The Effect of Programming Competency on Success in Undergraduate Team Projects in Computing Science

Thesis by

Marie Devlin

In Partial Fulfilment of the Requirements

for the Degree of

Doctor of Philosophy

Newcastle University

Newcastle upon Tyne, UK

(19th May 2015)

Abstract

As part of the Centre for Excellence in Teaching and Learning Project, Active Learning in Computing (CETL ALiC) Newcastle University, in partnership with Durham University, developed a Cross-Site Software Development Activity in their Stage 2 Software Engineering modules (FHEQ level 5) and both universities carried out this activity during the academic years 2005/06 to 2008/09. This initiative involved 'Companies' of Newcastle and Durham students working in partnership to develop a software solution together throughout the academic year. This initiative was risky because assessment and marking of deliverables for the project was conducted between staff at both sites. Each module had differing assessment weightings, learning outcomes and taught content. Therefore it was imperative that CETL ALiC staff kept a close eye on assessment outcomes during the project to ensure that no students were disadvantaged by the Cross-Site work.

This thesis outlines an initial review of assessment carried out at Newcastle University, the findings of which led to some concerns about fairness in attainment between students on different programmes at Newcastle due to student perceptions about the 'higher' value of programming skills and the 'lower' value of soft skills. These findings were the motivation for the deeper investigations into the assessment framework used in the Software Engineering Team Project (SETP) at Newcastle University that are presented in this thesis. The investigations show that student perceptions of the value of technical roles in the project teams led to students in non-technical roles being awarded lower peer percentage weightings, which in turn meant they achieved lower overall marks for the module.

The thesis introduces remedial work in the form of competency matrices that was carried out in an attempt to address this problem. This remedial work led to the development of the Student Appraisal Method, a 360 degree feedback method of formative assessment that is presented at the end of this thesis. This method of assessment can be generalised for other disciplines and should ensure students become more aware of their own personal competency development in team projects in the future and that they make better

judgements about the contribution of their teammates, irrespective of whether their role is technical or non-technical during Software Engineering projects.

For my mum Mavis, the best teacher of all.

Acknowledgements

I would like to thank my supervisor Professor Chris Phillips for his support and patience all the way through this project (it took me quite a while to get this far). Thanks too for all the ice cream, the opportunities to go to conferences in interesting locations and for all the nagging to get the work finished. I would also like to thank Dr Lindsay Marshall for all his support and for fighting with me about ideas on a daily basis, which is always good fun. Thanks too for all the coffee and cake when it was needed. A big thank you also goes to all the support staff in Computing Science. Your patience and efficiency has been much appreciated. Thanks for putting the technical infrastructure in place to make the team project work and for not batting an eyelid when I make strange requests for things like hula-hoops and footballs. I would like to thank my friends Debbie, Shirley, Mairin, Heather and Shane, just for being my friends but also for supporting me through everything I do and for putting up with me when I whinge - kudos to you all. I must say a big thank you to my colleagues in CETL ALiC: Prof. Liz Burd, Dr Sarah Drummond, Janet Lavery, Prof. Janet Finlay, Jakki Sheridan-Ross, Andrea Gorra, Andrew Hatch and Phyo Kyaw. A special thank you also to Tim Smith for setting up all the equipment for the CETL and getting us great deals. We had a great time during the CETL. It was hard work but I loved it and learned a lot from you all. Special thanks also to Dr Terry Charlton for putting up with years of team games and with me filling his car with loads of eggs, cardboard and bubble-wrap, all in the name of Active Learning. I would also like to thank all the students I have taught over the years for all their hard work and for being enthusiastic and eager to learn, passionate about their work and insatiably curious. I learned a lot from you all. Finally, the biggest thank you goes to my family for all their support and just for being brilliant.

Declaration

All work contained within this thesis represents the original contribution of the author. This study has given rise to a number of publications which are listed below. In particular, I make occasional reference to early work arising from the CETL ALiC project in Chapter 1and 2 (1, 2,). Most of Chapter 3 has been published in 3 and 4 and part of Chapter 6 has also been published in 5.

1. Devlin M, Marshall L, Phillips C. Active Learning in Computing: Engaging Learners in a Cross-Site Team Project. *In: SOLSTICE Conference 2006*. 2006, Edge Hill, Ormskirk: Edge Hill Centre for Excellence in Teaching and Learning.
2. Devlin M, Phillips C, Marshall L. Organised Chaos - Learning Outcomes from trialling Active Learning Methods in Computing Science. *In: International Conference in Engineering Education. New Challenges in Engineering Education and Research in the 21st Century***.** 2008, Pécs-Budapest, Hungary: Pollack Mihály Faculty of Engineering, University of Pécs.
3. Devlin M, Drummond S, Phillips C, Marshall L. Improving Assessment in Software Engineering Student Team Projects. *In: 9th Annual Conference of the Subject Centre for Information and Computer Sciences*. 2008, Liverpool Hope University, UK: Higher Education Academy, Subject Centre for ICS.
4. Devlin M, Phillips C, Marshall L. Assessment in Software Engineering- Towards a new Framework for Group Projects. *In: Proceedings of the ICEE & ICEER 2009 Korea International Conference on Engineering Education and Research*. 2009, Seoul, Korea: Se Yung Lim, Korea University of Technology and Education.
5. Devlin M, Phillips C. Assessing Competency in Undergraduate Software Engineering Teams. *In: IEEE Education Engineering Conference (EDUCON)***.** 2010, Madrid, Spain: Universidad Politecnica de Madrid.

Table of Contents

List of Tables and Figures

Chapter 1. Introduction

1.1 Problem Overview

In the academic years 2005-2008, Newcastle and Durham Universities, partners in the Active Learning in Computing (ALiC) project, part of the UK Centre of Excellence in Teaching and Learning (CETL) initiative, (CETL, 2005), introduced a collaborative learning model of Software Engineering to second year undergraduate students to reflect current industry practice by introducing cross-site software development between 'virtual teams'. A virtual team is a "group of geographically and/or organisationally dispersed co-workers that are assembled using a combination of telecommunications and information technologies to accomplish an organisational task" (Bell and Kozlowski, 2002). In this thesis the term 'Company' is used to refer to a collection of students from Newcastle and Durham (a virtual team) and the term 'Team' is used to refer to a collection of students from either Newcastle or Durham; two teams, one from Newcastle and once from Durham, form a company.

The motivation for this initiative was that the prevailing pedagogical model at the two universities reflected an unrealistic view of modern software engineering practice which involved the participation of co-located software designers, programmers, end-users and domain experts with easy access to each other's expertise and opinion i.e. students in the same classroom; the use of 'customers' in the form of lecturers and assignments that could be viewed as 'toy' development problems in terms of their applicability to systems that genuine customers would request.

Increasingly, in reality, cross-site software development ('off-shore development'), where projects are distributed between virtual teams which must work together, is becoming prevalent in the industry and has many attractions for employers, not least that it allows them to:

- develop software more cheaply in certain locations, by taking advantage of exchange rates and low labour costs;
- take advantage of time differences between countries to work round the clock on projects and deliver more quickly;
- have access to a greater pool of potential employees and their expertise.

All employers want employees who can communicate well and work in teams, take direction, yet are self-motivated, able to problem-solve and find things out, make intelligent judgements, test ideas and turn them into plans, meet deadlines and pay attention to detail, and therefore having some or all of these skills makes a person more employable (Ferguson, 2004). However, as corporate activity becomes more complex and dynamic, employers need workers who are adaptable and flexible enough to cope within fast-moving and demanding environments (O'Neal 2004; Bell, 2002). The use of virtual teams can help an organisation respond quickly to changes in their customer needs and many companies are turning to their use to make the most of knowledge and resources. Virtual teams provide access to the most qualified individuals for a particular job, regardless of their location and will play a major role in the organisations of the future (Bell and Kozlowski, 2002, Suchan, 2001).

These factors make it essential for UK HEIs to help undergraduate Computing Science students increase their employability by providing relevant and realistic learning experiences that will allow them to compete more effectively in the expanding global market (Ferguson et al., 2004; Devlin et al., 2006).

The pedagogical aims of the cross-site initiative between Newcastle and Durham were therefore to align students' group-work activities to their anticipated future work-based practices by providing an insight into Software Engineering in an industrial context, to make problem-solving more realistic, to allow staff and students to evaluate and use various technologies for cooperative working and to encourage the development of transferable skills such as communication, organising and team-working (Devlin et al. 2008a, 2008b). Skills and intended learning outcomes for a module encompassing this development at Newcastle were specified as initiative, adaptability, teamwork, numeracy, problem-solving, interpersonal communication, written communication and oral presentation and the assessment scheme was formulated around measuring student development of these skills during the module (MOF, 2014).

During each academic year of the project, teams were formed at Newcastle and each was paired with a corresponding team at Durham to form a virtual company. Usually the major project task was the design and implementation of a large software system (e.g. in 2005/6 the task was a tourists' guide application that could be loaded onto a PDA or mobile phone). Students worked together as a 'virtual company' across the sites and to facilitate this style of working, each

university provided video conferencing facilities and access to instant messaging and email addresses for teams. To share code and documentation, the teams were also provided with Subversion repositories and online document repositories in the School of Computing Science's Virtual Learning Environment - NESS (Newcastle eLearning Support System) (Devlin et al. 2009a).

1.2 Experiences with cross-site assessment

The changes in the module and the introduction of the cross-site aspect really stretched students' learning by making them more aware of the necessity for good communication and development practices and for professional working attitudes. However, this approach to teaching the module was not always wholly successful in terms of reducing student workload or in creating assessments that can clearly differentiate an individual's learning outcomes from the module (Devlin et al., 2008a, 2009a).

Assessment of teamwork could be relatively straightforward if we were simply addressing the tangible deliverables and products of teamwork, and if our marking criteria were just based on the standards of the discipline and not the personal characteristics of the participant. However, teamwork assessment invariably involves allocating individual marks for both *product* and *process*, which can prove problematic (Race, 2001). It is much harder to assess the processes involved in teamwork, as it is necessary to know the contributions of each team member to determine an individual mark. It is vital that each individual is assessed fairly so that those who significantly contribute are rewarded, and those who do not, will not benefit from the effort of their more conscientious colleagues. Accurate assessment of an individual is however, difficult in an environment that allows students to contribute at varying levels whilst also trying to ensure they gain the maximum benefit from the teamwork experience (Devlin et al., 2008a, 2008b). The assessment of virtual team deliverables therefore presented a challenge as staff had to agree on what was meant to be delivered and then how the process would be marked. This often meant compromising on format for the tangible deliverables, and always required in-depth discussion of marking criteria for both product and process assessment. What was necessary was to ensure that we had agreed comprehensive marking criteria coverage and, more importantly, to reassure

students that poor collaboration would not necessarily be detrimental to them as individuals. To aid in the individual assessment process and to mitigate the risk of unequal contribution, students at both sites also undertook self and peer assessment (only within their local teams).

Student feedback in the early stages of the project (verbal and informal feedback as well as feedback collected from standard module questionnaires), showed that there were a number of concerns about fair assessment, which is a common complaint in group projects. However, this was exacerbated when involving two universities working and assessing together. The addition of the collaboration to the respective Software Engineering modules meant that as well as the fear of unequal contribution, students felt they would be penalised if the cross-site interactions did not go well, or paradoxically, if they went very well, that too much collaboration would be viewed as cheating. To alleviate these concerns some changes were made to the assessments and their weightings in subsequent iterations of the module and variations were made in the coupling of marks allocated to teams at each site to strengthen collaboration but weaken dependency between sites. Unfortunately the students' perception that assessment was somehow unfair remained. Students reported good learning outcomes and experiences in questionnaires and set up to evaluate the module design but the assessment and their distrust of its accuracy tended to dominate the feedback received and began to overshadow some of the original aims and objectives of the work.

1.3 Research Motivation: Fair Assessment

As part of the ongoing CETL evaluation process, and with concern about the feedback received on assessment in mind, I conducted a brief review of assessment practices two years before and two years after the CETL ALiC implementation of cross-site Software Engineering between Durham and Newcastle (the academic years 2003/04, 2004/05, 2005/06 and 2006/07) of the grades received by Newcastle students and of the feedback we had received during that time. This work was conducted within the framework of a virtual team environment, but this was not, of itself, of great relevance to the study. I found that, in general, during all of these years, students who did not contribute to the coding of the product during the team project on average received lower grades, in comparison to those who did contribute. In total, over 4 cohorts,

there were 109 non-coders (34% of the total of completing students) and 215 coders. The data indicated that 57% of non-coders scored less than their team's average mark in comparison to 39% of coders. So, it appeared from these basic statistics, coders tended to perform better during the module, or at least, receive higher marks. I also looked at some of the qualitative feedback from students who took the module during these years and this showed that students felt there was unequal effort invested in the team project between coders and non-coders, specifically:

Coders often feel they contribute more to the project and do not get rewarded for it fairly.

Non-coders often feel they have not got enough work to do during the project and that their efforts are generally 'less important' than the efforts of coders.

As tutors, we emphasised to the students during the module that the 'product' (a piece of working software) was not the main focus and we believed this was borne out in how we assessed the module – the actual product solution was worth only 5% of the overall coursework marks, a very small percentage. Coding the product is certainly an important part of the Software Engineering process but of equal importance for learning are the other phases and the students' development of transferable skills such as communication, organisation, planning and teamwork. The assessment should therefore be based on students' overall performance and how well they achieve all the learning outcomes, irrespective of the role they take on during the project. The differences in attainment and student perceptions of the value of their efforts during the project caused concern and led me to ask questions about the 'fairness' of our assessment regime. These questions were the following.

1.4 Research Questions

1. What were the 'differences' between what coders and non-coders did during the project and how exactly had this impacted on their attainment / success in the team project that we had designed?
2. Had we missed something fundamental in the way we taught the background material or organised the assessment that affected student perceptions of the value of their role?
3. Was there a bias towards those who did the coding in our marking criteria for both product and process?

4. Had we fallen into the trap of "specifying assessment criteria for performance based on our perceptions of ability, rather than on what the students actually did" (Black and William, 1998)?
5. Did a student have to be 'good' at programming to do well?

1.5 Research Objectives

The objectives of this work are as follows.

1. To identify and examine the factors that might impact on attainment in software engineering team projects, including programming competency, using the Software Engineering module at Newcastle University as a case study.
2. To determine if programming competency is a primary predictor for team/ individual success in our project and if so, to recommend alternative assessment methods that can measure overall achievement and learning outcomes more fairly.
3. To use the results to:
    - Construct a framework for assessment in Software Engineering team projects that could also be generalised for other disciplines;
    - Find ways of improving the quality of assessment design in these team projects to ensure fairness and promote greater student learning;
    - Make recommendations on assessment methods that can help build individual and team confidence in the early stages and throughout team projects;
    - Create material to help students evaluate their performance *during* team projects, individually and as a team, so they can adjust their approach, if needed, to be more successful.

1.6 Primary Research Contribution

The main contribution of this study is the creation of an assessment framework for Team Projects in Software Engineering that could be adapted to other disciplines as well as the primary area of focus in Computing Science. This framework stems from an evaluation of the assessment approaches used in the Software Engineering Team Project (SETP) module in the second year of the

undergraduate programme in Computing Science at Newcastle. This review includes a statistical analysis of student learning outcomes and achievement in the module over the course of four academic years, as well as the qualitative results of student questionnaires, three Focus Groups on assessment and two (unsuccessful) experiments with peer assessment. In particular this framework sets out to make assessment a more formative, fair and reliable process, that focuses on all the required and desirable Software Engineering skills that a student should demonstrate and learn in Year 2 of their studies, not just the tangible products of their work or their technical competency in programming. This work also contributes to the debate on the merits of current classification systems for degrees in the UK HE community and to the general academic debate and effort on how to improve assessment and feedback which are areas of educational practice most UK HEIs seek to improve.

1.7 Structure of Thesis

Chapter 2 of this thesis provides a review of the relevant background literature used during this study. In Chapter 3 I provide an overview of the CETL ALiC Cross-Site Activity which was the primary motivation for this work. This review includes an overview of assessment methods and issues raised by students during the activity. Chapter 4 outlines the statistical methods used to evaluate the student data for this study. This chapter also includes a brief overview of the qualitative methods used to further investigate assessment issues. Chapter 5 details the results from both the quantitative and qualitative analyses carried out and summarises the findings and their implication for assessment on the SETP module. Chapter 6 outlines two experiments with peer assessment and then outlines the Student Appraisal Method that arose from these findings. The chapter also details how this method could be generalised to other disciplines. In conclusion of this thesis I summarise the work carried out and provide an insight into further work that will be carried out. Finally a full set of references and appendices are provided.

Chapter 2. Literature Review

2.1 Introduction

Assessment systems in UK Higher Education are generally underpinned by a theory of learning and a way of seeing the particular 'world' of the discipline being studied. Assessment systems in general outline both the standards and expectations of the institution in which the learner is situated, and those of practitioners of the discipline outside the institution, in the work place and wider social environment. Before evaluating the success of the existing assessment framework at Newcastle or making changes to it therefore, it is important to explore our knowledge about learners and also the theories of learning and teaching that have contributed to its adoption into institutional philosophy and teaching practice. It is also important to understand the levels of competency expected for newly qualified practitioners of the discipline on leaving university that may have contributed to its design.

In this chapter I briefly discuss the goals of higher education and define learner autonomy. I then focus on learner characteristics including self-perception and motivation and explore how these characteristics might impact on student attainment and the realisation of learner autonomy. I review some well-known theories of learning and assessment, giving a brief overview of their origin and examples of how they have contributed to the development of present-day instructional design and the current UK assessment 'culture'. I then detail some examples of what is currently considered to be good practice in assessment and feedback. This review includes an overview of previous work on assessing teamwork, including the issues of validity and reliability in terms of fair assessment of an individual's contribution to the team effort and in determining overall effectiveness when awarding a grade for the team, as a collective. I also explore definitions of competence and how Software Engineering and programming are currently assessed in the undergraduate Computing Science curriculum in the UK. Finally, I review previous work that focuses on how best to evaluate and construct an assessment framework.

2.2 Learner autonomy and motivation

The goal of higher education has often been debated and the definition fluctuates depending on the economic and political aspirations of the time in

which the question is being asked. The debate tends to focus on two important and oft-thought mutually exclusive philosophical arguments: (1) that the main goal is to facilitate a high level of personal development and intellectual growth for the student, and (2) that the primary purpose of Higher Education is to produce workers and practitioners to suit the needs of industry and society at the time and/or in the long term (Oxford, 2014) An easier way of looking at this is perhaps to focus on the motivations of the student and why they are undertaking a university degree. Each student will have their own goals in mind when signing up for a university programme. Their decision to pursue the programme may have been influenced by either the basic need for employment or for the (often viewed as 'higher' purpose) of intellectual challenge and stimulation, or both these things, but essentially, it is a matter of personal choice for the learner. Higher Education for the learner is an individual pursuit in a social landscape which makes its goals complex and contextual, but always essentially personal. As a teacher, I think my own view of the purpose of academic teaching in Higher Education is to help create autonomous learners, thinkers and practitioners of the discipline of interest that is in line with both the personal aspirations of those learners and the needs of society in general. For me, therefore, the two oft-debated arguments of purpose are not unrelated; they are sub-sections of a complex whole. There is also much debate about what being an autonomous learner actually means. For some it is simply having the ability to take charge of one's own learning (Holec, 1981), whereas for others the definition is wider in the sense that it depends on factors such as willingness and motivation to assume responsibility for the choices required (Littlewood, 1996). For Blondy, a self-directed (online) learning environment requires learners to establish their own learning goals and activities but also requires a curriculum that is focused on process versus content (Blondy, 2007). This means that tutors may need to give up their control of the course and allow learners to be empowered. In this respect, "traditional forms of higher education remain valid because students in these environments are used to expressing thoughts, ideas, opinions and solutions in written form, and reflection on the learning process is often as important as content" (Denicolo et al., 1992). However, there has to be a balance between what students do and what we teachers do to enable the growth of autonomy during a student's learning programme. Spratt et al. suggest, like Littlewood, that motivation is a key factor that influences the extent to which learners are ready to learn

autonomously and that teachers therefore might try to ensure motivation before they 'train' students to become autonomous (Spratt et al., 2002).

According to Torrance and Pryor, people's "attributions to achievement either to effort, intelligence or difficulty in learning activities, are both socially and individually constructed and affect how they respond to learning challenges and feedback from assessors" (Torrance and Pryor, 1988). This means that student motivation is affected by relationships between peers and with teachers and by individual experiences of learning. A learner's ability to develop their personal autonomy will be affected by their learning career and the experiences they have had previously. Their level of motivation and confidence will play a large part in determining how much they are able to exploit assessment criteria and feedback to improve their learning and achievement rather than just to get better grades. Students need to be aware of their own motivations and how they determine achievement and what is a good outcome for them personally. Students rely on feedback from teachers and its interpretation is often viewed as the teacher's responsibility rather than their own or something that peers can help with (Torrance and Pryor, 1988). For the majority of educators and students, *ability* is viewed as the most important determinant for success but some studies have shown that "measured ability on entry does not explain all the variance in eventual achievements" (Emler, 2001). Self-esteem can be a factor in this, as people with low self-esteem expect to fail. Emler showed that when performance is influenced by *effort* rather than by expectations of success or failure, there are few differences in achievements of low and high esteem individuals. People with high self-esteem can show greater persistence but, perhaps surprisingly, according to Emler, in such a way that "results in no consistent advantage".

Dweck also studied students' motivations and attitudes towards study and her theory is highly influential. The theory centres around four attributions: ability, effort, task difficulty and luck, and how often a person uses the same kind of attributions over time to explain their success/failure determines whether their attribution style is *self-enhancing* or *self-defeating* (Dweck, 1988).

Dweck divides students into two types based on the student's own theory about their ability.

- "Fixed IQ theorists – these students believe their ability is fixed and there is very little they can do to improve it. They believe ability comes

from talent rather than from the gradual development of skills through learning e.g. “I can’t do Math”.

- Untapped Potential Theorists – these students believe that ability and success are due to learning, and learning requires time and effort. In case of difficulty, these students believe they should try harder, try another approach, seek help etc.”

(Source: Dweck, 1988).

For Dweck, the most motivated and resilient students are the ones who believe their abilities can be developed through their effort and learning – the Untapped Potential Theorists. Dweck and Legget’s investigation of motivation and personality set out to identify behaviour patterns and link them to underlying psychological processes (Dweck and Legget, 1988). They described two patterns of student behaviour.

- The Helpless Pattern – characterized by the student avoiding challenge and a worsening of their performance when faced with obstacles. Students following this pattern did so, not because they lacked skill – the author’s research shows that those who avoid challenge and follow this pattern are initially equal in ability to those who seek challenge and show persistence. Their study also showed that “those most concerned with their ability behaved in ways that impaired its functioning and limited its growth.”
- The Mastery-Oriented Pattern – students who follow this pattern seek challenging tasks and generate effective strategies when they are faced with obstacles.

(Source: Dweck, 1988).

Overall the results of the Dweck and Legget’s 1988 study showed that students viewed effort and ability as inversely related, where high effort implies low ability and low effort implies high ability. The authors also differentiated between *performance goals* and *learning goals* – those with learning goals were more likely to view effort as a means for activating their ability for mastery whereas those with performance goals use an inference goal that says effort, even when it accompanies success, signifies a lack of ability.

According to Butler, an organizational environment and assessment regime "that focuses on ability rather than development and learning" can help to "perpetuate the myth of fixed intelligence" (Butler, 1988). Butler's results and those of Black and William (1998) support the view that a pre-occupation with grade attainment can lower the quality of performance. Some students will perform at a less than optimal level because they believe that is all they are capable of. They may have been told this before in previous educational experiences and it is the level they believe they deserve.

This work on motivation and student perceptions about attainment is relevant to the current research effort on the evaluation of the assessment regime used in the Newcastle SETP because the studies show that these are all factors that can influence the performance of students, irrespective of the assessment design or methods used by teachers.

These factors may have influenced the non-coders mentioned in the initial study outlined in Chapter 1. To clarify if these factors have impacted on attainment and influenced students I will need to investigate students' attitudes and perceptions about themselves as learners and their views on assessment in the module. I also need to look at some of the theories about learning that are commonly used in UK universities to create curricula and design assessment for students as these will have contributed to the design of the original assessment regime in the module.

2.3 Theories of Learning

2.3.1 Constructivism

Arguably, one of the most popular and widespread theories of learning still used as a basis for defining teaching and learning approaches today is constructivism. Its origins can be traced to the work of 18th century philosopher Giambattista Vico (1668-1744), who believed essentially that the only way of "knowing" a thing is to have made it and to be able "to account for the elements it contains and to trace the steps in putting them together" (von Glasersfeld, 1992). This philosophy made Vico a pioneer and moved attention from the "supposedly pre-existing world" (von Glasersfeld, 1992) towards the view where humans take a practical, "active involvement in the creation and acquisition of knowledge" i.e. they are 'builders and makers' of knowledge. Later educationalists and psychologists developed a more rounded idea of what

this constructivism comprises and these included John Dewey and Jean Piaget. John Dewey (1859-1952) was a liberal social reformer with a background in philosophy and psychology. He is held as one of the people most responsible for the success of the Progressive Education Movement (Reese, 2001). Progressivists base the curriculum around the experiences, interests and abilities of students and believe that students must learn by doing. Dewey stated that people learn well by interacting with others and also that our learning increases when we are engaged in activities that have meaning for us. To Dewey, education is a reconstruction of experience and an opportunity to apply previous experiences in new ways (Dewey, 1938). In a similar vein, Piaget, a developmental psychologist, argued that people produce knowledge and make meaning based on their experiences. Two key components of his position are assimilation and accommodation. Assimilation means the individual adds new experiences into their old experiences. This means they develop new outlooks, rethink misunderstandings and evaluate what is important – their perceptions are changed. Accommodation is where individuals have a view of how the world operates. When things do not operate in the way they expect or in new ways, they must accommodate and reframe their 'world view' with the results. Piaget focused very much on how learning occurs rather than what influences learning. He viewed the role of the teacher as very important and quite clear (and rather in contrast to more traditional views) i.e. they should function as facilitators who aid the student when it comes to their own learning. According to Piaget, the teacher must begin by asking questions rather than answering them. The student must come to his or her own conclusions, instead of relying on the teacher as the 'giver' of knowledge. Teachers need therefore to be in continual dialogue with students and should create learning 'experiences' that depend and focus on the needs of the student. He advocated that the teacher must challenge students by making them effective critical thinkers. Piaget's philosophy therefore emphasises learner-centeredness and advocates 'active discovery' (Koschman, 1996). This means that experiences and activities should be planned to allow students to explore, manipulate, experiment, question and to search out answers for themselves. For Piaget, learning is more meaningful if the student is allowed to experiment on their own and if teachers showed confidence in the students' ability to learn on their own. Within constructivist learning, the emphasis is on the learner rather than the teacher. The learner interacts with their environment

and constructs their own ideas, finds their own solution to problems and eventually becomes autonomous and independent. Learning is the result of the learner matching new information against information or knowledge already known or experienced to form meaningful connections and create new knowledge. This means that learning is heavily influenced by the context in which it takes place and also by the beliefs and attitudes of the learner. Constructivism therefore emphasises learning and not teaching. The learner interacts with their environment – gains an understanding of it and then constructs their own ideas and finds their own solutions. Learning is the result of individual mental construction and provides the opportunity for students to take responsibility for their own learning. Constructivism shifts responsibility for learning from the teacher to the learner, who is no longer seen as passive or powerless. The teacher becomes a facilitator rather than a dictator of learning. In constructivist models of learning, people learn by active construction of ideas and building of skills, through exploration, experimentation, receiving feedback and adapting themselves accordingly. "This leads to integration of concepts and skills into the learner's existing conceptual or competency structures" (Koschman, 1996). Constructivist pedagogies recommend that learners are supported or 'scaffolded' by expert tutors and environments that present new material and questions at the appropriate time.

2.3.2 Social Development Theory

Another highly influential theory is the work of Russian Psychologist Lev Vygotsky, who lived during the Russian Revolution. Vygotsky's work was largely unknown to the west until it was published in 1962. His theory is also viewed as one of the foundations of constructivism. It has three major themes: Social Interaction, the More Knowledgeable other (MKO) and the Zone of Proximal Development (ZPD). For Vygotsky, social interaction played a fundamental role in the process of cognitive development and social learning *preceded* development. The MKO refers to anyone who has a better understanding or a higher ability than the learner, with respect to a particular task – normally thought of as the teacher, but could also be peers, a younger person or even computers. The ZPD is the distance between a student's ability to perform a task under guidance from an adult and/or peer collaboration and the student's ability to solve the problem independently. According to

Vygotsky, learning occurs in this zone. He focused on the connections between people and the socio-cultural context in which they act and interact in shared experiences (Crawford, 1996). In contrast to a 'transmissionist' view of the teacher's role (i.e. the teacher transmits their knowledge to students who receive it *passively*), his theory promotes the idea that students take an active role in their learning. The roles of the teacher and student are transferred and learning becomes more of a shared experience for the student and the teacher.

2.3.3 Experiential Learning Theory

A more modern theory that borrows from both Dewey's Experiential Learning and also Constructivist pedagogies is Kolb's Experiential Learning theory (1984), which provides a descriptive model of the adult learning process. Kolb's theory emphasises the central role that experience plays in the learning process. It has its origins in the experimental work of Dewey and Piaget and forms a unique perspective on learning and development. He suggests that there are four stages in learning, a four stage learning cycle. This learning cycle is a central principle of the theory. The cycle process begins with Concrete Experiences, which provide a basis for reflection on that experience, (Observations and Reflections). These observations and reflections are then distilled into abstract concepts (Abstract Conceptualisation), where the person derives general rules for the description of the experience or the application of known theories to it. The next part in the process is the construction of ways of modifying the next occurrence of the experience (Active Experimentation), leading to the next Concrete Experience. The most direct application of the model is to use it to ensure that teaching and activities cover each phase of the process. The teacher can do this by asking questions that encourage the student to reflect, help them grasp or conceptualise the ideas and find ways of testing the ideas. The four quadrants of the cycle are associated with four different forms of knowledge. There is emphasis on developing students' skills and higher order thinking (analysis, synthesis and evaluation) and the emphasis is placed on students' exploration of their own attitudes and values. Experiential learning gives students greater autonomy and control over the subject matter and resources, their learning methods, pace of assessment etc. It focuses on experiential learning, or learning by doing. Sometimes the term Active Learning is used interchangeably with Experiential Learning. Kolb also

outlined four basic learning styles (developed by Kolb) Diverging, Assimilating, Converging and Accommodating and these are described as follows.

"Diverging – People who have this as their dominant learning style are best at viewing concrete situations from many different points of view. They perform better in situations that call for the generation of ideas – brainstorming, tend to be interested in people, to be imaginative and emotional and tend to specialise in the arts. These students also prefer to work in groups, listening with an open mind and like receiving personalised feedback.

Assimilating - These people are best at understanding a wide range of information and putting it into concise logical form. They are less focused on people and more on ideas and abstract concepts. They find it more important that a theory has logical soundness than practical value. They prefer reading, lectures, exploring analytical models and having time to think things through.

Converging – People with this learning style can solve problems and find solutions to practical problems, prefer technical tasks. They like to experiment with new ideas and to work with practical applications.

Accommodating – These learners are hands on and rely on intuition rather than logic. They prefer to take a practical and experiential approach. These learners like new challenges and to carry out plans and will rely on others for information then carry out their own analysis. They prefer to work in teams, set targets and actively try different ways to achieve an objective" (Kolb, 1984).

Kolb explained that various factors influence a person's preferred learning style. The learning style itself is the product of two pairs of variables depicted as lines of axis in Figure 1 each with conflicting modes at either end (Businessballs, 2012).

According to Kolb, we choose our approach to a task or experience by opting for either a) reflective observation i.e. watching others involved in the experience and reflecting on what happens, or b) through jumping straight in and just doing it ourselves (i.e. active experimentation) and at the same time, "we choose how to transform the experience into something meaningful for ourselves by either a) thinking, analyzing or planning (i.e. abstract conceptualization – thinking) or b) experiencing the concrete qualities of the world (concrete experience – feeling)" (Businessballs, 2012).

Outcomes of experiential learning are diverse e.g. a student may acquire a new skill or a group of students may develop a stronger social conscience as the result of the activity. In this sense, Kolb's theory also relates to Vygotsky in terms of the value of social learning, but it is seen as a side effect of, rather than central to the individual's learning.

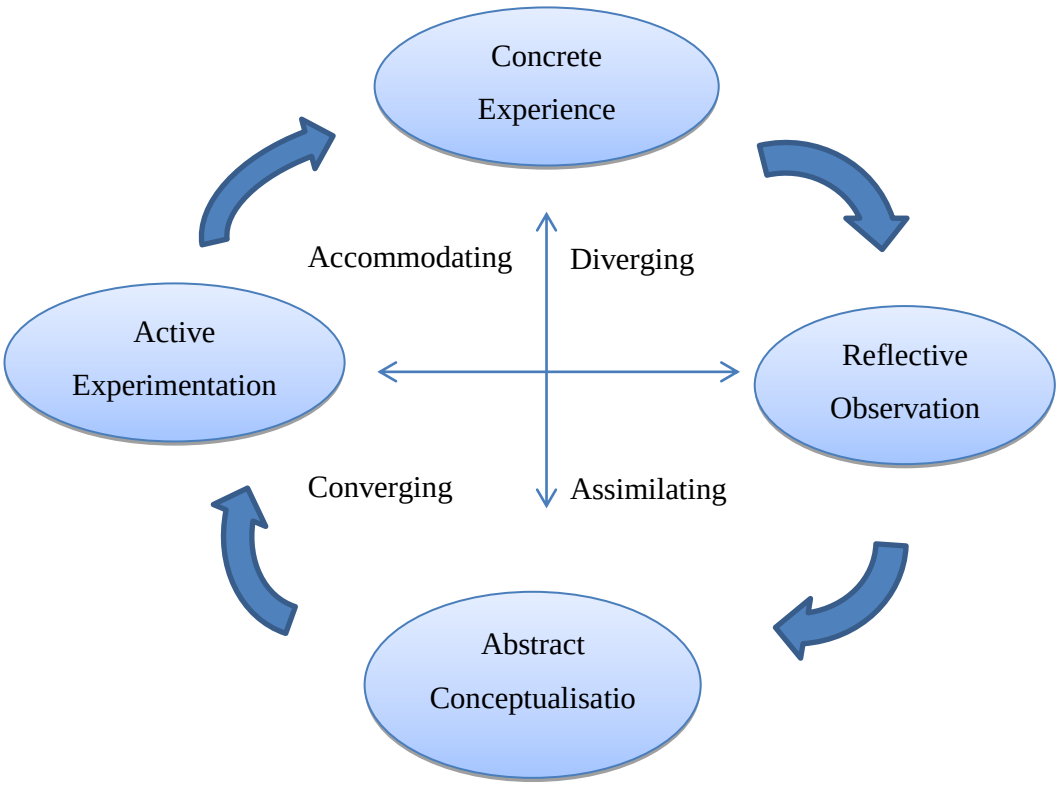

Figure 1: Kolb's Experiential Learning Cycle

The key to experiential learning really depends upon both experience and reflection and the quality of both aspects. If the experience is of limited quality and the reflection is also limited, then the experiential learning is also limited. If the experience is of good quality but the reflection limited, then the learning will also be limited. These factors need to interact meaningfully to enhance the learning. Experiential learning has a specific teaching style associated with it and that is *facilitation*. The teacher is viewed as an external motivator encouraging the interaction of the students' experience and reflection. Barriers to experiential learning include competing priorities e.g. workload and complexity could drain the student, as could personal or social problems.

However, the same could be said of these issues as barriers to most types of learning.

Our knowledge of how students learn and the learning process influences our approach to design learning and teaching for the SETP module at Newcastle. Undoubtedly, this knowledge, and our own particular views on learning, has an influence on how we set out to assess students and therefore impact directly on student attainment and achievement. The work of Dewey, Piaget, Vygotsky and Kolb and their contributions to the development of the theories of Constructivism and Experiential Learning have had a lasting impact on Higher Education today in terms of recognizing the importance of the human and social aspects of learning and the need to allow students to create their own knowledge and take an active part in their own learning. The facilitator role within Experiential Learning is also akin to the teaching approach that has evolved over the years in the SETP at Newcastle. The focus of the curriculum and learning design for the module has also become much more experiential and 'problem-based' over the years and part of the reason for change was due to the demands of industry. There was recognition that student assignments needed to be more authentic in terms of the relevance to employers and their similarity to the work of a software engineer in industry. Reflection on the effectiveness and impact of the particular teaching approach within the module are important to the current research study in terms of understanding where our assessment regime originated and the personal or institutional learning theories that support the learning design.

2.3.4 Constructive Alignment

A more recent learning theory that reflects and includes large portions of the theory of experiential learning and other views of learning presented so far, is that of Constructive Alignment. Constructive Alignment has emerged as the pre-dominant approach to developing teaching and learning in Higher Education in the UK today. The concept of Constructive Alignment was first introduced by Biggs in 1999 and the basic premise of this idea is that the curriculum is designed so that the learning activities and assessment tasks are aligned with the learning outcomes that are intended for a course or programme of study (Biggs, 1999). Teachers must define learning outcomes, and choose the learning and teaching methods that can lead to attainment of these outcomes

and specify what students need to learn to achieve the intended learning outcomes and then assess student learning and achievement of these. In other words, the learning activities and the assessment must be aligned.

There are a number of different ways of defining learning outcomes and these include: "A learning outcome is a statement of what a learner is expected to know, understand and or be able to do at the end of a period of learning", or "A written statement of what the successful student/learner is expected to be able to do at the end of the module/course unit or qualification" (Donnelly and Fitzmaurice, 2005). Generally, these statements are used to describe what students are expected to achieve and how they are expected to demonstrate their achievement (Kennedy et al. 2007). There is, however, still some debate in the educational community and in the educational literature on a standard way of defining learning outcomes to align European standards of Higher Education qualifications and whether learning outcomes *should* or *must* be achieved, how defining learning outcomes affect student learning and how learning outcomes are assessed (Directgov, 2010; Ofqual, 2011; Kennedy et al., 2007; Sweeney, 2010). In some studies the term 'competence' is associated with learning outcomes and defined as "a dynamic combination of attributes, abilities and attitudes", but there seems to be no common understanding of the term and the term learning outcomes is more commonly adopted (ECTS, 2005; Kennedy et al., 2007). Generally, learning outcomes are understood to be clear statements of what the learning is expected to achieve and how they are expected to demonstrate achievement. However Constructive Alignment can be difficult to achieve in practice and we can sometimes find learning outcomes we had not anticipated but are nonetheless valuable and valued by students. The fact that learning outcomes may also emerge over a period of time means teachers should allow for frequent modification of learning activity descriptions and perhaps use consistent methods to help review, identify and classify educational goals.

2.4 Instructional Design – Models and Taxonomies

2.4.1 Bloom's Taxonomy

Bloom's Taxonomy provides a way for teachers to classify educational goals in terms of their complexity and hence scaffold the learning for students to some degree. The taxonomy was originally produced by a group of college and

university examiners, along with Bloom, in 1956 and aimed to promote the exchange of test materials and ideas about testing and of stimulating research on examining and on the relation between examining and education. In the taxonomy educational objectives are arranged into a hierarchy of six levels with knowledge at the lowest level and evaluation at the highest level. Between these are comprehension (level 2), application (level 3), analysis (level 4) and synthesis (level 5) with evaluation at level 6. Bloom and his team claimed that all cognitive educational objectives could be located in this hierarchy. However users often disagree about where to locate their particular educational objectives, as some of the definitions of the levels are vague, and this causes problems e.g. the category where least detail is provided is application, which according to Bloom is, "the use of abstractions in particular and concrete situations and may include general ideas, rules or procedures, generalised methods, technical principles, ideas and theories which must be remembered and applied" (Bloom, 1956). No single theory of learning is represented in the educational objectives they tried to classify.

2.4.2 Anderson and Krathwol's Revision

An interesting revision of Bloom's work on defining educational goals has recently emerged and could be deemed more 'usable' for teachers seeking to determine learning outcomes for a module or programme of study. In the original Bloom framework all the categories were labelled as abilities and skills and for each of these, knowledge was deemed a pre-requisite. Each category presumes to build on the next and is a more advanced achievement. Anderson and Krathwol's 2001 revision retains six cognitive process categories – remember, understand, apply, analyse, evaluate and create (Figure 2). It involves a two-dimensional table with six cognitive processes and four types of knowledge and orders the cognitive process categories according to their degree of complexity (Figure 3). Anderson and Krathwohl define four different Knowledge types and these are as follows.

(i) Factual Knowledge is "knowledge that is basic to specific disciplines. This dimension refers to essential facts, terminology, details or elements students must know or be familiar with in order to understand a discipline or solve a problem in it."

(ii) Conceptual Knowledge is "knowledge of classifications, principles, generalizations, theories, models, or structures that are relevant to the discipline."

