Organised Chaos – Learning Outcomes from trialling Active Learning Methods in Computing Science

Marie Devlin*, Chris Phillips, Lindsay Marshall

Index Terms: *Active and Cooperative Learning, Team Work, Software Engineering*

Active Learning in Computing is a 5 year CETL project funded by HEFCE (UK). It involves a consortium of Universities from the North West of England, comprising Durham University as project lead and Newcastle University, University of Leeds and Leeds Metropolitan University as partners. The project aims to make Computing Science group and project work realistic and relevant to industry by introducing innovative teaching methods that foster independent learning and facilitate greater student engagement with the Computing Science curriculum.

As of 2008, we are approximately half-way through the project and are evaluating the impact of the active learning activities we have introduced on student learning outcomes in Undergraduate Computing Science. This paper presents a comparison of module results and student learning experiences on Newcastle's Software Engineering module over four academic years - 2003-2007, i.e. two years before and two years after introducing active learning via a year-long cross-site project between collaborating student teams at Durham and Newcastle.

In the first part of the paper we describe the module and compare the curricula and teaching approaches employed before and after active learning methods were introduced. We define our understanding and approach to active learning and outline the pedagogical philosophy that underpins it. We then give an overview of the activities and techniques we introduced during the latter two years of this study and briefly outline our experiences. In the next section of the paper we present the year-on-year results, but knowing that it is difficult to draw real conclusions from these we attempt to draw qualitative conclusions as evidenced by the free-format responses in questionnaire returns and student reflective reports. We then use these results to evaluate the impact our change to an active learning approach has had on student learning outcomes in the module and to provide some guidance for other practitioners should they wish to introduce similar methods to their own teaching.

Introduction

At Newcastle University, the level 2 Software Engineering module in Computing Science runs over one full academic year. It gives students the opportunity to work on a substantial software engineering project within a team. The module aims are to introduce issues regarding programming "in the large", including development models, project planning and management, team working etc. and to provide practical experience of software development in a simulated industrial setting. The intended learning outcomes for students are practical experience in issues such as team structure, task allocation, document preparation, project management and the design and implementation of a large software system, the ability to work as a member of a team, to fulfil appropriate roles and to apply these skills and other skills to the job at hand[1]. In recent years, it has become more commonplace to use virtual teams to develop software. Companies such as IBM, Microsoft, and Sun Microsystems now rely on these virtual teams to develop software for students with the ability to cope in dynamic environments and the flexibility to work in virtual teams on many projects at once [2].

[1] e-mail: marie.devlin@ncl.ac.uk

Active Learning in Computing (ALiC) is a Centre for Excellence in Teaching and Learning project (CETL) funded by the Higher Education Funding Council for England [3, 4]. It is a collaborative effort between four partner institutions: Newcastle University, Durham University (CETL lead), Leeds Metropolitan University and The University of Leeds. During the five year project, the ALiC team are exploring innovations in group and project work in Computing Science in order to better prepare students for the realities of working in the software industry. Due to the evolving practice of multi-site working in the software development industry and to increase the employability of our students, CETL ALiC staff made changes to the way we run our Software Engineering module at both Newcastle and Durham. We now run the module with cross-site software development teams forming virtual companies, emulating practice in the real world and giving students the chance to practice their transferable skills in a realistic scenario. Very few students will have had this experience during their degree studies. Here at Newcastle we wanted to evaluate how our introduction of active learning techniques has impacted on the student learning outcomes. In this paper we describe our teaching approaches before and after the implementation of active learning. We then outline the changes we made to our Software Engineering module and the active learning techniques we implemented. We discuss our results in terms of learning outcomes for students using quantitative module results and qualitative student feedback on their experiences. We then use these results to evaluate the impact of our introduction of active learning and in the final section of the paper we provide some guidance for others who wish to use a similar approach in their teaching.

