Collaborative Learning: Some Possibilities and Limitations for Students and Teachers

Matt Bower, Debbie Richards
Computing Department
Macquarie University

Collaborative learning has become recognised as a means of encouraging deep learning and a key technique in problem and experienced based learning. For Computing students collaboration is not only a learning strategy but a learning outcome. While this is not a new idea, there appears to be reluctance on the part of teachers and students to create and take those opportunities. This paper seeks to revisit the possibilities that exist for collaboration ranging from team based work to peer review in the hope of motivating a change in culture and practice. We include discussion of these strategies together with highlights from student surveys regarding student dispositions towards collaborative learning. We note that the perceived overheads and logistical difficulties, to students and teachers, will often discourage the use of collaborative tasks, but that the educational outcomes achievable through collaborative learning exceed those possible when students work in isolation. Particular attention is given to technological approaches for facilitating collaborative learning. While the discussions that follow relate to computer science education specifically, it is intended that many of the approaches and associated issues will apply to other learning domains.

Keywords: collaborative learning, groupwork, team-based learning, computing, computer supported collaborate learning.

Introduction

Collaborative learning is increasingly being recognised as a technique for improving learning outcomes (Beck, Chizhik, & McElroy, 2005; Chase & Okie, 2000; Hűbscher-Younger & Narayanan, 2003; Jonassen, Lee, Yang, & Laffey, 2005; Joseph & Payne, 2003; McDowell, Werner, Bullock, & Fernald, 2002). Based on socio-cultural learning principles (Vygotsky, 1978), collaborative learning allows students to progress beyond what they would have been able to learn alone by sharing mental models and observing the thought processes of others. As opposed to direct instruction, collaboration allows students to actively participate in problem solving processes by communicating about the conceptual representations relating to the task at hand. Collaborative approaches allow the tightly-coupled interactions required for rapid and complex concept formation to occur (Neale, Carroll, & Rosson, 2004). In collaborative learning the teacher often becomes a facilitator rather than the primary source of knowledge or control. Collaborative learning also has a range of generic skills benefits, including the development of general communication abilities, empathy, and social skills.

Employers are demanding that students have teamwork skills. A recent report based on an email survey conducted by the Australian human resources company Diversiti (Diversiti, 2006) involving 365 employees (12% response rate) across a large range of states and company sizes indicated that the most important personal characteristic of high performing IT professionals was being a team player (64%), followed by capability to perform and learn (61%), energy and optimism (46%), adaptable to changes (43%), passion for IT/Determination to succeed/multi-tasking and work balance (all 26%). Factors influencing hiring decisions placed more emphasis on other areas such as: motivation and results driven (98%), communication skills (98%) and demonstrated achievement in relevant skill area (96%). Nevertheless “cultural fit to team” scored 94%, ahead of other factors such as prior work experience (93%), references, and academic results. Results from the US are similar, with industry advising that while most graduates have satisfactory technical skills, their teamwork abilities are often deficient (Waite, Jackson, Diwan, & Leonardi, 2004). In the current climate of declining ICT enrolments delivering degrees that employees want and students need has never been more important.

From discourse with colleagues in the Computer Science Education field it appears that many computing subjects offer few opportunities for collaborative work. Since working in teams is a necessary skill to succeed in the IT industry and a good way to learn, we must ask ourselves why collaborative learning is not used more frequently. We offer the following reasons:

Fear of plagiarism or freeloading: inactive team members won’t be presenting their own work but will be presenting the work of their team members, which is unethical and also a potential source of friction between students.

Student Reticence: Students want to receive credit where credit is due. When they are part of a team, their individual efforts may not be recognised. As well, students (especially computing students) may want to avoid peer interaction either due to self-confidence, lack of enjoyment, or for the increased time cost involved in collaborating with others.

Effort: it takes a lot of effort to design activities that involve interaction and collaboration and even more effort to manage if a group or activity gets into trouble. This is exacerbated when academics are not comfortable or informed when it comes to implementing collaborative techniques.

Assessment: if people work in a team and produce a combined deliverable, it is difficult to accurately identify the contribution of each student and fairly apportion the marks. Some institutions, including Macquarie, place restrictions on the amount of groupwork permitted for this reason.

Technical support: technology is sometimes needed to assist the group to work due to differing study/work commitments, stages of life, and locations. Academics may avoid collaboration using online approaches due to unfamiliarity or because of the administrative overhead and risk of technical problems that it carries.

