Assessment in Software Engineering – Towards a new Framework for Group Projects

Marie Devlin[1], Chris Phillips[2], Lindsay Marshall[3]

[1~3]School of Computing Science, Newcastle University, UK

marie.devlin@ncl.ac.uk[1]

Abstract

In this paper we track the changes made over 4 years to the amount and type
of assessment in a Computing Science Software Engineering module based solely on group-work. Most of the changes have been made as part of our CETL project: Active Learning in Computing [1], which has attempted to introduce larger, more meaningful assessments to the module in response to feedback from the students, employers and staff involved. Our approach has not always been wholly successful in terms of reducing the student workload or in creating assessments that can differentiate an individual's learning outcomes clearly. However, our experiences have led us to recognise the need for an alternative approach to assessment in these types of projects. We review the assessment methods and types we used prior to the CETL project and those we use now and outline the experiences and feedback that prompted the changes we made. We evaluate the impact of these changes on students' learning outcomes and then describe our work towards an alternative assessment framework for group-projects in Software Engineering that aims to help staff recognise and measure student achievement more clearly and to help students get the most from their group
learning experiences.

Introduction

Since 2005, changes made to the Software Engineering module in the School of Computing Science, Newcastle University by the Active Learning in Computing initiative [AliC] have included the incorporation of cross-site team-work. Cross-site working is increasingly becoming more commonplace in real-world software development [2] and we have found that its introduction really stretches students' learning by making them more aware of the necessity for good communication practices and professional working attitudes. However, our approach has not always been wholly successful in terms of reducing student workload or in creating assessments that can clearly differentiate an individual's learning outcomes from the module. Our experiences and those of our students have led us to recognise the need for an alternative assessment framework for group-projects in Software Engineering that can help staff recognise and measure student achievement more clearly and to help students get the most from their group-learning experiences. In sections one and two of the paper we review our assessment practice before and since the introduction of cross-site working by ALiC and give an overview of our motivations for change, including student experiences and module feedback. In section 3 of the paper we evaluate the impact these changes have on students' learning outcomes and finally, in section 4, we outline our current work towards an alternative assessment framework for similar group projects in Software Engineering.

1. The Software Engineering Team Project

Since 2005, Active Learning in Computing partners Newcastle and Durham University (ALiC), part of the UK CETL initiative [3]) have introduced a collaborative learning model of Software Engineering to level 2 students that reflects current industry practice by being cross-site in nature and focuses on the development of technical and transferable skills. Teams are formed at Newcastle and paired with a corresponding team at Durham. Usually the major project task is the design and implementation a large software system – (e.g. in 2005 the task was a tour guide application that could be loaded onto a PDA or mobile phone). Students work together as a virtual enterprise across the sites. We provide video conferencing facilities and access to instant messaging and email addresses for teams. In order to share code and documentation the teams are also provided with Subversion repositories and online docu-

ment repositories in Newcastle's VLE NESS [4,5]. Skills outcomes for the module are currently listed at Newcastle as initiative, adaptability, teamwork, numeracy, problem-solving, interpersonal communication, written communication and oral presentation and the assessment scheme is formulated around measuring student development of these skills during the module [6].

2. Assessing Software Engineering Teams

Teamwork is necessary in higher education programs because it teaches students about working together, prepares them for the realities of the working world and allows them to develop skills in leadership, time management, negotiation and communication. However, assessing teamwork has always been a difficult task. Many higher education practitioners struggle to derive an individual's mark or provide visibility to the students as to how the individual marks are derived [7].

2.1 Assessment Before 2005

Originally, before ALiC became involved at Newcastle there were approximately separate 26 coursework assessments associated with the team project throughout the academic year. These coursework elements were to be completed on an individual or a team basis and consisted of a variety of products that were assessed, including a personal skills assessment, code and documentation, product demonstrations and team presentations. One reason for giving so many assignments during the module was to ensure that students kept working continuously throughout the whole academic year thus maintaining momentum of effort and interest in the project. Another was to ensure that the project was of substantial size to challenge a team of students and to force them to collaborate. The pedagogic approach used in the module design was essentially problem-based learning[8] because students had to make decisions on team roles, team management structure, task allocation and project management strategy. This went a long way to giving the students greater autonomy in determining the pace and style of their learning experience.
In terms of support and guidance each team was provided a staff monitor who observed their team interactions during a formal weekly meeting for assessment of the team working process. The presence of a monitor was to ensure that individual contributions to the team project could be judged objectively based on behaviour in meetings, actions taken and work produced by each individual in the team on a weekly basis. Figure 1 illustrates the types of assessments students completed.

