Negotiated Assessment Criteria and Peer Assessment in Software Engineering Group Project Work: A Case Study

Ian Utting
Computing Laboratory
University of Kent at Canterbury, UK
e-mail: I.A.Utting@ukc.ac.uk

The core second level course in Software Engineering for Computer Science students at the University of Kent (UKC) has, for many years, included a substantial design-and-build group project component. In recent years, this has been enhanced to address a number of issues crucial to students' development as reflective, professional practitioners
Problems (for both staff and students) in the implementation of these enhancements to group learning and the educational experience are addressed, and identified beneficial solutions are described. The value of this approach is situated firstly in terms of changes to the students' approach to subsequent, less constrained, project work, and secondly against the stated learning outcomes of the project; their improved technical and professional practices.

1 Introduction

In common with most undergraduate Computer Science programmes (in the UK), the University of Kent at Canterbury (UKC) runs a small-group design-and-build software development project as part of the second year of its three year degree programme. In our case, this project is primarily in support of the core Software Engineering unit. More unusually, the Computer Science degrees at UKC also include a group project in the final year of the programme.

Over recent years, we have seen both an increase in student numbers, and an increasing sophistication of the types of work undertaken by students in their final year project. In response to these issues, we have adopted a number of strategies designed to increase the value of the project for students both in terms of the technical outcomes of the work and in the extent to which we can encourage them to reflect on their maturation as professional practitioners in software engineering.

Particularly, we have enhanced those aspects of the work concerned with:

- Group allocation and formation.
- Technical and non-technical goal setting within the context of the taught material. These goals are used both to guide practice and as a basis for assessment.
- Students' critical evaluation of their own and others' achievements, in the context of moderated self and peer assessment.

Although the work is assessed as a group, it has proved possible to account for variations between individuals' efforts using a novel questionnaire based approach initially developed at the University of Exeter.

Many of the techniques we have drawn upon in this work have been identified by a national project on "Effective Project work in Computer Science" (project EPCoS), in which the author is a partner. EPCoS's goals are "To identify, make explicit and systematise existing best practices in Computer Science student project methods and techniques in order to make existing knowledge and experience readily accessible for the achievement of threshold standards in Computer Science graduates" (Fincher, 1998).

2 Group Allocation

Sensible allocation of scarce resources against competing requirements is one of the major lessons we would like students to learn from this project work. As such, we allow them to select groups and allocate tasks for themselves, but place a number of constraints in their way. The project that the students undertake has changed over time but always asks them to undertake the development of a small software system, focussing on the development processes involved rather than the end result. The systems are typically well specified (sometimes formally in, e.g., Z) and fall fairly readily into two halves. Each group of students develops one half of the system, integrating their work with that of another group, who are developing the other half.

This is the second piece of group in the software engineering course. Before attempting it, students have spent some 10 weeks working on a professional-issues case study undertaken in groups of about a dozen (Fincher & Utting, 1998). The fact that these (large) groups have been formed for an entirely different purpose – to reflect on professional (non-technical) responsibilities, and to generate a short group report each week – makes the task of dividing resources between the two halves of the development project an interesting one.

In a study of the factors affecting students' choice of the ways to break down their large-scale groups, Thorn (1998) observed that, despite the taught material in the course on resource allocation, the major factor influencing the students' choice was how much they "liked" the other members of the group. Nonetheless, in a survey of students' experiences of this work and its relevance to their final year projects, "experience of working in a group" and "allocation of tasks to group members" were the most-quoted advantages of having done the work.

3 Negotiating project goals and assessment criteria

It has often been observed that students are focussed strongly on software development as a code-writing activity. This is particularly problematic in the teaching of software engineering where limitations of available student time and effort make it difficult for them to develop software of a scale and complexity sufficient to require the techniques being taught. Surveys over time at UKC have suggested that, working from a simple specification of a small program, final year CS students write 15 lines of code per assessment hour. With a total available time per course of 120 hours, this means that even if nothing is done except coding the average student could produce only 1800 lines of code. A program of this size hardly requires the sort of design and project management techniques suitable for large-scale systems. Thus we need to constrain students into using apparently less appropriate techniques, in order to give them (at least some) practice in them before they move on to systems where their use will be critical.

