Peer group testing in software engineering projects

Nicole Clark, School of Computing, University of Tasmania
nclark@utas.edu.au

Introduction

Testing is important if you want to produce a quality product, but generally speaking student programmers have little enthusiasm for testing. Students perform a certain level of testing on any assignment work before submission but this is mainly superficial. There is no denying that testing is a crucial part of the software engineering process and this is why testing experience is a real skill needed by employers.

Over the last six years students in the *Software Engineering Project* course at the University of Tasmania have undertaken projects in teams with four or five members. Each team is collaborating with a different member of the IT industry to produce a unique piece of software. Since the year 2000, the students have conducted peer group testing sessions. The critique that the testers perform helps the development team to identify problems before assessment, hence increasing the quality of the work submitted. The testing sessions are also providing many different, but valuable, benefits such as serving as milestones, increasing learning, and increasing collaboration between teams.

Peer testing is one of the most cost effective ways of detecting defects in work products and improving the final quality of software. Peer reviews can be conducted as formal inspections (Fagan 1999) or informal walkthroughs (Yourdon 1989). The aim of both processes is to use a group of peers and a reading process to detect and locate defects in code. Wahl (2000) identified the importance of teaching students the basic elements of usability testing. Usability testing differs from reviewing the code as the testers are actually using the programs. In more recent times it has been identified that reviews of work products throughout the software lifecycle can be very beneficial. Barbosa et al. (2003) modified an *Analysis and Design* course to see the effects on their students' attitudes to testing by requiring the students to include testing-related practices in all phases of the development process.

Collofello (1987) from Arizona State University stated that it was not enough just to present lectures on reviewing and that it was necessary for the students to participate in a review process. Bailey et al. (2003) from the University of California investigated whether they could measure a change in attitude by the students toward software reviews. They concluded that students emotionally accepted reviews only after practice. Zeller (2000) has implemented an automated system *Praktomat* for managing the submission, test and mutual reviewing of students' programs. 63.5% of students confirmed that having their programs read and reviewed improved the quality of their programs. 61.5% of students felt that reading and reviewing other programs improved the quality of their programs. These findings were also backed by the results of the students.

Recently communication and interaction skills have been identified as being important and as educators we strive to find ways to incorporate teamwork and communication experiences into a course. Hilburn (1996; p.153) found that there were additional learning objectives to participating in a review:

- students get additional practice in reading Z specifications;
- students realise the value of precise, unambiguous and verifiable requirements;
- students get to see and study an alternative solution to the problem they worked on;
- students receive peer evaluation of their work and see the rather dramatic results that such assessment can produce; and

- they get practice in technical communication by articulating inspection results.

Similarly, Sullivan (1994; p.314) stated that 'reviews provide a human-interaction laboratory setting where students: hone teamwork and communication skills, master the peer review process and learn to learn from each other.' Sullivan (1994; p.316) incorporated reciprocal peer reviews in five different courses and made the following interesting observations:

- peer pressure tends to motivate producers to have their work products ready early enough for the reviewers to have time to review them; and
- the act of providing team members with a copy of work products seems to help students learn to share information.

Description of course

The *Software Engineering Project* course at the University of Tasmania provides students with the experience of working in a team and dealing with the associated problems of communication and team management. Many aspects of the development process are considered: requirement extraction; analysis; design; implementation; testing; and documentation. Projects are chosen such that each student can work both cooperatively and independently on parts of the development and experience integrating their work with that of other people. Every project allows the students to learn new technical skills, such as a different development environment, a new programming language, or different software packages. The projects could be in one or more of the following domains: object-oriented programming; virtual reality systems; online content systems; systems administration software; or artificial intelligence systems. The team works on the same project for the entire course.

The course is 25% of a student load for a semester; the course is run over two semesters of 13 weeks. In the first semester they complete release 1 (or a third of the project), in second semester they complete release 2 (the remaining two-thirds). In each semester they spend 6 weeks doing analysis and design and 6 weeks doing implementation, and 1 week doing documentation. The amount of testing corresponds to about 5% of the student's time and is weighted accordingly in the assessment scheme. Project courses at other universities sometimes represent more of a student load or a separate quality course is delivered so that more time can be spent on testing.

