Group Project Work from the Outset: An In-depth Teaching Experience Report

Martin Shepperd
Dept. of Information Systems & Computing
Brunel University, UK
martin.shepperd@brunel.ac.uk

Abstract

CONTEXT - we redesigned our undergraduate computing programmes to address problems of motivation and outdated content.
METHOD - the primary vehicle for the new curriculum was the group project which formed a central spine for the entire degree right from the first year.
RESULTS - so far this programme has been successfully run once. Failures, drop outs and students required to retake modules have been halved (from an average of 21.6% from the previous 4 years to 9.5%) and students obtaining the top two grades have increased from 25.2% to 38.9%.
CONCLUSIONS - whilst we cannot be certain that all improvement is due to the group projects informally the change has been well received, however, we are looking for areas to improve including the possibility of more structured support for student metacognitive awareness.
Keywords*: software engineering, student projects, groupwork.*

1. Introduction

This paper[1] is not intended as a scientifically rigorous study but rather an informal account of our experiences that we hope may (i) have some motivational value and (ii) offer some useful ideas and techniques for others involved in group project work in universities. Although this study is based in a computing department — which brings particular challenges and opportunities — many of the pedagogic principles are relevant to other disciplines such as engineering, design, business studies, psychology and architecture.

These ideas and experiences derive from a thorough redevelopment of the undergraduate curriculum for two undergraduate computing degree programmes (BSc Computer Science and BSc Information Systems) during 2008-2010 at Brunel University, London. The academic year 2009-2010 represents our first experience of the new scheme. At its heart lies a substantially increased level of project work (approximately a third of each year) acting as a spine to the entire programme.

Clearly the idea of students working as a group to carry out a software project is not new. Indeed group work is commonplace to computing and computer-related undergraduate programmes. The novelty lies in the extent to which group projects are used and the fact that they are seen as an integrating vehicle for all course themes from the outset i.e. Level One. In addition, we have deployed a number of specific implementation tactics that are not widespread but, we believe, effective.

The remainder of the paper is organised as follows. Section 2 briefly reviews previous innovative ideas on group projects for computing students. The following section provides the context in which Brunel decided to re-design its undergraduate computing curriculum. Section 4 describes in some detail the mechanics of our group projects and concentrates upon the Level 1 projects. Next, Section 5 presents our experiences from running the Level 1 group projects in 2009-10 and lastly, Section 6 concludes the paper with a review of what aspects of the group project seemed to work and where we might make improvements.

2. Related Work

The problems that undergraduate computing students have in learning to become competent software developers appear to be longstanding and endemic [9]. Even when this is not so there is often criticism from potential employers that students do not have good team skills and the ability to cooperate effectively [11].

A pioneering paper by Freeman, Wasserman, and Fairley [5] highlighted the need for computing students to not only have strong technical skills, but also problem solving and communication skills. They also pointed out the substantial differences between academic and industrial environments for software developers. For these reasons they

[1] This is an extended version of a paper presented at the 24th IEEE Conference on software Engineering Education & Training, May 22-24, 2011

argued strongly for the introduction of group projects into the computing syllabus. A particular innovation that they promoted is the software hut (the name suggests a small software house) where students are expected to 'sell' their software or solution to others and that some proportion of the marks are derived from this exercise. The idea is to introduce elements of marketing and competition, of negotiation and management and perhaps some sense of fun.

Another more recent example of work that has championed the use of undergraduate group projects is to be found at the University of Sheffield in the UK [6]. Here they take the idea of the software hut to its logical conclusion and actually have student groups working for external clients (though no money is involved). A detailed protocol is agreed but essentially the client gets to choose and keep the best software from 4 or 5 teams. This requires a good deal of maturity from the students so such projects are placed in the final year of their undergraduate programmes.

These approaches to student group projects can be characterised as founded upon the philosophy of a practicum [12]. In other words, by offering students experiences that mimic the "real world", the students will learn. We revisit this in the Discussion.

3. Background to Course Development

In 2008 the computing department at Brunel University decided to conduct a major review of their undergraduate programmes. This was prompted by:

- The limitations and frustrations of teaching large classes where even seminar groups could exceed 20 students
- Inconvenient timetabling due to a lack of large lecture theatres
- Uninteresting or dated modules[2] not reflecting current software development practices
- Lack of opportunities for students to display originality and creativity
- Students poorly prepared for their individual final year project
- Lack of connectedness or a sense of community, for example, even coursework is submitted and feedback received electronically via the university virtual learning environment (VLE)

[2]A module is a named unit of content e.g. Introductory Programming and typically 5 or 6 modules combine to make a level (or year) comprising 120 credits.