Taken together, these reasons present a persuasive disincentive to adopt collaborative approaches. In order for academics to utilise collaborative learning techniques they need to be aware of the range of opportunities available, convinced of their potentials, and provided with support to navigate possible pitfalls. In the next section results from a student survey provide substantial evidence supporting the use of collaborative learning techniques, as well as highlighting some of the associated issues surrounding implementation of such techniques. In the section that follows a range of collaborative strategies based on findings from the literature and our own experience are presented, as well as research results, for evaluative purposes.

Collaborative Learning – The Students' Perspective

In the US, Barker et al. (Barker, Garvin-Doxas, & Jackson, 2002) conducted a National Science Foundation funded ethnographic research to study the nature of the learning environment in computer science classrooms. They found that a culture of guarded behavior in an impersonal environment, and the presence of informal hierarchies of learners had led to competitive behaviors. They concluded that the defensive climate based on competitiveness rather than cooperation, judgments about others, superiority and lack of empathy inhibited collaboration, and that this culture needed to be addressed before effective learning could occur.

In order to ascertain our students' disposition towards collaborative learning, in 2004 we conducted a survey relating to various aspects of our teaching and student learning. A 30 item online questionnaire was issued to 196 of our undergraduate students in second and third year and yielded 103 responses. The instrument was comprised of both open ended and closed (seven point Likert scale) items. Closed response options ranged from very low (0) through to very high (6).

When asked to indicate "the importance of being able to work in a group for programming related roles" students indicated an average Likert scale rating of 5.2 out of a possible score of 6 (between 'high' and 'very high'). Students also strongly believed (average 5.10 out of 6) that they "need to be taught to work effectively with others". These responses were the highest scores of any questions on the survey instrument, and each was highly significant ($p < 0.001$), indicating the importance of providing opportunities for students to work in groups and learn about working in groups.

Students indicated that they spent an average of 28% of their time working with others. When asked how much time they would like to spend working with others the response was an average of 44%. This was a highly significant difference between the observed and desired amount of time spent working with others (paired t-test, $t = 7.78$, $df = 99$, $p < 0.001$). The 16% absolute difference becomes a more substantial figure when it is considered as a proportion; on average students wish to spend almost 60% more time working with others. Seventy-five percent of respondents believed that lecturers should provide more formal opportunities for working with others in the activities they set.

Students were also asked to rate their confidence in working with others, their enjoyment of working with others, and their general communication abilities. A positive linear correlation existed between the student's confidence in working with others and their self-perceived communication abilities ($\hat{\beta}_1 = 0.451$,

t = 5.07, p < 0.001). Further there was a positive linear correlation between communication abilities and enjoyment of group work ($\hat{\beta}_1 = 0.396$, t = 3.60, p < 0.001). These results indicate that communication abilities affect confidence and enjoyment when working in groups. By providing structured collaborative opportunities that support the development of communication skills, academics can improve student confidence and enjoyment when it comes to working with others.

Time was identified as a critical factor for our students. The majority of students spent more than the prescribed 12 hours a week per computing subject, with several indicating that they spent more than 25 hours per week. Some students may be reluctant to participate in group projects and the like for the potential administrative and communicative overheads that may be incurred. Given the time impoverished nature of today's learners, it is important that collaborative processes are not only designed to be valuable learning experiences but also effective in terms of the time-cost they involve.

When asked what proportion of time was spent in activities such as reading, studying, programming, debugging and so on, assignments were generally the major component requiring 20% or more of students' time towards their computing subjects. Surprisingly, many indicated that they placed little emphasis on reading the text book and only 58 of the students responded that they found the lectures to be at least moderately helpful or helpful in learning computing. The most common response (20/74) to an open-ended question concerning what students would find most helpful to learn computing was "practical application/frequent problem solving practice". If students feel that practical activities are most effective in developing their understanding then academics need to support that process. Prescribing groupwork activities that allow for collaborative programming provides students with a way to help one another. This may not only help relieve some of the time pressures on students but also support the development of important generic problem solving abilities by observing the generic troubleshooting strategies of others (Jonassen & Hung, 2003). In the survey, students identified these potential benefits. When asked as to the benefits of collaborative learning, they cited alleviate frustration when stuck (27) as the most popular reason, followed by better understanding (16) and seeing multiple perspectives (12).

Nearly eighty percent of students identified some ways in which learning computing in isolation was inferior to working with their peers. However, students also identified some disadvantages of working in groups. Reasons such as less distractions (28), can focus on concept formation/difficult problems (14), can choose own pace (13), more time efficient (12) and less conflict (5), were cited.

We note that respondents considered individual study to still be an essential parts of learning computing and thus our goal must not be to remove the opportunity for self-directed learning. As well, students may have some reservations about groupwork activities and display some reluctance in practice (Barker, Garvin-Doxas, & Jackson, 2002; Waite, Jackson, Diwan, & Leonardi, 2004). Yet based on the general feedback from this research there is solid support from students for academics to pursue collaborative learning activities.