(iii) Procedural Knowledge is knowledge that "helps students to do something specific to a discipline, subject, or area of study. It also refers to methods of inquiry, very specific or finite skills, algorithms, techniques, and particular methodologies."

(iv) Metacognitive Knowledge is "the awareness of one's own cognition and particular cognitive processes. It is strategic or reflective knowledge about how to go about solving problems, cognitive tasks, to include contextual and conditional knowledge and knowledge of self."

(Source: Anderson and Krathwohl, 2001).

This revision no longer claims that the process categories are in a hierarchy where the learner can only move to a higher level after mastering all the levels below. The grid emphasises the use of the taxonomy in course planning, teaching and assessment and in aligning these three elements. The original Bloom's Taxonomy was designed for Higher Education but Anderson and Krathwohl's version can also be used at primary or secondary school level. Anderson and Krathwohl recommend that their grid be used as an analytical tool so that the teacher can match activities and objectives to the types of knowledge and to the cognitive processes they outline. The dominant theme of Anderson and Krathwohl's work is the alignment of learning objectives, instruction and *assessment*. It is less concerned with *how* teachers teach, as it is their view that "most instructional decisions depend on the teacher's creativity and ingenuity" and therefore this framework can be used by many teachers, irrespective of their personal philosophical perspective on teaching. This revision and the grid (Figure 3) provided by Anderson and Krathwohl seem a useful starting point for evaluating assessment within the SETP module at Newcastle and should generate an initial way of evaluating the type of learning that currently takes place (and is/is not assessed). It could also help to define a set of assessment criteria that *should* be used within the module. The pedagogical *stance* of Anderson and Krathwohl is that cognitive performance can be improved through the alignment of learning objectives, assessment and instruction – very much like Biggs (Biggs, 1999). Biggs' views assessment as the most critical element teachers can get right to help students learn. He argues

that: “Assessment is almost certainly the most important single component in the system: get assessment wrong and you get everything wrong. We therefore need to be clear about why we assess, what we assess, how we assess, and who is involved in assessing” (Biggs, 2001).

Biggs’ work on assessment, and the work of Bloom, Anderson and Krathwohl on learning outcomes highlight a need to explore the influence theoretical work on learning and teaching has had on the assessment of the SETP module and also how our own motivations as teachers may have influenced its design. It may be that some aspects of the original aims of the module have been forgotten or eroded over time due to changes in priority and focus at school, institutional or national level. To do this, it is perhaps best to first explore the wider approach to assessment in Higher Education in the UK (i.e. the assessment ‘culture’) and then attempt to clarify and understand our own approach to assessment within the various contexts in which the SETP module takes place.

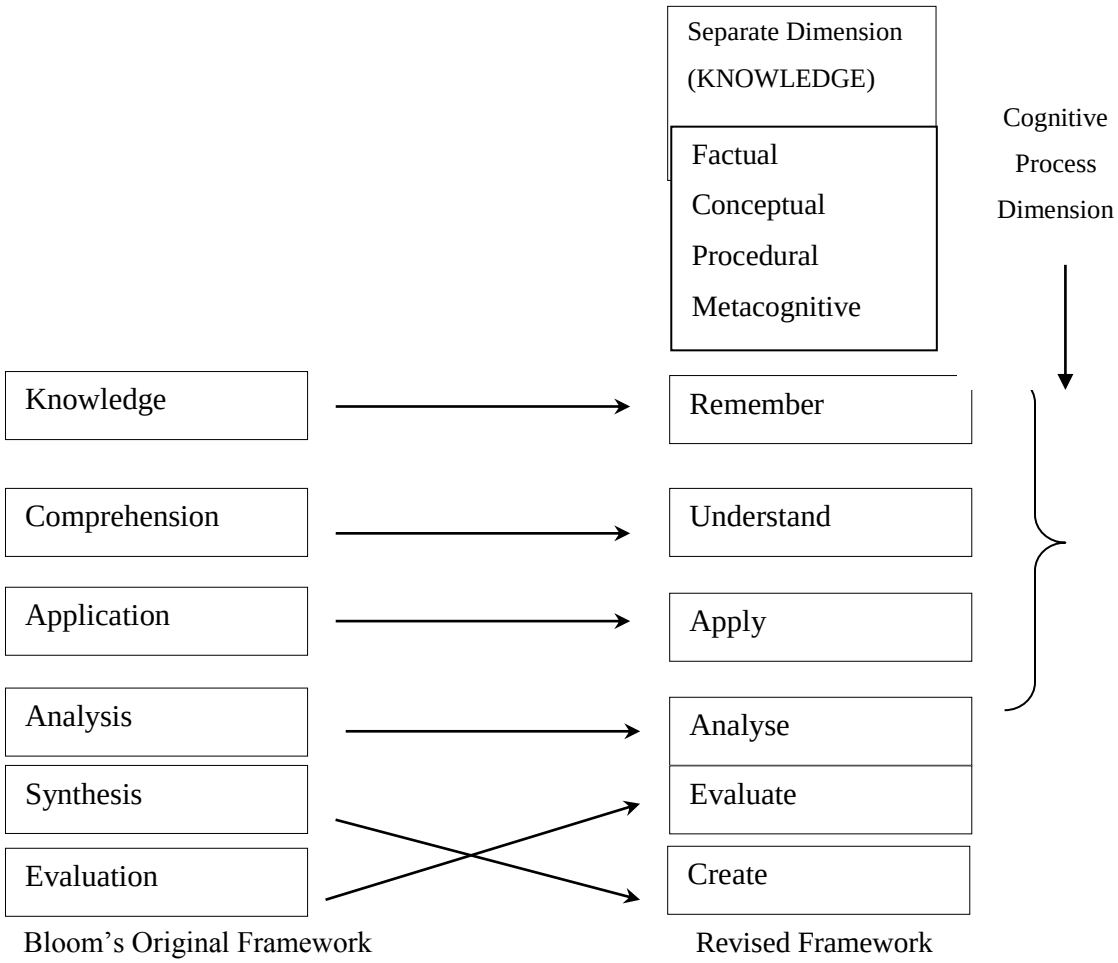

Figure 2: Anderson and Krathwohl's Revision of Bloom's Taxonomy

The Knowledge Dimension	Remember	Understand	Apply	Analyse	Evaluate	Create
Factual Knowledge						
Conceptual Knowledge						
Procedural Knowledge						
Meta-Cognitive Knowledge						

Figure 3: Grid for Matching Cognitive Processes and Types of Knowledge

(Source: Anderson and Krathwohl, 2001)

2.5 The Assessment Culture (Higher Education UK)

According to Balla and Boyle, assessment is very much a "value-laden activity" which is "surrounded by debates about academic standards, preparing students for employment and measuring quality" (Balla and Boyle, 1994; Entwistle, 1996). In institutional terms, it has a wide range of aims other than guidance for students in their learning e.g. summative decision making relating to grades and levels of award and the "derivation of quality and performance indicators or profiles for institutions or units within institutions" (Balla and Boyle, 1994). Previous work (Brown and Race, 1999; Freeman and Lewis, 1998; Yorke, 2001) has found that there are at least six main purposes to assessment:

1. To help select people in terms of their suitability to undertake something in the future;
2. To certify that a student has reached a particular standard or level of competence;
3. To assist learning by helping students see what they are achieving and their strengths and weaknesses;
4. To track progress;
5. To improve teaching by enabling us to make adjustments if our current approach is not effective;
6. To reassure stakeholders (such as industry) about quality and standards of teaching and learning.

Historical approaches to assessment in Higher Education were focused on the teacher's perspective of how well students had learned the material being taught and did not make allowances for student learning styles, any prior learning experiences or the assessment of skill and knowledge development over time. These approaches tended to produce a student who was curriculum-driven and very much used to a classroom in which instructors instructed and learners learned. Students grew used to working towards pre-set objectives and to being assessed, rewarded or penalised by teachers, rather than forming their own judgements about progress or by taking ownership of their learning. Thankfully, practices have changed somewhat in the last 30 years and teachers have found new ways to give students more control over their learning and to motivate them beyond the needs of merely passing the exam. However, as

Ecclestone argues, these changes have arisen partly as "defensive responses to resource pressures and criticisms of outdated teaching" (Ecclestone, 2000), rather than as a deliberate plan to overhaul assessment practices. Increasing student numbers, higher tuition fees and the subsequent demands for greater quality assurance mean that despite institutional and teacher efforts to make teaching and learning more 'student-centred', assessment still tends to follow a "unilateral agenda of authority" which is incompatible with the idea of a student that actively takes responsibility for their own learning (Boud and Falchikov, 2007).

Gibbs and Dunbar-Goddet (2007) reviewed the impact of assessment regimes on student learning. In their work, data were collected in 3 subject areas in three contrasting HE environments – "one pre-1992 institution, one 'elite' institution and one research-intensive institution", in the UK. They do not clarify what is meant by elite and research-intensive or the difference between these two. Their study confirmed that institutional differences have a significant effect on the nature of the prevailing assessment regime, which in turn, impacts on student learning. They outlined factors that have a strong effect on assessment cultures and these include:

- The value placed on scholarship of learning, teaching and assessment
- The extent of risk tolerated and therefore how much teachers can challenge students through assessment.
- Resource constraints which might lead to less relevant assessment tasks
- A strong focus on results as a means of quality assurance and enhancement, rather than the learning process, leading students to emphasise performance.
- Resources and systems are designed around the need to deliver material rather than creating effective learning opportunities.
- Incongruence between rhetoric of culture and reality

(Source: Gibbs and Dunbarr-Goddet, 2007)

An assessment regime therefore embodies many assumptions about what an education is and signals very much to students about the priorities of an institution or school in which they are based. As teachers we may have a tendency to frame assessment primarily in terms of the needs of the institution and accreditation bodies and focus on grades to classify students rather than

focusing assessment on the learning opportunity that it presents. This can lead to assessment regimes that focus on "demonstrating current knowledge but focus little on the process of learning and how students will learn after a particular instance of assessment" (Boud and Falchikov, 2007). Student expectations of assessment are very high and therefore our approach has a powerful effect on how students feel about their discipline and how they approach their studies. Differences in expectations formed from prior experience of assessment regimes at secondary school can also have an effect on students' attitudes and effectiveness in engaging with different assessment methods and approaches at university level and on their view of themselves as effective learners (Dweck, 1988; Torrance and Pryor, 1988).

The Burgess report (Burgess, 2007) highlighted many concerns about assessment practices and identified difficulties with the assumptions on which higher education assessment processes are based. The report states that the honours degree classification system is "no longer fit for purpose" as it cannot adequately describe the range of "knowledge, skills, experience and attributes of a graduate in 21st Century". Burgess argued that current systems concentrate on a single summative judgement that "results in a fixation on achieving a number that is considered 'good'" (Burgess, 2007). Burgess was convinced that a summative system (i.e. one that provides a final grade as an indicator of achievement e.g. first class or 2.1) "gives the appearance of 'signing off' a person's education with a simple numerical indicator, which is at odds with lifelong learning principles that we aspire to for our graduates" (Burgess, 2007). The report led to the development and pilot of the HEAR, (Higher Education Achievement Record) an extended academic transcript that includes skill descriptions as a key way of measuring and recording student achievement (HEAR, 2014). The HEAR could radically reform how we represent students' achievement by providing employers with a more detailed set of information about what a student has *done*, within the curriculum and outside it, in terms of skill development as well as knowledge acquisition. Reasons for this change are manifold but stem from an increased political emphasis on widening participation and skills, the transformation of the higher education experience, changes to the labour market and perceptions of what constitutes a worthwhile degree and good institutional practice. Burgess also found that the current approach encourages students and employers to focus on "one final outcome and perceived end point, rather than opening them up to the concept of a range

of different types and levels of achievement which are part of the on-going process of learning for all students that should continue long after their degree" (Burgess, 2007). Assessment is one of the most important forms of engagement that a student has with the institution, staff and their discipline. For students it should help them to determine where they are 'at' in their studies and what they need to do further to develop their skills and knowledge as a practitioner. Assessment thus forms part of the learning 'dialogue' between students and teachers, between students and the institution, and between students and the professional bodies and industry that they hope to become part of. It is therefore important for students and teachers that the purpose of assessment, and its wider context, is clear. Assessment practice naturally varies widely across the UK, both at institutional and discipline levels and in terms of whether assessment should be primarily *for* learning (formative) or *of* learning (summative) or a mixture of the two, which I will discuss in more detail in the next sections.

2.5.1 Assessment of Learning – Summative Assessment

Assessment *of* learning and achievement is often termed 'high stakes' or 'summative' assessment. According to the Quality Assurance Agency (QAA), an independent body that reviews the performance and standards of Further and Higher Education Institutions in the UK, the main purpose of summative assessment is "to measure student learning in a way that recognises it through the award of credits or equivalent (the combination of which can then lead to a named qualification)". Grades and classifications are therefore primarily performance indicators for the student, the department, the institution, employers, funding bodies and quality agencies, but the QAA emphasise that summative assessment "can, and does, facilitate student learning" (QAA, 2011). Some teachers argue that summative assessment has no real value to learning as it often takes place at the 'end' of a period of learning in the form of an examination and results in a number that really does not represent the learning that has taken place.

The balancing act between designing assessment that motivates and challenges the learner, but also tests the achievement, accredits learning and provides evidence to meet measures of quality, is a difficult one for teachers to accomplish. Some of the difficulties for teachers in ensuring that learning is a

major concern of assessment involve a battle against this emphasis on grades. When designing assessment teachers also need to consider the prior learning experiences of students in old-fashioned 'transmission' or 'banking' models of education as outlined by Freire (1970), the need to report on achievement and to quantify and validate what has been learned via some form of numerical judgement. Assigning marks to students means that student achievement is often "abstracted into just a few numbers" (McNamara, 2004), which can also cause difficulties for a student when they try to articulate the skills they have learned and how these skills have developed and changed during their course of study. Feedback received on work may also vary in quality and this may lead to a piecemeal view that might not help a student much in clarifying their overall proficiency as a practitioner of their discipline. Quality of feedback is further compounded by the fact that a lot about learning is indeterminate and 'fuzzy' rather than discrete and easily measured and therefore what has actually been learned is sometimes difficult to capture for summative assessment purposes and relies heavily on 'academic judgement'.

The importance of summative assessment as an indicator of standards for institutions and students (and other stakeholders such as employers, funding bodies and in comparison to nationally agreed frameworks etc.) means a lot of emphasis is placed on the *reliability* and *validity* of marking rather than on learning. Stakeholders need to be confident of measurements and qualitative information used to indicate level of performance. According to Balla and Boyle "the *validity* of a result or piece of information is the extent to which it is meaningful for a particular assessment or purpose" and "the level of consistency or replicability of data indicates reliability over time, equivalent task and observer" (Balla and Boyle, 1994). The QAA views these two principles as fundamental to the assessment process. It states that "Assessment is understood to be *valid* when it is testing precisely what examiners want it to test, bearing in mind the learning outcomes. *Reliability* in this context means that as far as possible, markers acting independently of each other but using the same assessment criteria would reach the same judgement on a piece of work" (QAA, 2011).

There are however, a few problems with these interpretations when it comes to the actual practice of assessment. HE institutions are expected to have learning goals that are "far more extensive and complex than mastery of the subject alone and are being held to account for student achievement in terms of these

goals e.g. employability" (Knight, 2002). Teachers often use a variety of assessment techniques that can lead to a wide range being used across a programme of study or even within one module. The variety of techniques often makes it difficult to compare performance between modules and to aggregate this into a 'score' at the end. Our search for evidence of reliability in these situations means that sometimes we tend to settle for assessment of the simpler forms of achievements that are more easily measured (Boud, 1995). More trouble arises when the idealised 'transferable' skills are to be assessed (e.g. skills such as leadership or communication) because it can be difficult to generalise about a student's ability i.e. "how the skill will transfer and manifest itself outside of a particular context or instance of learning and assessment" (Knight, 2002). Repeated observations to prove reliability of assessment are difficult, even within one module. Modularisation of programmes, differences between programmes within the same discipline, and wide variations between disciplines, exacerbate this issue. There is no common curriculum or common view of assessment. Knight argues that we need to critically reflect continually on assessment practice in our learning communities and with all the stakeholders involved, to ensure that our assessment methods are valid as assessment is a "communicative practice" (Knight, 2002). He also recommends that we should perhaps consider narrowing assessment to the achievements upon which we can be sure we will make reliable judgements. Some would argue that narrowing the range of skills and knowledge we assess might move us more towards the situation where students strategically focus on ways to get the best marks rather than considering the wider implications of their learning, and where skills development and really important instances of learning that give a more 'rounded' view of their achievements are ignored. Knight's reason for narrowing the assessment range is that what grades or degree classifications signify may not be very transferable because summative assessments are "usually silent on learning processes". Hopefully the HEAR will go some way to demonstrating a student's efforts and learning 'journey'. However, another way to assess complex skill development and learning is to find ways to examine the learning processes of a student continually throughout their learning experience, perhaps via *formative assessment*, before making a final judgment at the end, otherwise, simplifying what is assessed may lead to a simplification of what is taught and, indeed, what is learned.

2.5.2 Assessment for Learning (Formative Assessment)

Assessment becomes formative when the evidence is used to adapt the teaching approach to meet the learning needs of students (Black and William, 1998). Ideally formative assessment motivates students and helps them gain greater autonomy in their learning. Formative assessment strategies include questioning, requiring students to respond to feedback comments on their work (reflect) and peer and self-assessment. Summative and formative assessments have different "rules of engagement" because they have different intentions (Torrance, 2007). A key element of formative assessment is the use of feedback to feed *forward* and help students improve their performance over time by reflecting on their learning processes and goals. According to Hume and Coll, when using formative assessment, the teacher's theory of learning is important. Formative assessment centres mainly on *cognitivist* views that emphasise goal-setting, mental planning and the importance of organisation and it increasingly involves social interaction with peers (Hume and Coll, 2009). Formative assessment therefore fits very neatly into constructivist designs of teaching, particularly the areas of experiential and social learning outlined earlier. In formative assessment, teachers play a facilitator role to help students manage their own learning processes and they do this through the provision of active and engaging learning activities which typically involve open-ended problem-solving and the need for creativity and the use of a variety of assessment methods. However, it is vitally important when employing a formative assessment strategy not to focus on assessment procedures and practices or become too prescriptive and virtually 'coach' students to help them meet assessment criteria. It is important to keep focused on learning as the main purpose rather than compliance with procedures. Formative assessment is much more than a set of procedures as it takes a relational view of learning. Miller and Lavin argue that this requires us to "consider the interaction between all elements in the learning situation" (Miller and Lavin, 2007). Opinion is divided on whether formative assessment can sit comfortably alongside situations in which teachers have to make final accreditation judgements for qualifications i.e. high stakes summative assessment. Torrance and Pryor differentiate between convergent and divergent formative assessment. Convergent formative assessment tends to emphasise a linear view of the curriculum and assumes a relatively passive role for the student. In contrast, in

divergent formative assessment there is an emphasis "on the teacher finding out how students think, rather than concentrating on whether they are 'right' or not. Importantly, this type of formative assessment usually takes place and is a view of learning where students work together to create or construct new ideas. It allows students a greater say, both in the nature of their goals and their progression towards them. In this way the links between divergent formative assessment and experiential and social learning are clear. Also, the link between this type of assessment and the move towards greater student autonomy become evident" (Torrance and Pryor, 2007).

However, for formative assessment to be successful, tasks have to be justified in terms of the learning aims that they serve and can only work well if opportunities for students to communicate their evolving understanding are built into the planning. We need to get a clearer understanding of the student's thinking rather than steer them towards the expected answer. If we don't do this, students get the message that they are not required to think about their own answers and sometimes play a guessing game and try to work out what the teacher expects to see or hear. A key element of formative assessment is feedback that helps students to improve.

This element has its dangers as it can lead to students being more dependent on their tutors rather than less. Clarity in assessment procedures are important but, according to Torrance, this has also "underpinned the widespread use of coaching, practice and provision of formative feedback to boost individual and institutional achievement" (Torrance, 2007). The clearer the task of how to achieve a grade or award becomes and the more detailed assistance given by tutors and assessors, the more likely it is that candidates are to succeed. Too much clarity is in danger of removing the challenge of learning and reducing the quality and validity of outcomes achieved. When this happens assessment procedures completely dominate the learning experience and compliance with criteria comes to replace learning. Torrance argues that achievement is thus routinely defined in narrow terms i.e. that of securing the evidence and grades necessary to achieve an award. This may not necessarily mean achieving the highest grades available nor the depth of knowledge and skill needed for competent practice e.g. it may mean the minimum effort needed to pass.

At present it could be said that teachers are prone to spoon-feeding students in Higher Education (and indeed at other levels of education in the UK) when it comes to assessment. Students and teachers tend to focus on the pursuit of

grades and how to achieve the best grade possible. Students are allowed to draft and redraft assignments and often receive feedback on strengths and weaknesses and what needs to be done to improve their grade. Torrance states that "none of this support, even exam coaching, is necessarily inappropriate or unfair, in and of itself. Such practices are at the heart of professional judgements about the performance/competence interface which tutors and assessors must make" (Torrance, 2007). This behaviour is perhaps understandable in the context of results driven accountability that all HEIs face but can raise issues of equality and fairness. However the approach can be detrimental to learning and to enabling students to become independent learners. It really depends on the support culture and the pedagogical relationship between students and teachers that exists in the institution. As teachers, we need to ensure that we do not make learning objectives and teaching processes so explicit that we effectively 'dumb down' the learning process and call into question the validity and 'worthwhileness' of the outcomes students achieve. Assessment is most effective when its purpose is clear and where students can use the feedback from it in further work. Balla and Boyle state that good practice in assessment of student performance is associated with selection of the method "which matches the purpose of the assessment, the properties or characteristics being assessed and the objectives (intended outcomes) of instruction" (Balla and Boyle, 1994). Teachers also have to maintain the standards expected by the institution and the discipline so that the quality of graduates does not diminish. To do that we need to have confidence in measurements and in the qualitative information we use to indicate a student's level of performance. A degree programme is designed to equip students to learn for the long term, "to pave the way for a lifetime of learning where they will encounter little or no formal assessment or formal instruction in the same way again" (Boud and Falchikov, 2007). So we need to ensure that the correct attributes, skills and knowledge are taught and then assessed appropriately. We also need to maintain a balance in terms of the level of support provided to students so as not to impede increasing learner autonomy.

In the following sections I focus on a variety of current practices for assessing teamwork and methods that we might use to determine programming and software engineering competency. This review also includes an overview of the assessment criteria that can be derived from quality standards used in HE and

from the accrediting bodies of the computing industry. It is important to get an overview of current practice in these areas to inform the evaluation of the assessment regime in the SETP at Newcastle.

2.6 Assessing Teamwork

Teamwork assignments have become the norm in most HEI undergraduate programmes for the sound pedagogical reason that university education is not just about developing what people know and understand in isolation but about learning from and with others, for the benefit of society. Teamworking helps students "shift away from simple academic achievement to much broader goals – preparing them for their working lives." (Leik and Wyvil, 1996). Leik and Wyvil (1996), outline five benefits for students who work and are assessed in small groups (teams), these are:

- Students gain insight into group dynamics;
- Group assessments allow for the development of more comprehensive assignments than is possible for individual assignments;
- Group assessments develop students' interpersonal skills;
- Students are exposed to other points of view;
- Students are prepared for the real world.

(Source: Leik and Wyvil, 1996)

Other educational benefits of students working in groups are well-recognised. These include:

- Studying collaboratively has been shown to directly enhance learning;
- Employers value the teamwork and other generic skills that teamwork may help to develop;
- Teamwork enhances student understanding, students can learn from each other and benefit from activities that require them to communicate and discuss their ideas and knowledge;
- Teamwork provides an opportunity for students to clarify and refine their understanding of concepts through discussion and rehearsal with peers;
- Working with a team and for the benefit of the team motivates some students;

- Team assessment helps some students develop a sense of responsibility.

(Source: James et al. 2002).

However, group/team assessment makes students uncomfortable in general, especially if the assignment marks will have an effect on the classification of their degree at the end of their studies. With any assessment, the mark given is the final interpretation of learning achievement but often associated with this mark is the assumption, somewhat tacit, that if a student has performed well and achieved a high personal score, they have been highly engaged in the teamwork process and this level of engagement has contributed to their success (James et al. 2002). Teamwork can be a challenging way to learn because it requires the student not only to engage with the material but to work together with others to produce something that would not be possible to produce on their own. The task requires that they delegate and share tasks and therefore they share the responsibility for the quality of the work that is produced. Students are often sceptical about the abilities of some team members to produce work that meets their own personal standards and expectations. Students who have performed well in their studies up until the point of a team assignment are often the most reluctant to rely on others to produce the work needed to achieve the level of marks they may be used to receiving or feel they deserve.

Teamwork can also be difficult to assess because group learning cannot be 'captured' or measured as easily or in the same way as individual learning can, i.e. by the production of an assignment or body of work submitted by an individual. Teachers have to generate a summative mark that fairly represents the effort each student has put into the team activity and the level of success they have achieved, based on the expected standard for their level of study. However, "converting a student's contribution on a group task into a numeric grade is a complicated and problematic task" (Leik and Wyvil, 1996). It can be difficult to determine exactly what each student in a team has contributed to the products the team submit at the end of the activity. This is especially true if the activity takes place during more than one instance or over a long period of time. Often team activity on longer-term projects takes place outside of normal classroom sessions, away from the observation of the teacher. This lack of observation means it can be very difficult to determine if all team members are contributing effectively and equally to the team effort and the products/work submitted. Because of this lack of continuous observation, assessing individual

performance in teamwork fairly and accurately means we often need to measure the effectiveness of the team *process* in some way, as well as the quality of the product/s delivered at the end of the learning activity. To do this teachers rely on self-reporting from teams, often via a form of peer and/or self-assessment.

2.6.1 Using Peer Assessment

Peer assessment involves students evaluating each other's performance and contribution to a team task, against a set of agreed performance criteria. A peer assessment process benefits teachers as it provides an inside view into the parts of a team's process that are often not possible to be observed by the teacher. This makes team and individual effort easier to understand and assess, both formatively and in a summative manner. It provides a clearer picture than any tangible assessment product can of how the team works together and shares responsibilities for tasks (Linn et al., 1975). It also helps to increase teacher understanding of the team's approach to the task and their shared understanding of what is expected from them. Thus it helps teachers clarify if there are any misunderstandings or gaps in students' knowledge about taught course material or standards and to understand the assumptions that students make about the work they are doing (Topping, 2009). Some of the benefits of peer assessment to the student include being able to reflect on their own role in the team's performance and compare it to the contribution and effort of their teammates. This enhances the student's learning about the material and their understanding about different standards and ways of measuring performance and contribution. It also enhances their understanding of themselves in terms of their actions, attitudes and their strengths and weaknesses in team work and in the disciplinary work they are undertaking (Race, 2005). Peer assessment allows the student to receive feedback on their performance, from colleagues in the team who have worked closely with them, rather than from the teacher who is always an external observer and assessor. Peer assessment skills are valuable to students because they are transferable to the workplace e.g. when reviewing their own annual performance or that of a project team they are part of or are leading. Peer assessment and working in groups also allows students to free themselves from dependence on the authority of the teacher (Falchikov, 1995). However, peer assessment is not always effective and can sometimes even

backfire and detract from student learning and the aforementioned benefits of teamwork. Often the reasons peer assessment is not effective are based around three issues:

- Lack of explanation of the assessment criteria or the purpose of peer assessment;
- Not allowing students to practise peer assessment before they undertake it for real;
- Selecting a peer assessment method that is inappropriate to the task or too complex for students to understand.

Freeman et al. (2006) outline that we need to address these design and support issues in peer assessment if they are to allow students to improve their ability to make judgments on what constitutes good teamwork. We need to engage the students with the assessment criteria, the assessment process and with giving and receiving feedback. This will help students to understand the assessment criteria and how they should be applied. Leik and Wyvil outline several methods for assessing group contributions that are used in HE today (Leik and Wyvil, 1996) and these are outlined in the following sections.

Method 1: Multiplication of Group Mark by Individual Weighting Factor.

This method is based around the allocation of a group mark by the tutor to the work produced by the group and manipulation of this mark to derive a mark for individuals in the group, i.e.:

$$\textit{Individual student's mark} = \textit{Peer assessment factor} \times \textit{Group mark}$$

The peer assessment factor allows for a percentage of the group mark to be given to every group member and the rest of a student's marks to reflect the individual contribution made by that student. Students assess themselves as well as their peers. This is similar to what currently happens in the SETP.

Goldfinch adopted a similar form of peer assessment. In this interpretation students are presented with a form (figure 4) and asked to grade their peers using the following rating scheme:

- A mark of 3 for better than most of the group in this respect
- A mark of 2 for about average for this group in this respect
- A mark of 1 for not as good as most of the group in this respect
- A mark of 0 for no help at all in this respect

- A mark of -1 if the individual was a hindrance to the group in this respect

Source: Goldfinch, 1994)

Write the names of the other group members in the blank boxes on this row:				
Level of enthusiasm/ participation				
Suggesting ideas				
Understanding what was required				
Helping the group to function well as a team				
Organising the group and ensuring things get done				
Performing tasks efficiently				

Figure 4: Goldfinch 1994 Peer Assessment Form

(Source: Goldfinch, 1994)

Conway et al. (1993) used a variant of Goldfinch's method that was more task-related but their grid does not include any self-assessment and does not attribute marks alongside the rating scheme. This variation can be seen in Figure 5. The rating scheme or marking criteria used by Conway et al. were as follows*:*

- Did not contribute in this way;
- Willing but not very successful;
- Average;
- Above average;
- Outstanding.

(Source: Conway et al., 1993)

Group members' names			
(a) Literature search			
(b) Analysis of literature			
(c) Writing a report			
(d) Group presentation			

Figure 5: Peer Assessment form used by Conway et al. 1993

(Source: Conway et al., 1993)

This method is actually very similar to the Contribution Matrices used in the SETP module at Newcastle (as outlined in Chapter 3). The difference is that the matrix requires students to attribute a contribution *weighting* in Conway et al.'s version, whereas the Contribution Matrices in SETP ask for a description of contribution *type* and we do not attribute a numerical value to these.

Method 2: Distribution of a Pool of Marks

This method allows students to split up a group mark as they see fit e.g. if a group is given 60% as a group and the group has three members i.e. 60 x 3, they may decide to allocate it as follows: student 1 gets 70% of this mark, student 2 gets 40% of this mark and student 3 gets 70%. The students may agree the criteria for distribution beforehand or not. Unless criteria are specified for the distribution of marks, this becomes a more 'holistic' form of peer assessment – i.e. students are using their judgment to arrive at an overall figure for the contribution of their peers rather than breaking the assessment criteria down into categories. A variation of this method was used in the School of Computing & Mathematics at the University of Huddersfield (Leik and Wyvil, 1996). In this new derivation, each student distributed the group grade individually as they saw fit and the results were averaged by the teacher to derive the overall individual result for each student.

Method 3: Group Mark Plus or Minus Contribution Mark

Another method outlined by Leik and Wyvil, (1996) is one where group members peer assess one another according to certain group working tasks and record whether the member's contribution was major, average or small. These evaluations are then converted into numbers as the example shows in figure 6. In this way it is possible to attach different importance weightings to the group work aspects e.g. report writing, presentations, coding, leadership etc. So a group member who makes a strong contribution in all areas would receive an average rating of zero and receive the group mark. A member who made little contribution in all areas would have 20 marks deducted from the group mark. The number of criteria can be varied, as can the associated penalties.

	Major Contribution	Some Contribution	Little Contribution
Leadership & direction	0	-1	-2
Organisation & management	0	-1	-2
Ideas & suggestions	0	-1	-2
Data collection	0	-2	-4

Figure 6: Method 3 matrix (Source: Leik and Wyvil 1996)

A variation of this method gives an average contribution a zero mark, a below average contribution receives a negative mark (-1 or -2) and an above average contribution receives a positive mark (+1 or +2). These are then added to the group mark to give an individual mark. However Leik points out that "the average of the moderated mark must equal the group mark to avoid everyone marking everyone else up" (Leik and Wyvil, 1996). This method is interesting and might be useful for the Newcastle Computing Science SETP in the future because it allows the teacher to specify a range of contribution terms for students to use (as in those outlined by Conway et al. in Method 2) which can then be converted into numerical values for summative assessment. It is important though that students are clear that a value of 0 is not a negative evaluation of their performance.

Method 4 – Separation of Process and Product

Method 4 divides the assessment of process and product for team projects, where a tutor or expert performs assessment of the product and the students themselves using peer and self-assessment perform assessment of the team process. In this method, which was outlined by Falchikov in 1988, peer assessment is performed using a questionnaire that analyses the group process and student performance in two areas: task functions and group-maintenance functions (Falchikov, 1988). Task functions include assigning roles performed during each task based on student behaviour e.g. *Information and Opinion Giver, Information and Opinion Seeker, Starter*, *Direction Giver, Coordinator*,

Diagnoser, *Feasibility Tester and Evaluator*. 'Group Maintenance Functions' include *Encourager of Participation, Harmoniser and Compromiser, Tension Reliever, Communication Helper, Process Observer, Standard Setter, Active Listener and Trust Builder*. Each group member then allocates a High, Medium or Low rating to themselves and their peers for each of these 'functions' This method is different from Methods 1 and 2 in that the mark awarded for the *process* is independent of the mark awarded for the *product*. This method might not be very practical for a SETP as there are a lot of roles that would require clear explanation to avoid misunderstanding. Students might also confuse the 'roles' here with the Software Engineering Roles they take in their team. However, the idea of adding professional behaviours to the list of criteria for assessment is one that could be transferred.

Method 5 – Equally Shared Mark with Exceptional Tutor Intervention

In Method 5 all group members receive the group grade, unless there is a problem with a group member, which results in the tutor being approached and made aware of the problem. Students are encouraged to write comments about the group process and the tutor reserves the right to penalise a group member whose contribution is seen to be defective. The tutor decides the penalty. Leik and Wyvil point out that an alternative is for the tutor to call a meeting and negotiate a distribution of marks within the group (Leik and Wyvil, 1996). This latter process is time-consuming and requires good negotiating skills on the part of the tutor. In the Newcastle Computing Science SETP students write reports about the group process at the end of the module. If problems occur with a group member, students usually approach the module leader and ask for advice to try and fix the problem. It has been rare for the module leader to intervene in peer assessment cases but it has happened and agreement has been reached after discussion with the team and the individual concerned.

In Leik and Wyvil's study students carried out peer assessment using both the *holistic* and *category-based* peer assessment methods highlighted here. Holistic methods assign a mark or grade after reviewing performance from a 'global' perspective whereas category-based is assessment based on a set of categories (outlined by the teachers in this case).

The holistic methods (Methods 5 and 2) led to a larger percentage of group members awarding equal grades to each other, than the category-based approach. It also led to greater differences within the groups between the mark

of the best performer and the student awarded the lowest grade. Leik and Wyvil concluded that holistic peer assessment supports the goal of fair measurement of an individual's contribution better than category-based assessment, in terms of reaching a summative mark, because it focuses more on actual behaviour and contribution rather than focusing on attributes and skills as in category-based assessment. Their study also showed that category-based assessment is best used for formative assessment purposes, as it is qualitative. They suggest that if possible, both approaches could be used and that it is beneficial to allow students to have input into the development of categories for the formative assessment portion (Leik and Wyvil, 2001).

This section has highlighted the important learning benefits of working in teams. Team work allows students not only to gain insight into how a team might work in practice, in the world of work, but also to the industry and university expectations and standards within their discipline and community of professional practice. It has also highlighted that assessing the contribution of individuals in teams fairly can be a difficult task. Team marks are often a bone of contention for students, especially those who view themselves as high performers. Team working can be difficult for those students who lack confidence in their skills. Often when such students are placed in teams, they feel pressured by peers to perform and they are perhaps not as strong as other members of the team. This pressure can be a good thing as it often motivates students to work hard but can also cause some to give up easily and let the stronger students take the lead and the responsibility for major parts of the task.

This section has also introduced some peer assessment methods that are currently used in higher education. Peer and self-assessment is an important element in our drive to engage students more with their learning, and to help them to become more autonomous and critically reflective about their own learning and performance. It allows students to reflect on their own personal skills development and to understand the purpose of assessment design and assessment criteria. Student worries about team assessment and the recommendations from Leik and Wyvil's study are an important point of reference for the current work as they are elements that need to be taken into consideration when reviewing the team structures and peer assessment methods used in the SETP at Newcastle.

2.7 Assessing Software Engineering Competency

So far this chapter has reflected on generic concepts of learning, teaching and assessment in Higher Education in the UK. To ensure that the current work considers all pertinent aspects of assessment for reviewing our SETP, we need to look specifically at how Software Engineering Competency is currently assessed within academic institutions and within discipline communities and industry.

2.7.1 Determining Competence

The definition of the term Competence differs between disciplines and institutions and is often associated more with the human resources function in industry where core competencies are defined for roles and job descriptions. Modern competency modelling definitions and approaches stem from studies carried out in the Aviation Psychology program of the United States Air Force during World War II. Colonel John Flanagan had to try and improve military flight training and bombing mission effectiveness. He asked his trainees and veterans to describe their activities exactly in terms of what they had done successfully and unsuccessfully and later he formalised this process into the Critical Incident Technique, which is a method of identifying critical job requirements. This technique defines a set of behaviours that contribute to the success or failure of individuals in specific situations. Flanagan developed this technique because he found "Too often, statements regarding job requirements are merely lists of all the desirable traits of human beings. These are practically no help in selecting, classifying or training individuals for specific jobs" (Flanagan, 1954). The modern-day core competence approach now outlines what is important for collective learning in an organisation and recognises "the complex interaction of people, skills and technologies that drives performance and addresses the importance of learning" (Delamare LeDeist and Winterton, 2005). However, Delamare LeDeist and Winterton stress that a rigid adherence to a generic list of competencies can undermine success. Boon et al. view competency as a "useful term, bridging the gap between education and job requirements" (Boon et al., 2002). Competence, in general, is a term that can be used to describe what people need to be able to do in employment, tasks that people do and personal traits and characteristics. We may not be able to identify a universally applicable set of competencies and getting agreement on

critical competencies can be difficult if we are to concentrate on competency for the purposes of a course or level of study. Most HE providers would view a pass mark in exams and coursework as sufficient evidence of reaching the required level of competence in a module of study but the issue for academics teaching a course of study is to identify the complex set of competencies that are important and therefore must be demonstrated by the student. It is then necessary to identify what needs to be assessed at relevant stages within the unit of study and how that assessment will take place. Jeris and Johnson argue that skill level is characteristic not only of a person but of a context and that competence is a function of the context in which it is applied (Jeris and Johnson, 2004). Dreyfus and Dreyfus agree with the idea that competencies are difficult to view solely as generic, stating that "attributes used in accomplishing work are bound to the work context, regardless of the level of competence attained and that in the work situations individuals acquire situational or context-dependent knowledge and skills" (Dreyfus and Dreyfus, 1986). Interestingly, some of the early literature on competency development in the United States focuses on a behavioural view of competency i.e. competency captures skills and dispositions beyond cognitive ability such as self-awareness, self-regulation and social skills which are behavioural, and therefore these skills can be learned through training and development. Boyatzis (1982) studied 2,000 managers holding 41 different positions in 12 organisations and proposed an integrated model of managerial competence, giving competency a broader definition including knowledge and skills alongside behavioural characteristics. Competency models are now widely used to align individual capabilities with the core competences required in an organisation.

One of the key research objectives of this current study (as outlined in Chapter 1) is to determine if programming competency is a primary predictor of team or individual success in the SETP module. To do this it is therefore important to determine what constitutes competency in programming, in industry and academic Computing Science and in Software Engineering programmes. This will involve determining if there is a common set of competencies or a competency model that is used and/or if these are currently appropriate for the module. Only then can I determine if the assessment methods and design adequately capture the skills, knowledge and behavioural characteristics that are appropriate to the level of study (FHEQ 5) within the Computing Science

programme at Newcastle and in the wider aim of preparing students to work in the Software Engineering industry.