This last factor was seen as especially serious, leading to demotivation and a lack of engagement on the part of the student body and also, (dare I say it) on occasions from the academic staff employed by a university that perceives itself as "research intensive". A consequence of this is that there are considerable pressures upon academics to generate significant research outputs and this is seen as the primary basis for promotion.

In addition, we had to consider what "sort of student" we wished to graduate. This was addressed from two perspectives. The first perspective considers the student's long-term aspirations and the role of the degree in supporting their career. The second comes from our perspective and reflects what we consider to be core components of computing as a discipline.

The reality is that the majority of our graduates see themselves as heading for a commercial career rather than an academic one. This should not, however, be interpreted as a dumbing down of the degrees to meet the needs of immediate employability. First, jobs are only a stepping-stone to where our graduates belong. Within 5 to 10 years of graduation they should have risen to leadership positions where they will carry significant responsibilities. This relates to SFIA[3] Level 5. These levels indicate degrees of autonomy and influence exercised by such staff within the organisation and in the Draft Version 4 of the framework they are defined as:

Autonomy: Works under broad direction. Full accountability for own technical work or project / supervisory responsibilities. Receives assignments in the form of objectives. Establishes own milestones, team objectives and delegates assignments. Work is often self-initiated.

Influence: Influences organisation, customers, suppliers and peers within industry on contribution of specialisation. Significant responsibility for the work of others and for the allocation of resources. Decisions impact on success of assigned projects i.e. results, deadlines and budget. Develops business relationships with customers.

Achieving these levels of autonomy and influence depends, not just on the degree, but also on gaining experience and maturity in more junior roles. Perhaps the most telling element of the SFIA level 5 definition, in terms of the preparation that comes from our degree, is its description of the ability to handle complexity:

[3]The Skills Framework for the Information Age (SFIA) provides a common reference model for the identification of the skills needed to develop effective Information Systems (IS) making use of Information & Communications Technology (ICT). See URL: http://www.sfia.org.uk/

Complexity: Challenging range and variety of complex technical or professional work activities. Work requires application of fundamental principles in a wide and often unpredictable range of contexts. Understands relationship between specialism and wider customer / organisational requirements.

From a discipline perspective, the core that anchors and distinguishes computing (or at least our Department's particular viewpoint) is:

> a focus on software artefacts that exploit the capabilities of networked computers for the benefit of some human activity or purpose.

This leads to two families of degrees: those that are driven by a primary interest in the creation of software artefacts (computer science) and those driven by a primary interest in their beneficial impact on human activity (information systems). Each includes the other perspective as an essential secondary element anchoring them in the ethos of the Department and distinguishing them from the related disciplines.

The Group Projects provide a spine for Levels 1 and 2. The groups comprise 3-5[4] students and involve a task that integrates much of the other material in the year. The group project for Levels 1 and 2 comprises 40 out of a total of 120 credits for the year and thereby gives the student the opportunity to engage with a task of significant complexity. Each year is a different project. Typically this will include software development, context, communication, team working, project management and the application of different research methods (e.g. market research, usability assessment, etc.). Each team is supervised by a member of academic staff (faculty) with regular contact (normally weekly), thus, we see the projects as the vehicle for traditional style tutorials.

The project spine is completed by a more traditional individual final year project (FYP). These are projects where the student tackles some problem working closely with an academic member of staff. Students have a good deal of freedom to choose a problem of mutual interest. As per Levels 1 and 2, the FYP comprises 40 out of 120 credits.

The Level 1 group project seeks to integrate skills covered by the concurrently running modules into a non-trivial, practical group task including a significant degree of programming and technical engagement. It is also intended to help the student gain confidence in their technical abilities. Although the overall course structure was designed to make the running of group projects easier, e.g. students have no choices for modules in Level 1 and restricted choice in Level 2, the concurrent scheduling of much material does present some difficulties. For instance, students do not encounter Java until several months into Level 1 yet ideally they would be capable programmers from the outset. We countered this type of problem by (i) by carefully ordering the group project tasks e.g. by commencing with html and css before moving onto Java (ii) providing very detailed tutorial materials and technical support in Lab sessions and (iii) by postponing some technical aspects e.g. relational databases until the Level 2 project.

The projects also served as spine in two other respects. It provided the possibility of assessing learning outcomes from other modules, for instance aspects of programming, report writing and communication. Second it provided some experience for students of the whole picture rather than a piece meal approach that can arise when material is allocated to individual modules. They were involved in the development and evolution of a software system over an entire academic year. This contrasts with seeing requirements capture, design, testing, etc. as standalone tasks.