By the time that they come to start the project component of this course, the students have already been introduced to the basic principles of managing software development. This allows us to negotiate with them (in a fairly abstract manner) the kinds of tasks that they will have to undertake in order to successfully complete the project, and identify the products of the project in which evidence for the completion of those tasks will be found. As part of this process, conducted as a group discussion, we also look at the relative weights of the different tasks identified (in terms of both scale and complexity), which leads to the weight they will each be given in assessment. Although the outcome of this discussion varies (marginally) from year to year, a typical result is shown in Table 1.

Table 1: Assessment criteria with weightings

Section	Group 1	Group 2
Planning	10	10
Quality Management	10	10
Design and Design Documentation	15	15
Implementation	15	20
User Documentation	15	0
Testing	15	15
Interface Specification	0	10
Records of Meetings	10	10
Integration	10	10
Total	100	100

An example of the description of what evidence might be found for these activities, and where, is given below:

> **Quality Assurance and Management**
> Evidence for these activities will mainly be found in the records of project meetings and also in the delivered QA documentation.
> In particular, it should be evident that:
> - The responsibilities of group members for quality issues have been identified.
> - Checks on the quality of deliverables have been performed and the results documented.
> - Standards (for coding and documentation) have been specified and adhered to.
> - Configuration management and source code control have been used where appropriate, to keep track of different code and document versions, and to identify those responsible for their modification.

The act of having negotiated such criteria with the students both encourages them to understand the real worth of the individual tasks (especially "implementation"), and provides them with early guidance on the way in which their work will be judged. The fact that the assessment criteria are aligned with software engineering “good practice” is of immense benefit to the learning process.

4 Encouraging critical reflection – peer and self assessment

Given the inevitable delay between submission of work for assessment and its return, and the competing calls on students’ attention, even the best crafted feedback can easily lose its relevance, at least until the next time similar work must be performed, and by then students’ understanding of what they had done and why has often faded. For this reason, many students focus their attention on the mark that they are eventually awarded, and much of the value of feedback in developing their understanding of their work is lost.

In order to address these problems we instituted a "peer assessment" approach to the assessment of this project work. Getting students to evaluate the work of others is important in developing their own ideas as well as for the feedback they receive. Knowing that they will have to judge the work of others against the developed criteria also serves to keep them focussed on those criteria in their own work.

The process adopted requires groups of students who have themselves undertaken a particular half of the project to meet and assess the work of another such group, writing a short report as well as providing marks. A member of staff then moderates the marks before returning them. This moderation process reveals no significant difference between the marks awarded by students and by staff, and no significant correlation between the mark eventually awarded to a particular group and the accuracy of that groups' marking of the work of others.

In early versions of this process, following the return of marked work there was a period during which the assessed group could appeal against the assessment, and a meeting was convened at which both sides could present their case for (and against) the assessment before a binding agreement was reached. Typically, this route was taken by some 15% of groups. Partly this was a result of the number of criteria against which the work had been assessed – with more criteria, there is more room for detailed disagreements – but more frequently it was a result of an unreasonable expectation of the accuracy of assessment. During the moderation process, a discrepancy of ±5% overall was judged as acceptable – well within the bounds of error to be expected when work on such a scale is marked by more than one assessor. Such variation is normally hidden from students, relying on their trust of the judgement of staff. Such trust is clearly suspended when students are judged by their peers, even though the variation in mark is no wider than would be expected from multiple staff assessors.

In the current version of the process, groups are asked to submit (along with their project deliverables) a self-assessment of their own work against the negotiated criteria. This was instituted to assist in their reflection, to give them more of a basis against which to judge the work of others, and to provide input to the moderation process. Unexpectedly, it also has the effect of reducing the number of groups who are dissatisfied with the mark awarded them by their peers from 15% to 5% (one group). So marked is this effect that even the added complication of a substantial cash prize for the best project group resulted in no increase in the number of disagreements.