2.7.2 Competency in Programming

When trying to define competency in terms of programming or the competencies that a programmer should have 'mastered', it can be difficult. Most programmers will have their limitations e.g. a limitation might be that they do not know the full power of the language or they are unaware of certain algorithms or may be unable to grasp a sufficiently large portion of the problem at one time. However, when a programmer is writing a program or learning to program, in Weinberg's view, their learning takes place "in the context of a particular machine, a particular programming language, in a particular working environment and a particular set of historical events that determine not just the form of the code but also what the code does" (Weinberg, 1971). This view of judging competence, in terms of programming context, fits well with those of Jeris and Johnson and Dreyfus and Dreyfus mentioned in the previous section. If we are to attempt to measure programmer performance and competency levels then we need to answer the following question - what measurements or criteria can we use to determine if one programmer is better than another? The answer to the question, according to Weinberg, is not as simple as we might wish because programming is "complex human behaviour" (Weinberg, 1971). Some assessors/evaluators might discuss years of experience or number of projects completed as indicators of programming competency but neither of these factors could be deemed to accurately describe how 'good' a programmer is at programming. One of the reasons for this is that, in real projects, programmers rarely program on their own. It can therefore be difficult to distinguish their particular or individual effort or impact upon a project (a difficulty replicated in assessing all team projects). Any measurement or criterion for assessment in Higher Education needs to take the problem context and working environment into account. It is difficult also to measure the 'goodness' of individual programs as there is rarely one solution to the problem being solved and problems tend to vary in terms of difficulty level. We can make comparisons of such elements as "scope of language covered, reliability of object code and execution-time monitoring" (Weinberg, 1971), however it is still very difficult to determine absolutely if one program is 'better' than

another in all aspects. More often therefore, we tend to evaluate programs and thus programmers, not in comparison to one another, "but with an holistic view of the situation in which their system or programs are being developed" (Weinberg, 1971).

In Higher Education, determining competency in a programming module is subject to the assessment of abilities, skills and knowledge that are deemed appropriate for the level of the module and the assessment regime in which the module takes place. The criteria for marking vary across programmes and programme levels in all HEIs. In first year modules assessment criteria are, of course, determined as measuring programming competency at a less advanced level than that of second year level and so on. However, the skills, abilities and knowledge which are assessed are fairly concrete and technical i.e. we tend to focus on knowledge and application of programming constructs that have been taught in class that the student must use to design programs that solve specific problems they have been given. There is however uniformly across all HEIs in the UK, little or no focus on teaching, learning and assessing programmer 'behaviour' or 'soft skills' such as planning, scheduling and managing deadlines. These skills tend to be associated more with other modules, such as Project Management, that are taught later than FHEQ Level 4 or 5 in most undergraduate Computing Science (CS) degree programmes. This is certainly the case at Newcastle. Programming is a creative process and this creativity often involves an individual setting their mind, skill and effort towards the solution of a problem. A lot of programmers prefer to work alone especially when designing something new. Most undergraduate CS programmes also begin teaching programming to the individual and focusing on an individual's skill and knowledge development. However, most software projects in industry are large projects that cannot be programmed by an individual on their own. Projects in industry are generally made up of teams of programmers and so it becomes necessary to teach students how to work together on larger scale program and system development throughout their programme, before they leave university i.e. to teach them Software Engineering.

2.7.3 Programming in Teams

Software Engineering is defined as "the application of a systematic, disciplined, quantifiable approach to the development, operation and maintenance of

software, that is, the application of engineering to software" (IEEE Standard Glossary of Software Engineering Terminology, IEEE610 1990) (SE2004). It concerns the development of large programs and is otherwise known as "Programming in the Large" (DeRemer and Kron, 1976). DeRemer and Kron made a distinction between programming-in-the-large and programming-in-the-small, although the borderline between large and small is not sharply defined. Typically a program of 100 lines is small and a program of 50,000 lines is considered large but generally "programming in the large" refers to a multi-person job that spans more than half a year.

According to Weinberg, the worst way to run a programming project is "to hire a horde of trainees and put them to work under pressure and without supervision although this is the most common practice today" (Weinberg, 1971). Aspects of this statement very much describe undergraduate team projects in software engineering. Students who have never experienced teamwork or team programming situations are put together, under pressure, to develop a software system. The one redeeming thing in contrast to this statement is that teachers do tend to supervise student teams quite closely. Weinberg also maintains that there is a complementary relationship between ability and schedule and we could almost produce a programme with *less talent* "if we are willing to allow a stretching of the schedule and if we have not dropped below the minimum competence" (Weinberg, 1971). Weinberg therefore could be said to define a form of minimum competency within a programming team as the ability to both work to tight deadlines and deliver the system, but his interpretation of competency is not precise enough to determine the quality of the system or to measure/assess an individual programmer's skill levels. When designing team projects in the Software Engineering curriculum at tertiary level, university teachers tend to base assessment criteria on a range of different standards and expectations including those specified for the discipline by course-accrediting bodies within industry and academic discipline communities and those specified by quality assurance bodies within UK Higher Education. Added to this, teachers must bear in mind other skills and attributes that are specified as part of all degrees by their institution e.g. at Newcastle we have the Graduate Skills Framework (Newcastle, 2013). In the following sections I will discuss these in broad terms and then use them to evaluate the SETP module outlined in the case study in Chapter 3.

2.7.4 Accreditation, Frameworks and Standards

The Software Engineering curriculum at most HEIs and its assessment are influenced by accrediting bodies and professional organisations and of course national and inter-national quality standards for the discipline at tertiary level, such as those defined by the Quality Assurance Agency in the UK (QAA), The British Computer Society (BCS), The Institute of Electrical and Electronics Engineers (IEEE),The Association for Computing Machinery (ACM) and the IET (Institute of Engineering and Technology).

In practice, according to these standards a software engineer analyses user needs and then designs, tests and develops software to meet those needs. The process involves tasks such as choosing technologies, designing the architecture for the system, creating algorithms, programming, testing, maintaining and evolving an organisation's computer systems. Among the skills that are needed is an ability to problem-solve, to communicate to a variety of audiences and to work well in a team and to pay attention to detail. Software engineers need to commit to life-long learning to keep their technical skills current and relevant because technologies change rapidly and business methods and social needs tend to change over time. Communication skills, leadership skills and team-working skills also need to be updated periodically as software engineering is "a social process as well as a technical one and a failure to recognise the issues involved in social interaction many result in a compromise on the technical quality of a project" (Layzell et al., 2000). The nature of the work of software engineers means that they have to be adaptable and flexible, learn on the job, cope well with solving problems and manage many things going on at once. Professional practice is generally "dominated by team collaboration" (Brodie et al., 2008). Programmers need to be able to work in teams, often in multidisciplinary teams, and it is in this arena that interpersonal or 'soft skills' as well as technical skills are most needed. The Association for Graduate Recruitment in the UK outlines the difficulty some firms have in recruiting students with suitable 'soft skills' as well as academic ability (AGR, 2014). It seems that these days employers expect more from students than just academic knowledge and skills. Many focus on additional qualities such as their proactivity in gaining 'extra' skills during their studies and in terms of demonstrating the skills they already have (Ford, 2007). Joseph et al.(2010) also support the need for students to acquire and demonstrate a

range of skills beyond the core curriculum and recognises that "there is a growing (and gnawing) awareness that technical skills alone are insufficient for success in IT, particularly in today's dynamic, distributed and complex workplace. Companies are exploring outsourcing and offshoring to become more flexible and contain costs while strategically leveraging IT. Consequently, IT professionals must acquire a broader set of skills beyond their technical skills" (Joseph et al., 2010). These broader managerial or interpersonal skills are generically labelled "soft skills" or, as Joseph et al. define them – "practical intelligence".

Teaching emphasis and provision for the subject of Software Engineering ranges across HEIs, from whole programmes on Software Engineering to elective streams in single honours Computer Science programmes and single modules in Computing Science degrees. Universities that offer whole programmes or single modules/streams for Software Engineering (both in the UK and abroad) generally recognise the need for students to develop skills and attributes that will ensure they cope well with the demands of the rapidly-changing environments and diverse teams that are needed in today's software engineering industry. A large number of universities address this need by providing some form of team assignment that requires the development of a software product and its supporting documentation. Typically these projects are also designed to teach students about transferable skills such as team working, leadership and communication, as well as help them to learn more about the particulars that characterise software development in the modern working environment

Some examples of team projects as an aspect of Software Engineering provision from HEIs in the UK include Birmingham University, who run a four-year MEng Software Engineering degree with team projects in the first and second year (Birmingham, 2010); York University who run a year-long 30 credit Software Project module during the second year of its BSc. (Hons.) Computer Science degree (York, 2010) and Queen Mary University of London, who offer a second year Software Engineering module as part of the BSc. Computer Science (G400) degree, where the emphasis is on "large-scale software engineering teamwork, covering design, systems analysis and team skills required by industry" (Queen Mary, 2010). All of these courses on Software Engineering are subject to external evaluation of their teaching quality and fitness for purpose by industrial and academic organisations in the

Computer Science disciplinary community as well as Higher Education quality inspections in the UK. Some of the most common and relevant standards and guidelines used to assess Software Engineering programmes are outlined in the following section.

2.7.5 QAA: Software Engineering-specific Skills and Abilities

The Quality Assurance Agency (QAA) Code of Practice and Subject Benchmarks for Computing inform assessment practices in all UK HE institutions that offer degrees in Computer Science. Subject benchmark statements are an "important external source of reference for higher education institutions. They provide general guidance for articulating learning outcomes associated with the programme but are not a specification of a detailed curriculum in the subject" (QAA, 2000; QAA, 2006; QAA, 2007). The Benchmark statements from the QAA describe the nature and characteristics of programmes in a specific subject and outline the general expectations about the standards for the award of qualifications at a given level. Standards outlined in the documents reflect practice of Computing Science in the UK (and hence Software Engineering as part of that practice) and enable the Learning Outcomes for a particular programme to be reviewed and evaluated against agreed general expectations about standards on a national level.

The QAA standards were drawn up by a group of subject specialists acting on behalf of the academic community and address five major topics – curricular issues, course design, learning, teaching and assessment and finally the benchmark standards themselves. They capture the "intellectual and practical attributes that ought to be developed by study of the subject of Computing to honours degree level and aim to reflect Computing as practised within the UK", (QAA, 2000; QAA, 2006; QAA 2007).

Computing is such a fast paced, rapidly changing discipline that frequent updating of these standards is necessary. Both the QAA 2000 and 2007 documents outline standards for Computing Science course design to ensure "an appropriate balance of theory and practice, including methodologies that ensure students will adopt a disciplined approach to their tasks" but the subject benchmark does not "prescribe any core of material guaranteed to be present in all courses". It does expect that the course should "be up to date in terms of developments in computing and current thinking on curriculum development

and delivery and it should take appropriate account of issues such as employability of its graduates and the needs of employers" (Section 4, Principles of Course Design, QAA 2007).

Part of the knowledge that the QAA standard states Computing Science students should have is "how different teams can be structured" and "approaches to group activity". This description is brief and quite generic in nature and is not directly or specifically related in the document to the curriculum area of Software Engineering. The document specifies that in practical coursework there should be "an opportunity for students to gain experience of working in groups and as an individual" and that the assessment strategy associated with the course be "clearly documented and will allow the Higher Education Institution (HEI) to show that graduating students meet the criteria set in this subject benchmark statement" (QAA, 2007).

Computing Science students and hence, most students studying Software Engineering, are expected to develop a wide range of abilities and skills and these are divided into three broad categories in the QAA documents. The categories are:

1. Computing-related cognitive abilities and skills i.e. abilities and skills related to cognitive tasks;
2. Computing-related practical skills;
3. Additional transferable abilities and skills that may be developed in the context of computing but which are of a general nature and applicable in many other contexts.

(Source: QAA, 2007).

The nature of the additional transferable skills mentioned in Category 3 is not specified. Further to this, the document states that "Cognitive, practical and generic skills need to be placed in the context of the programme of study as designed by the institution and that "the implicit interplay between these identified skills both within and across these categories is recognised" (QAA, 2007).

A student of a Software Engineering Single Honours programme is expected to develop the following practical abilities and skills:

1. The ability to specify, design and construct computer-based systems;
2. The ability to evaluate systems in terms of general quality attributes and possible trade-offs presented within the given problem;
3. The ability to recognise any risks or safety aspects that may be involved in the operation of computing equipment within a given context;
4. The ability to deploy effectively the tools used for the construction and documentation of computer applications, with particular emphasis on understanding the whole process involved in the effective deployment of computers to solve practical problems;
5. The ability to operate computing equipment effectively, taking into account its logical and physical properties.

(Source: QAA, 2007)

The QAA qualifies the extent to which students should acquire these abilities and skills as "dependent on the emphasis of individual degree programmes" where students are expected to deploy these "to a greater and deeper extent than someone who is merely an interested practitioner" (QAA, 2007). These statements are quite vague and the document does not offer specific suggestions as to *how* these skills and abilities should be *taught*, *developed* or *assessed*. The QAA specifies additional transferable skills that are expected from all graduates in Software Engineering and these are:

1. The ability to work as a member of a development team, recognising the different roles within a team and different ways of organising teams;
2. The ability to manage one's own learning and development, including time management and organisational skills;
3. The recognition of the need for continuing professional development as part of lifelong learning.

(Source: QAA, 2007)

These skills are also reiterated in the Higher Education Academy Student Employability Profiles (HEA, 2007).

2.7.6 British Computer Society: Skills and Abilities for Computing Science

Many HEIs seek British Computer Society (BCS) accreditation of their programmes as a form of industrial verification of their educational and practical standards. The BCS undertakes a programme of visits to HEIs to consider their programmes for accreditation leading to CITP, CEng, or CSci status. The society supports the benchmark statements established by the QAA in that they are "broad statements about standards for the awards of honours and Masters Degrees and "embrace the BCS definitions" of these qualifications. It views the undergraduate subject benchmark as an "excellent framework that the society and higher education can use to support the accreditations process". The Society also looks at a range of issues relating to the department in which courses are delivered as well as a range of programme-specific issues as their view is that "The quality of a programme depends not only on its content, syllabuses and assessment, but also on the environment in which it is developed, implemented and improved" as it requires evidence that "students are adequately supported by appropriate learning resources" (BCS, 2014). For a programme to achieve accreditation from the BCS it is expected to meet the requirements set out in the QAA Computing Benchmark statement for honours degrees and the Society specifically seeks evidence that the programme learning outcomes appropriately reflect the abilities and skills defined in the QAA benchmark statement. The BCS visit HEI Computer Science Schools in universities in the UK normally every five years to assess programmes. Their generic guidelines on the quality of programmes are relevant to the current study, as is their code of conduct (BCS, 2011) because they can be used as guidelines for designing modules and their assessment and also to evaluate how effective current assessment practice is in the module at Newcastle.

2.7.7 Institution of Engineering and Technology: Self-Assessment of Competency

The Institution of Engineering and Technology (IET) suggests a self-assessment method for determining competency for engineers who are already

working in industry. The IET also accredits some IT/ Computing courses at third level institutions in the UK (IET, 2014). The IET see no need to define absolute scales for competency but set measures and definitions that focus on areas needed for development. The competency categories or degrees of competency they define are as follows: "Category A - fully competent in the area; Category B - can demonstrate competence in most elements associated with the area; Category C - can demonstrate competence in some elements associated with the area.; Category D - unable to offer any evidence of competence in the area*"* (IET, 2014).

The IET also define more specific levels of competency that practitioners can use to gauge their degree of competency in some aspect of their work and these can be found in Figure 7.

Whilst these levels are useful, they are vague and open to personal interpretation. In terms of the current study they express qualities that are pertinent to the development of behavioural attributes of a professional software engineer. The difficulty is that these might be difficult to test in any undergraduate module using typical assignments. However, they could perhaps be used as a basis to formulate a self-assessment task for students to help them reflect on their work.

So far in this Chapter the literature on learning and assessment in relation to the SETP module has been reviewed, including a review of the assessment culture in Higher Education in the UK and the assessment frameworks and standards that have influenced its development. No review of the literature that relates to the current work would be complete without an examination of the curriculum for Software Engineering as it is from this that all our decisions about the module teaching, assessment and learning stem. The ACM and IEEE recommendations for the curricula of Software Engineering are discussed in the following section.

Level 1 • Performs the activity with significant supervision and guidance. • Performs basic routines and predictable tasks. • Little or no responsibility or autonomy.
Level 2 • Performs the activity in a range of contexts. • Supervision is only required in more complex circumstances. • Some individual responsibility or autonomy.
Level 3 • Performs the activity in some complex and non-routine contexts. • Significant responsibility and autonomy. • Can oversee the work of others.
Level 4: • Performs the activity in a wide range of complex and non-routine contexts. • Substantial personal autonomy. • Can develop others in the activity.
Level 5: • Can take a strategic view. • Applies a significant range of fundamental principles and complex techniques across a wide and often unpredictable variety of contexts. • Wide scope of personal autonomy.

Figure 7: IET Competency Levels (Source: IET, 2014)

2.7.8 Content of Software Engineering Modules

In 1998 The Educational Activities board of the IEEE Computing Society and the ACM Education Board set up a joint task force to review curriculum guidelines for undergraduate programmes in Computing Science. The activity relating to the Software Engineering section of this review was known as the Computing Curricula Software Engineering (CCSE). They defined a set of Knowledge Areas that were deemed core to the curricula and matched their respective learning outcomes to the levels of Bloom's Taxonomy (Bloom, 1956). The resulting body of knowledge is known as Software Engineering Education Knowledge (SEEK), (SEEK, 2003). The steering committee stressed that the core material selected was not a complete curriculum and the learning outcome levels were not necessarily limited to introductory courses early in an undergraduate degree programme. Instead, the curricula provide a foundation for a set of educational modules (or units) that make up the Software

Engineering Curriculum. The Knowledge Areas (KAs) they identified as core were as follows.

- Computing Essentials
- Mathematical and Engineering Fundamentals
- Professional Practice
- Software Modeling and Analysis
- Software Design
- Software Verification and Validation
- Software Evolution
- Software Process
- Software Quality
- Software Management
- Systems and Application Specialties

(Source: SE2004, 2004)

Details of the full content of these Knowledge Areas can be found at (SE2004, 2004). The final draft of the Software Engineering Body of Knowledge was published in 2004. The document now includes Knowledge Areas such as Software Engineering Economics (SE2004, 2004).

The document presents a set of high-level characteristics for Software Engineering graduates and also suggests how the knowledge and skills that are deemed fundamental to Software Engineering can be taught, including a set of skills that students should master as well as the knowledge content outlined. In SE2004 a graduate of an undergraduate Software Engineering programme should:

1. Show mastery of the software engineering knowledge and skills, and professional issues necessary to begin practice as a software engineer – these include ethics, professional conduct and societal needs;
2. Work as an individual and as part of a team to develop and deliver quality software artefacts;
3. Reconcile conflicting project objectives, finding acceptable compromises within limitations of cost, time, knowledge, existing systems, and organizations;

4. Design appropriate solutions in one or more application domains using software engineering approaches that integrate ethical, social, legal, and economic concerns.
5. Demonstrate an understanding of and apply current theories, models, and techniques that provide a basis for problem identification and analysis, software design, development, implementation, verification, and documentation;
6. Demonstrate an understanding and appreciation for the importance of negotiation, effective work habits, leadership, and good communication with stakeholders in a typical software development environment;
7. Learn new models, techniques, and technologies as they emerge and appreciate the necessity of such continuing professional development.

(Source: SE2004 Volume – 8/23/2004/15)

Learning outcome 2 is particularly relevant to the SETP at Newcastle as it further details the level of knowledge a student should have of team work including "an emphasis on the importance of such matters as a disciplined approach, the need to adhere to deadlines, communication, and individual as well as team performance evaluations" (SE2004, 2004). The steering committee also stresses that there should be a strong real world element to the work students carry out and that they should experience at least one major activity during their studies that involves producing a solution for a client.

This section of the literature review is relevant to the current work because it outlines the external and internal evaluation mechanisms and standards that are used to evaluate the quality of the teaching and learning in undergraduate Computing Science and Software Engineering degrees in the UK. These can be used as a source of comparison to the content and learning design of the SETP. The SETP is however but one module on the Computing Science degree at Newcastle and it is important to evaluate the extent to which one module can encapsulate the learning outcomes, best practice and required content outlined by these standards. The SE2004 and SEEK Knowledge Areas give a good starting point for evaluation of the module content in terms of relevance and breadth of coverage. This document gives a clearer overview of skills and knowledge that should be part of the teaching, learning and assessment in the SETP module.

Overall the formal standards outlined by accrediting and quality assurance bodies such as the BCS and QAA and from discipline committees from the ACM and IEEE are somewhat vague in terms of providing guidance on what teachers should teach and how they should assess that standards are met. Whilst this makes standards and frameworks more flexible for HEIs to implement, it can also make it very difficult to design a whole Software Engineering curriculum for a degree programme or a single software engineering module as part of a curriculum, perhaps more so in terms of the soft skills that are outlined. Testing knowledge is somewhat easier than testing relevant skills and whether those skill levels have been reached, especially in terms of soft skills. It can sometimes prove more difficult to design suitable assessments to capture skill development and learning as well as practical technical skills and the required body of knowledge.

2.8 Summary

This literature review has shown that we need to take the focus off marks and onto the learning needed for students to become autonomous learners and software engineers in the real world as the current focus on degree classifications and marks does not fit well with our aim to create lifelong learners (Burgess, 2007). To do this we may need to change current assessment practice and focus more on the characteristics, behaviours, skills and knowledge needed to demonstrate competency in such a practical subject as Software Engineering rather than relying on some judgement of innate 'ability' (Black and William, 1998; Dweck, and Leggett, 1988). Designing assessment schemes to do this is not easy. We need to make students aware of the level of difficulty involved in their work and how much effort and practice might be required to achieve a specific level of competence and one way we can do this is to build time for reflection into our teaching and also via the use of formative feedback (Kolb, 1984; Torrance, 2007). The literature review has also shown that students need help in assessing their own existing knowledge and competence in order to progress and improve and we can do this by scaffolding learning using taxonomies such as that outlined by Anderson and Krathwohl (2001). In classes, students need frequent opportunities to perform and receive suggestions for improvement and at various points they need chances to reflect on what they have learned, what they still need to know and how to assess

themselves and their peers. We need to "forge educationally strong links between learner needs, learning outcomes, resources, learning and teaching strategies, assessment criteria and evaluation" (Chickering and Gamson, 1987, 1991). Donnelly and Fitzmaurice, (2005) advocate that to do this, we must start "within the context of a theoretical framework" as well as adhere to expected standards outlined by accrediting bodies. However, as teachers, we also need to identify our own theory of learning and remember that there is no universal way of learning. We need a broad value system for our theory of learning and need to consider the learning experience in terms of a student's whole programme of study and their prior learning experiences. Teachers need to formulate their ideas about learners and why the subject matter they are teaching is important. They need to focus on learning rather than teaching and factor in ways in which students are able to demonstrate their learning and their progression towards learning goals. Taxonomies (such as those of Bloom, and Anderson and Krathwohl) provide frameworks for all intended types and levels of learning and a basis for designing appropriate and fair assessment tasks. Examples from the literature of higher education show that learning can often be neglected because of the distractions of institutional or 'political' concerns about assessment and the increasing pursuit of grades (Boud and Falchikov, 2007; Burgess, 2007). We also need to remedy the fact that assessment is not always considered an integral part of teaching and learning and that we may need to review the alignment between the two to improve our expectations of students and the way we set learning goals (Biggs, 2001; Brown, 2004). The development or adaptation of an assessment framework is "an act of scholarship and development" (Imrie, 1995) and that is one of the intentions of this research. Assessment should be for learning, rather than just the assessment of performance and good assessment is "an intrinsic part of teaching" which initiates and manages learning (Imrie, 1995).

In the following chapters, a case study of the SETP Module at Newcastle University is presented. The assessment methods used are then analysed to determine their 'fitness' for purpose, using both statistical and qualitative methods, bearing in mind the issues raised in this literature review and in fulfilment of the research objectives 1, 2 and 3 as outlined in Chapter 1.

Chapter 3: Case Study: The Software Engineering Team Project Module at Newcastle University

3.1 Introduction

This chapter examines the Software Engineering Team Project module (SETP) at Newcastle University that will be used as a case study for the current research. It outlines the format of the module and its assessment prior to 2005 and then details the changes made when it became a focus of the CETL project Active Learning in Computing (CETL ALiC, 2005-2010). The teaching approach and assessment methods introduced by CETL ALiC are described and a brief overview of student and staff experiences and feedback, including some initial work conducted on assessment at Newcastle, is presented. Finally the results and implications of the feedback and experiences of assessment for the current research work are discussed.

3.2 Module Design and Pedagogy

Prior to 2005, the Software Engineering module in the School of Computing Science at Newcastle University was taught in the manner of most undergraduate modules in the UK, i.e. via a series of lectures and practical laboratory sessions. The assessment regime for the module was also based on a typical team project model with the submission and examination of individual items of coursework from each student for partial fulfilment of the module assessment criteria and of collaborative work from small teams for the rest of the module. Pre-requisite modules of study for the module included (and still include) a combination of programming modules that involve problem solving, program design and development and data structures and algorithms, with a focus on Object-Oriented programming techniques. Over the years the module has evolved to have a strong team-based focus with increased emphasis being placed on the 'soft skills' required for working in a team of developers, as well as the technical skills needed for large software development projects. One reason for the more emphasis on soft skills is that the needs of the Software Engineering discipline itself have changed over time and the module has been

adapted to reflect industrial practice more accurately, with a view to helping increase student employability in a competitive global market. The modern industrial software engineering process "typically involves participation of software designers, programmers, end-users and domain experts and is essentially a team-based activity, involving a wide variety of stakeholders" (Layzell et al., 2000). Improvements in communication technologies and economic pragmatism also mean that many software development companies now find it more efficient to develop their products collaboratively across different geographical sites. This division of effort allows them to access skills and expertise across the globe and to take advantage of different time zones to develop software 24 hours a day. It also allows the industry to save money on travel and to manage their interests more easily even though they are distributed around the globe. So, for example, it is possible to manage a project in India from your desk in the UK and to use communication technologies to contact your development team. This change in practice necessitated an additional shift in focus when teaching and learning Software Engineering. Use of modern communication technologies and the ability to work in a distributed team are now part of the skill set undergraduate software engineers need to acquire so that they can be desirable employees upon graduation. Cross-site software development necessitates consideration of the following issues for teaching:

- Most cross-site interactions are dependent on technology to facilitate the collaboration;
- There will likely be time restrictions for communications between teams and on how often they can work together;
- There may be cultural differences in working practices between the sites involved in the interactions;
- Working relationships may be more difficult to develop and sustain than they would if teams were co-located.

For a graduate software engineer to be more employable in the current job market, it is in their interests not only to have acquired the requisite team-working and communications skills for working in a modern software engineering environment, but to have experienced or mastered the ability to develop software collaboratively under cross-site conditions.

3.2.1 Introduction of Cross-Site Development

Active Learning in Computing (ALiC) was a five-year collaborative CETL (Centre of Excellence in Teaching and Learning) funded by the Higher Education Funding Council for England (HEFCE) from 2005-2010. The CETL involved four consortium partners from the discipline of Computing Science and these were Durham University (project lead), Newcastle University, Leeds Metropolitan University and the University of Leeds.

The fundamental aim of ALiC was to identify ways of engaging students more in their learning of computing, through project and team work, and to enable these students to become more independent learners and more employable.

In 2005, in response to the perceived need for inclusion of cross-site software development experience in the teaching of Software Engineering, the CETL ALiC team extended the Software Engineering modules at Durham and Newcastle to include inter-institutional collaboration between student development teams. Teams were formed into 'companies' with each company comprising a team of students from Durham University and a team from Newcastle University (geographically separated by 18 miles). In the academic year 2005-06 we had 12 companies, in 2006-07 there were 12, in 2007-08 there were 10 and in 2008-09 there were 12. These companies collaborated over the course of the academic year to produce a software product and its associated documentation.

The pedagogical aims of the cross-site collaboration were to give students some experience of Software Engineering in an industrial context, to make problem solving more realistic and engaging, to allow staff and students to use and evaluate various communication technologies for cooperative working, and to encourage the development of transferable skills.

The intended knowledge outcomes of the respective Software Engineering modules were, and still are, an understanding of the issues that relate to planning and the execution of a team-based software development project. The intended skills outcomes were practical experience in issues such as team structure, document preparation, project management and the design and implementation of a large software system, the ability to work as a member of a team and to fulfil appropriate roles and apply these skills to the project at hand (Module Outline, 2014). The module, as part of the undergraduate degree structure at Newcastle, was outlined formally in the module description as

having the value of 20 credits (10 ECTS) out of the 120 credits allocated to Stage 2 (FHEQ Level 5) and involved 12 hours of lectures, 20 practical hours and 168 hours of private study (200 hours of student study in total). The practical hours were used for team meetings at Newcastle where teams could organise the work of the project. The assessment methods were based on 100% coursework. At Durham the credit weighting of the Software Engineering module differed in that it was essentially a double credit module (40 credits, 20 ECTS). There were also differences in the number of timetabled lecture and practical sessions, with Durham having more in-depth lectures on the subject and also more formally supervised and scheduled laboratory hours in their timetable to work on the project. Newcastle's laboratory work for the project was not supervised but staff members acting as 'monitors' attended one hourly meeting per week for their team/s. Monitors were in attendance to observe the teams and provide guidance to them throughout the project, if needed. Each Durham team had a project manager who was a third year Computer Science student studying a Level 6 Project Management module. These Level 6 students took responsibility for project management for the local Durham team, making recommendations for the co-ordination and allocation of tasks as well as being involved in the setting and tracking of internal deadlines. These project managers also met their teams on a weekly basis.

Lectures focusing on Software Engineering theory took place at Newcastle during the first 5 weeks of the first semester only and thereafter students were to meet in their teams and conduct the project work without formal lecture slots. The initial lectures gave a flavour of Software Engineering as a discipline and included an overview of Software Lifecycle stages and Process Models, Project Planning, Team Organisation and Structure, Project Management, Requirements Elicitation and Analysis, Design, Configuration Management, and Testing and Debugging.

The necessity for cross-site collaboration to complete the project placed a strong emphasis on students managing their own teams, communicating with their colleagues at the partner site, distributing tasks and responsibilities between the two halves of their company, and planning the project together, all of which emulate current practice in the software industry. This design also mapped directly to one of the fundamental goals of ALiC i.e. to introduce a strong element of independent learning that would allow students to practice and develop their skills with minimal time spent 'receiving' knowledge in a

static classroom environment. Cross-site teams had to define an organisational structure, choose their preferred software design methodology, plan the software design, estimate the effort needed, consider the schedule for implementation, and allocate the work fairly between sites. They also needed to plan for software integration across sites and the testing and final demonstration of their product. Throughout the whole process teams also had to produce reports, update module leaders on the status of their project and deal with any personnel and technical issues that arose during the development process. Cross-site companies had to arrange meetings in their common practical hours so they could collaborate and complete the work.

3.2.2 Supporting Technologies and Materials Provided

Communication and cooperation are an "inherent part of the social process of Software Engineering" and therefore access to good communication technologies was critical to the successful functioning of cross-site teams, especially as the task was complex (O'Neal, 2004; Johnston and Miles, 2004).

Video-conferencing equipment was provided for the companies to facilitate the collaboration (and to emulate communication conditions of globally dispersed Software Engineering work). Both sites used Access Grid software (Access Grid, 2014). Access Grid provides multimedia capability that allows the interconnection of a high number of geographically distributed groups and is often used by academic institutions in the UK. We used Access Grid because at the time Skype was unreliable for some connections and not as well-used or effective as it is now.

The preferred development environment for the software was the Eclipse IDE that also facilitated version control via Subversion (Eclipse, 2014; Subversion, 2014). The shared version control was very important as students at each site needed to be able to edit code simultaneously and also to see the edits and revisions made by others. Discussion forums and document repositories were also provide by Newcastle via NESS (Newcastle E-learning Support System), a customised online learning environment. NESS is a web-based system developed within the School of Computing Science at Newcastle by Dr Lindsay Marshall (NESS, 2014). It allows students to submit coursework, view results and to receive online feedback from their tutor. It also supports staff in the management of learning, teaching and assessment. Cross-site teams were also

provided with Wikis, FAQ pages and had access to instant messaging for communication.

The nature of the project, and ALiC's drive to encourage autonomy of the companies, meant that students were not explicitly taught how to use all of the communication technologies. They were given basic instructions but were expected to learn how to use them effectively as part of the project work. This approach was in line with the design of other group projects within the discipline (Liu et al., 2002).

3.2.3 Assessment Methods

The Software Engineering modules were re-designed to collectively fulfil the standard learning outcomes of both Durham and Newcastle's original modules. Table 1 illustrates the mapping between the learning outcomes at Newcastle and the project deliverables. It also denotes which deliverables were individual submissions (I), local team only deliverables at Newcastle (T) and cross-site company deliverables (C), which were shared assignments between the companies at Durham and Newcastle.

Learning Outcomes	Deliverables
Communication with customer	(C) – requirements analysis meeting Email, notes.
Problem solving, Requirements Analysis Use of initiative, planning, choice of software development model	Project Plan (C), Project Specification (C) Team Structure Essay (I)
Software Design, industry standards and practices for design notation	Project Design document (C)
Programming, testing, software development	Software source code and documentation (C), user manuals (C), Completed Project Specification and Design (C)
Adaptability, Leadership, inter-personal communication, reflection, cross-site communication and collaboration, work as a member of a team, fulfil roles, time	Team Reports (T), Personal skills analysis (I) Meeting minutes and observations (T), Team Contract (T), Project Log Books (I) and

management and organisation, Project Management	(T), peer assessment, individual reflective report (I)
Team and Company Communication	Team Presentation (T) Interim and Final Report (T), Minutes and Log Book (T)
Written communication skills	Team Report (T), Individual Reflective Report (I), Documentation (C)
Professionalism	CV and Covering Letter (I) Mock Interview (I)

Table 1: Learning Outcomes mapped to Project Deliverables

Initially, at the beginning of the cross-site work, it was decided that all shared deliverables for each company would form part of the summative assessment for the module at both sites and reports from the local teams and individuals would then be used to help determine individual effort. Making shared deliverables high in value in terms of assessment credit was intended to ensure good cooperation between companies as they would each receive the same mark for these aspects of the work i.e. they needed to collaborate well in order to obtain good marks for the company assignments. In practice, quite early on in the first semester of the cross-site initiative, it was realised that such a simple approach was not possible. Some students at each site felt they had contributed much more than others and deemed the approach unfair if their share of the work was particularly good but another section was perceived as bringing the overall assessment mark down. To overcome the problem of determining contribution from each site and each individual, a contribution matrix was designed by staff at Durham (Figure 8).

Sections	Joe (Dur)	Kirill (NCL)	Mike (Dur)	Tom (NCL)
1.0 Introduction	CMR	R	M	R
1.1 Purpose	CMR	R		R
2.1.1 PC Modules	MR	MR	CMR	CMR
2.1.2 PDA Modules	M	C	MR	C
3.1.1 PC Modules	CMR	CMR	R	R
3.1.2 PDA Modules	CM	MR	CM	MR
3.2 Inter-process deps.	MR	CM	MR	CMR
3.2.1 PC Modules	CMR	R	MR	CMR

Key:
C – Create
M – Modify
R - Review

Figure 8: A sample contribution matrix

The contribution matrix allowed students to specify the exact nature of each individual's contribution to a project deliverable, for both local team deliverables and cross-site company deliverables. The module leaders specified that a contribution matrix should be included at each site for every company submission. The matrix provided the opportunity for each team member to describe their contribution in terms of the action they had taken on the deliverable e.g. creating (C), modifying (M), editing or reviewing (R) and for each company to specify which half of the company had contributed to each section of the deliverables. The matrix example in Figure 8 illustrates clearly those sections that were completed by Newcastle and those completed by Durham (student names have been changed). Teams at each site did not have to agree with the contributions in the matrices but staff at each site compared them and students were made aware that this would happen. The team project also provided an opportunity for students to evaluate their own performance and the performance of others in their team and cross-site company. It is common for most university courses to include some form of peer assessment in teamwork scenarios and staff felt this was an important aspect of the work that would allow students to reflect on their performance as a cross-site unit as well as on their local performance. Durham and Newcastle used different peer assessment methods as part of their team projects and these are outlined in the following sections:

3.2.4 Peer Assessment Methods

Newcastle: Percentage Sharing

At Newcastle, an holistic percentage-sharing form of peer assessment was carried out at two intervals during the academic year for the project. Teams were asked to share 100% between their team members based on their contribution to the team effort, once during each semester. The exact share for each person in the team was to be discussed openly in a formal team meeting and the agreed percentages noted down and then submitted to the module leader. Coming to an agreement publicly in this way often proved difficult for students and was quite emotive so staff monitors were to provide guidance to students on how to conduct the exercise, including allowing all student opinions to be heard during the discussion.

Durham: Self and Peer Ranking

Durham students completed four self and peer-assessment tasks throughout the duration of the project. Each student was asked to place themselves and their team members on a grid of 15 places (a low value being for the most contribution). In this way, they were able to demonstrate exceptional or non-exceptional contribution. This process forced students to evaluate their own performance in comparison to other team members. Project managers were also tasked with completing peer rankings for each of their team members.

Cross-site Percentage Sharing

In addition to these preferred methods of self and peer-assessment that had already been in use by each site prior to the cross-site initiative, each company was asked to divide 100% between each half of their development team (i.e. between their respective team members at Newcastle and at Durham). The CETL team were interested to see how students perceived the contribution of their local team in comparison to their colleagues at the other site. Unlike the other peer assessment methods used, this cross-site percentage sharing was not intended for summative assessment purposes for the project. Students were told that they did not need to confer with the other site to derive the percentages but they could if they wanted to.

3.2.5 Formative Assessment and Feedback

As part of the learning process, each company had to submit a draft version for each of the two major written deliverables (Project Requirements and Design documents), and feedback was provided. This feedback was a combination of the comments from coordinating staff at both Newcastle and Durham and came in the form of comments on the draft document and as verbal feedback to each team at their own site. Other methods of formative feedback used during the project were comments and advice from monitors (Newcastle) and project managers (Durham) during weekly meetings. At Newcastle each team was given an overview of their progress based on their average grade for team and company deliverables throughout the year. This meant that teams knew if they needed to make more effort as final marks and weightings were not calculated until the end of the project.

3.3 Calculating a Final Individual Mark

All Company deliverables had common marking schemes at Durham and Newcastle and were essentially double marked. Durham students' team marks were calculated in the same way for company deliverables. The differences in module credits meant that Durham students had to take an exam and other individual assessments as part of their Software Engineering module. The cross-site team project only constituted 75% of their coursework mark, with the other 25% spent on assignment tasks unrelated to the team project and with the overall weighting value for the module of 60% coursework and 40% exam.

Newcastle students were not given explicit marks for collaboration but elements of their coursework did depend on their interactions with the Durham half of their Company. They had, for example, to compile reports on what effects using the software on differing hardware would mean for the user and this involved a comparison of features and functionality across sites. Newcastle students also had to report on how collaborations had gone in a presentation and in both their individual reflective report and the final team report at the end of the module.

Durham students had to compile a personal diary of all meetings, either local or cross-site, logging items agreed and any other issues and concerns that had arisen. In addition, each student had to produce a legacy report where they discussed the team project, primarily from a local perspective e.g. team

dynamics, how improvements could have been made and an overview of their own contribution to the project. A section of this report also contained discussion on the impact of the cross-site collaboration on the work.

Team and individual marks for process and product were combined and calculated at Newcastle to form the overall module mark. This was carried out by separating deliverables and processes into team deliverables, individual deliverables and individual and team effectiveness marks (observations from monitors). The components A, B, C and D were then multiplied by a student's contribution weighting which was derived as an average of their peer assessment marks. The peer assessment marks were moderated by Team Monitors and adjusted if deemed unfair in comparison to information recorded on contribution matrices and from direct observations.

The component assignments for A, B, C and D were assessed and combined as follows.

Mark A was the team mark worth 25% of the overall module mark. Deliverables that contributed 70% of this mark were:

- Team contract (25% of 70%);
- Interim Team Report (20% of 70%);
- Final Team Report and Log Book (25% of 70%).

The remaining 30% of Mark A was allocated by the team monitor based on their view of Team Effectiveness throughout the project and was assessed by observation of the process in meetings and reading reports. Areas that were considered when assigning this mark were e.g. good distribution of effort, professional behaviour in and outside of formal meetings, how the team followed and updated the project plan, how the team dealt with issues such as absenteeism, communication and problem solving.

Mark B was an individual mark that constituted 15% of the Module Mark. It was based on individual deliverables. Deliverables that contributed to the Mark B were the Individual Reflective Report and Individual Log book (5% of the Module Mark). The remaining 10% of Mark B was allocated by the team monitor based on their view of each student's individual effectiveness during the project and was assessed by observation of their performance in the team process. Areas that were considered when assigning this mark included e.g.

each student's contribution in meetings, their contribution to the technical and non-technical deliverables (e.g. documentation, presentations) as well as organisational and leadership aspects of managing the project and whether the student was constructive, proactive, contributed to decision making and completed their tasks on time, was reliable.

Mark C was a team mark that constituted 40% of the Module Mark. Mark C was based on the following tangible team (and Company) deliverables:

- Final Project Specification and Design Document (10% of the Module Mark);
- Prototype Demonstration (10% of the Module Mark);
- Implementation (Design, source code and associated documentation) (10% of the Module Mark);
- Final Product Demonstration (10% of the Module Mark).

Mark D was an individual mark and was worth 20% of the overall Module Mark. Mark D was the mark awarded by the Module Leader for the following individual deliverables:

- Personal Skills Analysis (10% of the Module Mark);
- CV and Covering Letter (5% of the Module Mark);
- Mock Interview (5% of the Module Mark).