Opportunities for Collaborative Learning

Given the value of collaboration identified by industry, students, and educators (below), we offer for consideration a range of potential options that involve some form of collaborative learning. The categories include research results from educational literature, to indicate the range of opportunities available and the relative merits of each. It should be noted that some of the techniques overlap to some extent and fall under multiple categories. To act as a point of reference for other educators, we discuss some of the options we have explored and issues we have encountered in our own implementations at Macquarie University.

In-class Collaborative Activities

The most direct and supervised way to encourage collaboration is through in-class collaborative activities. Several approaches are recommended in the literature.

Beck, Chizhik, & McElroy (2005) investigate the use of cooperative learning in introductory computing classes. The specific roles assigned in cooperative learning tasks (such as program reader, method executor, facilitator) provide a structure to support more equal involvement of all students. Roles also highlighted the different aspects of the programming process, and reducing the scope of responsibility provided a form of reductionism to help students avoid the cognitive overload that is often experienced when first learning to program. They found that there was a statistically significant difference between the

cooperative learning group and the non-cooperative learning group on the final exam ($p = 0.010$). As well while the grades throughout the semester decreased for the non-cooperative learning group, they improved for the cooperative learning group (a highly statistically significant different in slopes, with p-value = 0.0073).

A conversational classroom environment is recommended by Waite, Jackson, & Diwan (2003). They describe how transforming the classroom into a more conversational environment (both between students and with the professor) lead to a doubling of the percentage of A grades. This transformation from traditional transmission approaches to a more engaged and participatory environment promoted the development of shared understandings. This collaborative model requires students to adopt a more active, responsible approach to their education. Waite et al. identify the two primary resources for implementing their approach as being (1) techniques for creating interaction and (2) techniques for creating a sense of presence. They also identify two behaviors – persistence and commitment to emergence – that are critical to creating and sustaining the system as a whole.

Van Gorp & Grissom (2001) suggest collaborative activities such as:

- Code walkthroughs – where students step through provided code to predict output
- Group code writing tasks – such as "write psuedocode to simulate 500 coin tosses"
- Group debugging – where student teams find syntax and logical errors in a program
- Lecture note reconstruction – whereby students reconstruct lecture outlines from memory.

They report that the amount of student perceived collaborative activity in the course was positively correlated with student grades.

At Macquarie we have also experimented with the strategy of in-class student presentations. In our year long industry and group based unit, students meet weekly for an hour with the unit convenor to present topics from the textbook to one another. Students present for 15 minutes and the task is worth 10% of their total assessment. For a number of years these presentations were considered boring and poorly attended. In the past two years, by adding a mark for the level of engagement and making this activity peer assessed there has been a major turn around. Students have been treated to quizzes, games, movie tickets and chocolate bars by inventive presenters. This highlights the way in which simple changes to the structure and assessment of collaborative tasks can positively affect performance.

As well, automated response technology has been incorporated into some lectures (Braiding, Richards, & Vaughan, 2006). Handheld keypads and supporting software are used to elicit, evaluate and present student responses to questions posed by the teacher in lectures and tutorials. The anonymity provided by the system, as opposed to asking students to vote for a solution by raising their hand, encourages shy or unconfident students to participate. The group results provide immediate feedback to the student as well as the teacher. The system can be programmed to automatically collect individual responses, show solutions and prepare selected statistics for student and teacher diagnosis.

Macquarie University is also using the Adobe Breeze platform to facilitate in-class collaborative approaches to learning (Bower, 2006). Since the text chat does not interfere with the lecturer's audio broadcast, students can hold discussions about the material during an instructors' presentation. It is also possible for all students to respond to a lecturer question simultaneously, improving both the degree of involvement and the efficiency of collaboration. Under this form of collaboration students feel that they both collaborate more and learn more as opposed to face-to-face (Bower & Richards, 2005).

Other educators also report the benefits of using technology to assist in class collaborative approaches. Simon et al. (2004) investigated the use of Tablet PCs to promote active learning in computer science. They point out that such technology has the advantage of allowing instructors to select answers rather than students – the technology has the ability to filter out identities more easily than in a face-to-face classroom. As well, they note that sharing of different students (or groups of students) spontaneous attempts to solve problems "provided an opportunity to point out common mistakes and allowed comparison of different approaches to the same problem" and that "the instructor also received immediate feedback on whether or not students understood" (p. 215).