Figure 1: Assessments pre 2005

Assessment	Type	Assessment	Type	Assessment	Type
Strengths Essay	I	Presentation	T	User Manual	T
Review	I	Contract	T	Documentation	T
Design	I	Criteria	T	Software	T
Structure	I	Trading Results	T	Percentages	T
Group Structure	T	Testing Strategy	T	Presentation	T
Contract	T	Project Plan	T	Interim Report	T
Draft Documentation	T	Testing Strategy	T	Peer Assessment	T
Sales Plan	T	Report	I	Software Demo	T
Flyer	T	Log Book	I		

Individual coursework (I) was mainly undertaken at the start of the module in preparation for team-working. Students were asked to review their skills using a cut down version of Belbin Team Roles [9] and to write an essay on their strengths and weaknesses. This assessment was designed to help them recognise their strengths and weaknesses and the skills they already possessed and could bring to the project. In terms of discipline specific knowledge students were introduced to the stages of the software engineering process i.e. requirements analysis, design, implementation, testing and maintenance and evolution. Prior to 2005, assessment largely focused on tangible evidence of team work – the work products. Decisions were made about individual performance and participation in the team process largely based on the judgement of the academic monitor who gave each team member an individual efficiency mark

at the end of the project, and by peers who derived marks using two instances of peer assessment during the project. At this time, reflection by the students on what they had learned and experienced during the project was mostly left to the end after all the work had been done. A large proportion of reflection and individual marks relied on the individual report and individual elements of coursework submitted at the start of the project. This meant that a student could pass the module mainly based on individual work and did not have to rely heavily on their team working mark. The impact of this was that some individuals had less incentive to perform well and participate fully in team work during the project, relying on their individual work to get them through and those that did participate had more anxiety over how individual marks were awarded. If 'slackers' could get good marks and could pull a team mark down by not working, it was not fair on the other team members who were performing well.

2.2 Assessment Practice 2005-Present

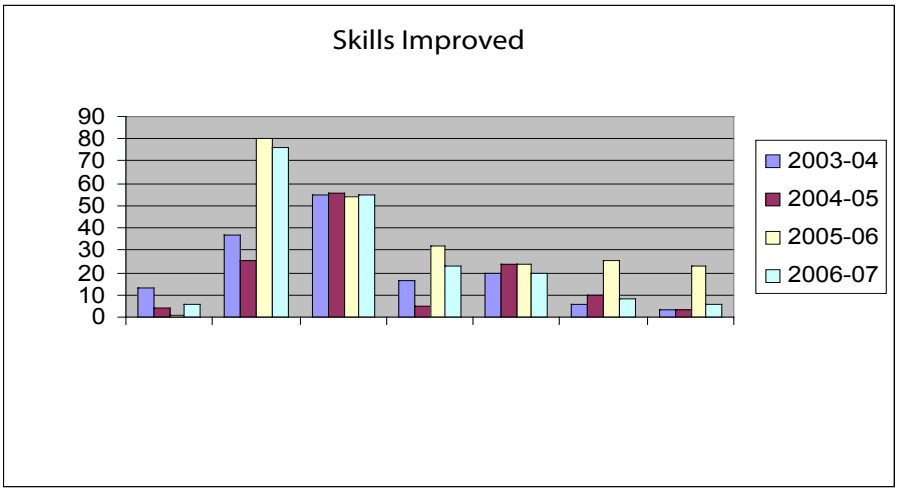
The introduction of cross-site team working with another university and reliance on an off-site team made students even more anxious about how marks would be awarded. CETL ALiC retained the use of many of the modes of assessment for the team project that were already in place and introduced new assessment practices to ensure fairness to students. We reduced the number of deliverables to 15 during the project by making some products more substantial and getting rid of others in order to try to avoid over-assessing the same skills repeatedly during the project e.g. presentation skills were originally assessed 3 times during the year and we reduced this to one assessment. We introduced the use of formative assessment [10] for the larger technical deliverables such as the requirements and design documents and allowed students to get feedback on a prototype system before their final code submission. This allowed students to improve and make changes to their work before it was marked and aimed to improve their awareness of how they were progressing. Formative assessment encouraged students to reflect on what had been achieved and helped them learn what improvements were needed during the project, rather than at the end. We also managed to get real employers involved in the module. They gave our students industrial-strength case-studies and problems to work on and also participated in the formative feedback on their work products. This aspect gave the project tasks authenticity and lets students see more clearly what the assessment criteria meant in terms of professional software engineering. Students were also provided with more information on the weight and importance of peer assessment and what was being measured or looked for. We gave more weight to the team working aspects of the project – both the deliverable products and the processes involved. We kept the peer assessment processes but also introduced the use of a contribution matrix [11] for all team deliverables that illustrated exactly what each individual had contributed to each product. Issues of non-contribution or poor contribution were dealt with largely by comparison of contribution matrices for teamwork effort and the use of an escalation procedure where non contributing students were reported for 'non-progression' to their degree program director if they were pulling the team down during the project. Offending students had to agree to improvements in future conduct. All contribution matrices had to be agreed by every team member before submission.