3.3.1 Weighting for individual effort

To come up with the final individual mark for the module for each student, the team monitor reviewed contribution matrices and peer assessment scores to make a judgement of a student's overall effort in relation to their team members for all team deliverables and processes. This weighting (derived from the 100% sharing in the peer assessment exercises by dividing each average score by 10) was then used as a multiplier on all team deliverables and then the final Team Project Mark (*M* in the following equation), was computed as:

$$M_i = B \times 0.15 + (A \times 0.25 + C \text{ x } 0.40) \text{ x } w_i / \max_i w_i + D \text{ x } 0.20$$

Here w_i is the student's weight allocated by Team Monitors and $\max_i w_i$ is the maximum weight given to any member of the team. Thus a student with the

maximum weight got the full Mark A and Mark C marks awarded to the team, a student with half the maximum weighting got half the Mark A and Mark C marks etc. until all team member weightings were allocated. Professor P.A. Lee and Dr C. Phillips from the School of Computing Science at Newcastle devised the original assessment scheme and use of peer assessment weights in this way.

3.4 Feedback and Experiences of Students and Staff

Throughout the CETL ALiC cross-site initiative between the academic years 2005 to 2010, we gathered feedback from students and staff in a number of ways. We used Focus Groups, module questionnaires, observations from team meetings and anonymised reports and log books from teams to gather information on the student experience and to assess the effectiveness of our work in light of the original aims of the CETL ALiC project. We compared module quantitative results across sites and also compared student learning outcomes and performance with results and experiences at the end of each year with results from the modules in the years prior to the cross-site initiative.

3.4.1 Project Issues

Feedback across both sites and across all years indicated that there were issues that needed to be addressed in our module design. These issues presented themselves in a number of ways but can be categorised as falling under one of the following three categories of Communication and Coordination, Technical, and finally, Assessment.

3.4.2 Communication and Coordination Issues

Staff purposely did not specify that teams should meet face to face before the cross-site work began. Some companies did choose to meet of their own accord. Those that did not were unfamiliar with each other when work began and had not built up any form of relationship or rapport before their first online meeting. A major consequence of this was that the majority of companies found it hard to view their off-site team as being part of the same company. This lack of relationship meant that students were not greatly motivated to help each other across site and often found it hard to respond in a timely fashion to help each other solve problems. This is similar to a reported problem in industry

where cross-site work introduces delays, with a significant slow-down of work in geographically distributed sites (Hersleb, Mockus et al., 2000).

Students also found it hard to schedule meetings because of differing programmes of study within the cohort and the cross-site element exacerbated this issue, even though the CETL ALiC team had aligned their respective timetabled hours for the project.

Each set of students assumed that the content, delivery and emphasis of the Software Engineering module at both sites were exactly the same i.e. the practical work had the same objectives and deliverables and the same deadlines. This was true for all the company deliverables but not for the individual assignments associated with the module. The emphasis at each site was in fact different. At Durham the emphasis was primarily on the production of a complete requirement specification followed by the design and implementation of the software i.e. a standard waterfall model of development whereas at Newcastle, it was on early implementation and prototyping.

3.4.3 Technical Issues

Students experienced connection difficulties in the video conferencing sessions. The majority of these issues were mainly due to the use of inferior hardware that resulted in patchy audio, poor images from the small webcams, server crashes, and loss of video or audio during meetings. There was also a marked lack of contingency planning on the part of staff and students if things went wrong during the video-conferencing. The reality of communication difficulties overshadowed the students' interest and enthusiasm for cross-site working and often left them feeling demotivated. Students thought that video conferencing was to be the main form of communication and therefore mandatory. Staff did not convey strongly enough that communication was the most important aspect and that video conferencing was just one way to ensure this. A full overview of Communication and Technical issues experienced by the students during the project can be found in (Charlton et al., 2009).

3.5 Assessment Issues

The cross-site work put assessment and team assessment in particular, more sharply into focus. Teamwork assessment invariably involves allocating an individual mark for both *product* and *process* and, as mentioned earlier, it is the

individual and team marks for process contribution that often proves problematic to derive. With either co-located or cross-site work, it is vital that each individual is assessed fairly so that those who contribute significantly are rewarded and those that don't contribute do not benefit from the effort of their more conscientious colleagues (non-contribution is a well-recognised problem in student groups where a member of the group contributes little or nothing to the group's activities). Working across sites / universities made addressing these issues more imperative. It was very important to make the assessment methods clear to the students at both sites to reassure them that a poor collaboration between two teams would not necessarily be detrimental to their overall marks for the module.

The cross-site percentage sharing exercise turned out to be quite problematic. Some of the companies decided that they would confer for this distribution and consequently the discussions turned out to be quite heated. There was considerable disagreement over which site had contributed the most effort. The CETL ALiC team had some idea that a few collaborations had not been as productive as was hoped and the sharing of percentages across sites bore this out, with several companies completely disagreeing over the appropriate distribution. Students worried about the balance in workloads because Durham's Software Engineering module was a double credit module and they wondered if they should take on double the amount of work Newcastle did for the project. In a similar vein, at Newcastle, the teams were mainly made up of Computing Science (CS) and Information Systems (IS) students - two different programmes that run within the School of Computing Science. The difference between these programmes is that IS students take modules from the Business School as part of their course whereas CS students concentrate on modules provided within The School of Computing Science. During the project most IS students from Newcastle reported that they did *all* the documentation for their team whilst the CS students tended to report they did more of the technical work (Charlton et al., 2009). The CETL ALiC team felt that it was important that whilst students were encouraged to work to their strengths, this should not have precluded them from improving on skills they viewed as weak or in need of improvement. The idea of a student being pigeon-holed because of the focus of their degree programme was worrying.

The introduction of contribution matrices helped to reassure students, to some extent, that all their efforts were taken into account. During the course of these

projects in the past, it had been noticed that students at both sites tended to view the coding of the system as the most important part of the Software Engineering process and that *soft skills* such as organising meetings, project planning and management etc. were often viewed as less crucial to the overall team performance. Completion of the contribution matrices not only helped to reassure students that their efforts were recorded but also made some realise the importance of all types of effort to the process, regardless of whether the task was writing code, project management or writing documentation. There were, however, arguments among Newcastle teams and mention of team members who had kept all the work to themselves so they would get a greater share of the marks. Staff at Newcastle also observed early on in the CETL ALiC project that during the first peer percentage sharing exercise, teams tended to divide the 100% quite evenly across all team members whereas the second set of peer percentages allocated during or just after the rather difficult implementation and delivery phase (towards the end of the project), reflected a greater difference between percentages awarded to team members. At this point they tended to be more inclined to debate and actively discuss the distribution of the 100% locally. It was the second peer assessment, the summative exercise, which seemed to cause the most friction between teams. Feedback from the module at Newcastle on peer assessment was quite negative as can be seen in Figure 9:

> "I was not happy with my original peer percentage for the second semester which was 14.4, mostly because I felt like I had done more work than some others. After listening to my objections, my mark was increased by the team. However, this meant that two other students lost some marks as a result. Obviously they were not happy about this. Eventually, we all (reluctantly) agreed on a mark that was fair. Personally I don't think this is a good system as it can be abused easily, for example, two members could unfairly rate each other. Also, there is only one role for IS students to play. This means that they will never get high marks."

Figure 9: Student feedback on peer assessment

Students needed reassurance that the assessment methods used by staff were reliable and fair. As there had been so much negative feedback on assessment and its fairness within local teams and across sites, each site decided to review its assessment methods for the programme in more detail, to determine if the

methods introduced by CETL ALiC had in some way disadvantaged the students taking part, in comparison to previous cohorts who had not taken part in cross-site work. As part of the CETL mid-term review in 2007-08 I analysed student grades, team reports, module feedback and individual feedback reports from the years 2003-2007 i.e. from two years before CETL ALiC started. I found that, in general, students who did not contribute largely to the coding of the product during the project received lower grades, on average, across all years, even those prior to CETL ALiC.

As can be seen in Table 2, there were 108 non-coders (33.5% approximately of the total number of completing students, across all years) and 212 coders. The data indicated that 57% of non-coders and 39% of coders scored less than their team's average mark. These figures were quite interesting considering that the coding effort and software product were worth only 5% of the total module marks available. The reason the product is worth so very little in the module assessment percentages is that the module leaders wanted to emphasise that all aspects of Software Engineering are important, not just the end product but the process. But the questions the CETL ALiC team asked were, were these basic statistics reasonable in terms of what any student can hope to attain in the module and were non-coders just naturally 'weaker' students?

Implement	1st (70+)	2.1 (60-69)	2.2 (50-59)	3rd (40-49)
No	26	42	31	9
Yes	67	91	46	8
Total	93	133	77	17

Table 2: Marks of Coders and Non-Coders from Newcastle

Student feedback also indicated that IS students were often given a lower peer percentage mark as their tasks and contribution were not deemed as strong as the coding effort made by the programmers from CS and this seemed to be borne out in the overall attainment in terms of marks for these students across all the years reviewed. It was felt that these basic statistics indicated there was something more going on with assessment in the module and the CETL ALiC work, which had prompted the review, had unearthed a possible weakness in the assessment strategy and design for the Team Project module.

3.6 Implications from the Initial CETL ALiC Review of Assessment

CETL ALiC staff believed that the learning outcomes for the module (detailed earlier in Table 1) were sufficiently broad to cater for a range of abilities amongst the student cohort and that terms such as "*practical experience in design and implementation"* (MOF, 2014) should cater for all the processes associated with the design and implementation of a system, including documentation and other 'non-coding' aspects. If a student demonstrated that they had achieved these learning outcomes (to varying levels, of course), then the differences between what coders and non-coders did during the project, should have been largely irrelevant. One of the main emphases of the module is teamwork and the complex processes involved in developing a software product. Those who taught the module deliberately wanted to emphasise all aspect of Software Engineering i.e. that it is not just about coding and there is so much more involved in meeting the goal of delivering a quality software product.

The results from the analysis of marks and feedback made the CETL team wonder if coders were perceived as 'better' Software Engineers, or even as 'better' students, by the students themselves and whether the design of the module and its assessment serve to reinforce this perception. The results also raised questions about the effectiveness of the assessment regime and its fairness. The CETL ALiC team found that there were a number of questions that arose e.g. what skills and learning were really being assessed by all these deliverables and were we capturing the *process* accurately enough to assess it? Is a mark for the project a sufficient indicator for an employer in industry to determine the extent of a student's Software Engineering skills? Is the mark generated by the assessment methods used during the module a true reflection of what the student has learned about Software Engineering? And most importantly, was there a bias towards coders in the assessment methods used or in the way they were marked and if so, did peer-assessment contribute to or further exacerbate this bias?

These initial results provided the motivation for the current research as they highlighted a need to conduct a more in-depth analysis of the effectiveness of the assessment methods used during the module and a deeper evaluation of student learning outcomes and experiences. These results also illustrated that peer assessment and student perception of the value of technical and non-

technical contributions to the project were particular areas that merited examination.

3.7 Summary

This chapter has provided an overview of the SETP module at Newcastle during the CETL AIiC project. It has also outlined the findings from an informal review of the assessment of the SETP module at Newcastle's School of Computing Science that were highlighted when the CETL ALiC initiative undertook cross-site team work with Durham University. These findings stemmed from both student feedback and staff experiences during the module and were the origin of the research questions outlined in Chapter 1 of this thesis. The following chapter outlines the methods used to further evaluate the assessment regime for the module at Newcastle in a bid to find answers to these questions.

Chapter 4: Methods

4.1 Introduction

This chapter outlines the sources of evidence used during this research, details how the data were gathered and structured, and describes the methods used in analysing the data. A description of each of the sources is given and an overview of the statistical methods that were used to evaluate the quantitative data is presented. The techniques for gathering qualitative data and the methods used for evaluating these sources are also presented. Finally, limitations of the data and data collection methods used during this study are outlined.

4.2 Sources of evidence:

To begin evaluating the assessment regime used in the SETP module in more detail, a more detailed review of results and student learning outcomes needed to be collected. It was decided to take the results from two years prior to the CETL ALiC initiative (academic years 2003/04 and 2004/05), and to compare these with the outcomes and results from the first two years of the CETL ALiC project (2005/06 and 2006/07), as in the preliminary analysis from the CETL review. There had been notable differences in the conditions prevailing when assignments were set for students during these two time periods. Thus it was felt that it would be easier to determine if the cross-site work had unduly influenced student experiences and resulted in a negative fashion and/or exacerbated the perceptions of staff or students about coders and non-coders.

The datasets considered most relevant to the current enquiry and assessment regime evaluation were as follows.

1. Summative module marks for all students completing the SETP at Newcastle for the academic years 2004/05-2006/07.
2. Peer assessment results from the Team Project at Newcastle for the academic years 2004-2007. These comprised two sets of peer assessment values for each year from the 36 student teams that participated in the SETP module.

3. Individual Reflective Reports from 2003/4-2006/07 for completing students. There were 322 reflective reports in total, each averaging 1500 words.
4. Focus group results and feedback on experience with an employer as focus group host from 2003/04-2006/07. This focus group comprised 24 students, 12 from Newcastle and 12 from Durham University (1 student per team from each site).
5. Focus group results and feedback with CETL ALiC staff (a PhD researcher who was not involved in assessing students), and representatives from all student teams from the academic year 2007/08. This focus group comprised 20 students, 10 from Newcastle and 10 from Durham.
6. Focus group results and feedback with representatives from all student teams with an employer as facilitator from the academic year 2008/09. This focus group comprised 24 students, 12 from Newcastle and 12 from Durham (1 student per team from each site).
7. Student programming marks from first year programming/technical modules at Newcastle. The results from two modules were used: Programming and Data Structures (a CS module) and Web Development (an IS module). The reason for the choice of these two modules was to ensure that the programming competency and achievement level (be it conventional OO programming or Web coding) was considered for all of the Computing Science and Information Systems students from the first year of their degree.
8. Module evaluation questionnaire results. These comprised free text comments from the module questionnaires for the Software Engineering module at Newcastle (two sets per year) for the academic years 2004/05, 2005/06 and 2006/07.

A review of the module design was also conducted. This included an evaluation of the teaching approaches in terms of the application of learning theories, its adherence to professional and academic standards and the design of assessment, assignment marking criteria and learning outcomes. The review of the assessment marking criteria and learning outcomes from all the assignments was performed using Anderson and Krathwohl's revision of Bloom's taxonomy (Bloom, 1954, Anderson and Krathwohl, 2001).

4.3 Preparation of Data for Statistical Analysis

The data sets involved in the quantitative statistical analysis were student record data and student reflective report data (Individual Reflective Report Assessment).

4.3.1 Student Marks Data

The final summative module marks for all completing students for the academic years 2003/04-2006/07 were collated. In total there were 325 students in this data set. The only students from the four cohorts not included in the data were those that did not complete the module. The reason for not including the students that did not complete was that in each case there was an incomplete set of marks and most non-completing students had dropped out of the module before the end, so they did not complete the required reports and evaluations that could be used – in particular there was no reflective report and no second set of peer percentages for or from these students. The marks data were anonymised and all records for students who did not finish the module removed. Duplicate records were also removed (students repeating the year due to non-completion the previous year). Failing or repeating student reports were retained. The data fields as follows: an identification number (a cut down version of the student ID number, year, team number, team average, a module mark and individual component marks for all assignments. Peer assessment marks for Semesters 1 and 2 and overall weighting were also included. Student record data were updated to include the programming marks from the aforementioned programming modules from the first year of study for all those students who completed the SETP. These marks were transferred to one of the spreadsheet columns in SPSS.

4.3.2 Individual Student Reflective Report Data

322 Individual Reflective Reports were mined for the following information – the nature and description of each student's role in the project and the areas where they had taken part or contributed in terms of requirements analysis, design, implementation, documentation, testing, organisation and leadership. Teams were identified as to whether they had worked with Durham or not. Each report was also mined to determine student learning outcomes in terms of the skills they stated they had learned, developed or improved during the

project, for example, organisation skills, leadership skills or technical skills such as design, coding and testing.

These results were then codified in the SPSS spreadsheet for the purpose of analysis – with 1 for Yes and 0 for No for each of the roles and tasks undertaken and for the skills learned, developed and improved. If a student did not mention a role or skill that was undertaken or learned in their report then that role or skill was given a value of 0. A copy of this spreadsheet can be found in Appendix A.

4.3.3 Reliability Analysis of the Reflective Report Coding Mechanism

To determine the reliability of the coding of student individual reports in terms of their role and activity declaration a reliability analysis using Cohen's Kappa (κ) was performed. Cohen's Kappa (κ) measures inter-rater agreement for categorical or nominal scales when there are two raters involved. In this case the binary coding method used in the original coding of the individual reports for the SETP module was repeated by another member of academic staff from Newcastle University School of Computing Science. The analysis was performed using a sample of the individual reports from two teams (13 students in total).

Cohen's Kappa has five assumptions that must be met (Laerd, 2014) and these are detailed in the following section along with an explanation of how well these are met in the current study, making it suitable for performing Cohen's Kappa.

1. The response is measured on an ordinal or nominal scale and that the categories are mutually exclusive.

In the original coding of student responses, participation in a role or activity was categorised as a 1 for Yes and a 0 for No and these values indicated that a student had carried out an activity/role (1) or they had not (0), so these categories are mutually exclusive. These values were then translated to did (1) and didn't (0) to make them suitable for the reliability analysis. So, the study design meets assumption 1 for running Cohen's Kappa.

2. The response data are paired observations of the same phenomenon, i.e. that both raters assess the same observations.

In this case, both raters reviewed the same student reports and used the same rating method. So, the study design meets assumption 2 needed to run Cohen's Kappa.

3. Each response variable must have the same number of categories and the cross-tabulation must be symmetric.

In this case response data from student reports for the activities and roles of Research, Design, Implement, Test, Organise, Lead and Document were translated into two columns, one column for each rater, and the 1's and 0's were translated into Did and Didn't respectively. This meant that the responses of both raters were measured on a dichotomous scale. With this translation, the study design meets the third assumption needed to run Cohen's Kappa.

4. The two raters are independent.

In order to remove the potential for bias, the second coder in this study performed their analysis in another room, using anonymised reports and an anonymised spreadsheet to fill in their values. The student records were also arranged in a different order from the original spreadsheet and the second coder had never viewed the spreadsheet or data that resulted from the original coding. This design ensures that the study meets the fourth assumption needed to run Cohen's Kappa.

5. The two raters are fixed – specifically selected to take place in the study.

In this case the second rater was specifically selected to take part in the study as they had taught on the SETP module before and understood what the reports were detailing and the general format of the SETP module. With this selection, the study design meets the fifth and final assumption for performing Cohen's Kappa.

Result: SPSS generates two tables of output for Cohen's Kappa: Cross-tabulation and Symmetric Measures.

The Cross-tabulation table (Table 3) helps us to understand the degree to which the two raters (the researcher and an academic) agreed or disagreed on judging whether a student participated in an activity or took on a certain role in the SETP module. The Symmetric Measures table presents Cohen's Kappa which

is a statistic that takes into account *chance agreement*. This means the likelihood that if both raters in this study were to guess randomly about each student's behaviour, they would end up agreeing by chance. We don't want this chance to make agreement appear better than it actually is. Therefore Cohen's Kappa measures the proportion of agreement *over and above* the agreement expected by chance (i.e. chance agreement).

Table 4 shows that 91 pairs of participation values were compared between the raters i.e. each rater indicated a 1 or a 0 for participation/non-participation in the 7 role and activity categories (Research, Design, Implement, Test, Organise, Lead and Document) for the same set of 13 students. For the 91 pairs of values compared, both raters agreed that students did participate in the same set of activities and roles in 49 cases and that in 24 cases students did not participate in the same set of activities and roles. There were only 18 cases of the 91 examined (i.e. 7 + 11) on which the two raters could not agree. The percentage of agreement between raters was 80% (49 + 24/91). This is quite a high percentage and illustrates that the method used to code student responses from the reports was quite reliable.

In Table 4 Cohen's Kappa (κ) is .573. This is the proportion of agreement over and above chance agreement. Cohen's Kappa (κ) can range from -1 to 1. Altman shows (1999) that a kappa (κ) of .573 represents a *moderate* strength of agreement, in line with the guidelines outlined by Landis and Koch (1977). Furthermore in table 4 as p <= .001, the kappa (κ) coefficient in this study means that p < .0005 and is therefore statistically significantly different from 0. This result is reassuring as it means the agreement between the two raters on the codes assigned to student responses, was based on more than a random guess and that the coding method used was valid.

		John		Total
		Did	Didn't	
Marie	Did	49	11	60
	Didn't	7	24	31
Total		56	35	91

Table 3 Cross-tabulation

		Value	Asymp. Std. Error[a]	Approx. T[b]	Approx. Sig.
Measure of Agreement	Kappa	.573	.089	5.491	.000
N of Valid Cases		91			

Table 4: Symmetric Measures

The coding method for the data obtained for individual reports was found to be reliable and valid, so now both the student marks data and the report data were prepared for preliminary statistical tests.

4.3.4 Preliminary Statistical Tests

The most commonly used statistical procedures are known as parametric tests (Vassar, 2014) and are based on an assumption of a normal distribution of data. Parametrical tests make assumptions about the population from where the data has been drawn. Normal distribution is the term used to describe a symmetrical bell-shaped curve on a histogram of values, which has the greatest frequency of values in the middle, with smaller frequencies towards the extremes. If the data are not normally distributed then it will be necessary to use non-parametric statistical tests to perform the analysis. Parametric statistics are more powerful and make assumptions about the data that are more stringent. To determine whether the data gathered from student assessment records and student reports were normally distributed the data needed to be tested for Skewness and Kurtosis.

- **Skewness** is the test of the symmetry of distribution i.e. whether there is bunching of the data at either end of the scale depicted by the distribution curve e.g. in marks. Negative skewness values indicate a clustering of scores at the high end or right hand side of a distribution graph (i.e. the higher end of the scale) and positive skewness indicates a positively skewed distribution i.e. where the scores are bunched together at the lower end of the scale (Pallant, 2010).
- **Kurtosis** is a method of describing the distribution of data around the average – it is concerned with the 'peak' of the distribution curve, rather than the extremes. Kurtosis values below 0 indicate a distribution that it relatively flat – i.e. too many cases in the extremes (Pallant, 2010).

According to Tabachnick and Fidell in large data samples (i.e. those with 200+ cases), "skewness will not make a great difference in the analysis and Kurtosis can result in an underestimate of the variance, but this risk is again reduced with a large sample" (Tabachnick and Fidell, 2007, p80).

The Shapiro-Wilk test can be used to find out whether a sample came from a normally distributed population. This is normally used for relatively small sample sizes. This test compares the scores in the sample to a normally distributed set of scores with the same mean and standard deviation. If the test statistic result is:

1. Non-significant i.e. $p > 0.05$ (p is the probability of obtaining a result close to the one observed) this means that the distribution of the sample is not significantly different from a normal distribution and you can run parametric tests such as t-tests and ANOVA (Analysis of Variance) and Pearson Correlation.
2. Significant i.e. $p < 0.05$ then the distribution is significantly different from a normal distribution and you should run non-parametric tests such as Kruskal-Wallis and Mann Whitney and Chi–square. These tests are deemed to be less sensitive than parametric tests in determining differences between groups (Pallant, 2010).

To overcome distribution abnormalities or skew and kurtosis on the samples a researcher needs to transform the data in some way to make it more normalised and easier to analyse statistically using SPSS. The methods that can be used are:

1. Log transformation – takes the log of a set of numbers and squashes the right tail of the distribution to reduce positive skew.
2. Square root transformation – takes the square root of large values and brings them close to the centre, again reducing positive skew.

There is some argument about whether a researcher should perform these transformations and by doing so, 'manipulate' the statistical data to fit. In the case of the student data for this study it was decided to use these tests for normality even though the data set was quite large.

4.4 Common Statistical Assumptions used in this Research

Before outlining other statistical tests that were used it is important to detail a set of assumptions that statisticians make about data that are common to all the techniques used in this research and these are as follows.

- **Related pairs** – each subject must provide a score on both variables X and Y – (related pairs) and both pieces of information must be from the same subject (Pallant, 2010, pp. 125). In this case, the subject, means the person whose response data is being statistically tested.

  Independence of observations – observations that make up the data must be independent of one another. Each observation or measurement must not be influenced by any other observation or measurement. According to Stevens "This is particularly problematic when studying the performance of students working in pairs or small groups. The behaviour of the group influences all group members" which is a factor in this particular research study in terms of the peer assessment data. In this case, Stevens recommends using a more stringent alpha value ($p < .01$) (Stevens, J, (1996) in Pallant, 2010, pp.125, 126).
- **Linearity** – The relationship between two variables should be linear.
- **Homoscedasticity** – This stipulates that the variability for all scores X should be similar for all values of variable Y. In this case Pallant recommends that the researcher review the scatter plots that are produced and these should show "a fairly even cigar shape" along their length (Pallant, 2010).

4.5 Quantitative Statistical Methods used in this Research

4.5.1 Correlation

Correlation determines if there is a link between two data items or data sets. This statistical test "describes the relationship between two continuous variables in terms of both the strength of the relationship, and the direction" (Pallant, 2010, pp.129).

There are two types of correlation that can be evaluated in SPSS. The first is a simple *bivariate* correlation that assesses the relationship between two

variables. The second method will allow you to assess the correlation between two variables whilst 'controlling' for another variable.

The statistic that is generated in a bivariate correlation is known as *Pearson's product-moment correlation* (*r*) (Pallant, 2010, pp.123). The statistical significance of *r* is also provided. The test determines if there is a positive or negative correlation between two variables (indicated by + or – before the value). A positive correlation means that as one variable increases, the other does too. A correlation of 0 means there is no relationship between the two variables and a negative relationship means as the value of one variable increases, the other decreases. This statistical test was deemed relevant to the evaluation of the student results as it could show if there was a relationship between role (for example, programmer) or Course (IS and CS) and a student's final mark in the SETP.

An important element of determining correlation is to determine if there was a sampling error that might have affected the results of a correlation test. The tests for this are known as the test of Correlation Coefficients.

4.5.2 Correlation Coefficients

These statistical techniques test the probability that the difference in the correlations of two groups would occur as a function of a sampling error, when in fact there was no real difference in the strength of the relationship between the two e.g. between the Module Mark and Programming Score for CS and IS students. This step cannot be done by SPSS so requires a calculator. The value obtained is assessed using a set of decision rules to determine "the likelihood that the difference in the correlation noted between the two groups could not have been due to chance" (Pallant, 2010).

Assumptions made with this technique are that the *r* values for both groups were obtained from random samples and that the two groups of cases are independent (not the same participants tested twice). SPSS ensures this by allowing a user to split cases. This technique also assumes that the distribution of scores is normal and the data set has more than 20 cases in each group. For the purposes of the calculation using a calculator the researcher must convert each of the *r* values into a *z* value (see table of *z* values in Appendix B).

4.5.3 Multiple Regression Technique

Multiple Regression Technique can help us learn how well a set of variables is able to predict a particular outcome and which variable in a set of variables is the best predictor of an outcome. The variables tested are usually continuous variables but they do not have to be. Dichotomous variables (e.g. yes or no) can also be used as predictors (Psychstat, 2013). This test was deemed relevant in terms of finding out whether programming competency (programming score) is a predictor of good marks in the SETP.

4.5.4 Hierarchical Multiple Regression (Sequential Regression)

In this test, the independent variables are entered into the equation in an order specified by the researcher. They are entered in steps or blocks and each independent variable is assessed in terms of what it adds to the prediction after the previous variables entered have been controlled for. When all the variables have been entered, the overall model is assessed in terms of its ability to predict the dependent measure e.g. to predict a good mark for the module. The contribution to this of each block of variables is also measured (Pallant, 2010, pp.149).

This technique was deemed relevant, for example, to evaluate if a student's role choice was a positive predictor of Module Mark. It is common for students to take on more than one role during the course of the project and this statistical method could be used to determine if the more technical roles in the team project resulted in higher marks. It could also be used to determine if the student's course (IS or CS) was a predictor of a good Module Mark**.**

4.5.5 Logistical Regression

There are many situations where the dependent variable is categorical. Unfortunately the Multiple Regression Technique is not suitable for when you have categorical dependent variables. Logical regression allows you to test models to predict categorical outcomes. In this research, for example, Mark Range is a categorical variable, which has several associated categories i.e. 1 (First Class), 2 (Second Class, First division), 3 (Second Class, Second Division), 4 (Third Class) and 5 (Fail). For this test, the 'predictor' variables can either be categorical or continuous or a mix of both in one model. The default procedure in SPSS is known as the *Forced Entry* method. In this test, all

predictor variables are tested in one block to assess their predictive ability while controlling for the effects of other predictors in the model. Other techniques in Logistical Regression allow you to specify the potential predictors from which SPSS can pick a subset that provides the best predictive power. These stepwise procedures have been criticised in both logistic and multiple regression because they can be heavily influenced by random variations in the data (Pallant, 2010, pp. 168) but were deemed relevant in this case because of the need to determine what variables in the data can be used to predict a good grade.

There is an assumption about the data associated with the Logistic Regression technique and this is Multi-collinearity. In this test an ideal situation is if the predictor variables are strongly related to the dependent variable but not strongly related to each other. There is no test for multi-collinearity in SPSS. To determine if multi-collinearity is occurring between the variables selected, Pallant recommends that the researcher focus on the coefficients table that is produced in SPSS and the columns labelled *collinearity statistics*. *Tolerance* values that are very low (less than .1) indicate that the variable has high correlations with other variables in the model. The researcher "may need to reconsider the variables to be included in the model if this is the case and remove one of the highly inter-correlated variables" (Pallant, 2010, pp. 169).

4.5.6 Two-Way ANOVA

A two-way ANOVA is a 'between-groups' analysis of variance. This means that there are two independent variables and 'between groups' means that there are different people in each group. This allows us to look at the individual and joint effects of the two independent variables and one dependent variable. For example, in the student data gathered for this research, the dependent variable might be peer assessment score, peer assessment weighting or module score and the independent variables might be 'Course' and 'Implement' (i.e. whether a student contributed to the implementation by programming). This test allows us to test for the main effect for each independent variable and to test whether there is any interaction effect between them. "An interaction effect occurs when the effect of one independent on the dependent variable depends on the level of the second independent variable" (Pallant, 2010, pp. 265).

4.6 Qualitative Methods used in this Research

4.6.1 Focus Groups

Three Focus Groups were conducted, taking place in the academic years 2006/07, 2007/08 and 2008/09 respectively. It was decided that these Focus Groups would give students from both Durham and Newcastle an opportunity to discuss the Software Engineering Team Project in more detail than they could via module questionnaires. The Focus Groups were facilitated by guest employers who had acted as customers for the project, and a PhD student who had interest in the cross-site work but no active involvement in the assessment of students. It was felt that a facilitator who was neutral in terms of assessment would make students feel more at ease when speaking about their project experiences. Students from each team at Durham and Newcastle were invited to an informal one-hour session and their responses were recorded on paper anonymously. The questions asked were designed to be high level and not to lead the students in any way in terms of focusing on assessment only. The topics covered areas such as how they felt about the overall project experience, and working with employers, about difficulties faced and what they felt they had learned. Detail of the questions asked can be found in Appendix C.

4.6.2 Module Evaluation Questionnaire Responses

Module questionnaires are conducted in the School of Computing Science at Newcastle at least once per iteration of a module. A 10-credit module normally runs for one semester (12 weeks). Module questionnaires are usually conducted towards the end of a module (normally during weeks 10 and 11 of the 12 week period). As the SETP module is a 20-credit module, it runs over two semesters. This means that the module questionnaires are conducted twice each time the module is run. This can be very useful for the teacher in terms of being able to recognise problems from the feedback early in the first semester and remedy them as much as possible for the second semester. Module questionnaires at Newcastle are conducted anonymously. Students filled in the questionnaires online in the NESS system (NB: as of 2013/14 the system in use is EvaSys). All the questionnaire feedback responses for the SETP module during the years 2003/04, 2004/05, 2005/06 and 2006/07 were reviewed during this study.

4.7 Merits of Statistical Tests Used

The quantitative statistical methods used in this study each have their own merits and the selection of those statistical methods was based on their suitability for the nature and type of quantitative and qualitative data collected. Specifically, the merits of the main statistical tests carried out (other than preliminary statistical tests made on the data and the codifying method used for the qualitative reports) are as follows.

- **Correlation and Correlation Coefficients –** The correlation method allows us to make predictions and therefore if two variables are related, we can predict one using the other. Correlation does not measure cause and effect, merely relationships. The advantage of Pearson Correlation Coefficient is that it allows us to calculate the correlation of data even if the data is formatted in an interval/ratio scale.
- **Multiple Regression Technique –** This technique predicts the unknown value of a variable from the known value of two or more variables (also called predictors). The technique is used when one is interested in predicting a continuous dependent variable from a number of independent variables. The method is very flexible as the independent variables can be numeric or categorical and we can discover the collective effects of the independent variables and explore the interplay among each factor on predicted outcomes. We can also measure the amount of variation in the dependent variable that can be attributed to the variables in the model and how much of the variation is unexplained. This technique also measures how well prediction of behaviour matches with actual observation of behaviour.
- **Logistical Regression and Hierarchical Regression –** The Logistical Regression technique allows us to determine the order of importance of variables and to select useful subsets of variables to examine in a stepwise manner. Hierarchical Regression is good for correlated variables and is used to analyse the effects of a predictor variable after controlling for other variables. Hierarchical regression is usually better at predictability than Logistical Regression as the order of variable entry is determined by the researcher before the analysis is carried out. Both techniques can suffer from error sampling problems but the

likelihood of this is reduced in the Hierarchical technique because there is more interaction between the researcher and the data (Lewis, 2007).

4.8 Limitations of the Data and of this Study

Pedagogic research is often criticised for using non-rigorous methods to gather and analyse both quantitative and qualitative data. Student feedback and the results of experiments during discrete instances of learning are often viewed as being subjective and of being too specific to one cohort and instance of time, making it hard to replicate results. Therefore the validity and reliability of quantitative and qualitative data in this type of research can be difficult to verify. However, this type of data is still invaluable to a teacher and serves the purpose of informing practice and providing guidance and that is the overall intention of this current research. Using the Individual Reflective Report that has been a common assignment throughout all the years of the SETP module, and attempting to quantify and evaluate responses to this in a structured way using statistical analysis, was viewed as one way to make the evaluation more 'scientific' and reliable. With the quantitative data also, different markers assessed students throughout all the years included in this study and their influence and judgements are difficult to measure. The SETP itself changed over time, the module evolved each year, the problem scenario given to students changed, the cohorts changed the module leaders changed, all during the course of the study. However, the assignment outline, assessment rubrics and the high level learning outcomes remained the same for the module during all the years of the study, so these are viewed as common indicators of quality that the students' performance was measured against.

4.9 Summary

In this chapter I have outlined the methods that were used to evaluate the effectiveness and fairness of the prevailing assessment framework in the SETP module from 2003/04 to 2006/07. This outline includes details of the data collection process and the nature and limitations of the data used. The statistical techniques discussed should allow me to objectively evaluate the student learning outcomes from the module and determine if there is a bias in terms of grade achievement towards students who programmed and those who did not, or towards students with higher programming competency in the module during

this time period. The results of the qualitative evaluation should provide a global picture of the module's effectiveness in terms of teaching, learning and assessing Software Engineering competency and in terms of the overall learning objectives and learning theories used in the module. Finally the limitations of the data and data collection methods used in this study are acknowledged and steps to ensure validity and reliability of the results are outlined. The combined results, regardless of their limitations, should allow me to meet the research objectives of this study as outlined in Chapter 1 of this thesis. These results are presented and discussed in the next chapter.

Chapter 5: Results

5.1 Introduction

This chapter outlines the results from the statistical analysis and qualitative techniques used in this study that were outlined in Chapter 4: Methods. The basic characteristics of the data are presented and then the statistical results are presented along with discussion of the main findings relevant to the research study i.e. the impact of programming competency on success in the SETP and its implications for the module assessment strategy. The qualitative results of three Focus Groups and questionnaire results that outline student experiences of the module are then presented, including student views on assessment. The module is then reviewed in terms of the assessment design and learning outcomes using Anderson and Krathwohl's revision of Bloom's taxonomy (Bloom, 1954, Anderson and Krathwohl, 2001) and in comparison to academic and industrial standards of assessing Software Engineering competency as outlined in Chapter 2 (the discussion on learning theories and how the assessment climate affects the module design will be discussed later in Chapter 7). The implications of the results for the assessment of the SETP module at Newcastle University are then discussed at the end of the chapter.

5.2 Quantitative Data Collection and Preparation

322 sample academic records from the student population taking the Software Engineering Team Project Module at Level 5 (2nd Year) were taken from the years 2003/04-2006/07 (Tables 5 and 6).

		Frequency	Percent	Valid Percent	Cumulative Percent
Valid	CS	185	57.5	57.5	57.5
	IS	132	41.0	41.0	98.4
	Other	5	1.6	1.6	100.0
	Total	322	100.0	100.0	

Table 5: Description of the Sample Population by Course

From each of the three Courses (Programmes) offered in the School of Computing Science at Newcastle for Undergraduates, there were 185 Computing Science students, 132 students from the Information Systems course and 5 students from 'Other' courses (Table 5). The term 'Other' courses means students who are taking the Team Project module as part of another programme from another School within the Faculty (Faculty of Science, Agriculture and Engineering (SAGE)), for example, Computer Systems Engineering (School of Electrical and Electronic Engineering). All students were in their second year of degree studies and all students were studying for an Honours degree. There were differences in entrance qualifications between the groups for their first year of study as Information Systems courses require A-level results of one grade lower than Computing Science students i.e. Computing Science students require AAB- ABB, whereas Information Systems students require A-level grades between ABB-ABC (typical offer).

		Value	Count	Percent
Standard Attributes	Label	<none>		
Valid Values	2003-04		98	30.4%
	2004-05		79	24.5%
	2005-06		79	24.5%
	2006-07		66	20.5%

Table 6: Distribution of the Population by Year of Study

Of the population selected for the study, 145 participants had taken the SETP module with the CETL ALiC implementation of working with Durham University and 177 had not. The number of students taking the programmes at Newcastle had fallen from 98 in the academic year 2003/04 to 66 in 2006/07 (Table 6). The module marks for the test sample population were examined

(Tables 7 and 8). Most students tend to do well on the module traditionally, and the lower percentile for the years examined bears this statement out, with the 25th percentile achieving an average mark of 59%, just below a 2.1 performance. Of the 322 cases selected for analysis, the 75th percentile had an average mark of 71%, a First-class performance. A further breakdown of numbers per classification is shown in Table 6. Across all the years of the study, 93 students received a First-class mark for the module (marks in the range from 70-100%), 132 achieved a 2:1 grade (marks between 60-69%), and 76 obtained a 2:2 grade (shown in the table as 3, marks between 50-59%). Table 4 (Mark Range) also shows that 17 students received a Third class mark (marks between 40 and 49%) and 4 students from the sample population were recorded as a Fail (a mark below 40%) during the years of the study.

		Value
Standard Attributes	Label	<none>
N	Valid	322
	Missing	0
Central Tendency and Dispersion	Mean	64.34
	Standard Deviation	9.436
	Percentile 25	59.00
	Percentile 50	65.00
	Percentile 75	71.00

Table 7: Mark Percentiles for SETP module

		Value	Count	Percent
Standard Attributes	Label	<none>		
Valid Values	1	First Class	93	28.9%
	2	2:1	132	41.0%
	3	2:2	76	23.6%
	4	Third	17	5.3%
	5	Fail	4	1.2%

Table 8: Mark Range

One of the main research questions for this current study is to find out if students who do well in the First Year programming modules will do well in the SETP module in Second Year i.e. is programming competency a predictor for success? So, the next information that was needed was the programming scores for each student. Since the academic year 2006/07 at Newcastle, students on the Information Systems programme have taken a different set of programming modules to Computing Science students during the first year of their studies. For the first three years of this study 2003/04-2005/06, Information Systems students took the *same* programming modules as Computing Science students i.e. students began programming in Java. For the final academic year of the study, 2006/07, the Information Systems and Computing Science programmes were changed, with both cohorts taking a common module in JavaScript programming for Semester 1 of the first year; Computing Science students then took a module in programming with Java and Information Systems students took a module in Web Programming in Semester 2.

		Value
Standard Attributes	Label	<none>
N	Valid	322
	Missing	0
Central Tendency and Dispersion	Mean	66.52
	Standard Deviation	18.323
	Percentile 25	54.00
	Percentile 50	67.00
	Percentile 75	82.00

Table 9: Percentiles for Programming Score

It was decided to take the average programming score for each student from their first module in programming, irrespective of year, as this score reflects their achievement in programming in the first year of their programme and the assessment examines the same basic programming concepts for all students, irrespective of Course. As can be seen in Table 9, most students did well in the programming modules with the 25th percentile scoring an average of a 2:2 mark in programming, the 50th percentile scoring an average of 67% (2:1) and the 75th percentile scoring a high First class mark. Another related area to look at in terms of marks and programming was whether those who took on the main role of programming in the Team Project are given a higher weight in the Peer Percentage exercise because they have contributed largely to the programming effort during the SETP.