Research by Graciela (2006) indicates that in-class collaborative approaches may have positive effects beyond the completion of the subject in which they occur. They deployed a systematic in-class collaborative approach involving group discussions to review previous work (followed by a short lecture to present new material) and planned groupwork activities. This resulted in 70% of students who were exposed to the active learning experience in CS1 also passing CS2, as opposed to only 44% who weren't

exposed to the active learning approach in the previous subject. As well, the dropout rate in CS2 was only 10% for students who underwent active learning in CS1 as opposed to 25% for those who did not.

Peer Review

Learning can be assisted by reviewing the work of peers and conversely by receiving review from peers. Peer review allows students to integrate vertically with more advanced students and have students learn from exposure to more coding applications without the burden of having to program it themselves. As well as there being advantages of peer review for students, there are also several advantages for academics. For instance, Gehringer notes that their online submit and peer review system builds a databank of code relating to specific topics that can then be used as resources for future activities (Gehringer, Chinn, Perez-Quinones, & Ardis, 2005).

Students appreciate peer review approaches. Gehringer (2001) reports on a Submit-Review-Publish cycle for learning computing. The reviews allowed students to revise their solutions before final submission. Each submission was double-blind reviewed by multiple reviewers, in much the same way as many conference paper submissions are handled. The benefits, as with conference paper refereeing is that the reviewer is kept informed of the latest research in the area while also offering their insights and comments based on their expertise. Feedback from students indicated that the approach helped them to improve the quality of their work.

In other instances, systems have been specifically designed to provide the benefits that can be derived from peer review processes. The Collaborative Algorithm Representations Of Undergraduates for Self-Enhanced Learning (CAROUSEL) system was purpose built to facilitate peer evaluation and feedback of student created visual representations of computing algorithms (Hűbscher-Younger & Narayanan, 2003). Whereas previous attempts with using animations to improve student comprehension of computing algorithms had proved unsuccessful, researchers found a significant positive relationship between these constructive and collaborative feedback-based activities and algorithm learning.

At Macquarie we have often used in-class peer review, to offer both parties the opportunity to gain a better understanding of the concepts in a timely manner. This process allows the reviewee to use findings to immediately redevelop their initial submission, and the reviewer to apply the reflections to their own work. These capacities to build on the work of others, review code, conduct structured walkthroughs and various other formal and informal review methods are valuable system development and maintenance skills that our IT graduates require for the workforce.

As an extension to peer review, students' marks or comments can be either directly assigned or used to guide assessment marks allocated by a teacher. For instance, in the Submit-Review-Publish approach used by Gehringer (2001) the average of the blind reviewers marks was used. At Macquarie we have used peer assessment to assign marks in two third-year units. In one unit, students assess the presentations of other students. In the other unit, students assess the software solutions offered to a problem that has been posed to all groups. For quality assurance purposes the lecturer has continued to perform their own assessment. Astonishingly, the average mark based on the students' scores (15-40 students) tended to be within half a mark of that awarded by the lecturer, though on some occasions the deviations can be large. An alternative strategy to ensure equity is to have the same student/s mark the same question/task for the whole class to allow them to develop a more refined sense of the differences between solutions.

Adjunct Collaborative Frameworks

There are several ways in which academics can provide general collaborative frameworks to support their courses. For instance, Chase and Okie (2000) introduced a peer and cooperative learning framework in a first course for Computer Science majors. This involved appointment of an undergraduate peer instructor as a co-teacher, and careful formation of groups based on personality types and using team-building activities. Groups were responsible for their members and if anyone fell behind or couldn't attend classes the group were responsible for bringing them up-to-date. Each new lecture topic was followed by a group assignment, which was then followed by an individual assignment. The results at the end of the term showed an improvement from 56% receiving a conceded pass or below to 32%. Significantly, the rate for female students changed from 53% to 15%.

Scharff & Brown (2004) report on the efficacy of creating holistic Learning Communities as a means to improving learning outcomes. Their Learning Community approach involved integrating curricula of a course in logic with their introductory computer science unit in order to provide a mutually supportive framework for the logico-deductive aspects of the material. By identifying a common group of students

and prescribing common assessment tasks between the two subjects, student feedback indicated the learning community approach offered significant support to their learning and was a contributing factor to their improved understanding.

Mentoring is another form of student interaction that can provide benefits to both the mentors and their protégés. Mentors don't necessarily need to be much older or significantly more experienced, but they do need to have at least experienced the process that their apprentice is going through, such as having completed the unit of study successfully in the past. Protégés are provided with technical and social support, while mentors benefit from improved communication skills, confidence building and intrinsic rewards. In the mentoring program in our large first-Computing unit the students that volunteered to meet weekly with their two mentors reported that they found the experience helpful. A similar program at Sydney University found that 94% of mentors from one year were willing to volunteer for the program in the following year (Miller & Kay, 2002). Providing students with industry placements is another way to offer students the possibility for mentor relationships to be developed.