3. Impact of Changes on Learning Outcomes

In order to determine the effects of our assessment changes on the learning outcomes of students we conducted a comparative analysis of 324 final individual reports and approximately 40 team reports that had been completed by students from 2003-2007. The structure and instructions for the reports remained the same during all this time and therefore we were able to compare sections and responses across all the years of the study. We focused on recording the skills students said they had learned from the project over the years to see if there was a change in their awareness of the importance of skills and of monitoring their own learning development during the project. We derived their perception of skill and learning outcomes from what they reported. Figure 2 shows that students consistently discussed 7 main areas of improvement during the module, only two of which, analysis and design/code, correspond with stages in the software engineering process. The other 5 skills noted by students correspond with team and 'academic' skills that are part of the module learning outcomes we specified. Over the years 2005 – 2007 we noted positive changes in that a greater proportion of students described their learning in terms of skills learned, developed and practiced. However, we are still unsure if students have actually developed the skills they describe or have just become more clever at writing reports! A learning outcome we would like from this module is that students be able to articulate and assess their own their skill levels more accurately at the end so that they can focus on their future

learning and development needs for the rest of their programme and indeed for the rest of their career. We would also like to make it clearer to students how an individual's final marks for the module are derived and to make the criteria used for each assessment more transparent. Assigning marks to students [12] means that student achievement "is abstracted into just a few numbers" so it is difficult for a student to perceive what skills they have learned and how their skills have developed during the project based on a mark and feedback on lots of separate elements of coursework. Their level of proficiency is decided based on the quality of teamwork products, team presentations, peer assessments, and monitor observations which receive a grade at the end. Tutors cannot be sure as to what degree of proficiency these skills have been developed during the project because it is hard to work out what each individual has achieved overall, in qualitative terms. In their final reports students tend to use language such as "it was a challenge" or "it was a good experience – I learned a lot" but do not specify exactly what they have learned in skill terms and this is something we need to improve for their learning to be given appropriate value and for our teaching to be targeted more specifically at areas all students find difficult. We need to focus more on assessment for learning because currently the information from all the elements of coursework is still poor in terms of telling us anything about proficiency levels reached in a particular skill. It is therefore the assessment design that is weak and we need to find an alternative.

Figure 2: Skills students identified they improved during the project.



4. Proposal: An Alternative Assessment Framework

We propose that the competencies that should be measured via peer, self, formative and summative assessment are along the lines of those discovered by Turley and Bieman when conducting a study of exceptional and non-exceptional professional software engineers[13]. They identified 38 competencies including – helps others, willingness to confront others, responds to schedule pressure, focus on user/customer needs, team-oriented, writes / automates tests with code etc. We would use these in conjunction with the assessment of technical and team work products. Many of the behaviours Turley and Bieman identified with non-exceptional performance "can be viewed as the behaviours of inexperienced engineers" because a beginner "will be unsure of their own skills and capabilities" and therefore defining levels of proficiency or development in these behaviours should give our students more confidence as to how they have improved during the year. Ambrose [14] suggests we need to provide an holistic view of a person's competency and for this to happen students need to develop self-efficacy where they can make judgements not on what skills they have but what they can do with the skills they possess. Currently, we tend to get the students to evaluate their skills in broad terms at the start of the module [8] but we do not retain the focus on skills in other pieces of coursework throughout the year or at the end. Ambrose's work and the work of Marakas et al [15] in the area of measuring competency suggest antecedents to self-efficacy include verbal persuasion by a credible mentor, social comparison, (by observing someone else performing similar tasks) and the degree and quality of feedback and perceived effort can all

enhance or decrease self-efficacy beliefs. So, it is with this in mind that we now propose an alternative way of assessing Software Engineering Team Projects that takes perceived effort and competency development more into account and gives students early feedback as to their progress so they can correct poor behaviours. We are currently looking at the work of Smith and Smarkusky who use Competency Matrices [7] for self and peer assessment. A competency matrix captures team knowledge and skills in various categories (they identify Process, Communication, Interaction, Contribution and Responsibility). These matrices then allow an assessor to assign a numerical range of proficiency in each specified competency – individuals are evaluated by selecting a class rank to indicate the baseline competencies expected of the individual – and in this work, peers assess whether an individual has met the expectation, exceeded the expectation by various amounts or requires (varying amounts) of improvement. An example of numerical ranking would be a student gets 1 if there is much improvement needed or a 5 if they far exceed the expected competency. At present we are still defining the precise competencies required by undergraduate software engineers and we need to refine and target those skills and competencies that are measurable and practicable within the confines of one module. So far areas we have identified some areas that hold potential for competency measurement and these include:

- Professionalism – account of a student's overall behaviour towards others in the group, towards their customer and to others outside the group, whilst getting the job done.
- Problem-solving – the degree to which a student proposes workable solutions to technical, planning and people situations, discusses possible solutions with team-mates or negotiates with others to find solutions to problems encountered by the team – in a practical, cost efficient and time efficient way.
- Communication and interpersonal skills – the degree to which as student expresses thoughts ideas, opinions and solutions orally or on paper, via presentation etc.
- Time management – the degree to which a student can cope changes in requirements, schedule and workload pressures
- Team-oriented behaviour – the degree to which as student is competitive on behalf of the team, pursuing team objectives.

However our work in identifying the precise competencies needed and how these map to assessment design is still ongoing.

5. Conclusion and Future Work

In this paper we have given a brief overview of the changes we made in assessment during our Software Engineering team project module. Our work on contribution matrices to reassure students that individual efforts were taken into account has gone some way to help students feel more confident about being assessed in teams. We have also introduced larger, more realistic assessments based on industrial case studies and reduced the number of times particular skills are assessed during the module, reducing the student workload. However, students still do not find it easy to identify what skills they have learned and developed at the end of the project. We have therefore proposed an alternative approach to assessing students in Software Engineering Team Projects that combines the use of competency measurement with the skills identified as those possessed by successful software engineers in industry. We are currently working on assessment tools that can be used by students to identify their baseline skills and measure their own competency development throughout their team project. We are also proposing these tools can then be used for peer and tutor assessment to give a more holistic view of an individual's overall competency development.

References

01. CETL AliC, http://www.dur.ac.uk/alic, accessed 15/03/09
02. Ernest Ferguson, Clifton Kussmal, Daniel D. McCracker, Mary Ann Robbert. (2004) Offshore Outsourcing: Current conditions and diagnosis, Proceedings of the 35th SIGCSE Technical Symposium on Computer Science Education, Virginia, USA, 2004. pp 330-331, ISSN: 00970-8418
03. HEFCE CETL initiative http://www.hefce.ac.uk/Learning/TInits/cetl/, accessed 15/03/09
04. NESS, https://ness.ncl.ac.uk:444/, accessed 15/03/09
05. Subversion, http://subversion.tigris.org/, accessed 15/03/09
06. Team Project Module Outline, Newcastle, http://www.cs.ncl.ac.uk/modules/2008/CSC2005, accessed 15/03/09

07. Harold H Smith, 111, Debera L Smarkusky. (2005), Competency Matrices for Peer Assessment of Individuals in Team Projects in Proceedings of SIGITE '05, 155-161
08. Lyn Brodie, Hong Zhou and Anthony Gibbons. (2008), Steps in developing an advanced software engineering course using problem based learning, Engineering Education, Vol. 3, Issue 1, 2-12
09. Gibbs, G., Learning in Teams: A Student Manual, (1981), Oxford Centre for Staff Development, 1994, ISBN 1 873576 20 X, which is in turn based on Management Teams, R.M. Belbin, Heinemann,.
10. Baroudi, Z. M., (2007), Formative Assessment and its role in instructional Practice, Postgraduate Journal of Education Research, Vol. 8(1), 37-48
11. Devlin, M., Drummond, S., Philips, C., Marshall, L., (2008), Improving Assessment and Feedback in Computing Science Team Projects, 9th Annual Conference of the Subject Centre for Information and Computer Sciences, Liverpool Hope University, White, H. (ed.), Higher Education Academy, Subject Centre for ICS, 133-139
12. McNamara, R. A. (2004). Evaluating Assessment with Competency Mapping in Proceedings of the Sixth Conference on Australasian Computing Education, Vol.30, R. Lister and A. Young (eds). ACM International Conference Proceeding Series, Vol. 57, Australian Computer Society, Darlinghurs, Australia, 193-199
13. Turley, R.T., and Bieman, J.M, (1995), Competencies of Exceptional and Non-Exceptional Software Engineers, in Journal of Systems and Software, (28(1): 19-38
14. Meta-cognition and software developer competency – construct development and empirical validation, Paul, J, Ambrose, Issues in Information Systems, Vol. VIII, no.2, 2007. pp273-279
15. Marakas , G. Yi, M.Y., & Johnson, R.D., (1998). The Multilevel and Multifaceted Character of Computer Self-Efficacy: Toward Clarification of the Construct and an Integrative Framework for Research, Information Systems Research¸ 9, (2), 126-163