Students are asked for feedback on the course, including the testing process. This feedback is provided in a number of ways such as: class discussions during lectures; team exit interviews; one-on-one discussions or emails with the lecturer; and class surveys using Likert scale questions and general comment questions. This feedback has been used to analyse the testing sessions.

Testing in 2000-2002

In 2000 there were 41 students and 9 development teams, by 2002 there were 108 students and 23 development teams. Each project was given a testing team to perform a software review. The testing team consisted of 3 or 4 people from different teams and one person from the development team who acted as the 'Author'. The lecturer formed the testing teams and each team was made up of people who were developing projects similar to the project being tested. The testing was done in week 12 in semester 1 and week 11 in semester 2 – this was one week before it was due for submission. All projects had been integrated by this stage.

To motivate participants to take their review responsibilities seriously, students were informed at the commencement of the session that they would not cause the development team to lose marks by pointing out errors or problems at this stage, and that by doing so they would probably increase the grade given for the quality of software. They were also told they needed to do the best that they could for the team they were testing as there was another team of students doing the same for their project. There was a reciprocal nature to the reviews as there was a lot of overlap between testing teams and

development teams. This established a level playing field and created an atmosphere of egoless teamwork as discussed by Sullivan (1994).

The testing sessions took place in a 2 or 3 hour tutorial. Each project was reviewed for 50 minutes by a testing team. A number of projects were tested twice at each tutorial by different testing teams. By the end of the year each student could have tested between 2-4 projects depending on the year. No person tested the same project twice. Each person was assigned a role within the testing team: Author, Moderator, Recorder or Inspector. Each student had the opportunity of playing a different role at each testing session; for example, different people from the development team acted as the author in each testing team. The lecturer was present during the session to answer any questions about the process and record attendance. Attendance at the reviews was worth 2% of the final grade.

Each review had the following format:

- overview (5 minutes) –the author describes the purpose of the program;
- demonstration (5 minutes) – the author gives a quick demonstration of the program;
- examination (35 minutes) – testers do a combination of usability testing and code reviewing; and
- exit Decision (5 minutes) – discuss handover status of project.

A defect recording log as described in Humphrey (1997) was used to record the defects. The review process closely followed that of an informal walkthrough (Yourdon 1989). During the examination period testers were particularly told to do the following:

- identify problems or suggest changes for the GUI;
- suggest code optimisations or simpler code;
- identify potential uncaught errors in the code; and
- suggest where more code comments were necessary.

The testing session had the following positive learning outcomes.

- The students were enthusiastic about the reviews. There was a definite buzz of excitement in the air. The students had fun.
- Judging by the number of defects on the logs testing teams were typically able to find 80-100 defects per session; though these were not necessarily distinct defects. This demonstrated to the students the importance of testing.
- A large number of problems with the interfaces and a significant number of potential errors caused by user input such as out of range errors were identified and later fixed. This increased the quality of the final product and highlighted to the programmer that they had not adequately tested their own code.
- The session gave them an opportunity to really talk to someone about the issues they had faced, and in some cases were yet to overcome. Just talking to someone who was willing to listen relieved a lot of stress. In some cases the testing team was able to suggest a number of possible solutions to outstanding problems. This reduced the anxiety levels of quite a number of students.
- Having a testing session at the end of release 1 provided an opportunity to share development ideas that could be used while implementing release 2.
- The reviews served as an early milestone for the integration of work; reducing stress the following week when it was due for actual submission.
- The sessions really encouraged the community aspect of the class. The students got to see what many of their class mates were doing. There was an atmosphere of sharing knowledge and ideas.

The testing session had the following negative aspects.

- 50 minutes wasn't long enough to familiarise themselves with an entire product that up till this time they knew nothing about.
- The defect recording logs only pointed out the negative aspects of product. This made the sessions more critical than supportive and therefore stressful. Some authors started to get defensive when there were so many people pointing out defects and began to put the blame on other team mates.

Testing in 2003

Many students in 2002 during the exit interviews asked for more collaboration with other teams. They wanted to have a better idea of what other teams were doing and the approaches they were using. Also at the conclusion of the 2002 testing session in semester 2 a number of students approached the lecturer saying they had received some really good ideas from their testing team, but wished they had received them earlier, particularly ideas of about alternative approaches they could have taken.