		Value
Standard Attributes	Label	<none>
N	Valid	322
	Missing	0
Central Tendency and Dispersion	Mean	.1517
	Standard Deviation	.03147
	Percentile 25	.1400
	Percentile 50	.1500
	Percentile 75	.1700

Table 10: Peer Assessment Weight

It was found that all weightings were available for students across all years. The weighting is the score given to an individual student by their Team Monitor, based on an average of the two Peer Percentage exercises undertaken by each team, one during each Semester of the SETP module (as outlined in

Chapter 3 Case Study). However, a breakdown of the two Peer Percentage Scores for the academic year 2003/04 was not recorded in the source system (NESS) and was therefore unavailable for detailed study so statistical studies involving the raw individual Peer Percentage scores have a smaller sample population. As can be seen in Table 10, the average peer weighting of team members was .14 for students in the 25th percentile, .15 for students in the 50th percentile and .17 for students in the 75th percentile of all cohorts across all years of the study.

5.3 Preliminary Statistical Testing

The sample population data for both Programming module scores and SETP scores were examined to determine normality. Normality describes a distribution of data that resembles a bell-shaped curve on a Histogram, with the greatest frequency of scores located in the middle of the curve and smaller scores at either of the two extremes of the curve (Pallant, 2010, pp.59). The Kolmogorov-Smirnov statistic shown in Table 11 assesses the normality of the distribution. A non-significant result (significance value of more than 0.05) indicates normality. In Table 9, it can be seen that the SETP scores (Mark) are significant for two of the three courses being examined (CS – Computing Science at .046 and IS –Information Systems with a significance of .016) and that the Programming Score (ProgScore on the table) is significant at .003. This would usually indicate that the data violates the assumption of normality. However, violation of the assumption of normality is common in larger samples (200+) (Pallant, 2010, pp.63) and therefore it was felt that rather than remove outliers and artificially manipulate the data to become more 'normal', it was best to leave outliers and raw scores in the analysis, as this is the true picture of the scores achieved by students. The higher and lower scores at the extremes of the curve or the bunching of scores in the middle, whilst being classed as outliers or skewed in statistical terms, are of interest to this study. The Shapiro-Wilk test for skewness in the distribution (as outlined in Chapter 4) is not significant for Computing Science students nor for students in the category 'Other', for Mark (module mark for the SETP module) but for IS students, it is, as there is a skew (significance .001) which suggests that the distribution is not symmetrical. For the variable Programming Score (ProgScore) there is some clustering at the higher end of the distribution for CS

students, which is significant (Significance .000). However, according to Pallant (Pallant, 2010, pp 57) “if the distribution is perfectly normal you would obtain skewness and kurtosis value of 0” which is rather an “uncommon occurrence”.

These small violations of the assumption of normality are common in larger samples, and again, of interest to the researcher, so the outliers were left in the study and the data was not ‘artificially’ normalised for the purpose of further statistical
analysis.

		Kolmogorov-Smirnov			Shapiro-Wilk		
	Course	Statistic	df	Sig.	Statistic	df	Sig.
Mark	CS	.066	185	.046	.987	185	.099
	IS	.087	132	.016	.959	132	.001
	Other	.215	5	.200	.962	5	.818
ProgScore	CS	.083	185	.003	.955	185	.000
	IS	.069	132	.200	.984	132	.113
	Other	.284	5	.200	.887	5	.341

Table 11: Tests of Normality

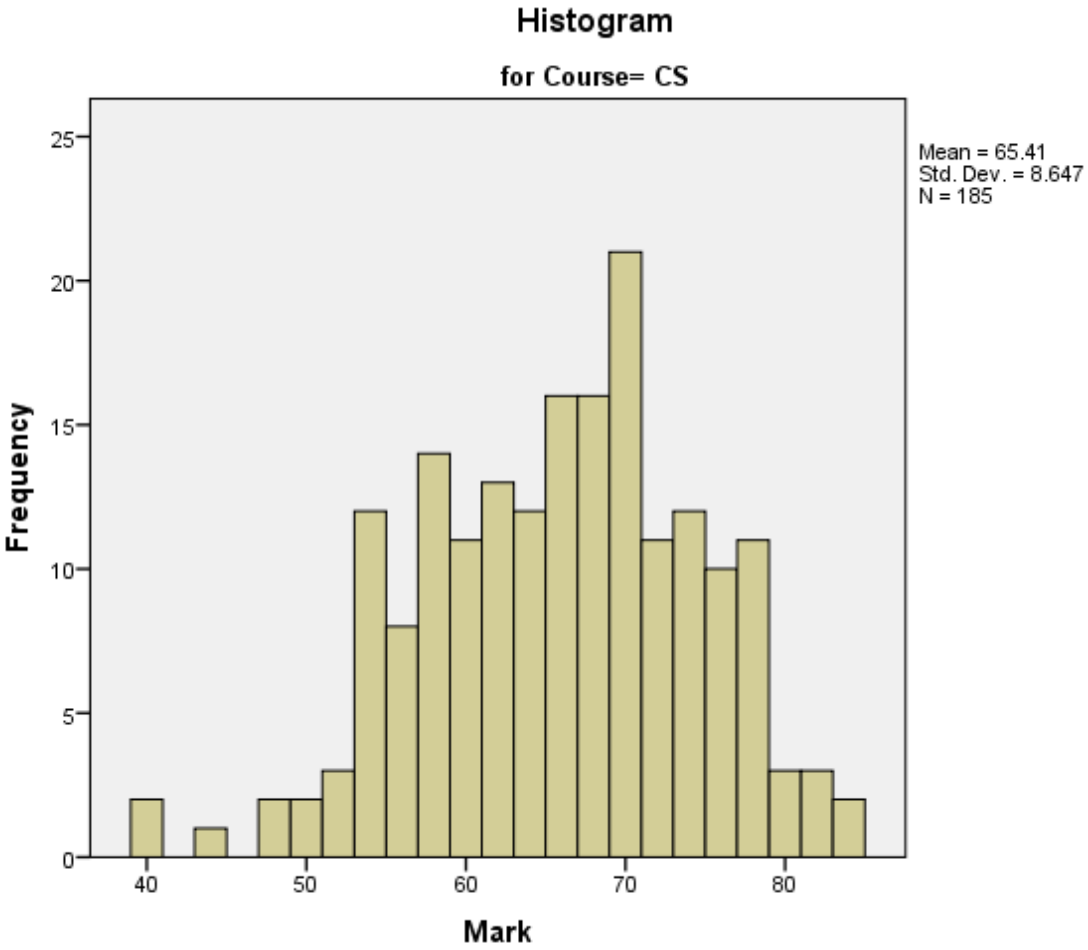

Figure 10: Distribution of Module Mark for CS Students

Figure 10 shows the distribution of marks for the Computing Science students on the SETP module. There are several scores that could be termed as outliers on the left-hand side of the graph but all of these are above the module pass mark of 40%. At the other end of the scale, on the right-hand side of the graph, there is a large cluster of scores above the start of the First class threshold of 70%. The average mark for Computing Science students for the module, during the years 2003/04-2006/07 is 65.41%, a 2:1 mark.

Figure 11 shows the distribution of marks for the Information Systems student sample taking the SETP during the same years of interest, 2003/04-2006/07. This graph again illustrates a number of low scores on the left-hand side, but the range is greater, beginning at a Failing mark of just under 30% and ranging to a set of First Class marks just below 90%. Again, for IS students, the average mark for the module is in the 2:1 area at 62.91, slightly lower than for the CS marks.

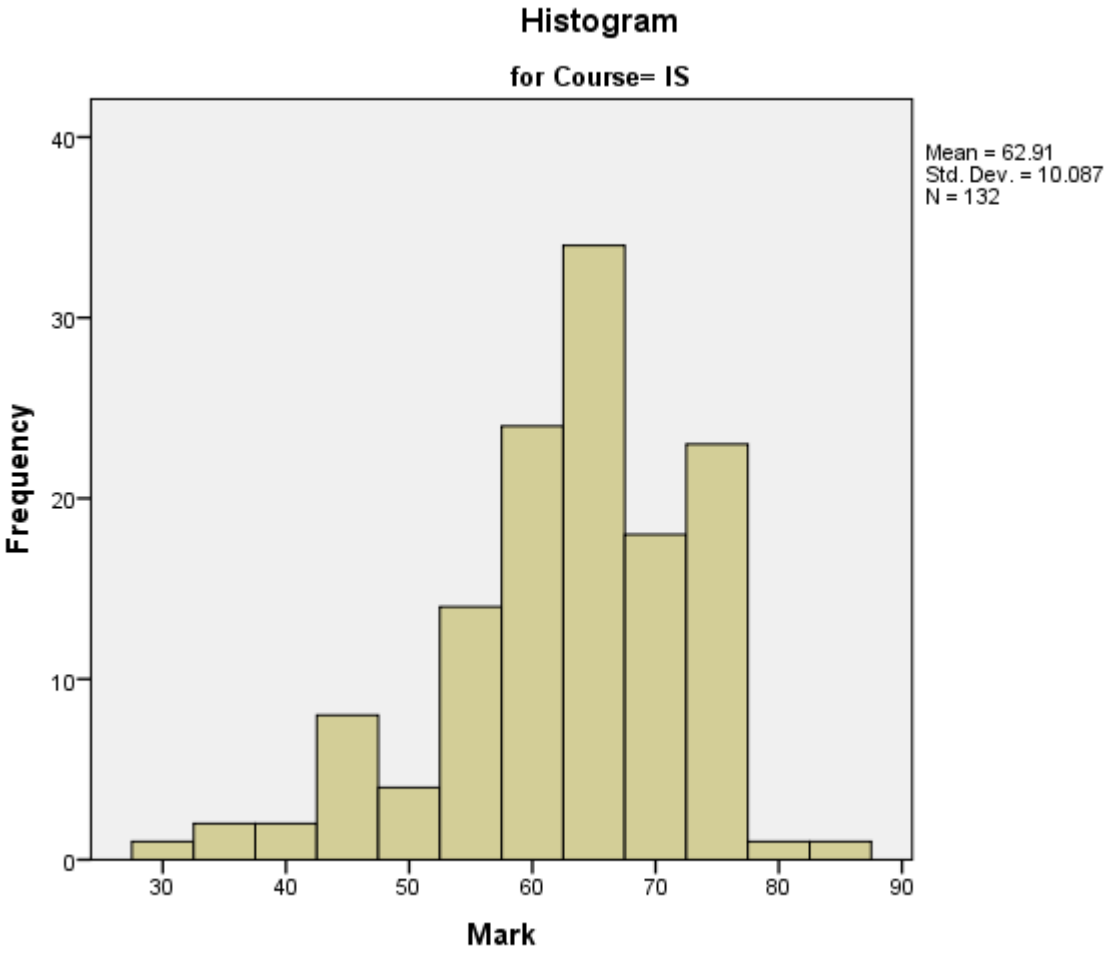

Figure 11: Distribution of Module Marks for IS students

There are some outliers but again it was felt important to keep them in the sample to reflect all scores and to test more accurately if marks were over inflated for programmers during the module. The next result to look at in terms of preliminary statistical tests and to determine the normality of the distribution was the Q-Q plot. In a Q-Q Plot "A reasonably straight line suggests a normal distribution" (Pallant, 2010, pp.63). In this plot, the observed value for each score is plotted against the expected value from the normal distribution. Both of the plots for CS and IS have reasonably straight lines in the plots (Figures 12 and 13) and show a distribution that was viewed normal 'enough' for the use of the stronger parametric statistical analysis techniques available is SPSS (version 19.0). The plot for the "Other" courses was not reviewed as there was not a big enough sample.

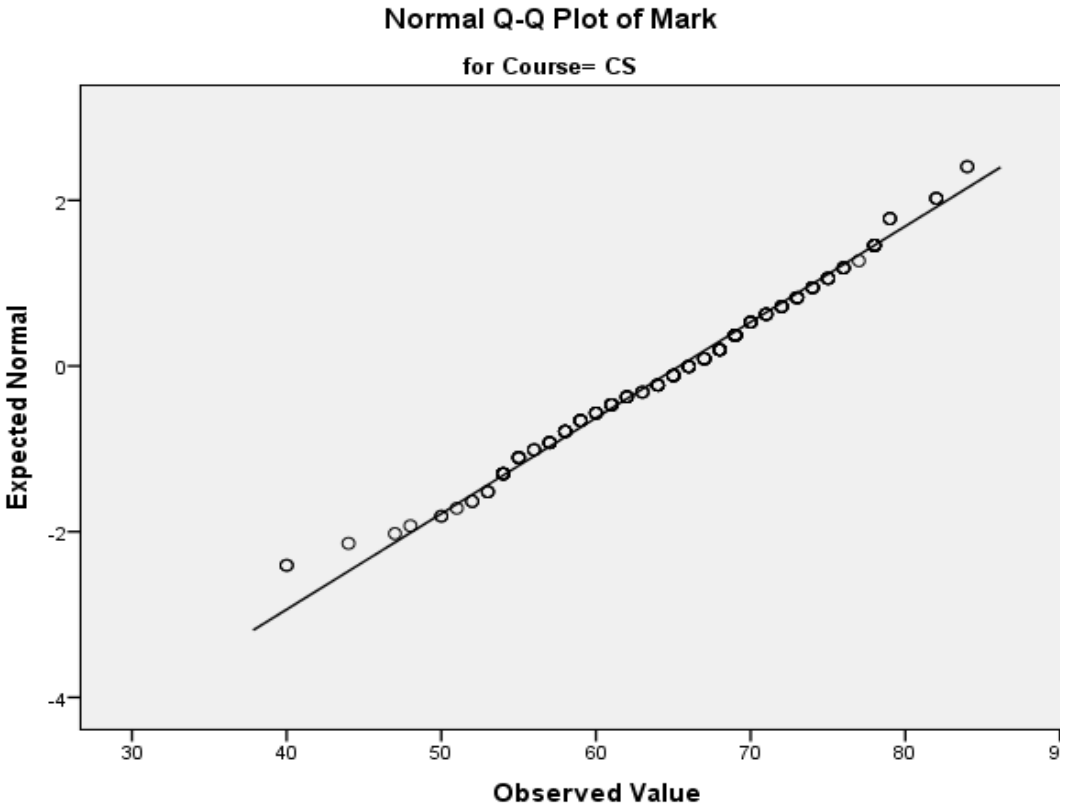

Figure 12: Plots for CS Marks

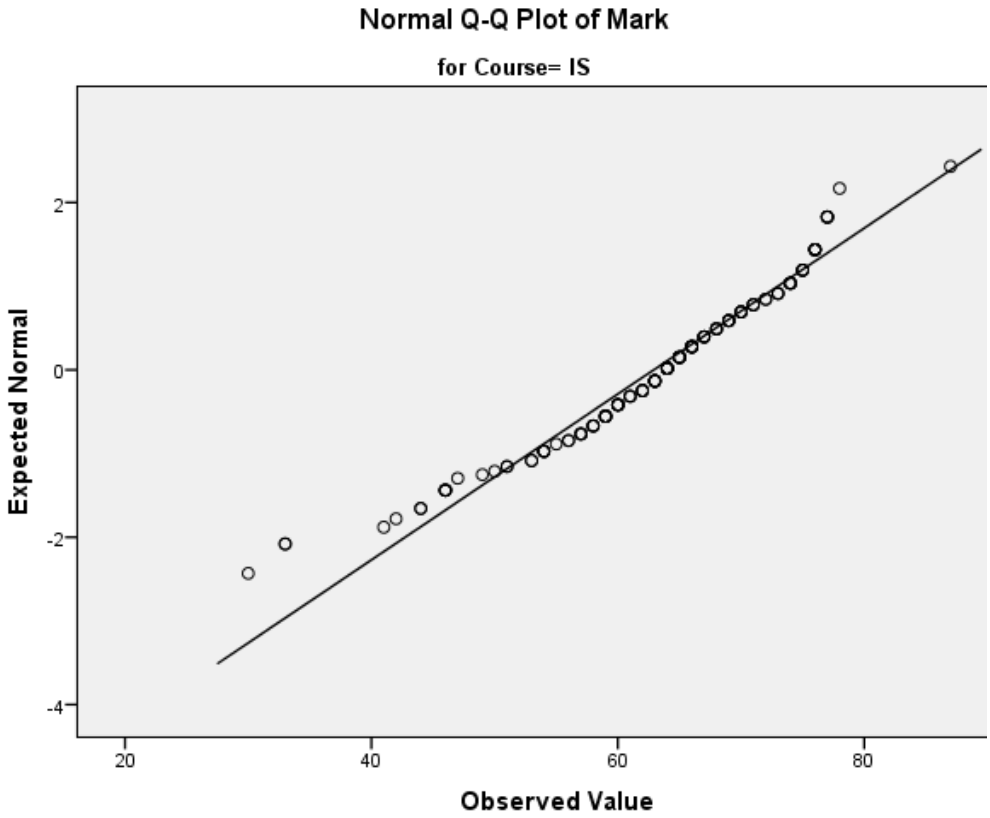

Figure 13: Plots for IS Marks

5.4 Statistical Analyses

5.4.1. Correlation

The Pearson correlation coefficient (*r*) is designed for interval variables (continuous variables) such as the variable Mark (the overall module mark of students on the SETP module). It can also be used if you have one dichotomous variable such as Yes/No e.g., in the student Individual Report data where students identified what tasks and roles they undertook, such as 'Implement'. The Spearman Rank order correlation is designed for ordinal level or ranked data e.g. Mark Range, where the range of achievement of a student is indicated by 1st Class, 2:1, 2:2, 3rd Class, Fail.

		Mark	Implement
Mark	Pearson Correlation	1	.122*
	Sig. (2-tailed)		.029
	N	322	322
Implement	Pearson Correlation	.122*	1
	Sig. (2-tailed)	.029	
	N	322	322

Figure 14: Correlations

Correlation measures the direction of the relationship between two variables indicated by a correlation coefficient value. If there is a negative sign in front of the result then there is a negative correlation between the two variables. In this case, the correlation between Mark and Implement was measured and as Figure 14 shows there is a correlation of .122 between the mark a student obtained on the module (Mark) and whether a student implemented or not (Implement). A correlation of 0 indicates no relationship, a correlation of 1 indicates a perfect positive correlation and -1.0 indicates a perfect negative correlation. In this case there is a small positive correlation between the two variables Mark and Implement. The nature of the effect size of the correlation was measured using the following criteria as outlined by Cohen (1988): Small r = .10 to .29, Medium r = .30 to .49, Large r = .50 to 1.0

Result: The relationship between Implement and Module Mark (Mark) was investigated using the Pearson product-moment correlation coefficient. Preliminary analyses were performed to ensure no violation of the assumptions

of normality, linearity and homoscedasticity. There was a small, positive correlation between the two variables, r = .12, n = 322, with a higher Module Mark being associated with Implementation.

5.4.2 Correlation Coefficients

The statistical difference between correlation coefficients for other groups of variables from the student data were then tested to see if they were significantly different i.e. is the difference big enough to be considered significant? The correlation coefficients between groups (Computing Science and Information Systems students and students on "Other" courses, and their respective Module Marks (Mark)) were tested. The significance test assesses the probability that the difference in the correlations of the two groups would occur as a function of a sampling error, when in fact there was no real difference in the strength of the relationship between two continuous variables. SPSS cannot do this step I so needed to perform it using a calculator (Pallant, 2010, pp.139).

Course				Mark	ProgScore
1	Spearman's rho	Mark	Correlation Coefficient	1.000	.268**
			Sig. (2-tailed)	.	.000
			N	185	185
		ProgScore	Correlation Coefficient	.268**	1.000
			Sig. (2-tailed)	.000	.
			N	185	185
2	Spearman's rho	Mark	Correlation Coefficient	1.000	.249**
			Sig. (2-tailed)	.	.004
			N	132	132
		ProgScore	Correlation Coefficient	.249**	1.000
			Sig. (2-tailed)	.004	.
			N	132	132
3	Spearman's rho	Mark	Correlation Coefficient	1.000	.205
			Sig. (2-tailed)	.	.741
			N	5	5
		ProgScore	Correlation Coefficient	.205	1.000
			Sig. (2-tailed)	.741	.
			N	5	5

Figure 15: Comparison of Correlation Coefficients for each Course

In Figure 15 the strength of the correlations for each group are indicated and each course is identified as follows: 1 = CS, 2 = IS, 3 = Other.

Result: Using this test we can see that there is a positive correlation between Mark and Programming Score (ProgScore) for all groups, although the correlation is greater for CS than for the other two groups, but only just. The correlation between programing marks and module mark for CS students was .27, while for IS it is slightly lower at .25.

To test if the difference between the two sets of students is significant I needed to test the statistical difference between these two correlation coefficients**.** This significance test assesses the probability that the difference in the correlations of the two groups would occur as a function of a sampling error, when in fact there was no real difference in the strength of the relationship between the two (Mark and ProgScore) for CS and IS.

Assumptions needed for this test are that the *r* values for both groups were obtained from random samples and that the two groups of cases are independent (i.e. not the same participants tested twice). The distribution of scores is normal and there are more than 20 cases in each group. In this case, the assumptions are not violated. The *r* values of each group were then converted into *z* values i.e. the two values for *r* obtained in the previous test were to be converted into standard score form (referred to as *z* scores) and these are as follows:

CS *r1* =.268 N1 = 185 IS *r2* =.249 N2 = 132

Next I found the *z* value that corresponds with each of the *r* values then constructed the following equation:

CS z1 = .277 IS z2 = .255

$$Zobs = \frac{z1 - z2}{\sqrt{\frac{1}{N1-3} + \frac{1}{N2-3}}}$$

(Source, Pallant 2010)

If $-1.96 < Z_{obs} < 1.96$ the correlation coefficients are not statistically significantly different.

Result: The result of this equation is that the Z_{obs} value is 0.1748 which means that Module Mark (Mark) and Programming Score (ProgScore) are not statistically significant in terms of Course, which is a 'good' result for the module leader and their approach to marking. This result illustrates that the Module Mark is not adversely affected by the students' programme of study and nor is the Programming Score.

5.4.3 Further examination of Programming Score and Module Mark by Year

There was a need to investigate further into Programming Scores and Module marks, by academic year, to see if the CETL ALiC intervention had made any impact on the correlation between the two sets of scores or had maintained some form of marking consistency, despite the changes introduced into the SETP Project module. The results of the correlation test can be viewed in Tables 12 and 13.

Result: The results of the raw correlation value for each year between the two variables (to 2 decimal places) for Mark and Programming Score was as follows:

.20 in 2003/04

.47 in 2004/05

.22 in 2005/06

.31 in 2006/07

Table 12: Correlation results for Programming Score and Year

These values showed that there was a significant correlation between Module Mark and Programming Score across all years but with a more stringent lower alpha value of .01, the correlation that was most significant is highlighted one year before the CETL ALiC intervention in 2004/05 (correlation of .479, Significance .000). It was felt, that while these statistical correlations were interesting, it was difficult to determine if the results reflected a natural difference in cohort ability from year to year or if these results were because of the assessment methods used or some other factor not taken into consideration. It was not easy to discern from comparing correlations across years and the result was difficult to generalise. In an attempt to clarify, a comparison between the Marks and Programming Scores in terms of correlation coefficients for the academic years 2003/04 and 2004/05 was performed and gave a Z_{obs} value of -2.019, which is statistically significantly different. What this shows is that there

was more of a variance in Module Mark and Programming Score in 2004/05 than there was in 2003/04, even though the population was smaller. The test does not show a 'cause' for this and does not illustrate if a high Programming Score could be said to have led to a High Module Mark (Mark) conclusively across all years of the study.

Year			Mark	ProgScore
2003-04	Mark	Pearson Correlation	1	.207*
		Sig. (2-tailed)		.041
		N	98	98
	ProgScore	Pearson Correlation	.207*	1
		Sig. (2-tailed)	.041	
		N	98	98
2004-05	Mark	Pearson Correlation	1	.479**
		Sig. (2-tailed)		.000
		N	79	79
	ProgScore	Pearson Correlation	.479**	1
		Sig. (2-tailed)	.000	
		N	79	79
2005-06	Mark	Pearson Correlation	1	.225*
		Sig. (2-tailed)		.046
		N	79	79
	ProgScore	Pearson Correlation	.225*	1
		Sig. (2-tailed)	.046	
		N	79	79
2006-07	Mark	Pearson Correlation	1	.315**
		Sig. (2-tailed)		.010
		N	66	66
	ProgScore	Pearson Correlation	.315**	1
		Sig. (2-tailed)	.010	
		N	66	66

Table 13: Correlations by Year

5.5 Multiple Regression Technique

An exploration of other variables that might impact on Module Mark was needed. A Multiple Regression technique was used with Mark as the

dependent variable and using independent variables Team Average and Programming Score, for example, how much of the variance in Module Mark can be explained by Team Average and Programming Score? There is a lot of output generated from SPSS for this test. Tables 14 and 15 describe the variables used in the test - Team Average Mark (Team Av), Average Module Mark (Mark) and Average Programming Score (ProgScore):

Descriptive Statistics

	Mean	Std. Deviation	N
Mark	64.34	9.436	322
TeamAv	63.98	5.683	322
ProgScore	66.52	18.323	322

Model	R	R Square	Adjusted R Square	Std. Error of the Estimate
1	.604[a]	.365	.361	7.541

Table 14: Model Summary Tables

		Mark	TeamAv	ProgScore
Pearson Correlation	Mark	1.000	.545	.278
	TeamAv	.545	1.000	.029
	ProgScore	.278	.029	1.000
Sig. (1-tailed)	Mark	.	.000	.000
	TeamAv	.000	.	.302
	ProgScore	.000	.302	.
N	Mark	322	322	322
	TeamAv	322	322	322
	ProgScore	322	322	322

Table 15: SPSS output from Multiple Regression

Model		Sum of Squares	df	Mean Square	F	Sig.
1	Regression	10440.677	2	5220.339	91.803	.000[a]
	Residual	18139.745	319	56.864		
	Total	28580.422	321			

Table 16: ANOVA

Result: The Multiple Regression test showed that there is a high correlation between Mark and Team Average (.545) as expected, but a fairly low correlation between Programming Score and Mark (.278), and an even lower correlation between Programming Score and Team Average (.029). The lowest correlation between Programming Score and Team Average, is heartening, in the sense that this statistic would support the idea that there is more to the team's overall average score than being a good programmer or at least, having a good score in programming the year before the team project.

In the Model Summary Table (Table 14) the value R Square was checked to assess how much of the variance in the dependent variable Mark is explained by the model, which includes the variables Team Average and Programming Score. In this case the value is .365 or 36.5 of the variance. The Adjusted R Square, in the same table corrects this value (for small samples) "to provide a better estimate of the true population value" (Pallant, 2010, pp.161). To assess the statistical significance of the result the next table to look at was the table labelled ANOVA (Table 16). The model in this example reaches statistical significance (Sig .000; this really means $p < .005$). The researcher needs to check which of the independent variables in the model most contributed to the prediction of the dependent variable Mark. This information is found in Coefficients Table 1 (can be found in Appendix B). To do this the column Standardized Coefficients and the Beta value of each of the independent variables were checked. The variable with the largest Beta value is Team Average (.537), with Programming Score at .262. The significance of both values was also checked. If the Significance value is less than .05, then the variable makes a significant contribution to predicting the independent variable Mark. In both cases, Team Average and Programming Score gave a

Significance value of .000, which means both variables make a significant contribution to predicting an individual student's Mark in the SETP module.

Result: Multiple Regression was also performed on the variables Course and Programming Score to see the impact of these variables on a student's Mark during the module. The coefficients table for this test can be found in Appendix B. In this case the variable with the largest Beta value was Programming Score (.135), so this makes the strongest unique contribution for explaining the dependent variable Mark between the two, when the variance explained by all other variables in the model is controlled for. The value for Course was very low (-1.219) so it did not make a strong contribution.

5.6 Hierarchical Multiple Regression

Hierarchical Multiple Regression was performed using other variables from the students' Individual Reflective Reports. These variables were indicators of the roles that students had performed during the project and included Test, Research, Implement, Design, Organise, Lead, Document). The analysis was designed to test if a student's role was a significant predictor of their overall achievement on the module, in terms of the Mark they received at the end.

Model	R	R Square	Adjusted R Square	Std. Error of the Estimate	Change Statistics				
					R Square Change	F Change	df1	df2	Sig. F Change
1	.151[a]	.023	.010	9.387	.023	1.843	4	317	.120
2	.261[b]	.068	.047	9.211	.045	5.071	3	314	.002

a. Predictors: (Constant), Test , Research, Implement, Design

b. Predictors: (Constant), Test , Research, Implement, Design, Organise, Lead, Document

c. Dependent Variable: Mark

Table 17: Impact of Other Variables on Mark: Model Summary

Table 17 illustrates Model 1 and Model 2. Model 1 refers to the first block of variables that were entered (Test, Research, Implement and Design) and Model 2 includes all the variables that were entered in both blocks (Test, Research, Implement, Design, Organise, Lead, and Document).

In this test we check the R Square values in the table. After the values in Block 1 have been entered, the overall model represents only 2.3% of the variance in Mark. After the variables in Block 2 have been entered, the model *as a whole* explains 4.7%.

The Second R Square Value includes all the variables from both blocks. To find out how much the overall variance in our dependent variable Mark is explained by Organise, Lead, Document roles we need to look at how the column labelled R Square changes. On the line marked Model 2, the R Square Change value is .045 – this means that Organise, Lead and Document adds 4.5% of the variance in Mark. This is statistically significant as indicated by Sig. F. Change value for this line .002 (< .05). Next we look at the Coefficients Table 2 (Appendix B) in Model 2 Row for this test. This summarises the results with all variables entered into the equation. Looking at the Sig. column, we can see that there is one variable that makes a unique statistically significant contribution <.05 – Lead (with a value of .006) i.e. whether a student led their team or not.

Result: This result means that leading a team is statistically significant and the best predictor of variance in marks from all the values tested. Whether a student implemented (i.e. programmed) is less of a good predictor of module mark with a value of .125.

5.6.1 The Effect of Adding Other Role Variables

It was decided that Hierarchical Multiple regression should be retried, again with the addition of the variables Course and Programming Score, to find if these were better predictors of a variance in Mark than the roles taken by students during the project. The coefficients table for this test can be found in Appendix B.

Again, the R Square values in the Model Summary table were checked.

After the first set of variables has been entered, the model explains 8.2% of the variance in Mark. In Block 2 the variables Research and Design have been added and the model as whole explains 9.2% of the variance. After the Block 3 variables have been added the model explains 12.7% of the variance, which is not a high score. Then the R Square Change column was reviewed and it was noted that Course and Programming Score are the strongest predictors of Mark with Lead, Organise and Implement being the strongest predictors after that.

Statistically significant contribution was made by both Course and Programming Score and this is indicated in the Sig. F Change column where the significance is .000 and signifies that Model 1 is statistically significant.

Result: Hierarchical Multiple Regression was used to assess the ability of Programming Score and Course to predict Mark on the SETP module and explained 8.2% of the variance in Mark. At Step 2, with added variables Research, Design and Implement, the total variance was 9.2%. With the addition of Organise, Implement and Lead in the final model, only Programming Score was found to be the most statistically significant predictor of variance in Mark for the module.

5.7 Logistic Regression

Logistic regression was used to find out what factors could be used from the student data to predict the likelihood that a student will take part in programming during the SETP.

This test needed one categorical (dichotomous) dependent variable (implement) (Table 16: Implement: Yes/No, Coded 1/0) and two or more continuous or categorical predictor variables (independent) and I chose Course (CS/IS coded 1=CS and 2 = IS, 3 = Other) and Programming Score (ProgScore). SPSS produced the following tables when this test was run (Tables 18, 19 and 20):

Original Value	Internal Value
No	0
Yes	1

Table 18: Dependent Variable Encoding

			Parameter coding	
		Frequency	(1)	(2)
Course	CS	185	.000	.000
	IS	132	1.000	.000
	Other	5	.000	1.000

Table 19: Categorical Variable Coding

Omnibus Tests of Model Coefficients

		Chi-square	Df	Sig.
Step 1	Step	97.976	3	.000
	Block	97.976	3	.000
	Model	97.976	3	.000

Table 20: SPSS output from Logistic Regression Test

The Omnibus Tests of Model Coefficients (Table 20) gives an overall indication of how well the model performs. This is referred to as a 'goodness of fit 'test. For this set of results, we want a highly significant value to indicate a good fit (the .Sig value should be less than .05). In this case the value is .000 therefore the model with our set of variables as predictors is a good one. The chi-square value was 97.976 with 3 degrees of freedom, meaning that the model is a good fit and that Course is a good predictor of whether a student implements or not.

Step	Chi-square	Df	Sig.
1	13.571	8	.094

Table 21: Hosmer and Lemeshow Test

The results for the Hosmer and Lemeshow Test (Table 21) also support our model as being worthwhile. This test, which SPSS states is the most reliable test of model fit available in SPSS, is interpreted differently to the Omnibus Test previously outlined. For the Hosmer-Lemeshow Goodness of Fit Test, poor fit is indicated by a significance value of less than .05, so to support our model we need a value greater than that. The chi-square value for this test is 13.571 with a significance level of .094. This value is larger than .05 and therefore indicates support for the model. The table Model Summary, Table 22, gives another piece of information about the usefulness of the model:

Step	-2 Log likelihood	Cox & Snell R Square	Nagelkerke R Square
1	314.210[a]	.262	.363

Table 22: Model Summary

			Predicted		
			Implement		Percentage
	Observed		No	Yes	Correct
Step 0	Implement	No	0	109	.0
		Yes	0	213	100.0
	Overall Percentage				66.1

Table 23: Model Table

The Cox & Snell R Square and the Nagelkerke R Square values in Table 22 provide an indication of the amount of variation in the dependent variable explained by the model (from a minimum value of 0 to a maximum value of approximately 1). These are described as *pseudo R square statistics*, rather than the true R square values seen in multiple regression output. In this case, the two values are .262 and .363 suggesting that between 26.2% and 36.3% of the variability is explained by this set of variables. The next table to consider was the Classification table (Table 24). This provides an indication of how well the model is able to predict the correct category for each case. In Block 0 without our variables the model correctly predicted 77% of cases which is quite good.

Classification Table

			Predicted		
			Implement		Percentage
	Observed		No	Yes	Correct
Step 1	Implement	No	84	25	77.1
		Yes	49	164	77.0
	Overall Percentage				77.0

Table 24: Block 0 Output

		B	S.E.	Wald	df	Sig.	Exp(B)	95% C.I.for EXP(B)	
								Lower	Upper
Step 1[a]	Course			68.634	2	.000			
	Course(1)	-2.406	.290	68.632	1	.000	.090	.051	.159
	Course(2)	-1.483	.946	2.460	1	.117	.227	.036	1.448
	ProgScore	.017	.008	4.494	1	.034	1.017	1.001	1.033
	Constant	.806	.568	2.012	1	.156	2.240		

Table 25: Variables in the equation

Table 25 gives information about the contribution of importance of each of the predictor variables. The test used here is known as the Wald Test (Pallant, 2010). We scan down the column labelled Sig. looking for values less than .05. These are the values that contribute significantly to the predictive ability of the model.

The *positive predictive value* is the percentage of cases that the model classifies as having the characteristic that is observed in this group. To calculate this we need to divide the number of cases in the predicted = yes cell (164) by the total number in the predicted =yes cells (25+164) and multiply by 100 to give a percentage = 86.7%. Therefore the positive predictive value is 86.7%, indicating that of the people predicted to have implemented, the model accurately picked 86.7% of them.

The negative predictive value is the percentage of cases predicted by the model not to have the characteristic that are actually observed not to have the characteristic. So that is 63%.

Result: This test showed that there are two major factors that could be used to predict whether a student Implements or not during the SETP and they are (1) whether you are a Course 1 student (CS student) and (2) the programming score from the 1st Year of studies (ProgScore). Its predictive value however is not 100%.

5.8 Investigating the impact of Role Choice on Module Mark and Mark Range

Next it was decided to further investigate Roles taken by students during the project to find out if there were particular roles taken by students from IS and

CS courses, and what impact did each kind of role have on their final Module Mark. There were assumptions that could be made about the answer but the use the self-reported data from the Individual Reflective Report assignment would help to clarify these. In the SETP module, students are encouraged to take on a number of roles in their team throughout the year. This is so that they can get as much experience as possible of all the stages/areas in the software engineering process and they can reflect on which areas they prefer to work in and where their skills are best used for the good of the team. A summary of the roles students reported in taking on during the project during all four years is outlined in Table 26:

Role	CS Yes	CS No	IS Yes	IS No	Other Yes	Other No
Research	159	26	116	16	4	1
Design	168	17	114	18	5	0
Implement	162	23	48	84	3	2
Test	138	47	90	42	4	1
Organise	73	112	55	77	3	2
Lead	75	110	48	84	2	3
Document	130	55	124	8	3	2

Table 26: Roles Taken by Students for Each Course during the Project

Table 26 illustrates that the majority of students (279/322, 86.6%) conducted Research and 78.8% took a role in Documenting during the project. The area of the project where most students took part was Design with 287/322 or 89.13% of the sample population contributing to this area. Testing was also a role that a large proportion of students participated in, with 228 of the 322 students in the sample population (70.8%), stating in their reflective reports that they performed testing. The roles that fewer students took on during the project were Leadership (125/322 or 38.81%), Implementation (110/322, 34.16%) and Organisation (39.75%) and there were fewer IS than CS students who took on a programming role during the project, 48 across all the four cohorts (48/132 IS students, 36.3%).

Result: Interestingly, fewer students took on the roles that are best at predicting the variance in student marks for the module, i.e. higher marks (as described earlier in section 5.6).

5.8.1 The Effect of Role Choice and Mark Range (Classification of Mark awarded)

The next thing to check was whether a student's role choice had any impact on their final classification mark at the end of the module. Ideally the answer would be that, irrespective of the role taken on by a student during the module, the chances of getting a good mark would be the same for all students. If the assessment regime for the module is fair, then role should not matter, only performance in that role. Students are encouraged in the module to view the whole Software Engineering process as important and not just the technical aspects. There is no denying that programming and the ability to program are essential to any team seeking to create software for a customer, but in Software Engineering the development methods used, the requirements analysis, the design, the organisation, leadership, planning and testing and how the team works together are all crucial to ensure that a high quality product are delivered to the customer on time and within budget. Given this, the whole process and all the roles in the team should be considered as equally important in the assessment process. Overall as can be seen in Table 27 there were 93 students from the sample population (28.88%) who received a First Class mark (70+) in the Software Engineering Team Project module (during the academic years 2003/04 – 2006/07). Of these 93 students, 86% had conducted Research for their team, 93.5% had taken part in the Design process, 72% had contributed to Implementation, and 71% had contributed to the Testing effort in their team.

The percentages of these 93 students who achieved a First Class mark who indicated 'Yes' for other roles were 55.91% for Organise, 49.46% for Lead and 74.19% for the Document role. So, from these figures it can be seen that the areas of the software engineering process these students contributed to the most were Research and Design. What is reassuring is that the majority of students indicated they took part in all parts of the project.

Role	Mark Range					Total
	First Class	2:1	2:2	Third Class	Fail	
Research						
No	13	20	9	1	0	43
Yes	80	112	67	16	4	279
Total	93	132	76	17	4	322
Design						
No	6	16	12	1	0	35
Yes	87	116	64	16	4	287
Total	93	132	76	17	4	322
Implement						
No	26	42	31	9	1	109
Yes	67	90	45	8	3	213
Total	93	132	76	17	4	322
Test						
No	27	38	20	4	1	90
Yes	66	94	56	13	3	232
Total	93	132	76	17	4	322
Organise						
No	41	75	39	11	3	169
Yes	52	57	37	6	1	153
Total	93	132	76	17	4	322
Lead						
No	47	82	50	16	2	197
Yes	46	50	26	1	2	125
Total	93	132	76	17	4	322
Document						
No	24	30	9	2	0	65
Yes	69	102	67	15	4	257
Total	93	132	76	17	4	322

Table 27: Module Mark Classifications by Role for the sample population

NB: The researcher bore in mind that the indication of roles and contribution from each student was 'self-reported' in the Individual Reflective Report assignment. This assignment was part of the coursework assessment for the SETP module and students received marks for this work. As an historical and common source of information from students for all years of the study, the reports proved a very useful and amenable source for analysis. However, other sources of information about roles and contribution (Contribution Matrices and Team Structure documents which were collaboratively constructed) could have been used to verify individual student claims but anonymity would have had to have been removed for the purpose and it was felt that this would prove an onerous process with no guarantee of gaining useful additional data at the end. There is a possibility, of course, that some students exaggerated claims about

their contribution and role during the project, but the Individual Reflective Reports were deemed the best source available for analysis and comparison (from a historical perspective, certainly).