4.1 Group and individual marks

For simplicity's sake, a single mark is awarded to an entire group. Normally, this is appropriate, as all members of the group have, to a first approximation, contributed equally to the success of the project. In occasional cases, however, a particular individual has spectacularly failed to do so. In this past, this has been catered for by a penalty clause allowing a group to allocate a mark of 0% to one (or more!) of their members on the grounds of their non-contribution, thus giving them some leverage during the course of the work. Such a decision must be taken unanimously by the group, including the individual concerned. This draconian step has only once been taken, and then against a student who was in the early stages of deciding to withdraw from the University. There has been, to date, no mechanism for rewarding conspicuously effective contributions to the work of a group.

In order to provide a less dramatic mechanism for accounting for the differential input of particular students, we are adopting (from Spring 1999) a mechanism identified in project EPCoS based on practice at the University of Exeter (Milne, 1998). Each member of a group fills in a small questionnaire assessing the contribution of all the groups' members (including themselves) against their expected contribution to each of the assessment criteria and to the work of the group as a whole. This will allow the identification of individuals who have under- or over-performed, and appropriate modification can then be made to the overall mark awarded to that individual. The intention is not to differentiate in any great detail between group members, but only to recognise particularly strong or weak contributions by varying the mark by ±5 to 10%, thus encouraging a more even application of effort by the group, and ameliorating the occasional bad feeling among those who feel that they have "carried" other members.

5 Conclusions

The project work described addresses a number of issues in students' development as software engineering practitioners. At base, it represents a fairly typical example of such work as practised in undergraduate Computer Science programmes. A number of mechanisms have been described by which students can be encouraged to reflect on the affective goals of such work, rather than on the superficial goals of producing working software:

- The use of groups formed for a separate purpose as means of providing constraints on available resources.
- The negotiation with students of assessment criteria (and appropriate weightings) derived from taught software engineering principles.
- The use of these criteria to drive students' efforts during the work, discouraging them from adopting simpler (but inappropriate) strategies.
- The use of (group) self- and peer-assessment to encourage reflection on the work produced and the principles learnt.
- The (proposed) use of individual self- and peer-evaluation to modify marks awarded to groups, recognising the differential input of individuals without detailed tracking of effort.

6 Acknowledgements

Project EPCoS is funded by the Higher Education Funding Councils for England (HEFCE) through their "Fund for the Development of Teaching and Learning" programme.
The prize for the best overall performance in the project is provided by Philips Research Laboratories in Redhill, UK.

REFERENCES

Sally Fincher, *Effective project work in computer science: Project EPCoS*. In Mette Knudsen and Torben "Sopper" Vinther, editors, Project Work in University Studies, volume 2, pp 195-203. Roskilde University, Denmark, September 1997.

Sally Fincher and Ian Utting, *Entraining students in professional issues: challenging their structures of knowledge*. In 6th Improving Student Learning Symposium: Improving Student Learning Outcomes, September 1998.

K. Thorn, *A Window on Group Formation Factors*. In Mike Holcombe et al., editors, Projects in the Computing Curriculum, pages 217-224. Springer-Verlag, July 1998.

W. Milne, *Moderation Using Student Input*, Project EPCoS internal document, http://www.cs.ukc.ac.uk/national/EPCOS/data_archive/bundles/bundle1.htm, 1998.

Ian Utting is a Senior Lecturer in the Computing Laboratory at the University of Kent at Canterbury in the UK, and Director of the UKC Authorized Academic Java Campus. His research interests are in the engineering of large-scale distributed systems, and in the teaching of Computer Science. He teaches Object Oriented Design, Software Engineering and Distributed Systems. He is currently working as a member of the HEFCE FDTL-funded project EPCoS
The Computing Laboratory at UKC is ranked as one of the top 10 in the UK for the teaching of Computer Science, and has recently become the first Sun AAJC in Europe.