Teaching Approaches

Prior to 2005, Software Engineering at Newcastle was taught in the manner of most level-2 Software Engineering modules, with lectures, laboratory-based practical sessions and assessments with an individual or group focus. The module had gradually evolved to have a strong team-based focus. The main reason for this change is that the Software Engineering discipline itself had adapted in order to emulate industrial practice. Newcastle students worked together in local teams throughout the year to produce a software product and the associated documentation at the end of the year. Lectures for the module were 'front-loaded' during the first five weeks of semester one and then a series of guest lectures from employers took place throughout semester two. Students were told to meet in their teams at least once a week to work on the project. There were no formal laboratory practicals but practical time was set aside in the timetable and lab space made available so that students could work on their own and manage their time themselves. A staff member was appointed for each team as a 'monitor' who observed team-interactions for the purpose of assessment.. The lectures for the module gave a flavour of Software Engineering as a discipline and provided an overview of the software development process including team structures and process models and the stages of the Software Lifecycle i.e. requirements analysis, design, configuration management, project management, project planning, testing and debugging. During the lectures we put the material in context, highlighted important concepts and key features and offered a perspective of the discipline [1].

From 2005 onwards, with the advent of CETL ALiC and our focus on the introduction of active learning in group and project work, the module was again modified. We decided to retain the series of lectures in semester one and the format of guest lectures from industry in semester two. We also retained the use of monitors and the available timetable slots and lab-space for teams to conduct independent laboratory work. We made sure that the laboratory times were the same at both Newcastle and Durham so students had time available to collaborate across sites. The teams at each site formed 'virtual companies' each made up of a team, comprising on average 6 students from each site. We retained the informal and independent laboratory time at Newcastle, whereas at Durham practical sessions were supported by demonstrators who could assist with technical questions. Students were given access to a shared repository for code and documentation and communication technologies in order to collaborate. The technology provided included applications such as video

conferencing, Skype – (VOIP technology), MSN as well as shared company and local team forums for communication via our VLE – Newcastle E-learning Support System, (NESS), as shown in Fig. 1. [5, 6, 7].

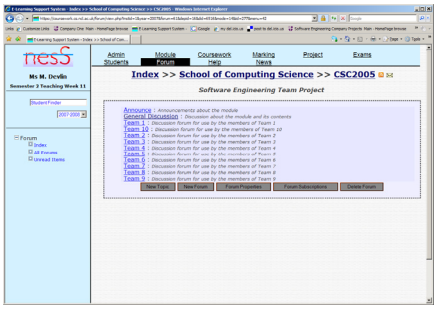

Fig. 1. NESS: **N**ewcastle **E**-learning **S**upport **S**ystem

The ALiC project is really about promoting change and we want to determine effective teaching and learning approaches that promote greater learner autonomy, embrace learner diversity and increase students control in determining the pace and style of their learning experiences [3]. For us, as for many tutors who implement active learning activities, there is no hard and fast definition of 'Active Learning'. It rather depends on the discipline, the group of learners, and the degree of emphasis that tutors want to place on it in their teaching. However, we have adopted a broad working definition – we are in agreement with Denicolo et al, that active learning has "four distinctive features – a search for meaning and understanding, greater student responsibility for learning; a concern with skills as well as knowledge and an approach to the curriculum which looks beyond graduation to wider career and social settings" [8]. An Active Learning approach gives student greater control over the subject matter they are taught, the teaching and learning resources they can use and the learning methods that they employ. It focuses on experiential learning or learning by doing. In our implementation of active learning, students learn through tackling relevant 'real world' problems and reflect on their learning by participating in discussions, writing reflective reports and exploring their own attitudes and values. David Kember and Doris Y. P. Leung outlined a survey in which Hong Kong University found their part-time students perceived a significantly higher development in 8 out of 9 capabilities than their full-time counterparts – the reason being that part time courses were more likely to have more teacher-student interaction and more likely to employ active student involvement – active learning approaches. Their results suggest that a "Strong effect on the development of graduate capabilities may come through employing active learning approaches" [9]. Computing is a practical subject by its very nature and we still conduct necessary laboratory sessions for programming etc. within our CS program. However students tend to sit in rows and work on their own during these sessions and this does not really reflect how a software development team would work in reality in the industry where programmers collaborate to create software. Traditional modes of teaching and learning such as essay assignments or lectures where students passively listen to theory are also becoming outmoded when it comes to helping students practice their communication, organisation, planning and research skills [10]. We believe that an active learning approach to the CS curriculum as defined in our design of the Cross-Site Software Engineering Team Project should help students to develop their employability skills as well as the specialist Software Engineering skills required of their discipline and chosen profession.

