2 shows the raw marking histogram from the first assignment. Most tests had a mark of +4 for a correct answer and –1 for a wrong answer. The results show the classic 3 hump outcomes – those students who really tried (80%-100%), those who tried a bit (50-70%), and those who have no idea (under 10%). This graph was extremely useful in splitting the class early in the semester to give students the help they needed. In general those with 80% and above did not want or need help. Those on 50%-70% need help in reading the specification and translating that to tests, and some help on programming. Those in the bottom group needed intensive help coming to grips with C++ and just getting a program to compile.

At heart the QtAutoTest tool requires that the student generate a text string that can be compared to an ideal text string. This immediately raises the possibility of use in subjects such as mathematics where a numerical answer is required, or chemistry where a text string such as H2SO4 may be the answer. It would be interesting to pursue these possibilities.

Operation & Student Acceptance

Experience in first and second year has shown problems with student acceptance and general operation that can be easily overcome. Common student complaints have included-

- If I make “small” mistakes then I get a low mark.
- While the program didn’t work at all a lot of the code was right so I should not get a poor mark.
- I can’t remember the syntax and so can’t get anything right.
- It’s not fair to have so many test conditions and make us think of them.
- I memorized the solution and didn’t have to know anything.
- The guide did not give any hint you required this test.
- There is no human to check up on me or ask questions so I can cheat easily.

Overcoming these problems requires careful design of the whole process before the activity and attention to student motivation and expectations.

The following process design issues were found to be important-

- Students may try to hide solutions on the test machines so consider limiting machine access, randomising which machine students use, and re-imaging machines.
- Students may seek help during a test and so network access must be blocked. Standard test conditions such as no mobile phones and no talking must be enforced.
- Ensure there is no mismatch between student and lecturer expectations of the solution. At minimum use standard test traceability by ensuring that every test in the test vector files relates to a request for functionality in the problem template that the students see. It is wise to get other staff or tutors to try the problem or check the test vectors and problem template.
- Ensure the problem is within the capability of students and does not use background knowledge that instructors assume but students do not possess. This is best achieved as above by getting tutors or staff to test the problem, check the template, and check the test vectors.
- Ensure all language demands have been well covered in lectures and one help session.

One of the most important activities is setting student expectations before any test occurs. It was found important to emphasise that a significant part of the test is thinking about test conditions not just writing code. To aid the students coding methodology the Extreme Programming lifecycle (Jeffries 2007) has been introduced, especially the ideas of deciding tests before writing code and a cycle of writing small amounts of code then compiling and testing. Standard engineering programming in industry has been introduced as a driving force in the education process, particularly where "little" problems that cause massive product recalls are not acceptable. An excellent example of these problems is the recent Microsoft X-Box debacle where they have set aside US$1 billion for service in the last quarter of 2007, Taub (2007).
Other matters discussed with the student body included-

- How the auto-tester worked and what type of things would cause it to reject an answer.
- How to move between test mode and delivery mode by using test macro statements.
- How cheating now would lead to serious problems in later tests and the exam.
- How programming skills would be required in courses and careers such as Electrical Engineering that on first glance contain no engineering computing.

Conclusion

Student motivation is a key factor in the acquiring of key skills and knowledge in a university program. Without appropriate skills and knowledge graduates may well have trouble securing a well payed and interesting job in their chosen field, or may be incompetent if they do secure such a position. Both reasoning and experience in the School of Electrical and Computer Engineering (SECE) at RMIT suggest that we cannot rely on student's long term vision and motivation to acquire skills and knowledge and so the education program must be redesigned more along the lines of managed learning. In such an environment the student has little choice but to acquire skills and knowledge as there are solid reasons clearly stated to students, and immediate consequences for students in the short term if they do not participate.

The consequences of a failure to create a managed learning environment can be seen in the experience at SECE, where some 70% of 2nd year students were not able to solve a simple introductory problem under test conditions because in the previous year, and in an otherwise excellent first year course it was possible to plagiarise. In first year, short term motivations had over powered long term goals and pushed students into plagiarism.

Managed learning requires a significant amount of clear and unbiased testing. In the funding starved environment of Australian universities this is a real problem. In many areas of engineering and computer science it is possible to move a large part of the marking burden to automated marking

systems and so radically reduce costs. Automated tools do not solve the problem in themselves. The instructor must carefully plan the whole education process paying particular attention to student motivation and student expectations. If the tool, planning, and setting of student expectations are all successfully implemented then the outcome of a better educational outcome at a lower cost can be achieved.

References

Ashworth, P., Bannister, P., et al. (1997). "Guilty in whose eyes? University students' perceptions of cheating and plagiarism in academic work and assessment." Studies in Higher Education 22(2): 187-203.

Australian Bureau of Statistics (ABS)(2005), *4102.0 - Australian Social Trends, 2005: National and state education and training summary tables,* Accessed at http://www.abs.gov.au/AUSSTATS/abs@.nsf/94713ad445ff1425ca25682000192af2/aee9d847b01c801cca25703b0080ccc0!OpenDocument on 10 August 2007.

Pintrich, P. R. and E. V. De Groot (1990). *Motivational and self-regulated learning components of classroom academic performance.* Journal of Educational Psychology 82(1): 33-40

Radcliffe, Pj (2004), *Automatic Testing of Software in University Courses*, 15th Annual Conference of the AAEE, pg 147-154.

Jeffries, R. (2007) XP Programming, accessed at www.xprogramming.com on 15/8/2007.

Taub E. A. (2007), *Microsoft to spend $1.15 billion for Xbox repairs*, (The New York Times) FRIDAY, JULY 6, 2007, Published: Thursday, 5 July 2007, 21:11 GMT 22:11 UK) Accessed at: http://news.bbc.co.uk/go/pr/fr/-/2/hi/business/6275728.stm on 10 August 2007.

Wiedemeier, P. D. (2002). "Preventing plagiarism in computer literacy courses." The Journal of Computing in Small Colleges 17(4): 154-163.

Acknowledgements

To the many students who do not plagiarise and do extend their skills and knowledge to become competent professionals.

Copyright statement