For these reasons and to increase the testing experience for the students incremental testing sessions were introduced. In 2003 there are 142 students enrolled in the course; this means there are 32 development teams. The testing teams were formed using a similar process to that of 2000. Each project was reviewed for 50 minutes by a testing team 3 times each semester, approximately every 4 weeks. The testing team was the same for the entire semester, though the testing teams were changed for the second semester. The project manager from the development team acted as the author for the testing team. So each student tested two projects, but they tested each project 3 times. Testing was performed at the end of the analysis, design and implementation phases of each release. The design phase included the development of a number of prototypes.

Testing was worth 5% of the final mark and each student was assessed on the **level** of participation averaged over the three sessions. In first semester the lecturer was present doing an evaluation of the level of participation. In second semester the students performed the evaluation. Each student got 2% for attendance and up to 3% for the level of participation.

Interestingly, Collofello (1987) stated that the instructor should not be present at the reviews as it will encourage serious negative behavioural factors but he doesn't state what they will be. The results of our study have been mixed on this issue. If the lecturer is present but only to answer questions about the process there is no noticeable negative behaviour. When the lecturer is present to evaluate the students, it takes the fun out of the session – but it seemed to have no impact on other behaviours; though it did increase the work load for the lecturer.

Prior to the testing session it is now emphasised that defects in the program does not mean the authors are defective people. This practice was introduced in 2001 to try to minimise the blame aspect experienced in 2000. Testers are also encouraged to point out the positives aspects of the projects they are testing. A defect recording log is still used in each session, but it has changed so that testers can also identify the good things about the product to make the session less critical.

> Sullivan (1994; p.317) noted that:
> The prospect of having work-in-progress evaluated by peers can provoke anxiety. This is especially true when the process is unfamiliar, when the reviewers are virtual strangers, or when the reviewers lack incentive to take their responsibility seriously.

By participating in the testing process three times a semester, the students became familiar with the process and by having the same testing team each time they developed a relationship with the development team. This relationship and the reciprocal nature of the testing teams inspired the students to take the process seriously. All these features combined to reduce the anxiety level.

All the previous negatives from 2000 have now been eliminated and the iterative approach to testing has resulted in the following positive learning outcomes.

- Greater involvement with another project meant that by the end of semester some testers began to feel a sense of ownership in the other product. Often testing team members will do testing outside the assessed tutorial as they begin to feel part of another team. Having the same testing team involved from the beginning of the project allowed them to develop some familiarity with the product and achieve more testing during each session.

- The class began interacting in week 4 as opposed to earlier years where the first collaboration was in week 11 or 12. This has fostered greater collaboration within the class.
- Testing prototypes in particular allowed the testing teams to give the author ideas on how to solve a problem or different approaches that could be taken.
- It wasn't necessary for the lecturer to have knowledge in all the varied technical areas such as programming languages and development software. Also the lecturer lacked time to give feedback and advice on all work products for all projects before submission. The review sessions prevented this being a problem as the students were learning from each other.

Discussion

An increase in the quality of a work product
Many students felt that the work presented for testing was complete. On average a peer group testing session would produce 80-100 suggestions for a product. These results have demonstrated to students that the superficial testing they have been performing is not adequate; if the product had not been tested by the testing teams their clients would have been dissatisfied. The testing sessions are conducted approximately one week before the work is due for submission, giving the teams plenty of time to analyse the feedback and perform necessary corrections before assessment or handover to the client. Also since the students will actually be present while the work is being tested, there is a tendency to take more care to reduce the embarrassment during the session. The testing sessions apply a positive form of peer-pressure on the students, which is beneficial to the quality of their work.

Students work steadily on a work product
Students often need milestones to assist them with time management. In previous courses students have been able to leave assignment work until the week (in some cases the day) it is due. Since the students need to show their work to peers before it is actually due for submission, they are required to begin working on it earlier. The reviews serve as internal project milestones. Even though it is totally up to the teams to decide what they will have ready for a testing session large proportions of work products are implemented and/or integrated 2 weeks ahead of the due date, allowing them time to do some internal team testing before giving it to the testing team.