Our Implementation of Active Learning

At the beginning of the academic year 2005 and using our new active learning design, students were are placed in teams at both Durham and Newcastle (using a similar selection method as of 2003-2004 where we aim to get a good mix of skills, degree programs and programming strengths etc. in each team). On average from 2005 until now, we have 6-7 students in each local team at Newcastle and 5-6 students in each team at Durham. One team from each site makes up one half of each company and we normally have between 10 and 12 companies. The students are given a problem to solve and in 2005, it was to create an application that would allow a traveller to download interesting material about their holiday destination to a PDA or mobile phone and in 2006-2007 it was to create a training application for a runner to use on their PDA. The 2005 problem was designed so that students from each site could work on designing separate implementations of the applications, Newcastle for a mobile phone, Durham for a PDA and our only stipulation was that each application had to have a similar look at feel. In 2006-2007 we decided to make the need to collaborate stronger between the sites and therefore we stipulated that each company had to work on creating the same piece of software and integrate all their documentation and code deliverables.
We organise an initial ice breaker session in the form of Company Games and this is where the 'organised chaos' referred to in our title begins. The event is very practical and each company, when meeting for the first time, get to work together on a dummy set of problems that emulates the software engineering process in that they have to design, build and test a product – we usually get the students to construct something using paper, plastic wrapping, elastic bands and so on and they get to write documents together - usually in the form of short quizzes. Throughout the event the students have to meet deadlines, communicate, plan, design, implement and test. They have project restrictions i.e. their design very much depends on the amount of cardboard, plastic and other materials they have managed to win by scoring points in the quizzes. This initial event gets them communicating with each other and is a lot of fun for the students. It gets very noisy and chaotic because it takes place at a fast pace and students really enjoy it and always comment favourably on it.

Fig. 2. The video-conferencing suite at Newcastle

After this event is when the real work begins and often where more 'organised chaos' ensues. Students are responsible for the organisation and management of their team and company interactions. This is where we believe the cross-site element of our Software Engineering project is a very effective vehicle for improving the independent learning capabilities of our students because each company has to come to agreement on a wide variety of elements within the project and not just the technical aspects of creating the system. They have to define an organisational structure, choose a software design methodology, plan the project, estimate the amount of effort needed, assign tasks and roles, consider the schedule for implementation and testing etc. to meet deadlines etc. They also need to plan for software integration, testing and a final demonstration of their product. Whilst the end product is a document piece of software, the way the companies operate locally and in their dealings with

their off-site colleagues is an important aspect of this module and our assessment methods also reflect this – we focus quite heavily on the technical aspects related to the software engineering discipline but also the valuable employability skills that students need to gain in order to work in the software industry e.g. communication skills, problem solving and decision making skills, time and task management skills, leadership, analytical skills etc.

The companies are given a lot of technology support for cross-site communication including a dedicated video-conferencing suite at each site which was funded by CETL ALiC capital funding (Fig. 2). They are also given access to other communication technologies such as video conferencing, Skype, Google talk, NESS forums etc. to communicate as mentioned earlier and access to Subversion repositories to work on code concurrently at both sites, this really tests their communication and organisation skills [11]. It means that they have to learn to manage their time effectively and in order to be successful companies must communicate regularly throughout the project in order to complete their tasks and to create and integrate their software. Whilst the end product is a documented piece of software, the way the companies operate locally and in their dealings with their off-site colleagues is an important aspect of this module and our assessment methods also reflect this – we focus quite heavily on both the technical aspects related to the software engineering discipline but also the valuable employability skills that students need to gain in order to work in the software industry e.g. communication skills, problem solving and decision making skills, time and task management skills, leadership, analytical skills etc. Students are assessed on an individual, team and company level throughout the year, using a variety of methods including observation, presentation, formal requirements and design documents, team and company reports as well as project plans, prototype and full product demonstrations etc. We also include two peer assessment exercises throughout the year so students get to reflect on their contributions to the project and to compare the quality of their work with others. The many different ways of assessing students also contribute to the Active learning design and mean that students have to reflect on their experiences, their contributions, their skills and how they learn as well as what they learn, what they need to improve on etc.

