Peer Testing in Software Engineering Projects

Nicole Clark
School of Computing
University of Tasmania
GPO Box 252-100, Tasmania, 7001, Australia
nclark@utas.edu.au

Abstract

For the last six years, students in the Software Engineering Project course at the University of Tasmania have undertaken projects in teams of four or five members. Since 1998 peer testing sessions have been conducted in two different formats: paired peer code reviews and group peer inspections. The critique that the testers perform can help the development team to identify problems before assessment hence increasing the quality of the work submitted. The peer testing sessions also provide many different, but valuable, benefits such as serving as milestones, increasing learning, and increasing collaboration between students in different teams.

Keywords: Software Engineering, Education, Collaboration, Testing.

1 Introduction

Successful software engineering is not just about producing a working piece of software. The software needs to be of sufficient quality that the client will actually use it. Zeller (2000) states that there are two best practices known to improve quality: testing and reviewing. Bailey et al (2003, page 260) states:

"*Our role as university educators is to teach these best practices and change attitudes so that our students graduate as software engineers who believe in the use of these methodologies".*

Generally speaking student programmers have little enthusiasm for testing. Since 1998 we have investigated what style of testing enthuses students towards testing, demonstrates the importance of testing, and produces the best results as far as quality of product is concerned. This paper documents our experiences with peer testing during a software engineering project course.

Software Engineering Project at the University of Tasmania is a third year capstone project course undertaken by students who have nearly completed their Bachelor degree. The aim of the course is to provide students with experience developing a medium-sized computing project in a small team of four or five students. Each team is collaborating with a different client to produce a unique piece of software.

Since 1998 peer testing sessions have been conducted in two different formats as part of this course:

1. Paired peer code reviews – each student is partnered with a student from another team and they review each others code;
2. Group peer inspections – each project is given a testing team of students from other projects. These students inspect the software and other work products for defects.

Our experience with each format of peer testing has led us to make changes to enhance the learning experience for the students. Peer testing has had the following positive consequences:

- Students have seen the necessity of testing – the number of defects detected during a testing session convinced most students of the benefits of testing;
- Quality of a work product has increased – the testing sessions are conducted prior to submission of a work product giving the development teams time to make the necessary changes to improve quality;
- Students now work steadily on a work product – the students prepare items for the testing session preventing them from leaving it until the week or day it is due for submission;
- Community feeling has been created within the class – the testing sessions allow students to interact with their class mates increasing their interest in what has been achieved by others;
- Collaboration has increased between teams – the sessions allow teams to exchange ideas on similar problems or approaches for development;
- Student learning has increased – students learn from each other, not just the lecturer, particularly in the area of technical skills such as programming languages or software development tools.

This paper begins with a description of related work. Next, it describes the structure of the Software Engineering Project course. This is followed by an examination of the peer testing sessions conducted during the last six years justifying the above claims. The paper concludes with an elaboration of each of the benefits experienced from performing peer testing.

2 Related Work

Peer reviews are a cost effective way of detecting defects in work products and improving the final quality of software (Fagan 1999). 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. Over the years much research has been undertaken into the processes for the reviews (Humphrey 1997), the roles for people conducting a review and the behavioural factors involved in a review process (Weinburg 1984).

Zeller (2000) has implemented Praktomat an automated system 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.

Wahl (2000) identified the importance of teaching students the basic elements of usability testing by incorporating it into a project course. Usability testing differs from reviewing the code as the testers are actually using the programs.

Collofello (1987) outlines a practical approach for teaching about software reviews in a one-semester software engineering course. He stated that it was not enough to just present lectures and that it was necessary for the students to participate in a review process. Bailey et al (2003) investigated whether a change in attitude by the students toward software inspections could be measured. They concluded that students emotionally accepted inspections only after practice.

Reviews of work products throughout the software lifecycle can be very beneficial. Collofello (1987) identified a project course as being one course where reviews could be undertaken and listed four places where a review could be put into the lifecycle. Hilburn (1996) describes a course where a team of students develops a formal specification of the requirements for a software system. These specifications are inspected by another team of student. 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.