Increased collaboration between teams
Even by third year there are many individuals who do not know their class mates. While doing the project they get to know their team mates. The testing sessions allow them to meet even more people. The peer group review sessions have fostered a community feeling; the students now communicate more with other teams, taking more interest in what has been achieved by others. Many educators have experienced the 'silent tutorial' where students are too shy to say anything; these review sessions are buzzing. The testing sessions allow the teams to share ideas on similar problems. They have reduced the feeling of isolation some individuals feel when they have a problem that they can not overcome. Since each team is working on a different program there are no concerns about plagiarism. In fact teams are encouraged to share ideas and approaches to save time, just as it would be within a business. Numerous times at the end of a session, an author would approach the lecturer bubbling with enthusiasm saying that a member of the testing time had provided the solution to something that had been plaguing them for some time.

Increased learning
Every project requires the students to learn new technical skills. The students benefit from peer learning by working in teams which allowed them to learn new languages better and faster than they would have by themselves. The peer testing sessions also allowed cross-team learning by the sharing of ideas. Each testing team is made up of 3 to 4 people from other teams, each of those people benefits from working with 3 or 4 other people; so one project actually has a potential source of ideas on approaches from more than 20 people. Students learnt much more from each other than from the lecturer who did not deliver lectures on any technical aspects of the projects (only process). This

collaboration also had the added benefit of reducing the workload for staff because students sought support and advice from their peers.

In a survey conducted in 2002 and 2003 students were asked to register agreement on a Likert scale to the following questions.

- Testing our software in the peer group testing sessions helped us ensure that the software was ready for release.
    - 80% of the students responded positively.
- I found the peer group testing sessions a useful learning experience.
    - 83% of the students responded positively.

Conclusion

Since 2000 students have participated in peer group testing sessions in a software engineering project course. This style of testing gets students enthusiastic about testing, demonstrates the importance of testing, and produces the best results as far as quality of product is concerned. Our experiences with peer group testing have led us to make changes to enhance the learning experience for the students and we have had the following positives consequences:

- an increase in quality of the final product;
- students work steadily on a work product;
- increased collaboration between teams; and
- increased learning.

Interaction skills are important and as educators we strive to find ways to incorporate teamwork and communication experiences into a course. The peer group testing sessions have lead to increased learning for each individual in the areas of technical skills and communication skills.

Acknowledgements

I would like to acknowledge the contribution of the students who have worked on projects from 1998 to 2003. I would also like to acknowledge the contribution made by the clients involved.

References

Bailey, D., Conn, T., Hanks, B. and Werner, L. (2003) Can we influence students' attitudes about inspections? Can we measure a change in attitude? Madrid, Spain: 16th Conference on Software Engineering Education and Training, 260-267.

Barbosa, E. F., Maldonado, J. C., LeBlanc, R. and Guzdial, M. (2003) Introducing testing practices into objects and design course. Madrid, Spain: 16th Conference on Software Engineering Education and Training, 279-286.

Collofello, J. S. (1987) Teaching technical reviews in A One-Semester Software Engineering Course. In *Proceedings of the eighteenth SIGCSE technical symposium on computer science education*, **19**(1), 222-227.

Fagan, M. E. (1999) Design and code inspections to reduce errors in program development. *IBM System Journal*, **38**(2&3), 258-287.

Hilburn, T. B. (1996) Inspections of formal specifications. In *Proceedings of the twenty-sixth SIGCSE technical symposium on computer science education*, **28**(1), 150-154.

Humphrey, W. S. (1997) *Introduction to Personal Software Process; 4th Edition*. Addison Wesley.

Sullivan, S. L. (1994) Reciprocal peer reviews. In *Proceedings of the twenty-fifth SIGCSE symposium on computer science education,* **26**(1), 314-318.

Wahl, N. J. (2000) Student run usability testing. In *Thirteenth Conference on Software Engineering Education & Training*. Austin, Texas, 123-130.

Yourdon, E. (1989) *Structured Walkthrough, 4th Edition*. Prentice Hall.

Zeller, A. (2000) Making students read and review code. In *Proceedings of the 5th annual SIGCSE/SIGCUE ITiCSE conference on innovation and technology in computer science education,* Helsinki, Finland, **2**(3), 89-92.