The assessments tasks and our project design constitute active learning because they foster independence and place the emphasis on student responsibility for learning as well as the development and reflection on skills. We believe that our implementation of active learning also follows the majority of the principles of good practice in undergraduate education as outlined by Chickering and Gamson [12]. We believe our implementation "encourages contact between students and faculty" – at Newcastle we use staff monitors who observe formal meetings between teams on a weekly basis and can advise a team if they believe it is necessary. It also "develops reciprocity and cooperation among students" as they have to collaborate and communicate in order to achieve their aims and objectives. Our design "encourages student responsibility" for organisation, planning, task allocation etc. as well as reflection on skills, contribution and the team and company processes throughout the year. We try to give "prompt feedback" but this is somewhere we do not always reach our targets. We give formative face to face feedback to students, online feedback, hard copy feedback and weekly feedback from monitors where appropriate and are constantly working to improve student perceptions of the quality and timeliness of this feedback. The overall assessment for the module "emphasises time spent on task" as each student must account for all their time during the module and meet deadlines throughout the year. We also "communicate high expectations" – over the years our module has become quite competitive between the teams and we award a series of prizes for performance so all students want to do well and finally we believe our design respects diverse talents and ways of learning because of the variety of assessment methods we use and the wide scope for participation in many ways within each company. We feel all students' get to work in their own way and explore, experiment and learn at their own pace.

Results

As can be seen in Table 1, we use a variety of assessments that test the students' achievements in terms of both employability skills and the specialist Software Engineering knowledge that we teach them in the module. in the UK, we all write Learning Outcomes for our modules and these are a few sentences on the module specification that describe the skills and knowledge we aim to teach students in the module. We wanted to find out the impact of our implementation of active learning techniques and the introduction of the cross-site perspective in the module.

Table 1: Learning Outcomes for Software Engineering Team Project Module

Learning Outcomes	**Deliverables**
Problem solving, requirements analysis	Statement of work document – requirements analysis
Use of initiative, planning, use of software development models, problem solving	Project plan, log books, team reports
Software design, software development models, industry standards and practices for design notation	Project document - design
Programming, testing, software development	Software source code and documentation, user manuals etc. Project Document
Adaptability, leadership, interpersonal communication, cross-site communication and collaboration, work as member of team, fulfill roles, time management, organisation	Personal skills analysis, individual reports, meeting minutes and observations, team reports, Team contract, log books, evaluating own and others performance and individual reflective reports
Communication	Team presentation, written reports, talking to customer, use of technologies
Written communication skills, using industry-standard notation	Project Document, team, individual reports, coding, documentation

In terms of measuring the success of a module against the learning outcomes in quantitative terms, especially where skills were concerned, it was pretty difficult. We had a look at the overall assessment results year on year for the 2 academic years prior to our introduction of active learning, (2003-04, 2004-05) and the 2 years since their introduction, (2005-06, 2006-07). These quantitative results can be seen in Table 2. Overall there was a slight improvement in student achievement in terms of marks in 2005-06, the first year that we introduced the cross-site perspective but in the following year, 2006-07, there was a slight drop from the overall mean.

Table 2: Module Assessment Results 2003-2007

Year	**Mean**	**N**
2003-04	62.1495	99
2004-05	62.1699	83
2005-06	65.4086	81
2006-07	61.2015	68
Total	62.7574	331

We think we can explain the slight difference in marks in terms of the different software engineering problems and the coupling of the teams across sites in both years. In 2005, the student teams were loosely coupled across sites and the problem was fairly straight forward.

The dependency between the teams for the final software was quite loose, relying only on a similar 'look and feel' between the two products which were developed at each site. The task was to create a holiday application for PDA's and mobile phones. In 2006 the student teams were more dependent on each other and the task was more difficult – they had to deploy their software on real PDAs and use GPS receivers to download and upload training data. There was a slight difference in difficulty level between these two sets of assignments and we feel the marks achieved by students reflect this. So, whereas the first year of our implementation reflects an improvement in the average student mark in the module, the second year actually shows a slight drop in marks. The reason we increased the dependency between the sites in 2006 was that we found the students did not really need to communicate with each other very much and worked rather independently – which was not what we had planned. Initially our main reason for the loose coupling in 2005 was in order to ensure that no local team's assessment was jeopardised by a non-cooperative team at another site. We feel that the difference in marks in 2006-2007, (Table 2.), in comparison to other years i.e. 1.55 marks, is not significant enough to indicate that our changes in the dependency across sites impacted negatively on student achievement. It is also often very hard to distinguish between the actual student cohort themselves in these years as each individual is different and therefore the levels of achievement for each individual student will, of course, differ greatly from year to year.