Extended Team-Based Tasks

A common way that educators seek to incorporate collaborative learning is via the use of group-based tasks. A typical example might include students working together for a significant proportion of the subject to deliver a substantial artefact such as a software system and/or supporting documentation. Here we are concerned with groups that form to solve a large problem or perform an extensive task that will be assessed. There are a variety of approaches that can be adopted.

The University of Sydney Basser Department of Computer Science has experimented with Problem Based Learning (PBL) approaches to teaching programming (Kay et al., 2000), which emphasises:

1. open ended, authentic and substantial problems which drive learning
2. explicit teaching and assessment of generic and metacognitive skills, and
3. collaborative learning in groups.

In the approach students are presented with an authentic problem (for instance, implementing a supermarket checkout queue simulator) that is used both as a driving force to develop metacognitive skills (for instance reflection upon steps taken to solve the problem and delegation of time) and for the subject of groupwork (externalising knowledge, developing collaborative skills). Kay et al. (2000) found that over the two year period since the introduction of Problem Based Learning into their Introduction to Computer Science unit the mean examination mark improved from 63% to 91%. They also note some positive affects on student satisfaction with the unit, as measured through an open ended questionnaire they deployed.

Joseph and Payne (2003) tested the use of cooperative learning groups in an undergraduate computing course. All course requirements were to be met via in class and out of class interaction with their group, however, individual grades were awarded for each task. Group tasks included textbook problems, in-class activities and a group project, three in-class and one final examination. It was found that students with the highest activity scores in the cooperative learning groups also scored significantly better in the final exam, indicating that engagement is a critical factor in successfully implementing extended team based work.

Technology is becoming a popular mechanism to implement distributed collaborative projects. Hause et al. (2001) contrasted the collaborations of a high and low performing team that participated in the Runstone project. This project involved distributed teams of third year students from Uppsala University (Sweden) and Grand Valley State University (US) creating software in teams using email, IRC chat, and shared webspace. Cheng and Beaumont (2004) have also used multimodal Computer Supported Collaborative Learning (CSCL) techniques to distributed PBL (dPBL) tasks. Finally, at Macquarie we have experimented with the use of wikis to facilitate semester long group projects in our Master of IT course (Bower, Woo, Roberts, & Watters, 2006). The successfulness of these approaches as compared to face-to-face techniques is still a point of conjecture.

Macquarie University's computing degree includes a project-based unit in which students experience working in a team. This approach provides the benefits of situated learning, showing students the relevance of their studies and offering them industry experience. However, such experiences need to be carefully managed. Management is heavily dependent on the structure and context (for instance whether placement is for individuals or a cohort of students). Key factors to consider are how individuals/groups are allocated, how resources are distributed and how activities are monitored.

Often collaborative project tasks can carry an assessment weighting, which can be a concern in terms of fairness and equity. Strategies that we have found can address this include:

- private submission of an 'Individual Contribution Form' to more accurately determine the actual contribution by each student
- the use of interviews and student journals to determine the extent of student activity in a task
- allocation of members to the group based on aptitude (for instance, grade point averages) to avoid disadvantaging a more capable member of the group due to biases in ability.

Out-of-class distributed activities

Online technologies allow academics to design collaborative activities that do not require students to physically assemble or even be online at the same time. An advantage of these sorts of approaches is that the technology generally allows the teacher to track collaborations for assessment and review purposes. Typical technologies include discussion boards, wikis, and virtual classrooms.

Clark (2000) points out that discussion boards allow students to make more reflective, extended contributions than face-to-face discussions. By providing temporal flexibility to collaborative design activities students are able to carefully compose their thoughts and make more complex, interrelated responses. Clark notes how for these sorts of tasks more structured specification positively affected the quality of postings.

Wikis also allow students to perform asynchronous online collaboration, with the added capacity to structure, re-structure and interlink content. Learning designs can include collaborative artefact creation, the formation of micropedias and glossaries, and mini-problem solving activities. Macquarie University has experimented with the use of wikis for facilitating weekly extension activities, with students indicating that the wiki allowed them to form negotiated meanings that may not have emerged if they were working face-to-face (Bower, Woo, Roberts, & Watters, 2006).

Finally, advances in virtual classroom technologies allow students to collaborate on distributed tasks in a synchronous mode. Macquarie University is using the Adobe Breeze platform not only for online classes, but also to facilitate out-of-class collaborative activities such as group-programming and pre-tutorial preparation tasks (Bower, 2006). These approaches allow students to access immediate support and feedback while still being out-of-class and potentially separated by large distances.