Result: The review of the role and classifications data showed that 93/322 students received a first class mark during the project. The reassuring aspect of this was that the majority of these students indicated that they had taken part in most aspects of the Software Engineering Process. The results from this analysis also indicated that those who had received a 2.1 mark (132 students) had mainly participated in all the stages of the software engineering process too. However, it can be seen in table 27, the majority of students who received a 2.1 mark took part in areas such as Implementation and Testing (90 and 94 respectively) but fewer took part in roles that were found to be predictors of a good mark in Section 5.6 i.e. Leadership and Organise (50 and 57 respectively). Students who received a 2.2 mark in the module (76 students), played a large part in the Research (requirements gathering and analysis) and Design aspects of the project (67/76 and 64/76 respectively) but less of a role in areas such as Implement (45/76), Testing (56/76), Organise (37/76) and Leadership 26/76 respectively). In many ways the relationship of good marks in the module to the breadth of roles played and greater participation in all aspects of the Software Engineering Process during the module is very reassuring. This tells me that the more effort a student puts in to all aspects of the project (not just the programming), the more likely they are to receive a good module mark, and that is how it should be. The worrying aspect of these results are that fewer IS students took on roles that would give them higher marks. When assignments are marked for the module, a student's role is not an issue for the module leaders as it is the quality of the deliverables produced by the team and the individual that are assessed. However, there are two sets of marks missing from all the data analysed so far that may throw some light as to why a student's role is so important for gaining good marks**.** These two sets of marks are (1) the Individual and Team Effectiveness Marks and (2) the Peer Assessment marks that make up a student's weighting within their team (as outlined in Chapter 3 Case Study). The Individual Effectiveness and Team Effectiveness marks are awarded by Team Monitors and Peer Assessment Marks are awarded by a student's teammates. It is important that these sets of marks are considered in terms of their influence on an individual's overall success on the module.

5.9 Two Way ANOVA: The influence of Peer Assessment on Module Mark

It was considered important to check if there are any differences between the weightings awarded to students, based on their course, or whether they implemented or not during the module. For this test a Two Way ANOVA test was selected. This test requires two categorical independent variables and one continuous variable. In this case variables Course and Implement were used for the independent categorical variables and for the continuous variable the value Weight was selected. This would show if any difference in Weight was due to Course or whether a student had Implemented or not. Table 29 shows the average weight given to students from each course and whether they implemented or not. Interestingly, an IS student who implemented, on average, got a lower weighting than those who did not.

Result: This test shows, on average that those who implemented got a higher peer percentage weighting than their peers. Students from Other courses got the lowest marks for those who did not implement and IS students who implemented got a lower weighting than those IS students who did not.

F	df1	df2	Sig.
1.070	5	316	.377

Table 28: Levene's Test

Course	Implement	Mean	Std. Deviation	N
CS	No	.1500	.02132	23
	Yes	.1531	.03520	162
	Total	.1527	.03375	185
IS	No	.1510	.02905	84
	Yes	.1494	.02786	48
	Total	.1504	.02853	132
Other	No	.1450	.00707	2
	Yes	.1533	.02309	3
	Total	.1500	.01732	5
Total	No	.1507	.02725	109
	Yes	.1523	.03347	213
	Total	.1517	.03147	322

Table 29: Peer Assessment Averages

In Levene's test of equality of error variances (Table 28) the required Sig. level should be greater than .05 and therefore not significant. This suggests that the variance is equal across the groups. In this case the result was .377.

Source	Type III Sum of Squares	Df	Mean Square	F	Sig.
Corrected Model	.001[a]	5	.000	.155	.979
Intercept	.888	1	.888	884.979	.000
Course	.000	2	.000	.050	.951
Implement	.000	1	.000	.104	.747
Course * Implement	.000	2	.000	.172	.842
Error	.317	316	.001		
Total	7.730	322			
Corrected Total	.318	321			

a. R Squared = .002 (Adjusted R Squared = -.013)

Table 30: Tests of Between-Subjects Effects

The next thing to check is if one variable influences another and if the interaction is significant. In Table 30 it was necessary to look at the Course and Implement row to ascertain if the interaction between these two variables is significant. If it is, then it can be difficult to interpret the main effects. In this case the value is .842, so the interaction is not significant.

(I) Course	(J) Course	Mean Difference (I-J)	Std. Error	Sig.	95% Confidence Interval	
					Lower Bound	Upper Bound
CS	IS	.0023	.00361	.801	-.0062	.0108
	Other	.0027	.01436	.981	-.0311	.0365
IS	CS	-.0023	.00361	.801	-.0108	.0062
	Other	.0004	.01443	1.000	-.0336	.0344
Other	CS	-.0027	.01436	.981	-.0365	.0311
	IS	-.0004	.01443	1.000	-.0344	.0336

Table 31: Multiple Comparisons

The main effects are the simple effect of one independent variable. In the left-hand column of Table 31 Multiple Comparisons, the variable Course needs to be checked to see if there is a main effect for each independent variable by

checking the Significance column (Sig). If the value is less than or equal to .05, there is a significant main effect.

Result: In this case there is not a significant difference between weightings given to each student based on their course, but CS weightings are slightly higher - by .0023 on average (Tables 31 and 32).

Tukey HSD[a,b,c]

Course	N	Subset 1
Other	5	.1500
IS	132	.1504
CS	185	.1527
Sig.		.972

Table 32: Tukey's Honestly Significant Difference

5.9.1 The Impact of Individual Effectiveness and Team Effectiveness Marks

A check of the module data on NESS showed that the marks for Individual Effectiveness and Team Effectiveness were only recorded for the module from academic year 2006/07 onwards. Prior to this, the values were not explicitly added into NESS but were included in a student's mark. Given that the data to assess the effect of these marks on overall student performance was not complete for three years of the study, it was decided that these values could not be used in the statistical analysis. This was a real setback in terms of determining factors that influence student achievement on the module. However, the data is available for analysis for the years 2007/08-2013/2014 and therefore an analysis of the impact of these marks will be one area of future work.

5.9.2 Summary of Statistical Results

The findings from the statistical analysis can be summarised as follows:

1. There is a small correlation between Module Mark and whether a student implemented or not during the project (variable Implement). This means that a slightly higher module mark is associated with contributing to programming during the module.

2. There is a positive correlation between Programming Competency (ProgScore) from first year modules and Mark for the SETP. However neither the module Mark nor the Programming Score from first year are adversely affected by the students' programme of study (CS, IS, Other).

3. The correlation between Programming Competency and Module Mark was strongest the year before the CETL began.

4. A Multiple Regression technique showed that Team Average and Programming Score make a significant contribution to predicting a student's mark on the module but there is a low correlation between the two variables themselves. A second Multiple Regression test showed that Programming Score makes the strongest unique contribution to explain the module mark.

5. Logistic Regression showed that the factors most likely to predict who will implement are whether you have a good Programming Score and are a CS student or not. A student was more likely to implement if they were a CS student.

6. An analysis of student roles showed that students took on a variety of roles throughout the project and that good marks are related to the breadth of roles a student takes on.

7. Those who implemented got a slightly higher peer percentage weighting. CS students get a higher peer percentage weighting on average and the likelihood is increased because they were more likely to implement.

8. IS students who implemented got the lowest peer percentage weighting of all groups.

9. The CETL intervention and cross-site development work did not have a significant negative impact on student attainment in terms of final mark during the module, in comparison to marks achieved by students two years prior to the activities introduced.

5.9.3 Lessons Learned from Statistical Tests

The statistical tests used in the study allowed a wide variety of numerical data from differing sources to be explored which could be viewed as an 'offsetting' factor for the limitations of the data as outlined in Chapter 4. This means that a large number of factors about students and their achievement and skills were taken into account, including their previous programming experience before undertaking the SETP, the differences between their programmes of study and the variations in the initial level of programming skills between members of the same team. The statistical analyses also allowed factors such as role selection and the students' perception of the 'value' of each role in a Software Engineering team to be explored using peer percentages and a comparison of marks awarded to each role 'type'.

In terms of lessons learned from these analyses, the variety of tests used and their results indicate that the real value of statistical testing is its provision of some useful insights into the quantitative data but the main thing to remember is that these facts and figures do not tell the whole story on their own. None of the tests used determined 'cause' and 'effect' of the differences between students' achievement in the SETP module definitively and of course, the effectiveness of their interpretation is open to debate. With this in mind therefore the quantitative data analyses and results should be appreciated in the context from which they have been derived and the purpose or goal of their derivation. Whilst this view might limit their 'replicability' in a scientific sense, it ensures their applicability and usability for resolving the problem at hand, i.e. that of variability in assessment and achievement between programmers and non-programmers in the SETP. With this aim in mind some more qualitative data from the SETP module was collected via three Focus Groups which are detailed in the following sections.

Three Focus Groups using student representatives from teams during the academic years of the CETL ALiC Cross-Site work (the CETL was active during 2005-2010) were conducted. Two facilitators were guest employers who had acted as customers during the project and one facilitator was a PhD student from the School of Computing Science. None of the facilitators had an active involvement in the assessment of students. It was felt that a facilitator that was 'neutral' with regard to assessment would make students feel more at ease with

speaking about their project experiences. Students were made aware that their responses would be anonymous and that they would be recorded (on paper) and would be used for research and for the improvement of the module for future cohorts. For the purposes of this current study the focus was on responses from students that relate to assessment and feedback within the module. There were 12 participants in the first focus group (2006/07) from each site and Proctor and Gamble staff facilitated the session. Newcastle representation was equal in respect of course with 3 CS and 3 IS students. There were 20 participants in Focus Group 2 (2007/08) and these students were from both sites. Again there was a balance of students from the CS and IS courses, with 5 members of each course taking part (10 Newcastle students) and 10 Durham students. A staff member from IBM facilitated Focus Group 3 (2008/09) and 12 students from each site took part. A full list of focus group questions can be found in Appendix C.

5.9.4 Key themes emerging from Focus Groups

5.9.4.1 Poor Quality Feedback

From the three Focus Groups it became clear that feedback was not viewed as formative. The student participants made it clear that they tended to "just get the grade and move on" rather than base any of their decisions on feedback received for a deliverable or element of the SETP work. This illustrates that students tended to understand the feedback as summative and not something they could build on for improvement in future work on the module. This might indicate that the feedback was not of sufficient quality because student expectations and reactions to assessment were different from what the module leaders would have liked (Boud and Falchikov, 2007). Good feedback should feed forward and ensure that students know what is good about their work, what they need to do next and what they need to do to improve (Hume and Coll, 2009). This clearly was not happening.

5.9.4.2 Disagreements about Roles and Responsibilities

In Focus Group 1 concerns were expressed that simply forcing the role of project manager on IS or CS students at Newcastle in future iterations of the module would be unfair as everyone has their own strengths and weaknesses. Students stated that having to apply for the position of project manager would

"expose the person keenest to do it" and would probably be a better approach to selection for this role. In some senses the students' view on roles was not reassuring as it seemed they wanted staff to choose the best candidates for project manager rather than taking responsibility for their own learning choices and role selections (Littlewood, 1996; Holec, 1981). Focus group 2 also reported disagreements about roles. Some students reported that they had team members who dominated the conversation during meetings and were quite forceful in getting their opinions across. Some teams stuck to their roles for the first semester but team members had to switch roles during the second semester and help out in areas such as programming and testing when deadlines were approaching. Factors leading to these views might be that the SETP module did not make it clear to students about the value of each role in the Software Engineering Process.

5.9.4.3 Concern over levels of Staff Involvement

Wanting the teacher to take more of the responsibility for what happened in teams became a clear element of concern in all three of the Focus Groups as students also expressed concern about the variability in monitor support between teams and most wanted more help. In Focus group 2 non-contribution from teammates was viewed as a big issue among the participants. Students felt that there should be harsher penalties for students that did not do their fair share of the work allocated to the team. Teams were unsure what the penalties should be but some thought that students should be 'sacked' from a team for not doing any work or penalised via their marks. It was explained to students that peer assessment and weightings could be one method that the team could use to 'penalise' such behaviour but the students felt they did not want to affect the person's marks and that it was the role of module leaders to impose such penalties and not their responsibility. In Focus Group 3 some students felt that their teammates were prone to 'act up' in supervised meetings so that the monitor would think they had done a lot of work, whereas the students felt this was not always the case. In this group also students wanted module leaders to punish those students who did not pull their weight during the project and to monitor performance outside of formal meetings if they could. The module leader aimed to allow students to experiment on their own and to explore so that they could take an active role in their learning (Koshman, 1996; Crawford,

1996) but feedback from the Focus Groups illustrated that perhaps Piaget's idea that the role of module leader or teacher should be as a 'facilitator' was a bit too difficult for students to cope with as they had not encountered it to such an extent in any other module during their degree up until the SETP.

5.9.4.4 Distribution of Workload

Teams in Focus group 1 (2006/07) reported an 'unfair' balance in the importance of deliverables across sites and many voiced concerns over the apparent skew in workloads and accreditation between sites. These issues were largely resolved in later iterations of the module, with clearer information being provided about the assessment weight of deliverables at each site. Focus group 2 reported arguments had taken place about contributions to different elements of the coursework. They also disagreed about the quality of work produced by some of their teammates. Teams often had difficulty in allocating tasks to their team members, especially when some were deemed unreliable. Newcastle students undertook a skills assessment at the beginning of the project and felt this would have been beneficial for all teams involved i.e. their counterparts at Durham. On the whole, programmers felt they had been given an enormous amount of work to complete whereas others had *simpler* tasks (such as documentation) to complete. Some programmers felt that they had contributed the most to their team's product and therefore had the biggest contribution to the project. By the time Focus Group 3 took place (2008/09) students reassured by the use of Contribution Matrices but they still felt these did not accurately reflect the time and effort put into a piece of work or the difference in difficulty between one piece of work and another e.g. programming and documentation. All three Focus Groups showed that students thought programming was a 'high value' task in comparison to other deliverables or parts of the software engineering process. Work on documentation or organisation was viewed as being of lower value to the team effort or simpler in nature. The importance of instilling the notion that all effort contributed to team outcomes and that all roles were valuable to team success became more evident with subsequent iterations of the module. Students also clearly needed more guidance on allocation of work, managing their time and project planning. The module leaders had set out to emphasise the values of skills and roles in all parts of a Software Engineering Team but the message was not getting through. It was

apparent that the assessment weightings for code, documentation and organisational tasks needed to be reviewed so that skills and contributions of 'value' were reflected clearly in the workload and assessment practice (Knight, 2002), without producing a rigid set of competencies as this can undermine success, as outlined by Delamare Le Deist and Winterton (2005). It was also important to ensure that those in a less technical role did not become less motivated or feel less important in terms of their contribution (as in Dweck, 1988).

5.9.4.5 Concerns over Peer Assessment

Company peer assessment proved interesting across all three Focus Groups, but particularly Focus Group 3. The results showed big differences in the perception between teams in the same company. There were a few arguments about this and about peer assessment within teams. Students wanted anonymity when selecting percentages as they found the face to face meetings to discuss weightings very difficult. Group assessment makes students uncomfortable and is a challenging way to learn and challenging way to assess (Boud and Falchikov, 2007). The module leaders recognised that peer assessment was not being as effective as it could be because the purpose needed more explanation and students needed more practice, prior to completing it 'for real' (Freeman et al, 2006). It was also important to ensure again that students recognised the value of all roles in the SETP and the skills outlined in QAA (2007) were just as valuable. The challenge for teachers was to transfer these skills (aimed at a complete single honours programme in Software Engineering) could be distilled into one single module.

5.9.5 Results from Module Questionnaires

During the time period of this study (academic years 2003/04, 2004/05, 2005/06 and 2006/07) there were many positive comments about the SETP module. Students enjoyed the challenge of creating real applications for real customers. They enjoyed working in a team, making decisions and working through the full software lifecycle. Most of the positive comments were in connection with the improvement in their team, communication and technical skills. Many students reported on the benefit the module had been in interviews for placements and how they could see the benefit for their future careers.

However, despite our attempts to alleviate concern over assessment with the introduction of contribution matrices (as outlined in Chapter 3) the students were still worried. Peer assessment seems to have been an issue at both sites during the cross-site initiative that began in 2005/06. It did not show up in the formal module evaluations or in the Focus Groups to a great extent, but was reported by individual students quite often in their individual reports and in the module questionnaires. Some examples of the comments pertaining to assessment in these questionnaires can be seen in Figure 16.

> "The team monitor and module leader cannot really give justice by marks. Team meetings were usually all about people talking about pointless things or trying to show their best side for the monitor whilst in reality they did nothing."
>
> "The Peer Review Coursework is fully subjective and nobody can really check if the information written is correct or true and this could lead to confusion, marking problems, etc. We should be marked on the quality of our physical work (and fulfillment of our roles), rather than on something so subjective such as a peer review."
>
> "I am very concerned with how the marks are calculated as there doesn't seem to be enough of a frequent check in who's doing what, the team contribution matrix doesn't state how much each person does, just what so it's possible for other members of your team to be argue they should be in more slots even though they may have done 1% of the overall work. This module definitely needs more frequent checks on what people are doing and how they are behaving, adding percentages and providing a report on team members that doesn't have to be agreed by all members would be a useful way of commenting on teams."

Figure 16: Example of Module Questionnaire Feedback

Information Systems (IS) students also reported that they could not cope with some of the technical aspects of the module and felt that the project was more focused on the Computing Science (CS) programme (Figure 17).

"I think the way it is structured is unfair on information systems students. The work involves lots of programming which goes way beyond the teachings in information systems. Often during meetings and programming tasks it was like working in a foreign language like German and meetings sometimes were like this. At times you would be listening to a conversation about a programming problem and you really didn't know what was going on. This could be adjusted by having not an easier element to the project but one which was covered by teachings in information systems."

"Some groups had 6-7 Programmers whilst we has 3 able ones and one lazy useless one. Felt very unfair. Having 4 IS students in a group made it very difficult because only half the group could do the programming and high level design work - they often had too much to do while we couldn't even help. Didn't completely finish the app because of this."

Figure 17: Student Comment on Module Structure

5.9.6 Evaluation of Module Assessment using Anderson and Krathwohl's Revision

The assignments for the SETP were reviewed using Anderson and Krathwohl's revision of Bloom's Taxonomy as outlined in Chapter 2 (Anderson and Krathwohl, 2001). This process was by no means rigorous and it relied on my own teaching experience to determine which areas were covered by each assignment. For the purposes of the analysis I outlined all the learning outcomes and expectations for each assignment on the module and reviewed what was *expected* from students in terms of the types of knowledge they should either *use* or *learn* when completing the assignments and the cognitive processes involved in each task.

Result: The results from this analysis (Table 33) show that all Knowledge Dimension areas are covered by the assessment objectives. The module is quite rich in terms of the areas covered and in terms of the cognitive load on students even though it is only one part of the whole academic year for second year undergraduates and a small portion of their overall degree programme. However, the SETP is perhaps one of the most *authentic* modules in terms of simulating what a student will do in their career after they graduate.

Assignment	Objectives	Anderson and Krathwohl	
		Knowledge	Cognitive level
Requirement Analysis (Meeting)	Gather requirements from customer, review similar systems, evaluate possible technologies for solution	Factual	Understand Analyse Evaluate
Project Plan	To create a plan using deadlines and a defined software development lifecycle/methodology, allocate tasks and resources to the plan	Factual Conceptual Procedural	Remember Understand Apply Evaluate
Project Specification	To use an industry standard template to demonstrate technical and non-technical aspects of the software product. To practice writing requirements and using formal notations.	Factual Conceptual Procedural	Remember Understand Apply Evaluate Create
Personal Skills Analysis	To help students reflect on current skills and how they may interact in a software development team. To help them think, of their current strengths and weaknesses and how they can further develop their skills during the module. To help develop the ability to evaluate their own performance.	Factual Conceptual Meta-Cognitive*	Analyse, Evaluate Remember
Team Contract & Structure	To define a team structure and software development methodology for the project. To agree a code of conduct and outline standards and procedures for the team.	Factual Conceptual Meta-Cognitive*	Remember, Understand Apply Analyse Evaluate
Project Design Document	To outline the architecture for the system, decompose the system into components and highlight functional dependencies. To illustrate how requirements will be met and to what level, to illustrate how the system will work and how the user will interact with it.	Factual Conceptual Procedural	Remember Understand Apply Analyse Evaluate Create
Software Source Code	To produce well structured, well tested, maintainable to meet the requirements. Use of team standards and conventions in coding, including good error handling and consideration of usability issues.	Factual Conceptual Procedural	Apply Analyse Evaluate Create
Software Documentation	To produce test cases, evidence of testing, user manuals and include good commenting in the code itself.	Factual Conceptual Procedural	Apply Analyse, Evaluate, Create
Team Log Book	To learn about accountability within the team, defining tasks, contribution and quality assurance methods and noting responsibility for all aspects of the work.	Conceptual Meta-Cognitive*	Apply Analyse Evaluate Create
Individual Log Book	To keep a personal accurate record of all work undertaken during the project. To make a note of effort and results and account for time during the project.	Conceptual Meta-Cognitive*	Analyse Evaluate
Meeting Minutes	To note the teams' process, professional behaviour and approach to the task in hand. To provide insight into decision points during the project and their effect.	Factual Conceptual Procedural Meta-Cognitive*	Apply Understand Evaluate Analyse
Team Report (Interim & Final)	To reflect on achievements, areas for improvement, areas of success. To evaluate the teams' development process, to demonstrate what has been learned and could be used for the next development project.	Meta-Cognitive* Procedural Conceptual Factual	Analyse Evaluate
Team Presentation	To discuss ideas, prototypes, the development process and demonstrate achievements. Includes reflection on the team process.	Factual Procedural Conceptual Metacognitive*	Analyse Evaluate

Table 33: Assignment Match to Anderson and Krathwohl's Taxonomy

On balance therefore, the cognitive load on students could be viewed as an ideal *range* in terms of what teachers are trying to achieve in the module design and students will understandably execute the assignments to differing levels of performance. The results also illustrated that 7 of the assignments make use of or demand the use of Meta-Cognitive processes and knowledge (highlighted in Table 33 by use of a *). This shows that there is considerable opportunity for reflection and the assessment of that reflection in the module in terms of requiring a student to reflect on and evaluate their own performance and also in terms of reflecting on their team's methods and processes during development.

5.9.7 Evaluation using Professional Standards and Frameworks

For the final part of the evaluation of the assessment design and approach in the SETP, the assignments and learning objectives were reviewed in comparison to the standards set out in Computing Curricula for Software Engineering Education Knowledge as outlined by IEEE and the ACM. (SEEK 2003, SE2004). I decided to use these rather than the QAA and BCS standards set out in Chapter 2. The reason for this choice is that the QAA and BCS standards are assessed by Newcastle University in the School of Computing Science on a regular basis by these bodies. The BCS conducts an accreditation visit and inspection every 5 years and Internal Subject Review inspects programme quality in QAA terms also every 5 years. The Software Engineering Module is always included in these reviews and always seems to be well received in terms of aims, objectives, teaching methods and standards that are set for students. The module has never been reviewed using the SEEK curricula and so this was felt to be a valid and interesting step to take, to find out if the course content and associated assessment is well-rounded in its aim at teaching Software Engineering in view of these international curricula revisions.

In SEEK Bloom's attributes are specified using the following:

- Knowledge (K), Comprehension (C), Application (A)

The relevance of the topic to the core body of knowledge that a student on a single honours programme in Software Engineering should know at the end of their studies is highlighted as Essential (E), Desirable (D) or Optional (O) within each of the SEEK Education Knowledge Areas. With these classifications in mind, I compared the module taught content and assessment design structure with the SEEK units for a whole programme in Software

Engineering, even though the SETP is only one 20 credit module from the programme at Newcastle. The full set of tables that show the results of the evaluation in more detail can be found in Appendix D but for the purpose of summary reporting, Table 33 shows the high level Knowledge Areas (KA) from SEEK, the number of topics from each Knowledge Area unit that are covered in the teaching or practical elements of the module, and the number of these that are assessed. The SEEK Knowledge Areas for Specialities and their related topics is not taught in second year of the undergraduate Computing Science or Information System programmes. Students specialise in the third year of their studies. Therefore the SEEK Knowledge Areas for Specialities were not evaluated in this review of the teaching and assessment for the Software Engineering Team Project module.

Result: The analysis of the teaching and assessment on the SETP Module using the SEEK Knowledge Areas was interesting as it highlighted areas that are perhaps over-assessed (Software Modelling & Analysis, 17/20 topics). The Professional Practice Area is covered very well in terms of the number of topics covered in teaching and assessment, 10/15 and 8/15 respectively (Table 34).

Knowledge Area	Number of Units and Sub- Topics covered in module	Number of topics that are assessed.
Computing Essentials	2/4 Units, 11/41 topics	4
Mathematical & Engineering Foundations	2/3 Units, 3/21 topics	1
Professional Practice	3/3 Units, 10/15 topics	8
Software Modelling & Analysis	7/7 Units, 20/42 topics	17
Software Design	4/6 Units, 11/39 topics	7
Software Validation & Verification	2/5 Units, 13/35 topics	6
Software Evolution	2/2 Units, 2/13 topics	0
Software Process	2/2 Units, 3/14 topics	
Software Quality	2/5 Units, 3/28 topics	2
Software Management	5/5 Units, 16/31 topics	9

Table 34: SEEK Comparison Summary

Software Management is also covered quite well with 16 out of the 31 topics being covered in teaching and 9 of these 16 topics being assessed within the module. The review using SEEK also highlighted some Knowledge Areas that merit more consideration in both the teaching and the assessment, i.e. Software Quality with only 3/28 topics covered in the teaching and 2 of these 3 are assessed and Software Validation & Verification, with only 2/5 units and only 13/35 topics covered in the module.

5.9.8 Discussion of Results

It must be said that the ability to program is an essential skill for all Software Engineers i.e. to build software solutions. However, one of the main drivers for including a team project in Software Engineering to the IS and CS students, was to teach students about the whole Software Engineering process and the value of every role in a development project. Projects need more than just programmers. Given some of the feedback on issues and problems faced by students during the CETL years of this study it is good to see in the statistical

analysis that student attainment in the module was not adversely impacted by the CETL intervention and cross-site work. However the results highlight some real issues that need to be addressed to ensure fairer assessment on the module. The results show that student perceptions of the inequality of attainment between IS students and CS students during the CETL ALiC review were somewhat justified. Programmers did get higher marks during the module. The statistical analysis showed that marking procedures on the whole were fair in the sense that good module marks were not affected by a student's programme of study, but that programming score (from First Year) was the strongest predictor of whether someone would implement or not and those most likely to implement during the module were CS students. The statistical results also show that good First class and 2.1 marks in the module were closely related to the breadth of roles taken on by an individual, not just a programming role, and that those who led teams and/or took responsibility for organisation or documentation, also scored well. These results are particularly heartening as good participation and involvement in all aspects of the module are what is expected from students. It was hoped that all students would take part in the programming and as many of the other roles and aspects of the project as they wanted to, regardless of programming scores or experience. The module was intended to allow students to try any role they wanted and not to pigeon hole or penalise anyone for the role they chose.

However, one of the most worrying aspects of the statistical results was the fact that IS students who programmed tended to be treated poorly in the peer assessment, i.e. they achieved a lower peer percentage than those IS students who did not program. This was a particularly interesting result because it indicated a problem with perception about IS students' abilities and also perhaps with the assessment design and balance of tasks within the module. It also indicated that some roles and tasks were perhaps not valued as much or deemed as important as programming the actual software product. Also, the results show that CS students did slightly better in peer assessment and their peer mark was higher, particularly if they implemented. These results lead to the understanding that more work needs to be done on reassuring students about fairness of assessment, especially peer assessment, and on recognising the value of all contributions, regardless of the role a student plays in the project. The peer assessment marks and the student's weighting can be determined as favouring CS students in particular, more so than any of the

other assessment marks reviewed during this part of the study. This imbalance in marks, especially those of IS students who implemented, illustrated there are perhaps differences in perceptions about IS and CS students amongst the students themselves that need to be challenged. Perhaps this is, as Dweck noted, students pigeonholed themselves and others in the class in terms of what they could or could not do and the marks they feel they 'deserved' (Dweck, 2001). Indeed staff may have also influenced students' feelings about what they and others were capable of and this filtered down into peer assessment marks. It is difficult to know what caused this result or the extent to which staff influenced peer assessment weights because the Individual and Team effectiveness data was incomplete. The lack of this additional data is rather unfortunate as it could either strengthen or totally refute the other statistical results that show peer assessment was a factor that impacted on IS student attainment negatively during the study. Nevertheless, peer assessment and student perceptions about the value of all roles were clearly areas that needed reviewing and improvement.

The Focus Groups and module questionnaire results during the years of the study i.e. before the CETL (2003/04 and 2004/05), during the first two years of the CETL (2005/06 and 2006/07) and afterwards (2007/08-2009/10) highlighted many areas for improvement in the module, particularly in the area of the cross-site work. There were numerous problems relating to scheduling, technologies and communication between the two universities and these were eventually all rectified as much as possible up until the CETL work ended in 2010. The issues of fairness in workload and assessment remained to a large extent unresolved. The use of Contribution Matrices (as highlighted in Chapter 3) served to alleviate some of the perceptions about the fairness of assessment between sites but did not alleviate the problems of perception of effort between CS and IS students at Newcastle. The feedback from students once again highlights peer assessment and perception of role value and the value of differing contributions between those on the two courses as issues of contention, a finding that is supported by the statistical analysis results. Students also wanted anonymity when conducting peer assessments and some way of ensuring that non-contribution was penalised, preferably by some action from staff, rather than relying on data from contribution matrices and peer assessment scores. The Focus Groups and questionnaires also show that a programming role was still viewed by many students as one of the most

difficult and demanding parts of the SETP and the one area where contribution was deemed as being of more value, more so than any other of the 'simpler' areas of work in the Software Engineering process. These are elements that I decided to investigate and attempt to resolve.

The results from the Anderson and Krathwohl review of the module assignments and their learning objectives proved interesting in terms of the areas of concentration i.e. the knowledge types covered by each assignment and the cognitive levels and skills used and required to complete them. This review showed that there is a high cognitive load in the module and students are expected to demonstrate and use all of the knowledge areas and skills outlined by both Bloom and Anderson and Krathwohl (Bloom, 1954; Anderson and Krathwohl, 2001). This is perhaps too much to expect from a student in one 20-credit module during the second year of their degree. Nevertheless it is recognised that each student will attain, use and learn differing levels of knowledge and cognitive skill during the module and that perhaps aiming high is what we should do rather than narrow our expectations in terms of what is learned and assessed (as was suggested by Knight, 2002). The real challenge for teachers is to ensure that all these cognitive levels and knowledge areas are clearly recognised and verifiable in our assessment design. Also, in our teaching, assessment design and feedback we need to ensure that students are supported as much as possible by teachers to attain the levels we have specified, in balance with our goal for their learning autonomy (as outlined in Chapter 1). The final set of results from the SEEK Review of Module Content and Assessment showed that the module covers quite a lot of the Knowledge Areas, Units and Topics outlined by IEEE and the ACM as Essential for Software Engineering graduates. In some ways this result is very positive as it illustrates that the module covers a lot of the areas required, even though it is only 'worth' 20 credits of a student's degree studies. A broad range of issues and topics are covered in lectures but also practised in the team project itself. Also, the matching of Bloom's taxonomy to the SEEK Knowledge areas illustrates that the module relies on the full range of learning outcomes and cognitive areas covered by SEEK. However, the SEEK curriculum Knowledge Areas matching does not include the cognitive process dimension areas of Analysis, Synthesis or Evaluation from Bloom's Taxonomy nor the Evaluate and Create cognitive processes from Anderson and Krathwohl's revision, and therefore the evaluation of assessment for the SETP module could go one step

further to determine which of these areas are actually covered by SEEK in undergraduate modules of Software Engineering in the UK. However the review has proved to be of value as it has highlighted the need for more teaching on Software Quality and Quality Assurance Measures in the module and also some more concentration on Software Verification and Validation.

5.9.9. Summary

This chapter has outlined the statistical and qualitative results that were used to gain answers to the research questions outlined in Chapter 1. The review of module marks, student reflective reports and data from Focus Groups and questionnaires showed that there were areas of the SETP module that needed changing as programming competency, or more accurately, the perception of programming competency (and non-competency) had adversely affected (at least) the peer assessment portion of the marks of some students from the IS course and those of Other courses who had not contributed to the programming effort during their teams' project. The results showed that some changes were needed in the overall module design to ensure that all roles in the project were seen to be equally valued in assessment and also in practice during the project. This chapter also presented a review of the assessments and learning outcomes using Anderson and Krathwohl's Taxonomy Revision of Bloom and the results indicated that the assignments covered all knowledge areas and cognitive processes in that taxonomy. These results indicated that the module allowed students many opportunities to reflect on their learning throughout the module and that despite a high cognitive load on students, the assignments allowed potential for a range of learning and honing of cognitive skills and knowledge. Finally in this chapter a review of the module using the SEEK Knowledge Areas outlined by ACM and IEEE was carried out. The results of this review showed that some areas of the SEEK curriculum were not covered as well as others in the module (Quality Assurance, Verification and Validation) and these were areas that could be included. The results also showed that the SEEK curriculum specification assumes that students at undergraduate level will not reach the levels of learning in Bloom's taxonomy of Synthesis and Evaluation or those in Anderson and Krathwohl of Evaluate and Create. Therefore strict adherence to this curriculum as a guide for learning design may mean that the

full range of learning outcomes from modules such as the SETP may not be captured or valued correctly in assessment design.

The next chapter presents some experiments that were carried out to change peer assessment in the module in light of these results, where I changed the assessment from holistic to both holistic and categorical, in an attempt to help students focus more on the value of all roles and all contributions, especially those other than programming.

Chapter 6: A New Assessment Framework for Software Engineering Team Projects

6.1 Introduction

This chapter reviews the SETP at Newcastle University in terms of learning theories presented in the Literature review in Chapter 2 and the results presented in Chapter 5 which were the drivers for creating a new assessment framework for SETPs. The chapter then outlines a new assessment framework for undergraduate teams that was first formulated in 2010[1], the Student Appraisal Method (SAM). It then presents two experiments (based on a partial implementation of the SAM framework) that were carried out to try to improve student perceptions and consideration of the value of all roles and contributions within the SETP. The chapter then presents the results from each of these experiments and outlines their implications for future instances of the SETP module. Finally an overview of other aspects of SAM that have been implemented since 2010 is provided, along with some guidance on how the framework could be used by other teachers in their student team projects.

6.2 Drivers for Creating a New Assessment Framework for SETPs

In Chapter 2 a review of the literature illustrated that work from Dweck, and Torrance and Pryor showed there was a need for understanding student motivations and self-esteem, as these factors can impact student perceptions about their work, the assessment they undertake and their attainment (Dweck, 1988; Torrance and Pryor, 1988). It was therefore deemed important to consider these issues when designing an assessment regime for a module or programme of study. For the SETP this implied students' perceptions about themselves and their abilities (and the abilities of other students) could possibly influence their behaviour in peer assessment exercises. The findings in the

[1] Part of this chapter is a re-working of a paper published in 2010 (Devlin and Phillips, 2010) where the first ideas about a new assessment approach were indicated.

statistical and qualitative analyses presented in Chapter 5 also seem to support this conclusion as coders and Computing Science students were generally given higher peer assessment marks and students from the IS discipline were given the lowest overall peer assessment marks, even though they attempted programming.

An investigation of learning theories used in HE in the UK in Chapter 2 showed that the approaches most appropriate to the design and aims of the SETP module before and during the CETL ALiC years are those that advocate both social and experiential learning (i.e. those of Vygotsky and Dewey (Crawford, 1996; Dewey, 1938) and also those that take a constructivist approach and allow for continuous reflection and experimentation such as the work of Kolb and Biggs (Biggs, 2001; Businessballs, 2012). The CETL ALiC initiative aimed to make learning more 'active' and engaging for students so as to help them take more control of their learning, to enable them to learn by doing and to become responsible and autonomous learners, so these learning theories were deemed as the most appropriate to investigate and apply to the module (CETL, 2005). However, as also highlighted in the literature review, a teacher needs to formulate a learning theory and teaching strategy of their own and it was the facilitator role that stemmed from Dewey's theory of experiential learning (1938) that seemed most appropriate at the beginning of the CETL. This role proved to be the difficult to perform as it involved a lot of risk-taking on behalf of the teacher. It meant allowing students to make mistakes and experience being uncomfortable sometimes so that they could learn. This approach also meant that although learning in the module is scaffolded and structured, a lot of the time the students had to find their own way and work through difficulties themselves. The 'teacher as facilitator' approach aims to really benefit students in terms of their journey to becoming autonomous learners and in their sense of pride and achievement at the end of a module that involves teamwork, regardless of the marks so therefore it seemed appropriate and worth the risks and difficulties. In the implementations of this approach teams in the SETP were supported when they had a crisis and given guidance or ideas on how to resolve team and technical issues, but the students were never given the 'answer'. It was deemed appropriate that the teams should be resourceful and attempt to resolve their own issues first, before approaching the teacher to 'rescue' them. Teams at Newcastle were in full control of what happened during their project, in terms of the software development

methodology and team structure they used, in terms of the software they used to build their solution, and in terms of determining the requirements and designs for the solutions they had to produce. This teaching approach was and still is somewhat in contradiction to the culture of support that is inherent in a lot of Higher Education teaching today. Students in Computing are used to being given detailed assignments that set out exactly what they have to do, that specify exactly what the solution should do at the end, and exactly which technologies they should use to realise it. The main point of the module was for students to learn about different software engineering approaches and what worked and what did not.

In terms of assuring quality the module at Newcastle managed to strike a balance between the conflicting priorities of the need to classify student achievement and the need to view assessment as a learning opportunity for students (as outlined by Boud and Falchikov, 2007), but there was room for improvement in the overall learning and assessment design and in the use of peer assessment, as the results in Chapter 5 also illustrate. The new assessment design therefore needs to bear in mind the level of willingness and motivation that students might have at the beginning of a large team project and their self-esteem (Emler, 2001). It is important also to consider that students may need to learn how to take responsibility for their choices *gradually* during the project (Littlewood, 1996). The new learning design and assessment framework should include reflection on the learning process so that this gradual transfer of responsibility can take place. It must also allow for active discovery and experimentation and continuous feedback so that students may adapt their approach and explore their own attitudes and values, as advocated by both Piaget and Kolb (Businessballs, 2012; Koschmann, 1996). Ideally the assessment framework should also (if possible) provide tasks for different types of learning styles (as outlined by Kolb) and focus more on formative assessment of skill development for lifelong learning, in balance with the necessary summative measures of the institution (Burgess, 2007). For the particular case of SETPs it should also focus on the essential competencies and skills required of an apprentice Software Engineer (SE2004, 2004). With these drivers, results and requirements in mind, in the next section the new assessment framework for SETPs and student team projects (in general) is presented.

6.3 The New Assessment Framework: The Student Appraisal Method (SAM)

The findings from Focus Groups (outlined in Chapter 5) showed that students do not always use the feedback or marks they receive for assignments during the module to improve their next submission ("we just got our grade and moved on") or to improve their approach to the teamwork. Students also do not realise the value of the peer assessment exercises, especially when we include category-based assessment as a form of peer feedback. The results of the quantitative and qualitative analysis show that there is also a real need for more valuing of all roles in the project, especially in relation to the contribution of students from other disciplines. The results show that programmers from the CS course are often favoured with higher ratings in peer assessment exercises and this affects the mark achievements of "non-programmers" or those that program from other courses (IS students). With these results in mind, it was decided there was a need to give each student more personalised and relevant feedback on their competence as a software engineer, during the project, as well as at the end i.e. to use more forms of formative assessment, so that students can adjust areas of their approach (as individuals and as teams) before the final summative assessments.