Recently communication and interaction skills have been identified as being important for software engineers and as educators we strive to find ways to incorporate teamwork and communication experiences into a course. Hilburn (1996, page 153) found that there were additional learning objectives to participating in an inspection:

- "*Students get additional practice in reading Z specifications;*
- *While inspecting the 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;*
- *They get practice in technical communication by articulating inspection results.*"

Similarly, Sullivan (1994, page 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 incorporated reciprocal peer reviews in five different courses. Sullivan (1994, page 316) 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;*
- *The act of providing team members with a copy of work products seems to help students learn to share information.*"

3 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: problem specification, requirement extraction, system design, implementation, integration, testing, and documentation.

Projects are selected so that each student has the opportunity to work both cooperatively and independently on the project. 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. Each project requires the student to undertake significant personal learning, for example many projects require the students to teach themselves a previously unused programming language.

The course is 25% of a student load for a semester. Since 2000 the course has been run over two 13 week semesters; prior to that it was a one semester course. Each student is required to work for eight hours each week on the project. This means the student can spend in excess of 200 hours working on their project over the 26-week course. Judging from the timesheets which the students are required to submit – 95% of the students spend more than that. The amount of testing corresponds to about 5% of the student's time. Project courses available at other universities sometimes represent more of a student load or alternatively a separate quality course is run in conjunction with the project course so that more time can be spent on testing.

In the first semester students complete release one (or a third of the project), in second semester they complete release two (the remaining two-thirds). As the students receive a good grounding of the process and spend a lot of time learning the development tools in first semester, they are able to achieve twice as much in the second semester. The benefit of giving them a second iteration is that they gain confidence and familiarity with the process. They have greater knowledge of the domain and are also able to tackle more technically challenging aspects of the project (rather than concentrating on process and programming language). In each semester they spend approximately six weeks analysing and designing and six weeks implementing and testing, and one week doing documentation and handover.

During each semester students are asked for feedback on various aspects of the course, including the testing process. This feedback is provided in a number of ways:

- A class discussion during the final lecture
- Team exit interview in the final week of the semester
- One-on-one discussions with the lecturer
- Emails received during the course and after
- Class surveys using Likert scale questions and general comment questions
- Centrally controlled Student Evaluation of Teaching and Learning surveys

4 Testing Sessions

Since 1998 peer testing sessions have been conducted in two different formats:

- Paired peer code reviews
- Group peer inspections

The peer testing sessions are conducted during a tutorial. Prior to 2002 this was a two hour tutorial, since then it has been a three hour tutorial. The lecturer is present for the entire tutorial to answer questions about the process and record attendance. In the first semester of 2003 the lecturer also performed evaluation of each student's level of participation (discussed further later).

Interestingly, Collofello (1987) stated that the instructor should not be present at the sessions since 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 some of the fun out of the session (particularly for the lecturer) – but it appeared to have no impact on other behaviours.

The remainder of this section is a description of the processes and our positive and negative experiences with peer testing during the last six years. Section 5 provides a summary of our positive experiences.

4.1 Testing in 1998

Facts on the 1998 class
13 week course
31 students
6 teams, 4-7 members
Two teams had external clients
All Java language programs
All object oriented style of development

Although teams were encouraged to test their own projects, the assessed requirement for testing was participation in a peer code review session. Each student was partnered with a student from a different team. The lecturer formed the partnerships and where possible, the partners were doing similar work in different projects.

To motivate participants to take their review responsibilities seriously, students were informed that they would not cause the code author to lose marks by pointing out defects at this stage, and that by pointing out the defects now, they would probably increase the grade given for the quality of software.

The review process closely followed that of a walkthrough (Yourdon 1989). Apart from reading and understanding the code the testers were asked to:

- Identify problems in the GUI;
- Suggest changes for the GUI;
- Suggest code optimisations;
- Identify potential uncaught errors in the code;
- Suggest simpler code;
- Suggest where more code comments were necessary.

The testing was performed in a tutorial, with 50 minutes spent on each person's code. The tutorial was held in week 10, three weeks before the final product was due to be finished. Failure to participate in the testing session meant a loss of two percent to their final grade.

After overcoming everyone's initial apprehension of showing their code to someone else for criticism, students were enthusiastic about showing someone what they had achieved. The paired code review had the following positive outcomes:

- A large number of problems with the interfaces were identified and later fixed, hence increasing the quality of the final product.
- A significant number of potential errors caused by user input such as out of range errors were found by the testers. This highlighted to the programmer that they had not adequately tested their own code.
- The students were forced to explain their code, and realised the importance of code comments.
- 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. Simply talking to someone who was willing to listen relieved a lot of stress.