Programming in Teams

'Programming in teams' is a subset of both extended team-based tasks and in-class activities which deserves special attention because of its applied, skill based nature and its frequent deployment in the Computer Science curriculum. System development methodologies such as Extreme Programming (XP) and Rapid Applications Development (RAD) have been used in industry for some time now. Concepts such as pair-programming (used in XP), version control, change management, integrated development environment (IDEs), documentation, testing and debugging only start to make real sense when students work on programming projects that require teams. Research relating to the use of programming in teams (specifically, pair-programming activities) as a learning strategy has been invariably positive in the results they report, as indicated by the following examples.

McDowell et al. (2002) examined 313 students across two semesters to gauge the effect of pair programming. In the semester where pair programming was used to complete assignments, the scores that students received were significantly higher ($n_1 = 172$, $\bar{x}_1 = 86\%$, $s_1 = 14\%$, vs $n_2 = 141$, $\bar{x}_2 = 67\%$, $s_2 = 20\%$, two sample t-test $p<0.001$). As well, the course completion rate for the semester where pair programming was used maintained a 92% completion rate as opposed to the non-pairing semester which maintained a 76% completion rate.

Nagappan et al. (2003) also investigate the use of pair programming in their introductory computing course. They report that pair programmers were more self-sufficient, generally performed better on projects and exams, and were more likely to receive a grade C or higher for the course than their solo counterparts. As well, both students and lab instructors report a more productive and less frustrating laboratory environment. They note that this supports several previous studies which indicated that pair programmers produce higher quality code in about half the time of solo programmers.

Williams and Upchurch (2001) support the qualitative outcomes of pair programming activities. They note that pair programming not only enhances student learning and satisfaction, but reduces the amount of

support that students require from educators. They found that the pair-programming approach they implemented led to higher quality code as a result of the continuous review provided within teams. They also describe the team-building and communication skills development that the strategy encourages.

While pair programming is not necessarily an industry standard, designing, developing and testing code in groups is the norm. To this end we have incorporated group-based programming assignments at second year level and will explore providing this experience in first year. Based on our research into collaborative learning and the identified need to carefully select groups, provide clear and well designed task specifications, implement systems to ensure equal participation and fair discrimination between individual contributions in assessment, we have been able to design and implement a collaborative experience that the students deem positive. The feedback reported below relates to a second year unit which provided approximately 120 students with their first introduction to Object Oriented Programming using Java.

1. I learnt more by working in a group than if I'd done the assignment on my own. SD-5%, D-9%, A-53%, SA-33%
2. I enjoyed working in a group SD-5%, D-7%, A-57%, SA-31%
3. It was easy to get the group to work together SD-10%, D-22%, A-48%, SA-20%
4. It was easier to get the task done by working in a group than if I'd done the assignment on my own. SD-11%, D-18%, A-44%, SA-27%
5. I would like more assignments to involve group work. SD-14%, D-21%, A-45%, SA-20%
6. Everyone in my team did an equal share. SD-5%, D-17%, A-55%, SA-23%
7. Most people in my team did an equal share SD-3%, D-9%, A-56%, SA-32%
8. I think programming and testing works well with groups SD-4%, D-18%, A-43%, SA-35%
9. I think training to work in groups is needed SD-3%, D-13%, A-36%, SA-38%
10. I think choosing our own groups is best SD-1%, D-2%, A-31%, SA-56%

Table 1: Survey questions and student responses

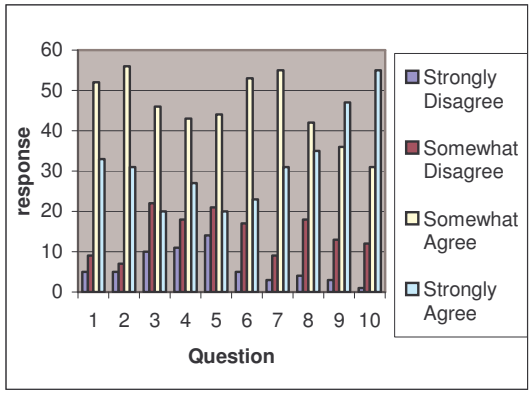

Figure 1: Table 1 as a bar chart. Clearly students indicated that working in groups was beneficial. Refer to Table 1 to look up the question

The participation rate and normalised average mark in the groupwork assignment were higher than the other two assignments, though it is difficult to compare the three assignments which sought to test different skills, knowledge and competencies at different phases of the course. However, simply because a team was involved, a bigger problem could be given and a more comprehensive solution produced. Taken together, better performance and increased student satisfaction demonstrate that programming in teams can operate successfully and improve educational outcomes.