In 2010 a new method of peer assessment was proposed that aimed to give students in these projects a more rounded view of their performance and the value of their roles. The idea was to adapt the approach used in the SETP module using ideas and concepts from the technique of 360 degree feedback or multi-rater feedback (Devlin and Phillips, 2010). 360 degree feedback is a developmental tool often used as a human resources tool to appraise employees and involves an individual being rated by managers, peers and sub-ordinates as well as undertaking a self-assessment (Fletcher, 1999; Tyson and Ward, 2004). Its increasing use indicates that today's organisations "value and reward self-awareness and sensitivity to input from colleagues" (Fletcher, 1999). The results from this form of appraisal have also been shown to improve performance and if performed 'correctly' can encourage learning (Kluger and De Nisi, 1996). For the SETP it was proposed that students undertake a similar form of performance appraisal for their work on the module that would outline their areas of achievement in software competency and would also highlight those areas that needed more development or improvement. Feedback on the

SETP module before this was piecemeal for all the separate assignments and did not give the students an overarching view of how they were performing during the project or their achievement in terms of skills and competency development as software engineers at the end. The peer assessment exercises using holistic feedback did not give students much idea of how they were progressing either or the purpose of peer assessment (Leik, and Wyvil, 2001; Brown, 2006; Freeman et al, 2006). It was proposed that students should be provided with competency matrices for peer assessment that would allow them to evaluate each other and themselves in terms of the competencies required of a novice software engineer including soft skills and technical skills and encompassing the social nature of software engineering. These competencies and skills should also match the learning outcomes for the module and those identified as essential to a software engineer by the QAA and SWEBOK, where possible, as outlined in Chapter 2 (QAA, 2007, SE2004, 2004; Layzell et al, 2000; Joseph et al, 2010). For this purpose the module team decided to use competency matrices such as those outlined by Smith and Smarkusky, an example of one such matrix can be seen in Table 35 (Smith and Smarkusky, 2005). These matrices could be used to capture learning and would give students early *formative* feedback on their progress so they could correct any poor behaviour or maintain their good approach and contributions to the team effort, as appropriate (Black and William, 1998). The use of these matrices would also allow students to track their progress and see how far they had developed since their own skill assessment at the beginning of the module after each instance of peer assessment. The competencies to be used were derived from the 38 competencies identified by Turley and Bieman when they conducted a study to find out what qualities, behaviours, skills and attributes distinguished exceptional and non-exceptional engineers (Turley and Bieman, 1994, 1995). The authors conducted a review of ten exceptional and ten non-exceptional professional software engineers to determine the competencies that are different between the two groups. Their research was conducted in two phases. In Phase 1 they identified critical competencies from a sample of 10 software engineers from five commercial research and development labs, at three sites of one Fortune500 company and during this phase they identified 38 essential competencies of software engineers.

Classification/ Rank	1st	2nd	3rd
Process			
Task performance	Consistently contributes to others tasks and delivers own tasks	Supports others in completing tasks	Does not deliver on own tasks.
Leadership	Exercises leadership skills regularly	Leads some tasks in the project e.g. Design	Prefers not to lead on major tasks but leads smaller tasks
Communication	Shares ideas regularly in meetings. Asks questions that direct conversation to project solution	Asks questions related to project basics.	Shares ideas when asked.

Table 35: Smith and Smarkusky's Competency Matrix Example

In Phase two they surveyed 129 software engineers to try to distinguish the competencies that are different between exceptional and non-exceptional software engineers. Their statistical analysis indicated that exceptional software engineers are distinguished by behaviours with an external focus e.g. a focus on their team or their customer. Exceptional software engineers are also more likely to maintain a big picture view of the project and to help other engineers. Turley and Bieman stated that many of the non-exceptional behaviours identified by them 'can be viewed as the behaviours of inexperienced engineers" because a beginner "will be unsure of their own skills and capabilities" (Turley and Bieman, 1994). The authors also concluded that no simple predictor of what makes an exceptional software engineer exists and that those that are exceptional have no more innate ability than average performers. However they did identify nine key work strategies that differentiate the 'stars' and these are: Taking the initiative, Networking, Self-management, Teamwork effectiveness, Leadership, Followership, Perspective, Show and tell - sharing ideas with others and Organisational 'savvy'. They also found that "the skills and strategies of the stars can be taught to average performers" (Turley and Bieman, 1994). Therefore, with this information in mind, defining skills and behaviours in a similar fashion to Turley and Bieman would mean that teachers of software engineering could define positive and negative behaviours and skills sets for students of software engineering (or their own version of them, in

line with the module objectives) and these could be used to evaluate progress throughout a team project. This approach could help students to gain more confidence and to adjust their behaviour if necessary. This approach would also improve the relevance of peer assessment reviews. However the number of competencies identified would need to be chosen carefully, as ideally feedback would be succinct and not onerous to complete and personalised for every student. Increasing student numbers might make this a huge task so a limited set of competencies was recommended. The use of competency matrices and a 360^0 feedback approach would also give students an insight into how their professional work would be assessed by employers and colleagues in their future careers. It would give a more competency and behaviour-focused (rather than role-focused) view of a student's progress and contribution during a SETP module. These matrices and the competencies associated with them could also be readily adapted to team projects in other disciplines. It would be feasible to identify generic skills and competencies associated with teamwork (as identified in some of the peer assessment tables and methods outlined by Leik and Wyvil in Chapter 2 (Leik and Wyvil, 2001). These generic skills could then be substituted in the competency matrices and used as a basis for evaluating the performance of students.

6.3.1 Components of the Student Appraisal Method

The Student Appraisal Method consists of three instances of competency matrices that are issued to students throughout the team project. The competencies that are used will vary from project to project and depend on the subject being studied. An example of the matrix developed for the SETP at Newcastle can be found in Appendix E. In the case of the SETP, the matrices are introduced throughout the software development cycle at crucial feedback points as follows.

- **Competency Matrix 1** – This matrix is used for *self-assessment* by the student *before* the project begins in earnest (i.e. top of circle in Figure 18) and ideally should be completed by the students before they meet together in their teams. The matrix should highlight the essential skills that are to be developed or used during the project. The aim of this matrix is to allow the student to assess what skills or experience they possess at that moment in time that will be needed to complete tasks

during the project, for example, programming, leadership, communication skills and design. The matrix should allow the student to highlight what they perceive is their current level of experience or skill for each element or skill. Skill levels on the matrix could be outlined in terms of *expert* level, *intermediate* level or *novice*, or the teacher can use phrases such as "I have no experience in this area" or "I have a little experience in this area" to describe experience levels. The idea is to get the student thinking about what they can already do and also about skills they might need to learn or improve upon during the project. This matrix should ideally be issued in conjunction with a report assignment that allows students to elaborate on their skill and experience levels and to illustrate their current understanding of what is needed for the project. Completion of this matrix and report gives the teacher very good insight into where students are starting from, their previous experiences and also any worries or concerns they may have about the team project. This matrix (as **Competency Matrix 4)** is reused at the end of the project to help the student write their final report and re-evaluate their skills once the project has finished.

- **Competency Matrix 2:** This matrix consists of the same set of skills and experience level descriptions as Matrix 1. This matrix is used for *both self and peer assessment* and should be issued part-way through the project (top right in Figure 18), when some work has been completed by the teams. Students should use this matrix to evaluate their teammates and themselves and to reflect on any skills they have improved or learned anew during the first part of the project. Matrix 2 should ideally be used in conjunction with an initial 'holistic' or numerical judgement from students about contribution to the team effort so far. In the case of Newcastle SETP the matrix is used in conjunction with the student allocating 100 percent (in terms of an effort judgement) between team members for the first part or first semester of the project. Familiarity with the matrix from the initial usage should make it clearer to students how they personally have progressed when they receive feedback from their peers but also evaluate their own performance. Competency Matrix 2 should always be preceded by some tutor feedback on elements of the project work so

that students can gauge their performance level using this feedback in conjunction with feedback from their peers.

- **Competency Matrix 3:** This matrix again consists of the same set of skills and experience level descriptions as those previously experienced by the students. This matrix is used towards the end of the project as part of the project review process (in the blue section/ top left of Figure 18). This matrix comes after formative assessment events that have provided feedback from experts and tutors. At this point students have received feedback from their peers, from tutors and from experts and they have also reflected on their own performance in light of their self-assessment at the beginning of the project. Towards the project end another reflection on their skills and achievements during the project should help them to evaluate their competency (and that of their peers) in terms of the project results but also their own contributions to the whole process. *Competency Matrix 3 is used for another round of self and also peer-assessment.* Feedback from this task should also assist each student in writing their final reflective report and help them to understand what they have learned and achieved during the project and what they still need to work on for the future and they can use the feedback from peers to fill in the final matrix, **Competency Matrix 4**.

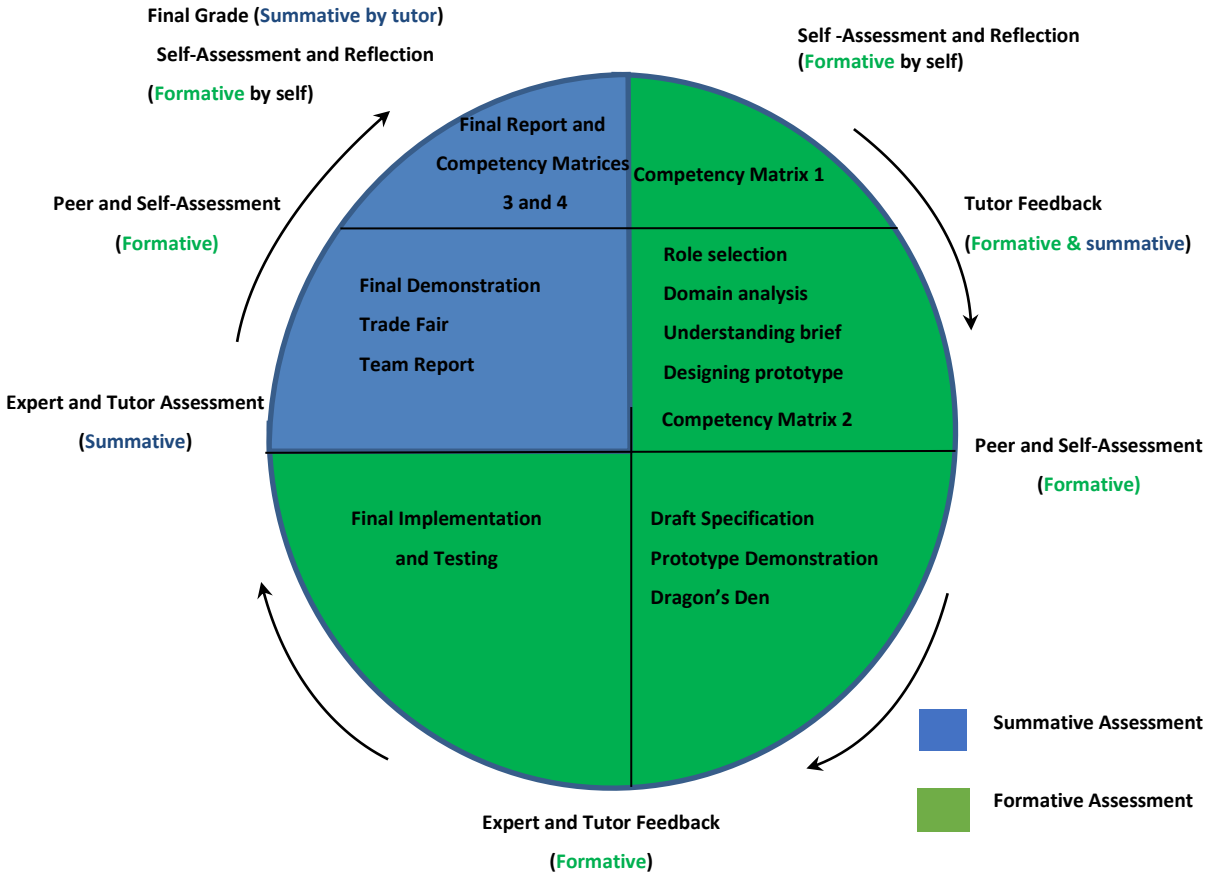

Figure 18: The Student Appraisal Method

An initial (and partial) implementation of these competency matrices was carried out in the SETP module, in an attempt to address all the issues raised by students and the findings from the review of the student data in Chapter 5. The next section outlines two experiments with peer assessment that involved the use of both holistic and category-based peer assessment and a set of competencies that students could use to evaluate themselves and their teammates.

6.3.2 Experiment 1 - Individual Category and Holistic Peer Assessment

In the academic year 2010/11 there were 163 students who took the SETP module and these students were divided into 22 teams (each with 6 or 7 team members). No Durham students were involved in this year. For peer assessment in this year students were asked to submit their individual views to NESS on how each team member had performed. They were asked to submit both a

holistic assessment i.e. the division of 100% between team members based on perceived contribution, and a category-based assessment of how each team member had performed their tasks. Some categories were provided to students as a guide to help them assess the contributions and performance. These categories were: Communication, Task Completion, Attendance at Meetings, and Participation. The students were told that these categories were suggestions and they could add their own categories for evaluation if they wished. Students were asked to justify their holistic percentage sharing via the category-based assessment, for example, " I gave X only 17% as he did not turn up to all the meetings and had not always completed the tasks he had been set for that week."

Results: There were two instances of peer assessment during this academic year, as in other years. In both instances students were asked to carry out the individual category-based and holistic peer assessment. The results of this experiment were disappointing as students invariably adhered rigidly to the suggested categories for the category-based assessment. They also did not elaborate much on these categories and submitted a very brief and rather unreflective category-based review for each member of their team.

The majority of students also made no real effort to elaborate on the reasons for their allocation of marks for the holistic peer assessment during this experiment. Feedback from students from the module questionnaires showed that they were not sure why they were conducting the category-based assessment and most said that they were unsure what to write. Some students in this year also misunderstood what was meant by the holistic division of 100%, as can be seen in Table 36:

	Communication (Out of 25 marks)	Task Completion (Out of 25 marks)	Attendance (Out of 25 marks)	Meeting Participation (Out of 25 marks)	Percentage /100%
Student A	25	25	25	25	100
Student B	24	25	25	24	98
Student C	22	25	25	22	94
Student D	23	25	25	23	96
Me	25	25	25	25	100
Student E	25	25	25	25	100
Student F	25	25	25	25	100

Table 36: Incorrect Example of Holistic Assessment

Table 34 shows that some students took the categories from the category-based assessment and used them as a method for calculating each student's effort out of 100% for the holistic peer assessment, so they did not come up with one percentage for the overall performance. In these cases the module leader had to rely on the team monitor to give an overview of each individual's performance to ensure that a fair weighting could be recorded for each team member. These results illustrated that students were unsure of what to do during the peer assessment tasks and that they had not given a clear explanation of what they needed to do in the case of both the holistic and the category-based peer assessment. However, some good examples of justification of the peer assessment marks were found during this experiment, as can be seen in Figure 19.

Student X Communication:

- X is extremely good at communication within the group both in person and using our Facebook Group.
- He has many ideas and is enthusiastic.
- Assigns tasks and make sure things get completed effectively.
- Challenges ideas regularly in order to do the best work possible

Figure 19: An example of a detailed category-based review

"To resolve any communication and participation issues we have, I am going to encourage the individuals within meetings and directly ask for their opinions. I am also going to talk to X with Z to make sure he attends on time. There may need to be a slight re-evaluation of roles and positions to make sure X and Y fit into the team better. Overall the team are working extremely well together and I am very happy with everyone's level of commitment and enthusiasm, I really like my team members a lot. On a personal level I am going to do more research to improve my own knowledge and results of team"

Figure 20: A student reflection on peer assessment, 2010/11

Some students also reflected on what they could do to improve the category areas that they thought were weak in their team as can be seen in Figure 20. The results from the first experiment therefore indicated that more work needed to be done on ensuring that students knew why they were carrying out the category and holistic-based assessment. They also illustrated that when students are given suggestions of categories, many will adhere strictly to the categories provided and not create their own categories for evaluating their own performance or that of their teammates. Many students will also not elaborate much on the categories provided unless they are instructed to do so. The results of this experiment were a little disappointing but they were also a learning experience provided more insight into what could be improved for next time.

6.3.3 Experiment 2 – Team Holistic and Category Based Assessment

In the academic year 2011/12 there were 184 students who took the SETP module and these students were again divided into 22 teams (each with 7 or 8 team members, some teams had 9 members). Again, no Durham students were involved. Teams were selected at random by the module leader (in previous years we had 'seeded' teams based on programming score but from 2010/11 onwards random team selection was felt to be more authentic). For peer assessment this year students were asked to submit their views as a team to NESS on how each team member had performed. They were again asked to submit both a holistic assessment i.e. the division of 100% between team members based on perceived contribution, and a category-based assessment of how each team member had performed their tasks. The same categories used in

Experiment 1 were provided to students as a guide to help them assess the contributions and performance, along with more detailed instruction on how they could also use categories of their own to arrive at their evaluations. These categories were as in Experiment 1: Communication, Task Completion, Attendance at Meetings, and Participation. Students were again asked to justify their holistic percentage sharing via the category-based assessment. The purpose of the peer assessment was explained to students and it was also suggested that the second face-to-face meeting for peer assessment be used to review the project outcomes and to aid each individual in writing their final reflective report.

Results:

The results from Experiment 2 were in fact much *worse* in terms of detail than those in Experiment 1. Teams typically did not spend much time discussing the category-based assessments and many complained that they found the whole process very uncomfortable. An example of one team's effort can be seen in Table 37:

Team Member	Positive attribute	Attribute to improve	Contribution
P	Participation	Technical Input	12.5
L	Technical input	Research	12.5
D	Task Completion	Technical input	12.5
I	Attendance	Participation	12.5
K	Organisation	Communication	12.5
A	Research	Organisation	12.5
F	Organisation	Communication	12.5
S	Communication	Attendance	12.5

Table 37: Example of team-based category assessment 2011/12

It was unclear what had gone wrong at first but it was found there was an issue with the instructions that had been given to students. Students said they wanted personalised feedback on their progress and did not want to discuss their performance weaknesses with other teammates in the face-to-face meeting. Students found it difficult to evaluate performance, as they had no other team

development experiences to compare this instance to. They said they found it difficult enough to agree on the holistic percentages and that category-based assessment seemed pointless. It was at this point that it was recognised there had also been a problem with Experiment 1. Each student had been asked to provide their feedback but these had not been sent back to each student afterwards because of a technical problem. The reason that this did not happen was the way the exercise was set up in the NESS system. The peer assessment exercises were set up as exercises that did not receive a mark. In the NESS system, at the time (and still), when an exercise does not receive any marks then the lecturer could not provide online feedback about that exercise. Therefore the students saw the exercise as a waste of time because they did not get the personalised feedback in time to make any changes or improvement in their behaviour. The second peer assessment was also viewed as a waste of time because it came at the end and meant that students could not make any improvements that might impact on their grade for the module and that was their main concern. They did not see the value of peer assessment for their future careers or in terms of feedback on areas they could improve for the next project and this was because it had not been explained it clearly enough. These results revealed a real need for a better way of conducting peer assessment, online. If the first category-based assessment was to be of value to students then they needed to get the feedback on their personal performance in time to be able to make changes for the next stage of the project, and if the second round was to be of value then students needed to understand what the implications were for their final module mark and for their future career. These experiments also illustrated more to me about all the peer assessments that had been conducted before in the module, both at the time of the cross-site work and before and after this work, at a local team level. Students at Newcastle had not had any feedback or justification for the holistic peer assessment marks they were each awarded and they had little or no feedback on their personal *performance* during the project, only that which was openly discussed in a meeting with peers, a meeting that made a lot of students uncomfortable. It became clear that the methods of peer assessment still needed to be adjusted in the module so that it became anonymous (as some students had requested in the Focus Groups discussed in Chapter 5) and also that it was viewed as a worthwhile exercise which provided useful and timely feedback to students and

helped them learn more about themselves and their growing competency as software engineers.

6.4 Implementation of more elements of the Student Appraisal Method

As well as experiments with category-based competency matrices for peer assessment during the module, the results of the statistical and qualitative analyses in Chapter 5 illustrated that students needed more types of feedback if the SAM assessment framework was going to work similarly to how 360 degree feedback and a competency appraisal process does in industry. Students needed to get a more *rounded* view of their progress and development with the help of different groups of people they interacted with during the project and the assessment marking criteria needed to be explained more fully. The tasks and assignments also needed to indicate the value of all roles in the software engineering process more strongly. To this end a number of changes in the design and assessment of the module have been implemented since 2010 and these are outlined in the following sections.

6.4.1 Discussion of Peer Assessment and Assessment in the Opening Lecture

Since the CETL ALiC work ended in 2010 the module leaders have presented an overview of the assignments and assessment procedures in the SETP in a lecture at the beginning of every academic year. This hour-long session includes an in-depth review of peer assessment and the impact of peer assessment marks on the overall module mark for students at the end. This session provides the opportunity for students to ask questions and also discuss their doubts and fears about peer assessment. This session is also used to explain the value of all roles in the project and to discuss how students need to be fair and professional when reviewing the contribution of their teammates. SAM requires that students are very clear about how they will be assessed, who will be assessing them, the marking criteria and what is expected of them, so this session is very important.

6.4.2 A Greater Balancing of Skills Required in the Problem Brief

The assignment briefs for the module have been reformulated over the years to ensure that the technical and academic skills of all students from the different programmes involved in the module are required to solve the problem. This

means that the module team have become more specific about the type of technologies that can be used to create the solution (whilst also attempting to allow students some freedom to choose). This is accomplished by providing a number of different application types and teams can select the one that suits them best, based on the skills available in their team. If students are to be appraised on their software engineering competency and the behaviours and skills associated with that, then *all* these skills need to be required in the problem brief and in the solution, in equal weight.

6.4.3 A Wider Variety of Assignment Types

Some different assignment types have been introduced into the module so that students from different backgrounds and with differing levels of technical and non-technical skills could contribute equally. These assignment types include the creation of a poster, the requirement for more evidence of testing and test case design, assignments such as meeting stakeholders and gathering requirements via interview etc. All of these involve the use of skills such as communication, presentation, research skills, documentation skills, graphics skills etc. and not just programming and design skills. Again for SAM to be effective this means that all the skills being assessed are required in the assignments and that there is a fair balance of skills required in each of the assignment types.

6.4.4 Clearer Overview of Marking Criteria and the Final Module Mark Calculation

During all the years of the CETL the equation for the final mark calculation (as outlined in the Chapter 3 Case Study) was not visible to students and was never explained. This meant that students did not know exactly the specific assessment values of the team and individual elements of the coursework for the module. Many students were puzzled by their final grade, because these marks were 'hidden' from view. This was so that it was felt that students would be inclined to give a narrower range of scores in their peer assessment if they could see the calculation. From the academic year 2011/12 onwards the calculation was available to students on Blackboard (a new VLE) and the exact weighting of each assignment for the module was detailed. It was explained clearly how team and individual marks worked in the module and this had

never really been done before. In some ways the fear of a narrow range of peer assessment marks has been borne out as in 2013/14 there were no IS students involved in the module and 7 of the 15 teams awarded all their team members the same mark. However, it is difficult to argue or corroborate whether this was because of the lack of IS students in the seven teams or because they had a team that contributed equally to every part of the work. On balance though, it was deemed best to be more open and transparent about assessment. Students are mark-oriented, no matter how much we aim for them to enjoy learning and see the value of what they are doing in the long term. So, for SAM to work, the assessment 'value' and what it means to their degree should be explained. Students should also be clear on what we mean by a first-class performance, and a second-class performance, and so on.

6.4.5 The Involvement of Software Engineers (from all roles) in Feedback

Local software engineering professionals were invited to take part in the final technical demonstrations of the teams' applications at the end of the academic year. The engineers were provided with a feedback sheet for each team and asked to comment on all aspects of each team's Software Engineering Process (they did not award marks). This feedback was given to student teams along with the mark assigned for the demonstration that was awarded by the module leader. This type of feedback was very much valued by the students and many commented that it made the project seem more real and that they really appreciated getting an industry view on their work. If the aim of a team project is to provide an authentic view of what students will experience in the workplace, then people from the workplace should be invited in to give their views. This is not always easy to manage but some employers are willing to give their time, especially if they can look at what students are doing and give comment on it, from their perspective. It is also important that students get a variety of feedback from a range of employers. The SAM idea requires teachers to invite expert practitioners in the discipline to comment on student novices and their performance.

6.4.6 The Introduction of Formative Assessment 'Events'

Two formative assessment events were introduced to the SETP so that students could get feedback on their abilities as software engineers and on their work (a)

at a very early stage in development from a panel of experts and (b) from the public and domain experts at the end of the project. The first event was Dragon's Den and is based on the TV series of the same name (Dragons Den, 2014). In the TV show entrepreneurs present their ideas for a product to a panel of would-be investors. The entrepreneurs describe their idea, the target market for the idea and also detail the technical and financial aspects of their product that they think will make their idea a business winner. A panel of Software Engineering specialists (from industry) and some stakeholders for each particular application were invited to take part. This event takes place very early on in the module when students are developing their prototypes and still working out the requirements for the application. In this event each team gives a presentation on their understanding of the requirements and then gives an overview of their ideas for the solution. Panel members ask questions and give feedback to students on the viability of their solution, their proposed software development approach, proposed technology choices and features of the application, their development plan etc. Students find this a rewarding experience and it gives them an early indication of their progress and elements of the work that they might need to adjust to be able to deliver an effective solution.

The second formative assessment event that was introduced was a Trade Fair at the end of the module. The idea originated from a Faculty-based lecturer in Enterprise and provided a great chance for our students to take part. In the Trade Fair event each team is given a stand that they must 'decorate' and a table to display their product. We invite the general public, academics from the School of Computing Science, all the industrial contacts, 'customers' and interested potential users of the product to the Trade Fair. Each team must demonstrate their product to each visitor at their stand. Visitors then fill in an anonymous feedback sheet for each team and this feedback is collated at the end of the event and sent to each team. This event gives students feedback on their product and also on how well they have fulfilled the brief. It provides an opportunity to really test their product 'in the field'. Again, students enjoy this event and put a lot of effort into it. They also really value feedback from potential users and from employers and other academics.

6.4.7 How SAM can be generalised for other Team Projects

SAM involves considerable effort on behalf of teachers for team project modules. However, the effort is worth it because students get a wide variety of feedback from different sources and a more rounded view of their achievements in Software Engineering. To implement SAM in other disciplines requires some thought on what a team project is trying to achieve. The method is perhaps more suited to practical science-based team projects in disciplines such as Engineering, Chemistry or Biology as it relies on very concrete definitions of skills, behaviours and attributes that are required of a practitioner of the discipline in industry. However, subjects in Humanities or Business could run formative assessment events e.g. poster sessions or student conferences that require the attendance of professionals and academics from the discipline. The competency matrices that are part of SAM could easily be adapted to list the expected skills levels or learning outcomes from such team projects and to multi-disciplinary projects (although defining competencies might be more challenging). Nevertheless, the principles of 360 degree feedback could be generalised to any discipline, with a little imagination.

However, SAM is not a rigorous assessment method designed to make marking team *processes* easier, rather it is an assessment method that is formative i.e. *for* learning, one that should help students to evaluate their own progress in terms of skills, knowledge and behaviour, and that of their peers, more easily and more effectively.

6.5 Summary

In this chapter an overview of the learning theories used in the SETP module was presented and some insight provided on the teaching approach that developed during the CETL ALiC years and afterwards. The chapter then introduced a new assessment framework for SETP that developed from the issues raised in the research on assessment in the module that are presented earlier in this thesis and from my experiences teaching the module since 2005. The chapter then outlined two ‘failed’ experiments using parts of the new assessment framework that were carried out in a bid to address the issues of role perception and contribution that were apparent from the earlier research the learning from these experiments was detailed. Other aspects of the SAM assessment framework that have been implemented since 2010 are then

outlined as well as how SAM could be adapted for other disciplines. The next chapter concludes this thesis, summarising the outcomes of the work so far and detailing further work that will be carried out.

Chapter 7: Conclusion

7.1 Review of Research Questions

In Chapter 1 of this thesis 5 questions to be answered by this research effort were presented. The first question focused on determining the differences between the work of coders and non-coders during the project and how this impacted on their attainment/success in the SETP. The quantitative analysis in Chapter 5 demonstrated that factors that affected student attainment were two-fold i.e. peer assessment weightings (based on student perceptions) and the role that a student took during the project. The analysis showed that more work was needed on ensuring students understood the value of all roles and contributions during the SETP. Question 2 was related to perception also as it focused on determining if the module leaders had missed something fundamental in the teaching of background material or in the design of assessment for the SETP that had affected student perceptions about the value of roles. The findings in Chapter 5 demonstrated again that more work needed to be done on helping students understand their own skills and contributions and also on teaching them to value the contribution of others. This is the same problem and required remedy needed as for Question 1.

The challenge for the third question was to ascertain whether there was a bias in the assessment marking criteria in favour of those who coded in the SETP. An exploration of the marks achieved by students from the IS and CS programmes and in the various roles in the project, over several years of the SETP, demonstrated that marking by module leaders was not biased in favour of one group in particular. However, some data was unavailable for exploration i.e. the marks from team monitors. So, the answer to Question 3 is inconclusive and needs to be explored further when more comprehensive data is available.

Question 4 queried whether module leaders had fallen into the trap of specifying assessment criteria based on perceptions of ability rather than what students actually did (as discussed by Black and William, 1998). This thesis has contributed to the debate on assessment design and criteria, including the value

of classifications for degrees, the importance of formative assessment and has demonstrated some of the value of focusing on skills and learning for life long development. The assessment criteria for the SETP were designed to measure what students did rather than ability. However there was a problem in the way assessment was explained to students, especially peer assessment, and the results in Chapter 5 showed clearly that this needed to be remedied to ensure more fairness.

The last research question in Chapter 1 was "Did a student have to be 'good' at programming to do well?" The research has demonstrated that this question was a complex one to answer as many factors contributed to student success during the SETP. The results of Focus Groups and statistical analyses showed that programmers were often given a slightly higher peer percentage and were sometimes more 'valued' during the module. The answer to this question was to challenge students' perceptions, rather than change marking or marking criteria.

7.2 Thesis Summary

This thesis describes a study undertaken to review the assessment regime of the Software Engineering Team Project Module at Newcastle University and to construct an assessment framework that acknowledges the validity and value of all roles in software development project teams and rewards all student contributions fairly. In Chapter 1 the thesis outlined the assessment review of the CETL ALiC cross-site activity implemented by Newcastle University and Durham University and the research problem of fair assessment of coders and non-coders in the student teams that emerged from this review. In Chapter 2 key areas of literature relevant to this study were identified from the work on experiential learning, learner autonomy and motivation, on instructional and assessment design methods used to evaluate teamwork in Higher Education in the UK today, and on competency criteria and standards determined by professional bodies in Software Engineering. In Chapter 3 an overview of the CETL ALiC cross-site activity was provided and the methods of assessment used by both Durham University and Newcastle University to evaluate the performance of cross-site companies were presented. This chapter also detailed the assessment issues faced by students and the CETL ALiC team during the CETL years 2005-2010. Chapter 4 introduced the quantitative and qualitative

methods used in this study to evaluate the effectiveness of the assessment methods used during the cross-site activity and on the SETP module before and after the CETL ALiC initiative. Chapter 5 outlined the results of these analyses and their implications for the assessment regime of the SETP. The findings show that students need more help in reviewing their skills development and achievements during the module and in recognising the value of all contributions and roles in peer review exercises. The statistical results illustrated that there were some misconceptions about the value of some roles during the project, especially those that were deemed non-technical. These misconceptions led to variations in peer assessment values that needed correcting. These corrections in the way peer assessment is performed as outlined in Chapter 6 aimed to ensure that students reward all programmers for their efforts, regardless of their programme of study (IS and CS) and to recognise the value of inputs to the team efforts in areas such as planning, organisation and documentation as well as more technical areas, especially programming. In Chapter 6 two experiments that sought to help students with peer assessment were outlined. These experiments provided an insight into student perceptions about their personal abilities and skills and those of their teammates but were not wholly successful in addressing the issues raised by the peer assessment results outlined in Chapter 5. Chapter 6 outlined the solution that was derived to address the wider issue of helping students review their personal Software Engineering competency during the SETP module (and upon its completion) using the SAM formative assessment framework. Chapter 6 also detailed aspects of the Student Appraisal Method that are currently in place and how these could be adopted by other universities for any module that involves team work.

7.3 Future Work

This thesis defined a competency matrix as part of SAM to be used (four times) at three stages of the SETP module during the academic year 2014/15. A copy of the self-assessment version that is currently in use in the SETP can be found in Appendix E. This competency matrix, in the form of Smith and Smarkusky's Competency Matrices, incorporates skills, abilities and behaviours expected to be developed by each student during the SETP module and are largely based on the learning outcomes for the module (Smith and Smarkusky, 2005). These

skills, abilities and behaviours are also based on some of those outlined in the study of Software Engineering professionals carried out by Turley and Bieman (Turley and Bieman, 1988) and also on aspects of the Software Engineering Education body of Knowledge (SE2004, 2004). In future iterations of the SETP the calculation of peer weightings and their contribution to a student's final mark will be derived using the marking criteria used by Conway et al. outlined in Chapter 2 (Conway et al., 1994) where levels of contribution are defined and then translated into marks. Students will be made aware of the value of each level of contribution. These matrices will be used four times during the module. The first time they will be used is during the Self-Assessment/Skill Evaluation exercise at the beginning of the SETP module. Students will use these to assess their individual skills and ability levels before being placed into teams. The second and third time the matrix will be used is to evaluate self and peers during the two peer assessment periods of the module (once per semester, one at the midpoint of the project and one near the end) and the final time the matrix will be used is as part of the Individual Reflective Report written by each student at the end of the module. Peer assessment matrices and their results will be sent out to students just after the first peer evaluation session (module mid-point) and just before they write their Individual Reflective Report at the end of the module. Students will be able to use their peer review matrices and their self- assessment matrix to help evaluate and reflect on their own performance and competency development as a software engineer.

7.4 Conclusion

To address the assessment issues raised during the CETL ALiC review of the SETP module and to ensure that students recognise the value of all contributions and roles during their team project, this thesis has outlined a new approach to assessment for team project modules that encompasses formative peer and self-assessment along with tutor, public, and professional reviews of student work. The work conducted in this thesis has helped to identify some ways in which tutors can help students to reflect and evaluate their own performance and that of their teammates more effectively via the use of competency matrices. It has also helped to identify factors that might impact student attainment in SETPs and reviewed the particular impact of programming competency on attainment on the SETP module at Newcastle

University. It has also determined that programming competency is not a primary predictor for individual and team success on the SETP module but it was a factor with *strong* influence in terms of students' peer assessment weightings and the value placed on each role in their development team. The SAM assessment framework outlined in Chapter 6 of this thesis has created a more rounded approach to assessment and feedback for students in terms of their competency development during the SETP module at Newcastle University. The SAM framework is essentially a set of formative assessment methods that rely on feedback from the full range of people and groups involved in and affected by the student development teams on the SETP module and their software products. SAM's aim is to provide a wider range of feedback on all aspects of the project including a team's professional behaviour, the quality of their software product and the effectiveness of their development approach. Parts of the SAM approach that were implemented have performed extremely well in this regard e.g. the formative assessment events outlined in Chapter 6, section 6.4.6. However, implementations of other aspects of SAM towards the end of this study have had varying degrees of success. The experiments in Chapter 6 showed that much more work is needed on developing the skills and abilities used in the competency matrices that are central to the SAM approach. Another future development of this work is to fully automate the self and peer-assessment process for the SETP module and to incorporate this into our existing virtual learning environment Blackboard. Student learning outcomes and module performance using the SAM framework will also be reviewed again at a later date.

References

Access Grid: http://www.accessgrid.org/ < accessed online 21/08/2014>

Altman, D.G. (1999*). Practical statistics for medical research*. New York. NY: Chapman and Hall/CRC Press.

Association of Graduate Recruiters, available at http://www.agr.org.uk <accessed online 17/07/2014>.

Anderson, L., W., Krathwohl, D. with Airasian, P., W. et al, (Eds.), (2001), A Taxonomy for Learning, Teaching and Assessing: A revision of Bloom's Taxonomy of Educational Objectives, New York, Longman, 2001.

ASKe Position Paper on Assessment, Available online at: http://www.brookes.ac.uk/aske/documents/ASKePositionPaper.pdf <accessed online 09 September 2014>

Balla, J., Boyle, P., (1994), Assessment of Student Performance: A Framework for Improving Practice, Journal of Assessment and Evaluation in Higher Education, Vol. 19, No. 1., pp.7-28.

The British Computer Society Code of Conduct and Good Practice, http://www.bcs.org/content/conWebDoc/1587 <accessed online 13 April 2011>

BCS Code of Conduct Summary, http://www.bcs.org/upload/pdf/conduct.pdf, <accessed online 17th July 2014>

BCS Accreditation Criteria http://www.bcs.org/content/ConWebDoc/52296, <accessed online 17th July 2014>

Bell B., S. & Kozlowski, S., W., J., "A Typology of Virtual Teams: Implications for Effective Leadership", *Group and Organization Management*, Vol. 27, No.1, 2002, pp.14-49.

Biggs, J, (1999), Teaching for Quality Learning at University, Buckingham: SHRE/OU Press.

Biggs, J, (2001), On Constructive Alignment, seminar notes to LTSN, University of Edinburgh, July 3rd 2001.

Biggs, J., (2003), Teaching for Quality Learning at University, Buckingham: SHRE/OU Press.

Biggs, J., & Collis, K., F (1982), Evaluating the Quality of Learning: The SOLO Taxonomy, New York, Academic Press.

Birmingham University
http://www.cs.bham.ac.uk/admissions/undergraduate/se.php, <accessed online 31/08/2010>

Black, P. & William D, Inside the Black Box: Raising Standards through Classroom Assessment by P *Phi Delta Kappan*, Vol. 80, 1998

Black, P., and William, D., Assessment in Education, Vol. 5, No. 1, 1998

Bligh, D.A. (1998), What's the use of lectures? (5th ed.), Exeter: Intellect.

Bloom, B., S., (1956), Taxonomy of educational objectives: the classification of educational goals by a committee of college and university examiners, New York, Longmans, Green, 1956-64.

Blondy L. C., (2007), Evaluation and Application of Andragogical Assumptions to the Adult Online Learning Environment, *Journal of Interactive Online Learning*, Vol. 6, No. 2.

Boon, J., van der Klink, M., (2002) Competencies: The Triumph of a Fuzzy Concept, Academy of Human Resource Development, Annual Conference, Honolulu, HA, 27th Feb -3rd March Proceedings, Vol. 1, pp.327-334

Boud, D, (1995), Assessment and learning: contradictory or complementary? *in* P.T. Knight (Ed.) *Assessment for Learning in Higher Education*, pp35-48 (London, Kogan Page).

Boud, D and Falchikov, N, (2007), Rethinking Assessment in Higher Education, Learning for the longer term, *Routeledge*, New York, 2007.

Boyatzis, R.E., (1982), The Competent Manager: A Model for Effective Performance, (NY:Wiley).

Brodie, L., Zhou, H., Gibbons, A., (2008), "Steps in developing an advanced software engineering course using problem based learning", *Engineering Education*, Vol 3, No 1, 2008, pp.2-12.

Brown S. and Race P., (1999) The Lecturer's Toolkit (London: Kogan Page)

Brown S., (2004), Assessment for Learning, Learning and Teaching in Higher Education, Issue 1, 2004.

Brown, S., (2006), Invitation to Assessment, Director of Quality, University of Northumbria, UK, in Deliberations, Available online at: http://www.londonmet.ac.uk/deliberations/assessment/brown.cfm, <accessed online 20/07/2011>

Burgess, (2007), Beyond the Honours Degree Classification: The Burgess Group Final Report, October 2007, Universities UK.

Businessballs, (2012), Available online at http://www.businessballs.com/kolblearningstyles.htm, <accessed online 12/06/2012>

Butler, R., (1988), Enhancing and undermining intrinsic motivation: the effects of different feedback conditions on motivational perception, interest and performance, Journal of Educational Psychology, 79, pp.474-482.

CETL initiative, (2005), http://www.hefce.ac.uk/whatwedo/lt/enh/cetl/, <accessed online 04/02/2015>

Charlton T, Devlin M, Drummond S., (2009), Using Facebook to improve communication in undergraduate software development teams. Computer Science Education, 2009, 19(4), pp.273-292.

Chickering, A. and Gamson, Z., (1987), Seven Principles for Good Practice in Undergraduate Education, AAHE Bulletin March 1987.

Chickering, A. W. and Gamson, Z. F. (1991), Appendix A: Seven principles for good practice in undergraduate education, New Directions for Teaching and Learning, 1991: pp.63–69.

Cohen, J.W. (1988), Statistical Power analysis for the behavioural sciences (2nd edition), Hillsdale NJ: Lawrence Erlbaum Associates.

Confessore, S. J. (2002), L'autonomie de l'apprenant dans les nouvelles situations de travail (Learner autonomy in the new workplace). In A. Moisan & P. Carré (Eds.), *L'autoformation, fait social?Aspects historiques et sociologiques* (Self-directed learning as a social fact? (Historical and sociological aspects) (pp.195-214), Paris: L'Harmattan.

Conway,R., Kember, D., Sivan, A. & Wu, M., (1993), Peer assessment of an individual's contributions to a group project, Assessment and Evaluation in Higher Education, 18(1), pp.45-56.

Coppit D, (2006), Implementing large projects in software engineering courses, Computer science education, Vol. 16, No. 1 March 2006 pp.53-73

Crawford, K, (1996), Vygotskian approaches in human development in the information era, Educational Studies in Mathematics September 1996, Volume 31, Issue 1-2, pp.43-62.

Crooks, T, J (1998), The impact of classroom evaluation practices on students, Review of Educational Research, 58, pp.438-481

Crooks, T., J (1988), Assessing Student Performance: Green Guide 8, Higher Education Research and Development Society of Australasia.