[4]We avoided larger groups since this made it easier to ensure students could not evade tasks about which they did not feel confident e.g. programming. In addition independent studies [10] have reported small team sizes of 3 to 6 are more satisfactory than larger groups (better bonding, less competition).

4. Mechanics

In this section we describe in more detail the operation of the Level 1 group project. The students were informed that the aim of this 40 credit module (i.e. one third of the first year) was to:

> integrate skills covered by the other Level 1 modules into a non-trivial, practical group task including a significant degree of programming and technical engagement. It is also intended to help the student gain confidence in their technical abilities.

The learning outcomes (LOs) for the module are as follows. Students should be able to:

1. plan, manage and track a non-trivial group activity.
2. take an open-ended problem and define and refine the requirements.
3. design, develop and test a modest piece of software (order of a few hundred LOC).
4. independently (i.e. individually) understand, modify and test a given piece of software
5. effectively present, communicate and market ideas and solutions to their peers.
6. create and use technical documentation.

7. understand and apply the principles of professional and ethical behaviour in a group context.

8. reflect and learn from their group project experiences.

A number of aspects of the Group Project are intentionally open-ended to encourage student creativity and develop open-ended problem solving skills. Typically the coursework includes some requirements analysis and understanding of the problem context, software design and development, communication, team working, project management.

This module is intended to be primarily practical in its nature; in other words students learn by doing and, most importantly, reflect after-the-event. We try to reassure students that on occasions they will be asked to do something new and that this might seem daunting. However, it is by actually engaging with a problem that they have the best opportunity to learn and it is interesting to note that many of them commented favourably upon this aspect of the module. We appreciate the challenges students face and so don't expect perfection and try to create an environment where they feel it is safe to experiment. This is reinforced by marking scheme that isn't punitive.

Level 1 students were allocated to groups of four with the goal of creating teams of mixed ability and experience (this process is described in the next section). Groups were then provided with a description of the problem. Teams then spent half the academic year researching into the problem domain and developing a proof of concept website to underpin the subsequent detailed development. An important textbook that we recommended to the students to provide guidance on surviving as a team is Levin [7].

Teams then endeavoured to sell their partial solution to other teams and to 'purchase' software from another team upon which subsequent development was required to be based. Marks for the first stage were related to the number of 'sales'. Students quickly appreciated that documentation and a good design gave them a competitive advantage. The more innovative offered additional blandishments such as support contracts. In addition, we awarded prizes to the best teams. Next more detailed requirements were provided and the second part of the development based on the 'bought' but incomplete code. The marking scheme was weighted such that students could recover from zero sales from the first stage but be challenged if they made an unwise 'purchase' (see Table 1).

Thus, the students had an opportunity to learn about team working, planning and monitoring, requirements, marketing, procurement, documentation, testing, maintenance and changing requirements in addition to the chance to develop a non-trivial piece of software.

4.1 Team Formation

Early on in the academic year, the students were placed in a team of approximately four students and assigned a Tutor (who is an academic member of staff). Initially we formed as many 4 person teams as possible, however where boundary problems occurred we created 5 person teams. Two students left or transferred course very early on resulting in two 3 person teams.

The teams were chosen by us. Our intention was to form teams of mixed experiences and abilities, e.g. we mixed BSc Computer Science and Information Systems students. The reasons for this were twofold. First, to provide more opportunities for students to learn from their peers and second, to better represents 'real' software development teams. A particular problem for a Level 1 group project is that the abilities of students may not be known in advance. We handled this by delaying team formation until Week 4 and also by asking the students to complete an on-line diagnostic test. A few students failed to do the test and this (rather than the actual score) turned out to be a very good predictor of future problems.

We also informed the students that a special pool of students would be created from non-attenders and non-participants (without valid reason) and used to form teams. This meant that if they were lazy and expected to be carried by their fellow students then they would be placed in a team with like-minded students. Our advice was therefore to engage in this important module right from the start. This was probably the single most important decision we made and turned out to be fully vindicated. Students who commenced with a laissez faire approach did not substantially change their behaviour. It also meant that to a large extent we were able to ring fence problems into a very small number of groups rather than experience widespread impact due to the difficulties of a few students. I should stress, however, such students were given, equal if not more support than others.

As far as possible team membership was stable throughout the year but if circumstances dictated (e.g. serious illness) we retained the option to make modifications. In this eventuality students could be compensated by the marking process. In practice we moved students between 4 teams (out of a total of 47) during Week 10. This was due to low levels of participation and external problems for six students.