- Because they were partnered with someone working in a similar area, their testing partner was able to suggest a number of possible solutions to outstanding problems. This reduced the anxiety levels of quite a number of students.

The session had the following problems:

- Since testers were only being shown a small part of an entire application it was not easy for the tester to put it into context.
- Some students had produced very little individual code and therefore had very little to show their partner.
- Due to the timing of the testing session most projects were not yet integrated so many changes were made to the code after the review.
- A few students felt the session was of little benefit for them, although they were willing to admit it was of benefit to either their partner or other people in their development team.

4.2 Testing in 1999

Facts on the 1999 class
13 week course
34 students
8 teams, 4-6 members
All external clients
Many different programming languages
All object oriented style of development, except one.

This year the peer code review was held in week 11, two weeks before it was due. All projects had been integrated by this stage. The students were told to give a short demonstration of the entire program, but to then concentrate on their personal section. This enabled them to put their contribution in context. They then showed their code to their partner. Again failure to attend the review session meant a loss of two percent.

All the positives from 1998 were still present, but there was a new problem. This year we introduced external clients and therefore there were many different programming languages used (not all Java as in 1998). It was difficult to pair students with someone doing similar work since the projects were now so varied and in differing programming languages, some of which had not been previously taught to the students. It was difficult for the tester to come to grips with a new language in one hour and to provide meaningful suggestions.

4.3 Testing in 2000

Facts on the 2000 class
26 week course
41 students
9 teams, 4-5 members
All external clients
Many different programming languages
Three virtual reality systems, remainder object oriented style of development.

To overcome the problem of pairing students with a person working in similar areas, each project was given a testing team to perform a project inspection rather than merely reviewing code. The testing team consisted of three or four people from different teams and one person from the development team who acted as the ‘author’. Each testing team was made up of people who were developing projects similar to the project being tested.

Again students were told at the commencement of the session that they would not cause teams to lose marks by pointing out defects. They were also told they should 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 because 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).

Each inspection 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 complete a combination of usability testing and code reviewing
- Review (5 minutes) – discussion of the handover status of the project to formulate an exit decision

Each person was assigned a role within the testing team: author, moderator, recorder or inspector. As suggested by Collofello (1987), each student had the opportunity to play a different role at each testing session. A defect recording log as described in Humphrey (1997) was used to record the defects.

The inspections were performed in week 12 of semester one and week 11 in semester two. Each project was inspected once by a testing team in first semester. In second semester each student participated in two testing teams, six projects were inspected twice. Different people from the development team acted as the author in each testing team. Thus by the end of the year each student had tested three projects (possibly once acting as the author for their own project). No person tested the same project twice. Attendance at the three inspections was worth 2% of the final grade.

The group peer inspection had the following positive outcomes:

- The students had fun which made them enthusiastic about the inspections. There was a definite buzz of excitement in the air as a result of the thrill of having other people really use their software and from nervous excitement from having people trying to find fault with it.
- The testing teams were able to find a lot of defects, typically 80-100 defects per program; these were not necessarily distinct defects. The number of defects identified demonstrated to the students the importance of testing.

- Having a testing session at the end of semester one provided an opportunity to share development ideas that could be used during semester two.
- The inspections served as an early milestone for the integration of work reducing stress the following week when it was due for actual submission.

The sessions had the following problems:

- 50 minutes wasn't long enough for the students to familiarise themselves with an entire product that 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 for the students.
- 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.

4.4 Testing in 2001

Facts on the 2001 class
26 week course
24 students - reduction due to a change to the structure of another degree.
6 teams, 4-5 members
All external clients
Many different programming languages

Group peer project inspections were continued and paired peer code reviews were re-introduced in first semester. The peer code review was held the week before the peer inspections.

Each student participated in two inspections each semester; four projects were inspected twice each semester. Different people from the development team acted as the author in each testing team. By the end of the year each student had participated in one peer code review and had inspected four projects (possibly once acting as the author for their own project). No person tested the same project twice.

Testing was worth 5% of the final grade: 1.5% code review, 1.5% first inspection, and 2% second semester inspection. Assessment was based purely on attendance.

Prior to the testing session it was emphasised to the testers that pointing out defects would benefit the development team. It was equally emphasised to the 'authors' that they shouldn't take the process personally – testers were not pointing out defects in them but rather defects in the program. This was to try to minimise the blame aspect from the previous year. Testers were also encouraged to identify the positives aspects of the projects they were testing.