Summary and Conclusion

This paper provides evidence supporting the use of collaborative learning, as well as a number of strategies that can be used. Issues relating to the various approaches have been discussed and comparisons have been made. Implementing group based tasks carries an intrinsic administrative overhead for teachers and students. Yet students need to have experience with working in groups if for nothing else to develop the generic skills required by industry. For that reason alone lecturers need to provide collaborative learning opportunities, which means understanding the various approaches, their disadvantages and their issues. However we have shown that in addition to developing generic skills, collaborative approaches also have the potential to improve learning outcomes. A summary of the possibilities and limitations is provided in Table 2 below. The summary is not intended to be definitive but hopefully presents a basis from which debate can occur.

No matter which approach to collaboration is adopted, it has become clear to us that success lies in the implementation, and not in the specific approach. However there are differences between approaches, and as such educators need to carefully match the collaborative approach to the learning requirements of the task. By presenting a range of collaborative strategies available and identifying the limitations and possibilities inherent in each it is intended that other academics will be both more inclined and able to integrate collaborative learning into their curricula from a more informed point of view.

	In-class Group Activities	Peer Review	Adjunct Collaborative Frameworks	Extended Team-Based Tasks	Out-of-class Distributed Activities	Group Programming
Plagiarism/ Freeloading	Potentially	No, if well structured	Potentially	Potentially	Low, since trackable	Potentially
Student reticence	Potentially	low	Low	Potentially	Low, due to flexibility	Potentially
Teacher effort	High	High	Initially high, lower ongoing	High maintain, lower marking	Low	Initially high low ongoing
Assessment discrimin. issues	Potentially	Low	No	High	Low	Potentially
Technological overhead	High if using collab. tech.	High if using online system	High if through collab. tech.	High if using collab. tech.	High	Low
Develops social networks	High	Depends on design	High	High	Depends on design	High
Troubleshooting Support	High	Low	Potentially	High	Potentially	High
Teachers communication	High	Medium	Medium	High	High	High
Meets industry needs/skill	Potentially	Review skills	Low	Project skills	High	Yes
Engagement with problem	High, espec. cooperative tsks	Medium	Low	High, unless freeloading	High, due to accountability	High, unless freeloading

Table 2: Summary of the issues and benefits of different collaborative learning strategies

References

Barker, L. J., Garvin-Doxas, K., & Jackson, M. (2002). *Defensive climate in the computer science classroom. In*, Proceedings of the 33rd SIGCSE technical symposium on Computer science education, pp. 43-47.

Beck, L. L., Chizhik, A. W., & McElroy, A. C. (2005). *Cooperative learning techniques in CS1: design and experimental evaluation. In*, Proceedings of the 36th SIGCSE technical symposium on Computer science education, St. Louis, Missouri, USA, pp. 470-474.

Bower, M. (2006). *Virtual Classroom Pedagogy. In*, Thirty-Sixth SIGCSE Technical Symposium of Computer Science Education, St. Louis, Missouri, USA, pp. 148-152.

Bower, M., & Richards, D. (2005). *The Impact of Virtual Classroom Laboratories in Computer Science Education. In*, Thirty-Sixth SIGCSE Technical Symposium of Computer Science Education, St. Louis, Missouri, USA, pp. 292-296.

Bower, M., Woo, K., Roberts, M., & Watters, P. (2006). *Wiki Pedagogy - A Tale of Two Wikis. In*, Information Technology based Higher Education and Training, Sydney, pp. 187-198.

Braiding, C., Richards, D., & Vaughan, A. (2006). *Fun and feedback at the press of a button. In*, Proceedings of the Australasian Society for Computers In Learning in Tertiary Education, Sydney, NSW.

Chase, J. D., & Okie, E. G. (2000). *Combining cooperative learning and peer instruction in introductory computer science. In*, Proceedings of the thirty-first SIGCSE technical symposium on Computer Science Education, Austin, Texas, United States, pp. 372-376.

Cheng, C. S., & Beaumont, C. (2004). Evaluating the effectiveness of ICT to support globally distributed PBL teams. In *Proceedings of the 9th annual SIGCSE conference on Innovation and technology in computer science education* (pp. 47-51). Leeds, United Kingdom: ACM Press.

Clark, M. (2000). *Getting participation through discussion. In*, Proceedings of the thirty-first SIGCSE technical symposium on Computer science education, Austin, Texas, United States, pp. 129--133.