To reiterate careful team formation was one of the most important decisions for this module and therefore we tried to use as many sources of intelligence as possible.

4.2 Group Project Tasks

Next we describe the coursework the students were expected to undertake during this module. As far as possible

we took an incremental approach rather than giving students a series of seemingly random and disconnected activities.

Table 1. Group Project Task Components

Task	Description	Group / Individual	% of Total Marks
T1	Initial software prototype	Group	10%
T2	Integration of components and documentation	Group	10%
T3	Implementation of additional functionality and documentation	Group	50%
T4	Individual implementation of an unseen change to group system	Individual	Pass/ fail
T5	Post-mortem style group review	Group	30%
T6	Assessment of ethical and professional behaviour	Individual	Pass/ fail

A difficulty we had was choosing something that was challenging for those who already had a strong background in programming but not impossible for those without prior knowledge. In the end we decided upon the development of a website suitable for new students at Brunel University. This commenced with some simple static html and css files (Task 1), integration of other websites into a single portal (Task 2) and the addition of dynamically generated webpages using Java and jsp (Task 3). More details may be found in Table 1 and the Appendix which contains excerpts from the student instructions. These tasks are representative of much modern software development and use a mix of typical technology. This was emphasised to the students as a motivator when some struggled with Task 3.

Note the intentional increase of mark weightings so that teams taking a while to "bed down" were not doomed to certain failure and retained the possibility of a good score. Note also the significant weighting (30%) attached to the post project review. Here the intention was to give substantial credit for the ability to learn from failure as well as from success.

We also decided to implement a version of the software hut [5] described in the Introduction as part of Task 1. The idea was to encourage an element of competition, to allow a degree of creativity since the task was relatively unconstrained and finally to help students view software both from the perspective of a developer and from that of a consumer. Each team developed a website, all of which we then hosted and each team could choose three others to incorporate into their Task 2 portal. Choices might be influenced by the quality of the content, implementation, etc. The teams were then ranked according to the number of times they were chosen and marks for Task 1 awarded according (see Figure 1).

Task 4 was designed to prevent weaker students hiding within a team and not developing their own technical skills. They were asked to make two small, unseen modifications to the system under exam conditions. The task was simply marked on a pass / fail basis and was not intended to be overly difficult, rather to set some minimum threshold.

Post project reflection is widely advocated (if not always effectively conducted) as a part of good software development practice [4]. In addition reflection is stressed as an extremely important part of experiential learning [2], so we incorporated a significant opportunity for groups to explicitly reflect upon their experiences. This is revisited and possible enhancements are discussed in Section 6.

Finally, the Task 6 assessment of professional and ethical behaviour was intended as a sanction in the event that a student completely undermined their team through lack of commitment or other unacceptable behaviour.

5. Results

To give some idea of scale and the running of the module Table 2 provides some basic logistics. As can be seen a significant number of staff were associated and this meant a substantial amount of effort was required to brief everyone and deal with ad hoc queries. With hindsight it would have been useful to prepare an information pack for each member of staff.

Table 2. Group Project Logistics

Students	191
Teams	47
Tutors	28
Graduate teaching assistants (GTAs)	10
No. of teams making all submissions	45
Weekly formal schedule	1 hr lecture + 1 hr tutorial + 3 hr lab session

We now consider the outcome of running the Group Project in more detail, particularly the themes of engagement and performance. We then review the results of a student feedback survey.

5.1 Student performance

In order to assess whether the new UG programmes have led to better performance by our students, we looked at the

results from this years Level 1 (2009-10) compared to the previous four years. The comparison is broken down into the two degree cohorts (IS and CS). The grades are overall year grade (120 credits) derived from five constituent modules. The group project when introduced represents 40 out of the 120 credits. The results indicate a promising improvement across all studies, in other words it is suggestive that the impact of the group project is not merely local. We do not yet have data to see the impact upon subsequent years i.e. Levels 2 and 3.

A summary of the results by year and by degree programme is given in Figures 1 and 2. Note that grades A to D represent passes, an E grade indicates a narrow miss with the opportunity of condonement or a retake. F is a serious fail. However, a threat to validity, is that a number of simultaneous changes were made so it is not possible to be certain that the improvement in performance is solely due to the introduction of group projects.