There were no new negatives this year, and all of the previous negatives were eliminated, except that there still wasn't enough time in the inspection session. Having both a code review and project inspection had the following new positive outcomes:

- The attitude of the students to the code review was exceedingly professional. Even though it was still a negative process (identifying defects) it was taken positively as students could see how this information would benefit them. Testers were able to make about 10-15 suggestions to their partner.
- The code review served as a milestone for each individual to complete their work. This caused less stress the following week when teams were integrating work for the inspection.
- The review and the inspections 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.

4.5 Testing in 2002

Facts on the 2002 class
Two 13 week units
108 students – increase due to change in structure of degree
23 teams, 4-5 members
All external clients except one
Many different programming languages

There were now too many students to hold the paired code reviews, but the group peer inspections were continued. Testing teams were formed in a similar manner to 2000.

Due to the sheer volume of students each project was inspected once for 50 minutes each semester. Each student only tested one project a semester, but not the same project. The project manager from the development team acted as the author for the testing team. Attendance at each testing session was worth 1% of the mark given in each semester.

All the positives from previous inspections were still present, but there were two new negatives:

- There was not enough emphasis on the importance of testing. Each student only performed two hours of assessed testing. Judging by the quality of the products they did spend considerably more time testing their own products.
- The suggested alternative approaches to the development processes were given too late to be really useful, particularly the suggestions given in week 11 of semester two.

In a survey conducted at the end of the year (to which 100% of students responded) 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.*
    - 84% of the class responded positively.
- *I found the peer group testing sessions a useful learning experience.*
    - 83% of the class responded positively.

4.6 Testing in 2003

Facts on the 2003 class
Two 13 week units
135 students
31 teams, 4-5 members
All external clients
Many different programming languages

Many students asked for more collaboration with other teams during the exit interviews in 2002. 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 two 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 for alternative approaches they could have taken.

For these reasons, and to increase the testing experience for the students, incremental testing sessions were introduced. Each project was inspected for 50 minutes by a testing team three times each semester, approximately every four weeks. The testing team was the same for the entire semester, although the testing teams were changed for the second semester.

Testing was performed at the end of analysis, design and implementation phases of each release. The design phase included the development of a number of prototypes. As Collofello (1987) noted the materials for inspection should be both code and other intermediate products to produce the best results. Different types of defect recording logs were used in each session to reflect the different types of work product. The logs were also altered to enable testers to identify the good things about a work product to make the sessions less critical.

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

To motivate participants to take their responsibilities seriously the students were told that they were being assessed by the lecturer who was present and that the lecturer would be reading the comments they wrote on the defect recording logs. The presence of the lecturer doing assessment in the first semester seemed to take some of the fun out of the session (it also made it a lot more work for the lecturer). In the second semester students assessed each other. Each tester evaluated the project manager (the 'author') as follows:

- Was the project manager on time for the session?
- Was the project manager organised?
- Did the project manager give an overview of the project?
- Did the project manager ask your opinion during the examination period?
- Was the project manager able to answer any questions you asked?
- Did the project manager conduct the review at the end of the session?

Each project manager evaluated each of their testers as follows:

- Was the tester on time for the session?
- Did the tester listen to your overview of the project?
- Did the tester give you verbal suggestions during the examination period?
- How helpful were the comments made by the tester?
- Did the tester dominate your time during the examination or review preventing you from talking to other testers?
- Did the tester write down any defects on printouts or logs?
- Did the tester write down any helpful suggestions on printouts or logs?

Sullivan (1994, page 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 testing teams to take the process seriously. All these features combined to reduce the anxiety level.

All the previous negatives have now been eliminated and the incremental approach to testing has resulted in the following positive learning outcomes:

- Having greater involvement with another project, meant some testers began to feel a sense of ownership in the other product by the end of a semester.
- Some testers actually do testing for the other team outside the assessed tutorial as they now 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 four as opposed to week 11 or 12 in earlier years. This has fostered greater collaboration within the class.
- Testing prototypes gave students ideas about how to solve a problem or different approaches that could be taken much earlier.

- It wasn't necessary for the lecturer to have knowledge in all the varied technical areas such as programming languages and development software.
- The lecturer lacked time to give feedback and advice on all work products for all projects before submission. The inspection sessions prevented this from being a problem since the students were learning from each other.