Cryer, P., and Elton, L., (1992), Learning Actively on One's own, Block B: Preparing Self-Instructional Materials Effective Teaching and Learning in Higher Education, Module 8 Sheffield USDTU, 1992.

Delamare Le Deist, F. & Winterton, J. (2005), What is competence? Human Resource Development International, 8(1), pp.27-46.

Denicolo, Pam, Entwistle, Noel and Hounsell, Dai, (1992), Effective Learning and Teaching in Higher Education, Module 1, parts 1 & 2, *What is Active Learning,* , CVCP Universities Staff Development and Training Unit, Sheffield, 1992

DeRemer, F. and Kron, H.H, (1976), Programming-in-the-Large Versus Programming-in-the Small, IEEE Transactions on Software Engineering, June 1976, Vol. 2, No. 2., pp.80-86

Devlin M, Phillips C., (2010), Assessing Competency in Undergraduate Software Engineering Teams. *In: IEEE Education Engineering Conference (EDUCON)*. 2010, Madrid, Spain: Universidad Politecnica de Madrid.

Devlin M, Marshall L, Phillips C., (2006), Active Learning in Computing: Engaging Learners in a Cross-Site Team Project. *In: SOLSTICE Conference 2006*. 2006, Edge Hill, Ormskirk: Edge Hill Centre for Excellence in Teaching and Learning.

Devlin, M., Phillips, C. and Marshall, L, (2007), Making Computing Science Students More Employable with Problem-Based Learning and Cross-Site Teamwork, *In International Conference on Engineering Education and Research* (iCEER) 2007, Melbourne, Australia, 2-7 December 2007 International Network for Engineering and Education Research, 2007 Notes : Proceedings on CD-ROM. Session : Industry, Problem and Project Based Learning. Paper no. 5. 11 pp.

Devlin, M., Drummond, S., Phillips, C. and Marshall, L, (2008a), Improving Assessment in Software Engineering Student Team Projects, In *9th Annual Conference of the Subject Centre for Information and Computer Sciences*, 26th-28th August 2008, Liverpool Hope University, White, H. (ed.) pp 133-139, Higher Education Academy, Subject Centre for ICS, 2008

Devlin, M., Drummond, S. and Hatch, A., (2008b), Using Collaborative Technology in CS Education to facilitate Cross-Site Software Development, *Journal of Systemics, Cybernetics and Informatics*, Vol. 6, Issue 6, International Institute of Informatics and Cybernetics, 2008, pp.1-6.

Devlin, M., Phillips, C. and Marshall, L, (2009a), Assessment in Software Engineering- Towards a new Framework for Group Projects. In P*roceedings of the ICEE & ICEER 2009 Korea International Conference on Engineering Education and Research*, 23-28 August 2009, Seoul, KoreaKim, H.S. (ed.), pp.13-18, Se Yung Lim, Korea University of Technology and Education, 2009.

Dewey, John, (1916), Democracy and Education; New York, McMillan, MW 9:179.

Dewey, J., (1938), Experience and Education, Kappa Delta Pi Lecture Series, Collier-Macmillan Books 1963, London.

Directgov, (2011), http://www.direct.gov.uk/en/EducationAndLearning/QualificationsExplained/DG_10039017, <accessed online 15/04/2011>

Donnolly, R and Fitzmaurice, M., (2005). Designing Modules for Learning. In: *Emerging Issues in the Practice of University Learning and Teaching*, O'Neill, G et al, Dublin AISHE, 2005, pp.99-110.

Dragon's Den http://www.bbc.co.uk/programmes/b006vq92 <accessed online 09/09/2014>

Dreyfus H and Dreyfus, S.E., (1986), Mind over Machine: The power of human intuition and expertise in the era of the computer (New York Free Press).

Drummond, S. and Devlin, M., (2006), Software Engineering Students' Cross-Site Collaboration: An Experience Report, *In 7th Annual Conference of the Subject Centre for Information and Computer Sciences*, 29th-31st August 2006, Trinity College, Dublin, Steede, H. and Hackett, J. (eds.), pp.95-100
Higher Education Academy, Subject Centre for ICS, 2006.

Dweck, C., S. and Leggett, E., (1988), A Social-Cognitive Approach to Motivation and Personality, Psychological Review, Vol. 95, No.2, pp.256-273.

Dweck, Carol, S., Legget, Ellen, L., (1988) A Social-Cognitive Approach to Motivation and Personality, Psychological Review, 1988, Vol. 95, No.2, pp.256-273

Eclipse, (2014) https://www.eclipse.org/ <accessed online 21/08/2014>

Emler, N., (2001) "Self-esteem: the costs and causes of low self-worth", (York, Joseph Rowntree Foundation and YPS.

Eneau, J., (May 2008), From Autonomy to Reciprocity, or Vice Versa? French Personlism's Contribution to a New Perspective on Self-Directed Learning, *Adult Education Quarterly*,, Vol. 58, No. 3, pp.229-248.

Ecclestone, K., (2000) Assessment and Critical Autonomy in Post Compulsory Education in the UK, Journal of Education and Work, Vol. 13, No. 2, pp.141-162.

ECTS Users' Guide (2005) Brussels: Directorate-General for Education and Culture. Available online at: http://ec.europa.eu/education/lifelong-learning-policy/doc/ects/guide_en.pdf <accessed online 15/04/2011>.

Entwistle, N, (1996), Recent research on student learning *in* J. Tait and P. Knight (Eds.), *The Management of Independent Learning*, pp.97-112 (London, Kogan Page).

Falchikov, N., (1995), Peer Feedback Marking: Developing Peer Assessment, Innovations in Education and Training International, 32(2), pp.175-187.

Falchikov, N., (1988), Self and peer assessment of a group project designed to promote the skills of capability, Innovations in Education and Teaching International, 25(4), pp.327-339.

Ferguson, F., Kussmal, C., McCracker, D., Robber, M., A., (2004), "Offshore Outsourcing: current conditions & diagnosis", *Proceedings of the 35th SIGCSE Technical Symposium on Computer Science Education*, Virginia, USA, 2004, pp.330-331.

Flanagan, John C, (1954), The Critical Incident Technique, Vol 51, No. 4 The Psychological Bulletin
Available at: https://www.apa.org/pubs/databases/psycinfo/cit-article.pdf <accessed online 17th July 2014>

Fletcher, C, (1999), The implications of research on gender differences in self-assessment and 360 degree appraisal, 9(1), pp.39-46.

Ford, L, (2010), "Graduates Lacking Soft Skills, Employers Warn", *Education Guardian*,
http://www.guardian.co.uk/money/2007/jan/30/workandcareers.graduates,
Jan 30, 2007 <accessed online 24/05/2010>.

Fowler, J., (2008), Experiential Learning and its facilitation, Nurse Education Today, 28, 427-433

Freeman, M., Hutchinson, D., Treleaven, L., Sykes, C., (2006), Iterative learning: Self and Peer assessment of group work,Proceedings of the 23rd ascilite conference: Who's learning? Whose Technology? pp257 265.

Freeman, R., and Lewis R, (1998), Planning and Implementing Assessment (London: Kogan Page)

Freire, P., 1970, Pedagogy of the Oppressed, Harmondsworth, Penguin, London.

Gibbs, G and Dunbar-Goddet, H. (2007) The effects of programme assessment environments on student learning, Report submitted to the Higher Education Academy, York. Accessible online at: http://www.heacademy.ac.uk/assets/York/documents/ourwork/research/gibbs_0506.pdf, <accessed online 18/04/2011>.

Goldfinch, J., (1994), Further Developments in peer assessment of group projects, Assessment and Evaluation in Higher Education, 19(1), pp.29-35.

HEA, (2006), Higher Education Academy Student Employability Profiles: Computing, Sept. 2006, http://www.heacademy.ac.uk/assets/York/documents/ourwork/tla/employability_enterprise/student_employability_profiles_apr07.pdf, <accessed online 13/04/2011>.

HEA, (2007), Higher Education Academy Student Employability Profiles, available at: http://www.heacademy.ac.uk/assets/documents/employability/student_employability_profiles_apr07.pdf <accessed online 17/07/2014>.

HEAR, (2014), Higher Education Achievement Record, available at: http://www.hear.ac.uk, <accessed online 17/07/2014>

Herbsleb, J. D., A. Mockus, et al., (2000), Distance, dependencies, and delay in a global collaboration in *Proceedings of the 2000 ACM conference on Computer supported cooperative work*, Philadelphia, Pennsylvania, United States, ACM Press.

Higgs, Bettie, McCarthy, Marian, (2005), Active Learning: From Lecture Theatre to Field Work in Emerging Issues in the Practice of University Learning and Teaching. O'Neill, G., Moore, S.., McMullan, B., (eds), Dublin, AISHE, 2005, pp.37-44.

Holec, H, (1981), Autonomy in Foreign Language Learning. Oxford: Pergamon

Hume Anne and Coll, Richard, K., (2009), Assessment of learning, for learning and as learning: New Zealand case studies, Assessment in Education: Principles, Policy & Practice, 16:3, pp.269-290

IET, (2014), Institution of Engineering and Technology, available at: http://www.theiet.org, <accessed online 17/07/2014>.

Imrie, B.W. (1995), Assessment for Learning: quality and taxonomies, Assessment and Evaluation in Higher Education 20(2), pp.175-189.

Jarvis, P., (2004), Adult Education and Lifelong Learning, third edition, Routeledge Falmer, London.

Jeris, L. and Johnson, K., (2004), Speaking of Competence; towards a cross-translation for human resource development and continuing professional education, Academy of Human Resource Development, Annual Conference, Austin, TX, Proceedings Vol. 2, pp.1103-1110.

Johnston, L. and Miles, L. (2004), Assessing contributions to group assignments, *Assessment & Evaluation in Higher Education*, 29(6).

Joseph D., Ang, S., Chang, R., H., L., and Slaughter, S., (2010), "Practical Intelligence in IT: Assessing Soft Skills of IT professionals", *Communications of the ACM*, Vol. 53, No 2, 2010, pp.1480-154.

Kennedy, D., Hyland, A., Ryan, N, (2007), Implementing Bologna in your Institution: Writing and Using Learning Outcomes: a Practical Guide, Quality Promotion Unit, University College Cork, 2007, pp.1-30, Available online at: http://www.bologna.msmt.cz/files/learning-outcomes.pdf <accessed online 15/04/2011>.

Knight, P., (2002), Summative Assessment in Higher Education: Practices in Dissarray, Studies in Higher Education, 27:3, pp.275-286.

Kolb, D., (1984), Experiential Learning: Experience as the Source of Learning and Development, Prentice Hall, Englewood Cliffs, New Jersey.

Koschman, T., (1996), Problem-based learning: A principled approach to the use of computers in collaborative learning in CSCL: Theory and Practice of an Emerging Paradigm – T. Koschman, Editor, 1996, Laurence Erlbaum Assoc.,Mahwah, NJ, pp.83-124.

Krathwohl, D., R., Bloom, B., S. and Masia, B., B., (1964), Taxonomy of Educational Objectives: The Classification of Educational Goals. Handbook II: Affective Domain. New York: David McKay Co., Inc.

Landis, J.R. and Koch, G.G. (1977), The measurement of observer agreement for categorical data. Biometrics 33, pp.159-74.

Laerd Statistics, (2014), Available at: https://statistics.laerd.com/spss-tutorials/cohens-kappa-in-spss-statistics.php, <accessed online 17/07/2014>

Layzell, O Pearl Brererton, Andrew French, (2000), Supporting Collaboration in Distributed Software Engineering Teams, in *Proceedings of Seventh Asia-Pacific Software Engineernig Conference, ASPEC*, 2000, pp.38-45

Leik, Mark, Wyvil, Michael, Farrow, Stephen, (1996), A Survey of Methods Deriving Individual Grades from Group Assessments, Journal of Assessment & Evaluation in Higher Education, Vol. 21, No. 3, 1996, pp.267-280

Lejk, M and Wyvil, M, (2001), Peer Assessment of Contributions to a Group Project: a comparison of holistic and category-based approaches, Assessment & Evaluation in Education, Vol. 26, No.1, 2001, pp.61-91.

Lewis, M, (2007), Stepwise versus Hierarchical Regression: Pros and Cons, Annual Meeting of the Southwest Educational Research Association, Feb 7, 2007, Available at: https://www.academia.edu, <accessed online 5th February 2015>

Linn, B.S., Arsotequi, M amd Zeppa, R., (1975), Performance Rating Scale for Peer and Self-assessments, British Journal of Medical Education, 9, pp.98-101.

Liu, J., Marsaglia, J., Olson, D., (2002), "Teaching Software Engineering to make students ready for the Real World", Journal of Computing Sciences in Colleges, Vol. 18, issue 2, (December, 2002), pp.43-50.

Littlewood, W., (1996), Autonomy: an anatomy and a framework. *System* 24(4), 2, pp.427-435

McInnis, James, R., and Devlin, M., (2002), *Assessing Learning in Australian Universities, Core Principles of Effective Assessment.* http://www.cshe.unimelb.edu.au/assessinglearning/05/index.html, <accessed online
16/08/2010>.

Miller, D. and Lavin, F., (2007), But now I feel I want to give it a try: formative assessment, self-esteem and a sense of competence, The Curriculum Journal, Vol. 18, No.1, March 2007, pp.3-25.

MOF, (2014), http://www.ncl.ac.uk/computing/current/module/CSC2022 <accessed online 21/08/2014>

McNamara, R., A., (2004), Evaluating Assessment with Competency Mapping in *Proceedings of the Sixth Conference on Australasian Computing Education, Volv. 30., R Lister and A. Young (eds), ACM International Conference Proceeding Series, Vol. 57, Australian Computer Society, Darlinghurst, Australia, pp.193-199.*

Moseley, D., Baumfield, V., Elliott, J., Gregson, M., Higgins, S., Miller, J., Newton, D., P. (2005), Frameworks for Thinking – A Handbook for Teaching and Learning, Cambridge University Press, 2005.

Natriello G, (1987), The impact of evaluation processes on students, Educational Psychologist, 2, pp.155-175.

Newcastle Graduate Skills Framework, (2013), http://www.ncl.ac.uk/quilt/assets/documents/str-gsf-framework.pdf,

<accessed online 17/07/2014>

NESS, (2014), https://ness.ncl.ac.uk/index.php <accessed online 21/08/2014>

Nias, J., (1979) Teaching in Groups in Higher Education: Types, Purposes and Techniques, In Aims and Techniques of Group Teaching, London Society for Research into Higher Education, 1st Edition, 1970 4th Edition, 1979 edited by Jennifer Nias.

Ofqual, (2011) European Qualifications Framework, http://www.ofqual.gov.uk/qualification-and-assessment-framework/eqf < accessed online 15/04/2011>

O'Neal, M., (2004), "Restructuring Computing Programs to Meet Employment Challenges", *IEEE Computer,* Vol. 37, No. 11, 2004, pp.29-34.

Oxford University (2014) Paper 1, Higher Education and Higher Learning, Institute for the Advancement of University Learning, University of Oxford: http://www.learning.ox.ac.uk/media/global/wwwadminoxacuk/localsites/oxfordlearninginstitute/documents/supportresources/lecturersteachingstaff/resources/resources/Higher_Education_and_Higher_Learning.pdf <accessed online 24/06/2014>

Pallant J, (2010), SPSS Survival Manual Open University Press, McGraw Hill Education.

Pintrich, P., R. (1999), The role of motivation in promoting and sustaining self-regulated learning, International Journal of Educational Research 31, Chapter 2, pp.459-470

Psychstat, (2013): http://www.psychstat.missouristate.edu/multibook/mlt08m.html, <accessed online 18/01/013>

QAA, (2000), QAA Subject Benchmarks, AR 009 4/2000, Quality Assurance Agency for Higher Education, 2000.

QAA, (2006), Code of Practice for the assurance of academic quality and standards in higher education, http://www.qaa.ac.uk/academicinfrastructure/codeOfPractice/section6/default.asp <accessed online 31/08/2010>

QAA, (2007), QAA 170 03/07, Subject Benchmarks, Computing, Quality Assurance Agency for Higher Education, 2007.

QAA, (2011), Understanding assessment: its role in safeguarding academic standards and quality in higher education – A guide for early career staff. Sept 2011. Available at: http://www.qaa.ac.uk/Publications/InformationAndGuidance/Documents/UnderstandingAssessment.pdf <accessed online 03/072014>

Queen Mary University, (2010), http://www.dcs.qmul.ac.uk/undergraduate/programmes/g400.php, <accessed online 31/08/ 2010>

Race, P, (2001), "A Briefing on Self, Peer and Group Assessment", LTSN Generic Centre (2001).

Race, Phil, (2005), Making Learning Happen – A Guide for Post-Compulsory Education, SAGE Publications.

Reese, W. J., (2001), The Origins of Progressive Education, History of Education Quarterly, Vol. 41, Issue 1, pp.1-24.

Schön, D., (1983), The Reflective Practitioner, New York: Basic Books.

Software Engineering, (2004), Curriculum Guidelines for Undergraduate Degree Programmes in Software Engineering http://sites.computer.org/ccse/SE2004Volume.pdf, <accessed online 18/07/2014>

Software Engineering Education Knowledge, (2003), Computing Curricula, Software Engineering Volume, Final Draft of the Software Engineering Education Knowledge, eds. Ann Sobel, April 30 2003.

Smith, Harold, H. and Smarkusky, Debera L., (2005), "Competency Matrices for Peer Assessment of Individuals in Team Projects" in Proceedings of SIGITE'05, pp.155-161, 2005.

Spratt M., Humphreys, G., and Chan, V., (2002) Autonomy and Motivation: Which comes first? Language Teaching Research 6(3), pp.245-266.

Steinaker, N, and Bell, R, (1979), The Experiential Taxonomy, Academic Press, New York.

(Stevens 1996) Stevens, J., (1996), Applied multivariate statistics for the social sciences (3rd edition.) Mahwah, NJ: Lawrence Erlbaum.

Subversion, (2014), http://www.eclipse.org/subversive/ <accessed online 21/08/2014>

Suchan, J. and Hayzak, G., (2001), "The Communication Characteristics of Virtual Teams: A Case Study", *IEEE Transactions on Professional Communication*, Vol. 4. , Number 3, Sept. 2001, pp.174-186.

SE2004, (2004), Software Engineering Body of Knowledge, executive editors, Alain Abran, James W. Moore; editors, Pierre Bourque, Robert Dupuis (2004) Pierre Bourque and Robert Dupuis Editors, *Guide to the Software Engineering Body of Knowledge - 2004 Version* IEEE Computer Society pp. 1–200. http://www.swebok.org. <accessed online 16/09/2010>

Sweeney, S., (2012), The Bologna Process and the European Higher Education Area. Available at: https://www.heacademy.ac.uk/sites/default/files/Simon_Sweeney_bologna_workshop.pdf
< accessed online 19/09/2012>.

Tabachnick, B.G. and Fidell, L.S. (2007), Using multivariate statistics (5th edn) Boston, Pearson Education.

Thanasoulas, Dimitios, (2011), Constructivist Learning, available at: http://www.seasite.niu.edu/Tagalog/Teachers_Page/Language_Learning_Articles/constructivist_learning.htm, <accessed online 03/072014>

Torrance, H, (2007), Assessment as Learning? How the use of explicit learning objectives, assessment criteria and feedback in post-secondary education and training can come to dominate learning. Assessment in Education: Principles, Policy & Practice, Special Issue: Assessment in Post-Secondary Education and Training.

Torrance, H and Pryor, J., (1988), Investigating Formative Assessment: Teaching, Learning and Assessment in the Classroom, Open University Press, 1988.

Topping, Keith, (2009), Peer Assessment, Theory into Practice, 48(1), pp.20-27.

Turley, R., T., and Bieman, J., M., (1994), "Identifying Essential Competencies of Software Engineers", *Proceedings of the 22nd annual ACM Computer Science Conference on Scaling up: meeting the challenge of complexity in real-world computing applications*: Phoenix, Arizona, United States, pp.271 - 278, 1994.

Turley, R., T., and Bieman, J., M., (1995), "Competencies of Exceptional and Non-exceptional Software Engineers, Journal of Systems and Software, vol. 28, issue 1, pp.19-38, 1995.

Tynjala, P., Salminen, R., T., Sutela, T., Nuutinen, A and Pitkanen, S, (2005), Factors related to study success in Engineering Education,
European Journal of Engineering Education, Vol. 30, number 2, pp.221-231.

Tyson, S and Ward, P, (2004), The Use of 360 Degree Feedback Technique in the Evaluation of Management Development, Management Learning, 35(2), pp.205-223.

University of York, (2010), http://www.cs.york.ac.uk/undergraduate/ug-courses/course-profile/, <accessed online 31/08/2010>

Varela, F. (1989). *Autonomie et connaissance* (Principles of biological autonomy). Paris: Seuil. (Original work published 1979).

Vassar, (2014), Vassar Stats Online book: Available at: http://vassarstats.net/textbook/parametric.html, <accessed online 18/07/2014>.

von Glasersfeld, E., (2007), Aspects of constructivism. Sense, Rotterdam: 91-99, available at: http://www.vonglasersfeld.com/139.2, <accessed online 03/07/2014>

Vygotsky, L, (1978), Mind in Society: the Development of Higher Psychological Processes, Cambridge MA: Harvard University Press.

Weeden, l P., Winter, J., Broadfoot, P., (2002), Assessment, What's in it for schools? (London, Routeledge, Falmer), 2002.

Weinberg, Gerald, M., (1971), The Psychology of Computer Programming, Computer Science Series, Van Nostrand Reinhold Company Inc., 1971

Weiner, B, (1992), Human Motivation, Metaphors, theories and research, Newbury Park, CS, SAGE Publications *in* Assessing Women in Engineering (AWE) Project (2005), Attribution Theory, AWE Research Overviews, - available at http://www.aweonline.org <accessed online, 31/07/12>.

Yorke, M., (2001), No. 1, Assessment: A guide for Senior Managers in Smith, B., Blackwell, R., and Yorke, M., (Eds) Assessment Series (York LTSN Generic Centre), available online at: http://www.heacademy.ac.uk/resources/detail/assessment/assessment_series, < accessed online, 03/07/2014>

Appendix A

Year	Number	Research	Design	Implement	Test	Organise	Lead	Document	Durham	Analyse	Communicate	Organisation
2003-04	20340519	1	1	0	1	1	0	1	0	0	0	1
2003-04	22064367	1	1	0	1	0	0	1	0	0	1	1
2003-04	10824436	1	1	1	1	1	1	1	0	1	1	1
2003-04	20333441	0	1	1	1	0	0	1	0	0	0	1
2003-04	20901044	1	1	1	0	0	1	1	0	0	0	0
2003-04	22048622	1	1	0	1	0	0	1	0	0	0	0
2003-04	20565433	1	0	0	1	1	0	1	0	1	1	1
2003-04	21269121	1	1	0	0	1	1	1	0	1	0	0
2003-04	23829596	1	1	1	1	0	0	0	0	0	0	0
2003-04	34121799	1	1	1	0	0	0	1	0	0	1	0
2003-04	37055312	0	1	1	1	1	1	0	0	0	0	1
2003-04	1293708	1	1	0	1	1	1	1	0	0	1	1
2003-04	3381573	1	1	0	1	1	0	1	0	0	1	1
2003-04	23195158	1	1	1	0	1	0	1	0	0	0	0
2003-04	33683973	1	1	1	1	1	0	1	0	0	1	1
2003-04	11565912	1	1	1	0	0	1	1	0	0	1	1
2003-04	23171668	1	1	1	1	0	0	1	0	1	1	0
2003-04	20175014	1	1	0	1	1	0	1	0	0	0	1
2003-04	20542652	1	1	0	1	0	0	1	0	1	0	0
2003-04	11916491	0	0	1	1	0	0	1	0	0	0	0
2003-04	34007596	1	1	1	1	0	0	1	0	0	0	1
2003-04	23740671	1	1	1	1	1	1	1	0	0	1	0
2003-04	22774112	1	1	1	1	1	0	1	0	0	0	1
2003-04	21073942	1	1	0	0	1	0	1	0	0	0	0
2003-04	12004481	1	1	0	1	1	1	1	0	0	1	0
2003-04	24201452	0	1	1	1	0	1	1	0	0	1	0
2003-04	22420116	1	1	1	1	0	0	0	0	0	0	0
2003-04	21129995	1	1	1	1	0	0	0	0	0	1	0
2003-04	20543866	1	1	1	0	0	0	1	0	0	0	1

Appendix B

Coefficients[a]

Model		Unstandardized Coefficients		Standardized Coefficients	t	Sig.	95.0% Confidence Interval for B		Correlations			Collinearity Statistics
		B	Std. Error	Beta			Lower Bound	Upper Bound	Zero-order	Partial	Part	Tolerance
1	(Constant)	-1.093	4.957		-342	.733	-11.445	8.059				
	TeamAv	.892	.074	.537	12.034	.000	.745	1.037	.545	.559	.537	.999
	ProgScore	.135	.028	.262	5.876	.000	.090	.180	.278	.313	.262	.999

a. Dependent Variable: Mark

COEFFICIENTS TABLE 1

Coefficients[a]

Model		Unstandardized Coefficients		Standardized Coefficients	t	Sig.	95.0% Confidence Interval for B		Correlations			Collinearity Statistics	
		B	Std. Error	Beta			Lower Bound	Upper Bound	Zero-order	Partial	Part	Tolerance	VIF
1	(Constant)	57.126	2.665		21.434	.000	51.883	62.370					
	Course	-1.219	.987	-.068	-1.235	.218	-3.160	.723	-.129	-.069	-.066	.946	1.057
	ProgScore	.135	.028	.262	4.746	.000	.079	.191	.278	.257	.255	.946	1.057

b. Dependent Variable: Mark

COEFFICIENTS TABLE 2

Coefficients[a]

Model		Unstandardized Coefficients		Standardized Coefficients	t	Sig.	95.0% Confidence Interval for B		Correlations			Collinearity Statistics	
		B	Std. Error	Beta			Lower Bound	Upper Bound	Zero-order	Partial	Part	Tolerance	VIF
1	(Constant)	63.579	2.181		29.158	.000	59.289	67.869					
	Research	-2.192	1.544	-.079	-1.420	.157	-5.229	.845	-.077	-.079	-.079	.992	1.008
	Design	1.442	1.739	.048	.829	.408	-1.979	4.863	.068	.047	.046	.934	1.070
	Implement	2.200	1.141	.111	1.929	.055	-.044	4.445	.122	.108	.107	.939	1.065
	Test	-.110	1.175	-.005	-.094	.925	-2.423	2.203	.013	-.005	-.005	.983	1.017
2	(Constant)	63.382	2.553		24.830	.000	58.360	68.405					
	Research	-1.807	1.521	-.065	-1.188	.236	-4.800	1.186	-.077	-.067	-.065	.984	1.016
	Design	1.076	1.715	.036	.628	.531	-2.298	4.450	.068	.035	.034	.925	1.081
	Implement	1.785	1.161	.090	1.537	.125	-.500	4.070	.122	.086	.084	.872	1.146
	Test	-.228	1.161	-.011	-.196	.845	-2.512	2.056	.013	-.011	-.011	.971	1.030
	Organise	1.804	1.045	.096	1.727	.085	-.252	3.859	.107	.097	.094	.968	1.033
	Lead	2.968	1.080	.154	2.747	.006	.842	5.093	.190	.153	.150	.951	1.052
	Document	-1.829	1.347	-.078	-1.358	.176	-4.481	.822	-.133	-.076	-.074	.901	1.110

COEFFICIENTS TABLE 3 - Hierarchical Multiple Regression

Coefficients[a]

Model		Unstandardized Coefficients		Standardized Coefficients	t	Sig.	95.0% Confidence Interval for B		Correlations			Collinearity Statistics	
		B	Std. Error	Beta			Lower Bound	Upper Bound	Zero-order	Partial	Part	Tolerance	VIF
1	(Constant)	57.126	2.665		21.434	.000	51.883	62.370					
	ProgScore	.135	.028	.262	4.746	.000	.079	.191	.278	.257	.255	.946	1.057
	Course	-1.219	.987	-.068	-1.235	.218	-3.160	.723	-.129	-.069	-.066	.946	1.057
2	(Constant)	57.259	3.240		17.673	.000	50.884	63.633					
	ProgScore	.136	.028	.263	4.786	.000	.080	.191	.278	.260	.256	.945	1.058
	Course	-1.130	.985	-.063	-1.147	.252	-3.069	.809	-.129	-.064	-.061	.943	1.060
	Research	-2.380	1.486	-.086	-1.601	.110	-5.304	.545	-.077	-.090	-.086	.994	1.006
	Design	1.960	1.626	.065	1.206	.229	-1.238	5.158	.068	.068	.065	.993	1.007
3	(Constant)	56.073	3.357		16.703	.000	49.467	62.678					
	ProgScore	.126	.028	.245	4.456	.000	.070	.182	.278	.244	.235	.920	1.086
	Course	-1.362	1.116	-.076	-1.220	.224	-3.558	.835	-.129	-.069	-.064	.713	1.402
	Research	-2.145	1.467	-.077	-1.462	.145	-5.031	.741	-.077	-.082	-.077	.991	1.009
	Design	1.689	1.657	.056	1.019	.309	-1.571	4.949	.068	.057	.054	.928	1.078
	Implement	.272	1.260	.014	.216	.829	-2.208	2.752	.122	.012	.011	.694	1.441
	Organise	2.064	1.024	.109	2.015	.045	.049	4.079	.107	.113	.106	.944	1.060
	Lead	2.664	1.041	.138	2.558	.011	.615	4.712	.190	.143	.135	.958	1.043

Table 11.1
Transformation of r to z

r	z_r	r	z_r	r	z_r	r	z_r	r	z_r
.000	.000	.200	.203	.400	.424	.600	.693	.800	1.099
.005	.005	.205	.208	.405	.430	.605	.701	.805	1.113
.010	.010	.210	.213	.410	.436	.610	.709	.810	1.127
.015	.015	.215	.218	.415	.442	.615	.717	.815	1.142
.020	.020	.220	.224	.420	.448	.620	.725	.820	1.157
.025	.025	.225	.229	.425	.454	.625	.733	.825	1.172
.030	.030	.230	.234	.430	.460	.630	.741	.830	1.188
.035	.035	.235	.239	.435	.466	.636	.750	.835	1.204
.040	.040	.240	.245	.440	.472	.640	.758	.840	1.221
.045	.045	.245	.250	.445	.478	.645	.767	.845	1.238
.050	.050	.250	.255	.450	.485	.650	.775	.850	1.256
.055	.055	.255	.261	.455	.491	.655	.784	.855	1.274
.060	.060	.260	.266	.460	.497	.660	.793	.860	1.293
.065	.065	.265	.271	.465	.504	.665	.802	.865	1.313
.070	.070	.270	.277	.470	.510	.670	.811	.870	1.333
.075	.075	.275	.282	.475	.517	.675	.820	.875	1.354
.080	.080	.280	.288	.460	.523	.680	.829	.880	1.376
.085	.085	.285	.293	.485	.530	.685	.838	.885	1.398
.090	.090	.290	.299	.490	.536	.690	.848	.890	1.422
.095	.095	.295	.304	.495	.543	.695	.858	.895	1.447
.100	.100	.300	.310	.500	.549	.700	.867	.900	1.472
.105	.105	.305	.315	.505	.556	.705	.877	.905	1.499
.110	.110	.310	.321	.510	.563	.710	.887	.910	1.528
.115	.116	.315	.326	.515	.570	.715	.897	.915	1.557
.120	.121	.320	.332	.520	.576	.720	.908	.920	1.589
.125	.126	.325	.337	.525	.583	.725	.918	.925	1.623
.130	.131	.330	.343	.530	.590	.730	.929	.930	1.658
.135	.136	.335	.348	.535	.597	.735	.940	.935	1.697
.140	.141	.340	.354	.540	.604	.740	.950	.940	1.738
.145	.146	.345	.360	.545	.611	.745	.962	.945	1.783
.150	.151	.350	.365	.550	.618	.750	.973	.950	1.832
.155	.156	.355	.371	.555	.626	.755	.984	.955	1.886
.160	.161	.360	.377	.560	.633	.760	.996	.960	1.946
.165	.167	.365	.383	.565	.640	.765	1.008	.965	2.014
.170	.172	.370	.388	.570	.648	.770	1.020	.970	2.092
.175	.177	.375	.394	.575	.655	.775	1.033	.975	2.185
.180	.182	.380	.400	.580	.662	.780	1.045	.980	2.298
.185	.187	.385	.406	.585	.670	.785	1.058	.985	2.443
.190	.192	.390	.412	.590	.678	.790	1.071	.990	2.647
.195	.198	.395	.418	.595	.685	.795	1.085	.995	2.994

Source: McCall (1990); originally from Edwards, A.L. (1967). *Statistical methods* (2nd edition). Holt, Rinehart & Winston

Appendix C

Focus Group Questions

Technologies

What technologies did your company use to communicate?

How well did these technologies help you work together?

Which technologies worked best/ did you prefer?

What technologies would you have liked to use that were not provided?

Did you have any experience of using any of these technologies before the project?

If so, what did you use it for?

If not, did you find it interesting to work with these technologies?

Scheduling

Did you find it hard to schedule time for local (Newcastle team only) and company meetings?

If so, why do you think it was difficult?

Did you find the schedule for deliverables easy to follow?

Did you miss any deadlines (individual, team, company)?

What part of the project was the most difficult to schedule/organise between yourself and Durham?

Module Content

Did you find the lecture materials helped you during the project?

If so, which materials did you find the most helpful/unhelpful?

Can you suggest topics that might have helped you more?

Did the guest lectures from companies help you cope with some aspects of the project?

Module Support

Did you find it easy to get help when you needed it?

Do you think you were given good support throughout the project?

What kind of support would you have preferred?

Is there anything that the lecturers could have provided before the project that would have been helpful e.g. examples, invited talks on different subjects?

Project Process

What part of the project was the most difficult for you as an individual, team, company?

Why do you think it was it difficult?

Did you stick to your designated role?

Did the team/company fulfil the terms agreed in the original contract?

How did you deal with disagreements in the company, local team?

Did your team structure work well?

Assessment

How was your experience of the peer review process?

Did the Company peer review go well?

Did you think the local peer assessment process go well?

If there were issues (facilitator should ask), how did you resolve these?

What was the most difficult/easiest assignment during the project?

Overall Experience

What was the best thing about the project?

What was the worst thing about the project? (Individual views, team view)

If you had to do it all again, what would you do differently?

What particular skills do you think you have learned/improved on because of this project?

Appendix D

SEEK Knowledge Area	Bloom's Level	Core Status Essential (E) or Desirable (D)	Method in Module - Lecture (L), Practiced (P)	Assessed Y/N
Computing Essentials:				
API design and Use	A	E	P	Y (as part of code)
Code reuse and Libraries	A	E	P	Y (as part of code)
OO Runtime issues	A	E	P	Y (as part of Design & Code)
Error Handling, Exception Handling	A	E	P	Y (as part of Design a& Code)
Construction methods for distributed software	A	E	L, P to a small extent.	N
Hardware/ Software Co-design	A	E	P	N
Test-first programming		D	L	N
Development Environments	A	E	P	N
GUI builders	A	E	P	N
Unit Test Tools	C	E	P	N
Maths & Engineering Fundamentals:				
Formulation of problems. Alternative solutions, Feasibility	C	E	L, P	Y
Value consideration throughout the Software Lifecycle	K	E	L to an extent	N
Generating System Objectives	C	E	L, P	N

SEEK Knowledge Area	Bloom's Level	Core Status Essential (E) or Desirable (D)	Method in Module - Lecture (L), Practiced (P)	Assessed Y/N
Professional Practice:				
Working in Teams	A	E	L, P	Y
Interacting with Stakeholders	C	E	L, P	Y
Dealing with ambiguity & uncertainty	K	E	L, P	Y
Communication Skills: Reading & summarising reading (code, documentation)	A	E	P	Y
Writing assignments, reports, evaluations and justifications	A	E	L, P	Y
Team & Group Communication	A	E	L. P	Y
Presentation Skills	A	E	P	Y
Code of Ethics and Professional Conduct	C	E	L, (to an extent, 1st year module)P	Y
Nature & Role of Software Engineering Standards	K	E	P	N (1st year module)

SEEK Knowledge Area	Bloom's Level	Core Status Essential (E) or Desirable (D)	Method in Module - Lecture (L), Practiced (P)	Assessed Y/N
Software Modelling and Analysis				
Modelling Principles - decomposition etc.	A	E	L, P	Y
Information Modelling (Class Diagrams)	A	E	L, P	Y
Behavioural Modelling (Use Case)	A	E	L, P	Y
Architectural Modelling	C	E	L, P	Y
Functional Modelling (Component Diagrams)	C	E	L, P	Y
Traceability	C	E	L, P	Y
Analysing Quality - Safety, Security	C	E	L, P	Y
Definition of Requirements	C	E	L, P	Y
Requirements Process	C	E	L, P	Y
Layers of Requirements	C	E	L	N
Requirements Characteristics	C	E	L	N
Requirements Management	C	E	L, P	Y
Relating Requirements to System		D	L, P	N

Engineering and Human Centred Design				
Elicitation Sources	C	E	L, P	Y
Elicitation Techniques	C	E	L, P	Y
Requirements Document Basics	K	E	L, P	Y
Software Requirements Specification	A	E	L, P	Y
Specification Languages (UML)	K	E	L, P	Y
Prototyping	K	E	L, P	Y
Acceptance Test Design	C	E	L, P	Y

SEEK Knowledge Area	Bloom's Level	Core Status Essential (E) or Desirable (D)	Method in Module - Lecture (L), Practiced (P)	Assessed Y/N
Software Quality				
Definition of quality	K	E	L	N
Roles of People, Process Methods, Tools & Techniques	A	E	L, P	N
Quality Planning	A	E	L	N
Assessment of product quality attributes	C	E	L, P (to a limited extent)	N
Software Management				
General Project Management	K	E	L, P	Y
Project Management Roles	K	E	L, P	Y
Evaluation and Planning	C	E	L, P	Y
Work Breakdown Structure	A	E	L	N
Task Scheduling	A	E	L	Y (limited extent)
Organisational Structures	K	E	L, P	Y
Meeting Management	A	E	L, P	Y
Building & Motivating Teams	A	E	L, P	Y
Conflict Resolution	A	E	L, P	Y
Change Control	K	E	L, P	N
Monitoring &	C	E	L, P	Y

Reporting				
Maintenance Issues	K	E	L	N

SEEK Knowledge Area	Bloom's Level	Core Status Essential (E) or Desirable (D)	Method in Module - Lecture (L), Practiced (P)	Assessed Y/N
Software Design				
Definition of Design	C	E	L	N
Context of Design within lifecycles	K	E	L	N
Interaction between design & requirements	K	E	L, P	Y
Design for quality attributes	C	E	L, P	Y
OO design	C, A	E	L, P	Y
Architectural Styles	C	E	L	Y (Limited)
Architectural Design	A	E	L	Y
Requirements Traceability in architecture	K	E	L, P	Y
Architectural Notations	C	E	L, P	Y
Design notations	C	E	L	N

SEEK Knowledge Area	Bloom's Level	Core Status Essential (E) or Desirable (D)	Method in Module - Lecture (L), Practiced (P)	Assessed Y/N
Software Verification & Validation				
Reviews	A	E	L, P	N
Walkthroughs	A	E	L, P	N
Inspections	A	E	L	N
Unit Testing	A	E	L, P	Y
Exception Handling	A	E	L, P	Y
Black Box Techniques	A	E	L, P	Y
Integration Testing	C	E	L, P	Y
Test Cases	A	E	L, P	Y
System Acceptance Testing	A	E	L, P	Y (limited)
Regression Testing	C	E	L	N
Analysing Failure Reports	C	E	L	N
Defect Analysis	K	E	L	N
Problem Tracking	C	E	L, P	N

SEEK Knowledge Area	Bloom's Level	Core Status Essential (E) or Desirable (D)	Method in Module - Lecture (L), Practiced (P)	Assessed Y/N
Software Evolution				
Basic Concepts of Evolution and Maintenance	K	E	L	N
Software Process Reengineering	K	E	L	N
Software Process				
Themes & Terminology	K	E	L, P	N
Modelling and Specification of Software Processes	C	E	L, P	Y
Lifecycle Models	C	E	L, P	Y

Appendix E

Level	Primary Strength	Secondary Strength	Weakness	I have not practised this skill yet
Responding to email				
Suggesting ideas to the team				
Completing Tasks Efficiently				
Helping the team to function (take notes, schedule meetings, book rooms, organise task lists, keep everyone aware of deadlines).				
Planning & Organisation				
Design of System				
Completing Documentation & Reports				
Organising meetings				
Leadership & Direction				
Presenting ideas to customer, team, module leaders				
Programming				
Graphical Design – GUI Design				
Poster Design				
Testing				
Commenting Code				
Conflict Resolution				
Professional Behaviour				