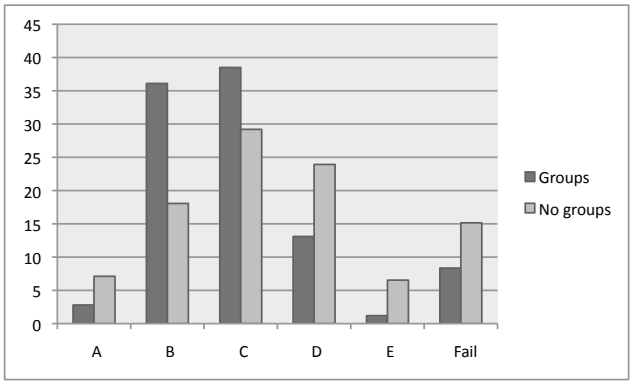

Figure 1. Level 1 Results - Comparing Group Projects With Previous Years

To summarise:

- Just in excess of 50% of CS students obtained an overall grade A or B average (cf 21-42% previously)
- 27% of IS students obtained an overall grade A or B average (cf 9-23% previously)
- 6.3% of CS students obtained a D grade or lower in 2009-0 compared to 17-24% in the earlier years
- 12.7% of IS students obtained a D grade or lower in 2009-0 compared to 16-28% in the earlier years

Note also the consistent superior performance of computer science students over information systems students (see Fig. 2).

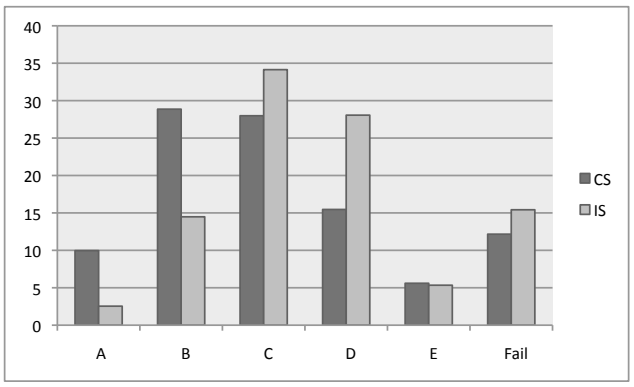

Figure 2. Level 1 Results - Comparing CS and IS Students for All Years

5.2 Engagement

Here we consider first student and then staff engagement.

Table 3. Coursework submission rates

Task	On time	Late	Missed
1: Team website	47	0	0
2: Integrated website	47	0	0
3: Dynamic website	45	1	1
5: Team review and presentation	45	0	2

As Table 3 indicates the module has a remarkable level of participation in terms of submitting coursework on time. Recall that this is a Level 1 module and typically a number (10% or more) of students struggle, drop out or change course without informing us. It is possible that the sense of not letting down one's team mates and the close relationship with a tutor has led to this much increased level of engagement.

Staff engagement is more complex to comment upon, since there is no assessment or formal feedback mechanism. However an immediate benefit is that all staff get to know at least 4 or 8 undergraduate students personally. This is in the context of many academics not knowing any undergraduate students either due to other teaching commitments or being involved lecturing large groups (150-200 students) with very limited opportunities for personal interaction. There were also many examples of staff providing a role beyond that of project tutor e.g. giving advice on future choices of modules.

Some problems inevitably arose. The tight integration with other modules did not always go exactly to plan, e.g. there were some synchronisation issues with the Introduc-

tory Programming module. This might be avoided in the future by more careful planning with all lecturers associated with Level 1 prior to the start of the academic year.

We also had group tutors with varying technical expertise due to the fact that the department embraces both computer science and information systems experts. This led to some apprehension, however, we encouraged students to use the lab sessions as their primary source of technical input. These were supported by our team of GTAs. A side effect of acting as tutor was learning by osmosis and it does seem, at least anecdotally, that some staff have gained in expertise and confidence over the duration of this project.

5.3 Student Feedback

It is customary to solicit student feedback for every module at Brunel University by means of a standard student feedback form that is distributed towards the end of the academic year. Many of the questions elicit a quantitative response in the form of a Likert scale, e.g. how would you classify the standard of your lectures $1 \cdots 5$? Since they are not specific to group projects and need to be interpreted relative to other modules, we focus instead on two open-ended questions:

5.3.1 What do you like most about the module?

We obtained 44 responses out of an original 191 students meaning a 23% response rate. This low rate was largely the consequence of the module finishing in advance of the rest of the course and therefore students being more difficult to locate. It is possible that those students who struggled the most or who were disaffected were under-represented, however, we take the view that we still have some useful and interesting qualitative feedback.

(i) Working in a team (15 responses)

> "Good to experience working in a team"

One respondent illustrated the dilemma of team work by stating "Working in a team" as the best thing but also "the group members" as the thing he or she liked least.