In the survey conducted at the end of release one (to which 96% of students responded) 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.*
    - 76% responded positively.
- *I found the peer group testing sessions a useful learning experience.*
    - 84% responded positively.
- *Testing sessions improved the quality of our deliverables.*
    - 83% responded positively.
- *Preparation for the peer group testing session served as a useful deadline for integration of documents and software.*
    - 88% responded positively.
- *I gained some useful knowledge for my own project by testing another project.*
    - 67% responded positively.
- *Testing the same project at each session increased my familiarity with the product allowing me to find more defects.*
    - 84% responded positively.

These numbers, and those from 2002, are significantly higher than those reported by Zeller (2000). In particular 83% felt that the testing sessions improved the quality of the work products. 88% of students appreciated the internal milestone provided by a testing session. 67% of students felt they gained useful knowledge for their own project by testing another project. This reduced the number of questions that they needed to ask the lecturer; students learning from students.

Students were also asked if they would like to continue to test the same project in second semester; 44% stated that they would, indicating that a large percentage of the class felt ownership of another project, or had established a relationship with other people. Practicalities, such as timetable and testing team size, meant that only 25% of students continued to test the same project.

5 Benefits of peer testing

To summarise, peer testing has had the following positive consequences:

- **Students have seen the necessity of testing**. Many students felt that the work presented for testing was complete. On average a paired code review would produce at least 10-15 suggestions for each person, a group peer inspection 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.
- **An increase in quality in a work product.** 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 anonymity of having it assessed while you are not around has been removed. 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 much earlier. The testing sessions serve as internal project milestones. Even though it is totally up to the individuals or teams to decide what they will have ready for a testing session large proportions of work products are implemented and/or integrated two weeks ahead of the due date, allowing them time to do some internal team testing before giving it to the testing team. The testing sessions provide an opportunity for all students to see the progress of other teams and compare their own progress.
- **Created a community feeling within the class.** 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 group peer inspection sessions, in particular, 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. The inspection sessions force the students to talk to each other.
- **Increased collaboration between teams**. The testing sessions allow the teams to exchange ideas on similar problems. In some cases this collaboration has led teams to overcome what seemed to be insurmountable hurdles. The testing sessions have reduced the feeling of isolation some individuals feel

when they have a problem that they can not solve. 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 team had just provided the solution to something that had been plaguing them for a long 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 faster and better 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 three to four people from other teams, each of those people benefits from working with three or four other people. So one project actually has a potential source of ideas on approaches from around 20 people. In the words of Sullivan (1994) "*The cumulative life experiences of the students are shared*". 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.

6 Conclusion

Since 1998 students have participated in peer testing sessions in our software engineering project course. Our experience with both paired peer code reviews and group peer inspections was extremely positive. The peer style of testing gets students enthusiastic about testing, demonstrates the importance of testing, and helps teams produce a quality product. Peer testing has provided the following positive learning outcomes:

- Students have seen the necessity of testing;
- An increase in quality in a work product;
- Students work steadily on a work product;
- Created a community feeling within the class;
- Increased collaboration between teams;
- Increased learning.

Interaction skills are important and as educators we strive to find ways to incorporate teamwork and communication experiences into a course. Peer testing has enhanced the learning experience for the students, particularly in the areas of technical skills and communication skills.

7 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 and supervisors involved.

8 References

Bailey D., Conn T., Hanks B., Werner L. (2003): Can We Influence Students' Attitudes About Inspections? Can We Measure a Change in Attitude?, *16th Conference on Software Engineering Education and Training,* Madrid, Spain, 260-267

Barbosa E. F., Maldonado J. C., LeBlanc R., and Guzdial M. (2003): Introducing Testing Practices into Objects and Design Course, *16th Conference on Software Engineering Education and Training*, Madrid, Spain, 279-286

Collofello J.S. (1987): Teaching Technical Reviews in a One-Semester Software Engineering Course, *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, *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*, Addison Wesley, 4th Edition

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

Wahl N.J., (2000): Student Run Usability Testing, *Thirteenth Conference on Software Engineering Education & Training*, Austin, Texas, 123-130

Weinberg G.M., Freedmand D.P. (1984): Reviews, Walkthroughs and Inspections, *IEEE Transactions on Software Engineering*, SE-10(1): 68-72

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

Zeller A. (2000): Making Students Read and Review Code. *Proceedings of the 5th annual SIGCSE/SIGCUE ITiCSE conference on Innovation and technology in computer science education*, 2(3), Helsinki, Finland, 89-92