We believe that the perceived skills achievement and student feedback on the module can reflect the impact of our introduction of active learning techniques more accurately. In previous work we have analysed the roles that students chose and their perception of the skills they felt they had at the beginning of the module before their project started and then at the end of the project [13, 14]. We elicited all our student feedback via individual reports, focus groups and module questionnaires. For the purposes of assessing the impact of the changes to the module we looked at feedback from these materials for the four years of our study – from 2003-2007 and we have broadly categorised the issues, experiences and achievements they reported into *Communication, Problem Solving, Software Engineering Practice and Project Management.*

1) Communication

Across all the years of our study the teams experienced difficulties in communicating with each other. Most teams conducted regular formal and informal meetings locally but when working outside the timetabled hours they found it difficult to arrange suitable meeting times or to contact all of their team members. A large number of teams also suffered from the non-participation of one or more team members. With the introduction of the cross-site perspective, communication between companies became much more challenging. The students had a lot of problems with the video conferencing technology in both 2005-06 and 2006-07 despite regular quality tests of the equipment and often had to resort to other communication methods e.g. telephoning, Skype, MSN or email. A large number of the teams chose to meet face to face on one of the sites when completing the larger project deliverables i.e. during the Design and Implementation phases. On the whole students found collaboration with a cross-site team was the most difficult aspect of their project but the majority felt they had benefited from the experience. They welcomed the opportunity to use new technologies and almost all students reported an increase in their communication skills at the end of the project..

> "*Having company members in Newcastle and Durham was very hard to timetable at first but after improving organisation of the team and finding out the most effective communication methods, the project became a lot easier."* Student Comment

2) Problem Solving

There were some common problems that teams and companies experienced in the four years of our study. These were: non–participation or absence of a team member, assigning roles and

delegating tasks, agreement on the design of their software system and integration of their code. The cross-site perspective mean that they had problems agreeing on site-specific responsibilities and also dividing tasks between the local team but felt that even though this was a problem they had learned more about their own skills and abilities during the project and the importance of identifying the correct person for the correct role in teamwork. Students also reported that they had learned about delegation, communication and professionalism and most of their learning took place when they experienced difficulties in working with one another. Not all problems were resolved but reflective reports suggest that students are more aware of what can go wrong from the people perspective of a team project and would know what to do if faced by similar situations in the future. The cross-site work from 2005 onwards also introduced issues of technical support and highlighted the different teaching approaches at Newcastle and Durham – at Durham students are supported in their practical sessions for the project, whereas at Newcastle they are not. Staff are used as monitors at Newcastle whereas at Durham, third year students undertaking their Project Management module act as project managers for the local teams. These differences often caused concern for Newcastle students especially as they felt they were 'thrown in at the deep end'. Across all the years of our study students experienced conflict of some form between local team members and with their cross-site colleagues. Most teams had identified risks at the start of the project during a project management workshop but when faced with the actual difficulty e.g. on losing a team member or poor communication between sites, many found it hard to cope. Between 2003-2005 students reported that they learned a great deal about team organisation, project planning, task allocation and team management. Students seem to have benefited most from trying to find their own solutions without calling on the assistance or intervention from monitors and module leaders but sometimes they did especially if they had someone in their team who was not pulling their weight. From 2005 onwards all students had added organisational difficulties such as booking meeting rooms, arranging video conferences and coordinating the completion and submission of deliverables across site but in general students coped very well with any conflict and technology or organisational difficulties that they faced and they found partial or full solutions for these. The reports indicate that students learned a lot about planning and organisation mostly through experiencing problems and many felt that their own problem solving, negotiation and time management skills had improved directly because they recognised that people were depending on them and they had to deliver so they pulled together when things went wrong.

> *"The only notable problem was that one member left our team. This meant work had to be redistributed, and due to the fact his departure was essentially near the end of our implementation phase, this proved to be extremely awkward. This problem had to be solved through efficient communication and effective team working on teams at both sites."*
> Student Comment

3) Software Engineering Practice

During 2003-2005 students reported on which parts of the Software Engineering process they had participated in, the challenges of each of the phases and to comment that they knew far more about the subject by the end of the project. However, the introduction of cross-site working seems to have emphasised more the importance of a positive professional attitude and a shared code of conduct, an agreement on the use of a clear software engineering model by all participants and an agreed standard of documentation and coding practice between sites. Students found Requirements Analysis quite easy during most of the project as tutors gave them fairly clear instructions. The use of professional notation for the Design phase of the project caused difficulty in all years as students were not instructed on the construction of UML diagrams during the taught part of the course. Students commented that they needed more instruction on design practices. The Integration and Testing phases were a major problem escalated by the cross-site work and some teams during 2005-2007 did not manage to integrate or test their code fully in time for project demonstrations which they found

frustrating and disappointing as they had worked very hard but they did learn from their experiences:

"*This project has taught me a lot about software engineering. It has broadened my knowledge about Java applications as well - realising the importance of having a data structure that is flexible and can be altered without ruining the whole behaviour of the program. It has also taught me how to manage a project and how important it is to take some time to design and plan the structure of the application before implementing the program. If we were to do this all over again we would have talked to our Durham team more and agreed what structure we were going to use for our implementation. In that way, the style would be more consistent and the code would have synchronised better.*" Comment from Student Report

4) Project Management

The Project Management module at Newcastle is an option for third year students and only a small part of the level 2 Software Engineering module. In year two students are given only a brief overview of project planning, possible team structures, team role selection and some general tips on teamworking and generally have to learn about project management by doing it. During the period of this study students experienced difficulties in task allocation, estimating how long work would take, setting realistic deadlines in order to get work completed etc. The cross-site work from 2005-2007 onwards seems to have exacerbated these difficulties and the students found it hard to assign tasks, to designate roles and had problems coordinating their joint efforts to fulfil their project objectives. In 2005-06 there was an issue with differing timetabling of practical slots and the deliverable schedules at Newcastle and Durham and these were corrected for the 2006-07 project. Student reports indicate that organising, planning and scheduling were major issues for all teams before and after our introduction of cross-site working. Students reported that they had learned an awful lot about how to organise their teams – (strong leadership, using phase leaders, splitting their team into specific sub-team functions etc.); how to allocate roles i.e. based on ability rather than preference and how to plan for deliverables – using 'soft' deadlines, providing company guidelines on the format of code and documentation etc.

Evaluation

We have had to revise our design several times to ensure that assessment is always fair because this is one of the major issues with a cross-site implementation between two universities. Our concentration on reflection throughout the year means giving students greater opportunities to express themselves in oral as well as written presentations, to work collaboratively with others and whereas we recognise we pay a lot of attention to employability skills and this might mean covering less ground in a technical sense within the module, our concern is for the quality rather than the quantity of students' learning. We focus on how the students learn as well as what they learn [8]. We feel that our implementation of active learning has had more of an impact on the quality of experience and skills gained by students rather than the quantitative assessment results indicate and this is what we set out to achieve. All too often students focus on their marks – because that is how we eventually classify them. Ideally, we would like students to also focus on their employability skills, experiences, values and attitudes as well as the knowledge they have learned in order to further develop as independent learners in the future and therefore somewhere within the curriculum we need to focus on improving their general skill achievements as well as the specialist skills and knowledge of the Computing Science discipline. Of course, our implementation of active learning is not perfect and it has had to evolve based on staff and student experiences and feedback over the past couple of years. We are still working on making it a valuable experience for all students and assisting them in any way we can. We

have had to modify our design several times especially in terms of assessment and we constantly review positive and negative comments about this work over the course of the project. We feel that students are getting a good experience even when they have a testing time of it – the fact is that students learn more by making errors, by practising and getting things right next time round and more especially when they are given the chance to reflect on their own experiences and what they have learned from them. Employers have also expressed a great interest in the cross-site aspect of our work because they think our approach is very realistic. Our design seems comparable with what happens in reality with teams based in different countries working together and using technology to communicate. Our students' experiences are also realistic as many employers have commented on similar experiences within their own organisations. In the last section of this paper we offer some general guidelines, based on our experiences and on feedback from students, for anyone who would like to implement a similar project.