> "I was very lucky and had a great team"

The interesting thing here is that the student saw this as entirely externally determined and therefore nothing to do with his or her contributions to team building. This contrasts with another respondent:

> "working in a team to be able to develop this skill"

It's easy to lose sight of the idea that education can be fun:

> "I really enjoyed working in a team "

and as a means of learning:

> " . . . because I think I learn a lot more rather than working individually"
>
> "Exploring coding in more details (sic) with a group get help and learn at same time"

and finally as a cooperative and team exercise:

> "Working in the team, share different tasks."

(ii) Task oriented aspects of the module (13 responses)
A total of 13 students (i.e. about 30%) mentioned some technical aspect or experiential learning of technical skills.

> "Designing a website from scratch using different programming methods"

was a common response, also:

> "Practical programming. Planning. Deadlines"

Here we see an appreciation of the experiential aspects of learning (this was a common theme) but also of the value of a project management and self-management skills.

Despite our efforts to achieve a high degree of integration with other modules this was either not recognised or not seen as important, so the following was quite unusual:

> "This module has motivated me to cope with all the other modules."

5.3.2 What do you like least about the module?

As previously indicated we had 10 blank responses both for positive and negative features. There were 9 with a positive responses without a negative (c.f. 5 with a negative response but no positive). Again we grouped the types of response.
(i) Disliked aspects of their team (8 students)
These respondents expressed some variant of the other team members didn't contribute appropriately.
(ii) Disliked aspects of the task (15 students)
The fact that more students had strong negative feelings about aspects of the task, than had about being placed arbitrarily in a team we found slightly surprising. Most remarks seem to relate to Task 3 (the overall task was split into a series of sub-tasks each with their own delivery date). Task 3 represented a substantial step up in difficulty.

> "Task 3 was a big step up and was quite a struggle."

Some commented upon specific activities they didn't enjoy such as "Creating the documentation" or specific technologies such as the web-server Tomcat. It is likely the latter stem from frustrating experiences rather than deep problems.

Two students commented specifically commented on what they saw as excessive task difficulty, e.g.,

> "Quite a lot of expectations of understanding everything."

Finally, there were comments on the provision of task information. Breaking the student project into sub-tasks added a considerable degree of complexity and some students reported they found this hard to understand. For instance one student responded:

> "Lack of information on time."

It's not exactly clear what is intended by this comment, however, it was a design decision to release task information as and when it was needed and it may be the student was meaning he or she would have liked more of a big picture at the start.

6. Discussion

As we indicated at the outset, this paper is not a rigorous scientific study. We have not conducted randomised controlled trials. We not only introduced group projects but also made other substantial changes to the curriculum. We cannot be certain that our observations can be solely attributed to the group work. Moreover, there is a lack of objectivity, since the author was intimately involved in the design and delivery of the group project module. Nevertheless, we are sufficiently encouraged by the changes that seem to have been wrought, to add our experiences to those of other educators before us.

First, we consider whether there are local circumstances that have facilitated this ambitious scheme of group project working. One feature is that the projects were designed in from the outset as opposed to being a modification of an existing scheme. We made the conscious decision to keep the structure as simple as possible consequently the students follow a quite rigid year-based plan which means that student cohorts are fairly similar. We also made a decision that students should all address the *same* problem for each year and resisted the temptation to allow students to tackle more business or computer science related problems depending upon their choice of final degree. Apart from ease of management, this enabled greater communication between the different 'flavours' of computing especially as we intentionally created heterogeneous teams. Where there is a high level of student choice (in timing or module selections) the managerial challenges certainly increase. I believe the key lies in creating heterogeneous teams such that the group contains the necessary expertise even if it is not fully located in each individual student.

Another obvious question is to what extent have the students actually achieved the intended learning outcomes. In a narrow sense the LOs are all assessed and so achievement, or otherwise, is measured by the considering the cohort grades. In a broader, and ultimately far more important, sense we don't yet know what the long term impact of the Group Projects are, but this something we will monitor carefully through such indicators as overall course grades and future employability.

A potential opportunity is for research projects based upon the data arising from our having almost 50 independent implementations of the same system. For example, we are now in the position to study the impact of different architectural design decisions upon such qualities as defect-proneness and understandability.

What are areas for improvement? One substantial issue to consider is that we have not considered how students learn and how they learn about learning, that is their metacognitive abilities. As mentioned previously, our implicit approach has been one of the practicum, i.e. trying to create an authentic situation within which experiential learning may occur [12]. This shows that we were not concerned with just programming ability but also other skills such as negotiation, planning, self management and so forth. Nevertheless, we do not articulate precisely what learning we expect to occur nor do we provide any means for checking whether it has been achieved.