Diversiti. (2006). Diversiti IT Hiring Influence Report. Retrieved 12th September, 2006, from http://www.diversiti.com.au/Diversiti_CMS/Content/Documents/Diversiti%20IT%20Hiring%20Influence%20Report%202006.pdf

Gehringer, E. F. (2001). *Electronic peer review and peer grading in computer-science courses. In*, Proceedings of the thirty-second SIGCSE technical symposium on Computer Science Education, pp. 139-143.

Gehringer, E. F., Chinn, D. D., Perez-Quinones, M. A., & Ardis, M. A. (2005). *Using peer review in teaching computing. In*, Proceedings of the 36th SIGCSE technical symposium on Computer science education, St. Louis, Missouri, USA, pp. 321-322.

Graciela, G. (2006). *A systematic approach to active and cooperative learning in CS1 and its effects on CS2. In*, Proceedings of the 37th SIGCSE technical symposium on Computer science education, Houston, Texas, USA, pp. 133-137.

Hause, M. L., Almstrum, V. L., Last, M. Z., & Woodroffe, M. R. (2001). *Interaction factors in software development performance in distributed student teams in computer science. In*, Proceedings of the 6th annual conference on Innovation and technology in computer science education, Canterbury, United Kingdom, pp. 69-72.

Hűbscher-Younger, T., & Narayanan, N. H. (2003). *Constructive and collaborative learning of algorithms. In*, Proceedings of the 34th SIGCSE technical symposium on computer science education, pp. 6-10.

Jonassen, D. H., & Hung, W. (2003). Learning to Troubleshoot: A New Theory-Based Design Architecture. Retrieved September, 2006, from http://www.coe.missouri.edu/~jonassen/Troubleshooting.pdf

Jonassen, D. H., Lee, C. B., Yang, C.-C., & Laffey, J. (2005). The Collaboration Principle in Multimedia Learning. In R. E. Mayer (Ed.), *The Cambridge Handbook of Multimedia Learning* (pp. 247-270). New York: Cambridge University Press.

Joseph, A., & Payne, M. (2003). Group dynamics and collaborative group performance. In *Proceedings of the 34th SIGCSE technical symposium on Computer science education* (pp. 368-371): ACM Press.

Kay, J., Barg, M., Fekete, A., Greening, T., Hollands, O., Kingston, J. H., et al. (2000). Problem-Based Learning for Foundation Computer Science Courses. *Computer Science Education, 10*(2), 109-128.

McDowell, C., Werner, L., Bullock, H., & Fernald, J. (2002). The effects of pair-programming on performance in an introductory programming course. In *Proceedings of the 33rd SIGCSE technical symposium on Computer science education* (pp. 38-42): ACM Press.

Miller, A., & Kay, J. (2002). *A mentor program in CS1. In*, Proceedings of the 7th annual conference on Innovation and technology in computer science education, Aarhus, Denmark, pp. 9-13.

Nagappan, N., Williams, L., Ferzli, M., Wiebe, E., Yang, K., Miller, C., et al. (2003). *Improving the CS1 experience with pair programming. In*, Proceedings of the 34th SIGCSE technical symposium on computer science education, pp. 359-362.

Neale, D. C., Carroll, J. M., & Rosson, M. B. (2004). *Evaluating computer-supported cooperative work: models and frameworks. In*, Proceedings of the 2004 ACM conference on computer supported cooperative work, Chicago, Illinois, USA, pp. 112-121.

Scharff, C., & Brown, H. (2004). Thinking Through Computing: The Power of Learning Communities. *Computer Science Education, 14*(4), 297-320.

Simon, B., Anderson, R., Hoyer, C., & Su, J. (2004). *Preliminary experiences with a tablet PC based system to support active learning in computer science courses. In*, Proceedings of the 9th annual SIGCSE conference on Innovation and technology in computer science education, Leeds, United Kingdom, pp. 213-217.

Van Gorp, M. J., & Grissom, S. (2001). An Empirical Evaluation of Using Constructive Classroom Activities to Teach Introductory Programming. *Computer Science Education, 11*(3), 247-260.

Vygotsky, L. S. (1978). *Mind in Society*: Cambridge, MA: Harvard University Press.

Waite, W. M., Jackson, M. H., & Diwan, A. (2003). *The conversational classroom. In*, Proceedings of the 34th SIGCSE technical symposium on computer science education, pp. 127-131.

Waite, W. M., Jackson, M. H., Diwan, A., & Leonardi, P. M. (2004). *Student culture vs group work in computer science. In*, Proceedings of the 35th SIGCSE technical symposium on Computer science education, Norfolk, Virginia, USA, pp. 12-16.

Williams, L., & Upchurch, R. L. (2001). In support of student pair-programming. In *Proceedings of the thirty-second SIGCSE technical symposium on Computer Science Education* (pp. 327-331): ACM Press.