Guidelines for Practitioners

1. Cross-Site work can be difficult and therefore partners should be selected very carefully and factors that influence your decision should be finding a suitable 'window' in the curriculum, a comparable cohort size and comparable curriculum at both institutions. Planning is a crucial element and can take up to a year in advance and there are issues of scalability that need to be considered.
2. Assessment is the area that we believe you need to be most careful about. It can be difficult to design common assessments and marking schemes so that a non-performing team at one site does not jeopardise the marks of their counterpart team and the other site – we think we have found ways of doing this and these are the subject of another paper but techniques we have used are double marking across sites, peer assessment for both teams and companies, and contribution matrices for all joint deliverables. Students need to be reassured about assessment – especially when there is a cross-site element involved. Module marks should be achieved via a series of assessments, some individual assessments as well as joint so that students have a variety of ways to achieve.
3. Students need to be briefed and taught how to peer assess.
4. Students should have time to reflect on their perceptions about their own skills status before starting the project – so they can measure their own progress at the end.
5. You need to ensure there are choices with regard to the technologies that are used for communication. We found that video conferencing was perhaps not the best solution in all instances for students when working together.
6. Our experiences have taught us though that students prefer to work face to face with each other if possible and it can be prohibitive in terms of cost/ resource to facilitate even one face to face meeting nevertheless we have built at least one opportunity for this at the start of the project because students need to bond on some level before starting work.
7. The problem needs to be realistic and allow for all skill levels in your cohort. Some students are better at documentation, organisation, report writing than they are at programming and vice versa so a variety of tasks needs to be available within the project task.
8. Team selection methods need consideration. At Newcastle and Durham, we look at performances from year one to ensure an even distribution of skills in each teams. We believe this gives students a fighting chance in terms of delivering their software at the end. Naturally abilities differ between all individuals and it is difficult to predict how a team will perform together.
9. You need the support of your colleagues and your Head of School and you need to persist. We know that our project would not be possible without the wealth of experience and the patience and support of our Head of School and the dedication of

our team monitors, lecturing staff and technical support teams at both institutions because it is a large undertaking.

Conclusion and Further Work

We believe an active learning approach fits very well within the CS curriculum given the practical nature of the discipline and the realities of global software development within the industry our graduates will eventually enter. An active learning approach to the SE curriculum allows us to explore new ways of teaching and learning that involve students by allowing them to select and structure their learning materials, investigate and research, identify and solve problems, test out theories and procedures and evaluate what they have learned but it is not without its difficulties and we are constantly working on improving our implementation. In 2007-08 we have used a real customer for the project and this has furthered the realism of our approach. At the moment we are evaluating this experience and working on ways to improve the Cross-Site Project for next time.

References

[1] Module Outline CSC2005: http://www.cs.ncl.ac.uk/modules/2008/CSC2005

[2] Mary Z Last, "Understanding the Group Development Process in Global Software Teams", *Session S1F, 33rd ASEE/IEEE Frontiers in Education Conference*, Boulder, USA, Nov 5-8, 2003

[3] CETL ALiC – http://www.dur.ac.uk/alic

[4] HEFCE – http://www.hefce.ac.uk/

[5] SKYPE – http://www.skype.com/intl/en-gb/

[6] MSN – http://www.msn.com

[7] NESS- https://coursework.cs.ncl.ac.uk

[8] Pam Denicolo, Noel Entwistle and Dai Hounsell "What is Active Learning?" *Effective Learning & Teaching in Higher Education, Module 1, Parts 1 & 2, by, CVCP*, Universities Staff Development and Training Unit, Sheffield, pp3,1992

[9] David Kember and Doris Y. P. Leung, "The influence of Active Learning Experiences on the development of Graduate Capabilities", *Journal of Studies in Higher Education*, Vol. 30, No.2., pp155-170, April 2005,

[10] Jennifer Nias (Editor), "The Human Nature of Learning - Selections from the work of M.L.J. Abercrombie", *The Society for Research into Higher Education*, SRHE & Open University Press, pp71, 1993

[11] Subversion: https://subversion.tigris.org

[12] Arthur W Chickering and Zelda F. Gamson, "Seven Principles for Good Practice in Undergraduate Education", *AAHE Bulletin*, March 1987

[13] Devlin, M., Phillips, C. and Marshall, L., "Making Computing Science Students More Employable with Problem-Based Learning and Cross-Site Teamwork", *International Conference on Engineering Education and Research (iCEER) 2007,* Melbourne, Australia, 2-7 December, 2007, International Network for Engineering and Education Research, 2007

[14] Devlin, M., Marshall, L. and Phillips, C., "Active Learning in Computing: Engaging Learners in a Cross-Site Team Project", *SOLSTICE Conference*, Edge Hill, Ormskirk, Edge Hill Centre for Excellence in Teaching and Learning, pp 1-11, 3rd May, 2006