There has, however, been a good deal of research by cognitive and educational psychologists exploring the role of metacognition (that is thinking about thinking) upon learning [3]. In particular it provides an explicit mechanism to review progress by means of reflection. Our group project contained an opportunity for group reflection via a presentation (as Task 5). However, this suffered from two problems. First it may have been distorted due to it being assessed, a problem noted by Boud and Walker [1]. Second, the exercise was informal with little guidance being provided (other than the prohibition of personal and defamatory comments). An interesting empirical study of students in Higher Education by Mair [8] looks explicitly at reflection as part of the learning process. An important feature of Mair's work is the provision of scaffolding or spreadsheet templates to assist students in organising their reflections. This is a mechanism that we might profitably adopt for future implementations of our group project.

In summary, there are three main findings. Group projects can be powerful agents for change. We have seen much improved levels of engagement by the students and, in our experience, unprecedented levels of coursework sub-

mission and submissions by the deadline. Second, the devil really does lie in the detail. Communicating effectively with the large numbers of people is important. Designing an appropriate task (particularly in terms of getting the balance between the challenge and the manageableness of it) is important. And most important of all, careful allocation of students to teams. Third, our understanding of what students perceive and how they learn has not sufficiently central to the design and delivery of group projects. This offers scope for further improvement and refinement in future years.

Acknowledgments

I would like to thank my colleagues and all the students for their enthusiastic support and tolerance of the inevitable teething problems. I'm also indebted to Tony Elliman for relating the curriculum development to the SFIA, to Carol Elliott for providing the data on student performance and to Carolyn Mair for careful proof reading and much good advice on the organisation of this article.

References

[1] D. Boud and D. Walker. Promoting reflection in professional courses: the challenge of context. *Studies in Higher Education*, 23(2):191–206, 1998.

[2] E. Boyd and A. Fales. Reflective learning: Key to learning from experience. *Journal of Humanistic Psychology*, 23(2):99–117, 1983.

[3] S. Coutinho. The relationship between goals, metacognition, and academic success. *Educate*, 7(1):39–47, 2007.

[4] T. Dingsøyr. Postmortem reviews: purpose and approaches in software engineering. *Information and Software Technology*, 47(5):293, 2005.

[5] P. Freeman, A. Wasserman, and R. Fairley. Essential elements of software engineering education. pages 116–122, San Francisco, CA, 1976.

[6] M. Holcombe, M. Gheorghe, and F. Macias. Teaching XP for real: Some lnitial observations and plans. In *Extreme Programming Perspectives*. Addison-Wesley, 2003.

[7] P. Levin. *Successful teamwork*. Open University Press, Maidenhead, Berks, 2005.

[8] C. Mair. Using technology for enhancing reflective writing, metacognition and learning. *Journal of Further and Higher Education*, In Press, 2011.

[9] M. McCracken, V. Almstrum, D. Diaz, M. Guzdial, D. Hagan, Y. Kolikant, C. Laxer, L. Thomas, I. Utting, and T. Wilusz. A multi-national, multi-institutional study of assessment of programming skills of first-year cs students. *ACM SIGCSE Bulletin*, 33(4), 2001.

[10] M. Shaw and B. Harkey. Some effects on the congruency of members characteristics and group structure upon group behaviour. *Journal of Personality and Social Psychology*, 83:150–163, 1976.

[11] W. Waite, M. Jackson, A. Diwan, and P. Leonardi. Student culture vs group work in computer science. *ACM SIGCSE Bulletin*, 36(1), 2004.

[12] L. Williams and R. Upchurch. Extreme programming for software engineering education? In *31st ASEE/IEEE Frontiers in Education Conference*, Reno, NV, 2001. IEEE Computer Society.

Appendix: Level 1 Group Project Task Description

Overall Description of the Assessment

During week 4 you will be placed in a group of approximately four students and assigned a Tutor (who is an academic member of staff). The groups will be chosen by us. Our intention is to form teams of mixed experiences and abilities. The reasons for this are twofold. First, it will provide an opportunity for you to learn from your peers and second, it better represents 'real' software development teams.

If you are a good, hard working student you can expect to do very well in this module, however, a special pool of students will be formed from non-attenders and non-participants (without valid reason) and used to form separate groups. This means that if you are lazy and expect to be carried by your fellow students you will be placed in a group with like-minded students. Our advice is to engage in this important module right from the start.

As far as possible groups will be stable throughout the year but if circumstances dictate (e.g. serious illness) we may have to make modifications. In this eventuality you will be compensated (as appropriate) by the marking process.

You will be expected to participate in each major class of activity, so for example, avoiding programming will not be permitted. This is achieved by making Learning Outcome 4 mandatory. So each group member will be required to carry out a small, unseen maintenance task (T4) to their system in a controlled environment. It will be assessed on a pass/fail basis as it carries zero credits.

The entire group is expected to meet with their Tutor regularly (usually weekly but refer to the Module Calendar). Your Tutor is there to provide advice, guidance and encouragement but not resolve detailed technical questions which should be dealt with during the Lab Sessions. Your Tutor will be the best person to ask questions about the running of the Group Project.

The Group Project comprises six sub-tasks. Detailed information for each task will be released according to the calendar below. Note that some tasks are to be performed and assessed as a group whilst others are individual.

Task 1

You are required to design and develop a website that would be useful for new computing undergraduates at Brunel University. You are not restricted to academic aspects of university life, indeed we encourage you to think more broadly.

You have one month to develop the website. Then you must submit your website files (for details see below). We will then host them on our server such that they are visible to all the other teams. Each team will then select three other websites and for Task 2 incorporate all four (the three selected sites plus the self-developed site) into single site. Most marks for Task 1 derive from 'sales' so Teams are advised to consider what would be distinctive and attractive to other teams. We suggest you choose a relatively focused topic so that you can make a coherent website and market it more effectively to your colleagues.

You are asked to provide static html at this stage. Use an ASCII html editor such as gedit which is open source and available for windows, linux and many other platforms. Do not use the save-as-html facility in any of the Office products. This generates extremely verbose and difficult to maintain html that will cause you a great deal of difficulty in subsequent tasks. Likewise do not use Dreamweaver or similar editors that focus on the aesthetics of the page rather than the underlying html. If you can't view and understand the html you are writing don't use the editor.

You are strongly advised to use css to manage your formatting and to make it easier to integrate other websites in Task 2 and accommodate changes in Task 3. Avoid Javascript or other complicating technologies at this stage. Remember simple, robust and easy to modify is best.

Task 2

You have developed your own static html website for Task 1. The aim, now, of this task is to select three other websites developed by your peers and integrate them into a single site. Although the initial development for the "bought in" sites will have been accomplished by the other groups you will be entirely responsible for the final integrated website so choose carefully.

The first step is to evaluate the websites produced by the other teams during Task 1. We will host these on a single web server so that you can make systematic comparisons. You can also talk to other teams to help make an informed decision as to which three other websites you wish to incorporate into your site. It is likely you will wish to choose sites that address different topics to the one addressed by your own website. We also advise you to consider technical aspects. For example, is the html documented? How easy would the site be to modify? Have the developers made systematic use of css?

Task 3

The aim of this third group task is to extend the website you developed in Term 1 (Tasks 1 and 2). You will now have the opportunity to incorporate your knowledge of programming by adding Java and Java Server Page technology to your system so that it can generate dynamic html.

Your website should now be extended to provide information on the various modules available to students. The information will be accessed from a text data file that we provide on our server. This means you need to access this file, find the relevant information and then generate suitable html to be rendered by the web browser.

NB not all versions of web-based software are fully compatible, so we mandate the use of Firefox 3.5, the server Tomcat 6 and JDK 1.6. All testing and marking will be based on this environment. You are strongly advised to use Eclipse as your development environment.

Your programme should discover what modules are available (remember that this will be dynamic) and present the choices (code and name) to the user of the website allowing them to select a particular module in order to find more information (i.e. the remaining data) about their choice. You may choose how you accomplish this although you must generate html which will be rendered by the web browser.

Remember it is good practice to make your code as robust as possible. This is sometimes known as defensive programming. Please make sure you follow these requirements and resist the temptation to embellish your solution. You will not receive any credit for additional functionality e.g. deciding that your website would be improved by adding a horoscope facility or whatever! Moreover you run the real risk of reducing reliability, understandability and flexibility.

Finally you are warned that we will make some additional small changes to the system specification during the course of Task 3. For this reason good quality architecture, use of css and a general approach of designing for change are recommended.

Task 4

You are required to make the following two changes to your software system [the changes are unseen and are carried out under exam conditions].

1. Change the paragraph text size to an absolute size of 24 px for all pages in the site.
2. The "module information" link displays the module code and name of the modules. Please remove the functionality that displays the start time and description. Use comments rather than deleting source statements.