DURHAM UNIVERSITY

A Longitudinal Evaluation of the Impact of a Problem-Based Learning Approach to the Teaching of Software Development in Higher Education

A Thesis Submitted for the Degree of Doctor of Education from the School of Education, Durham University.

by

James Doody

December 2009

Table of Contents

APPENDIX L - ANALYSIS OF LEARNING SELF-REGULATION, PROGRAMMING SELF-EFFICACY, APPROACHES TO LEARNING, AND PREFERENCES FOR TYPES OF TEACHING SCORES L-1

List of Tables

List of Figures

Abstract

First year students on Computing courses at tertiary level find Software Development difficult: learner outcomes are poor, with high failure rates and low learner retention. A number of research studies have shown that novice programmers have low intrinsic motivation and low programming self-efficacy. One of the other possible explanations for the difficulties many learners have with Software Development is that it may be a Threshold Concept in Computing. The literature suggests that Problem-Based Learning (PBL) can improve the teaching of difficult concepts, and it has been promoted by professional and funding bodies as a teaching strategy that can improve learner outcomes and bring about positive changes in learner behaviour. The main aim of this research study was to establish the impact on learner outcomes and behaviour of a Hybrid PBL approach used in the teaching of an introductory Software Development module at an Irish tertiary level institution. Learners on the Software Development module are characterised by low prior attainment in State college entry examinations, and the majority are from low income socio-economic backgrounds. Learner outcomes and behaviours were investigated over four cohorts of learners using a large range of data sources. A randomised controlled experimental design was used to measure changes in attainment, programming self-efficacy, motivation, approaches to study and preferences for types of teaching. Questionnaires, data mining of learner activity and attendance logs were used to provide additional information about learner behaviour, and further analysis was undertaken using qualitative techniques such as classroom observations and interviews. Both qualitative and quantitative measures were used to confirm, cross-validate and corroborate findings. The study made significant discoveries about the strengths and limitations of the Problem-Based Learning approach in the teaching of Software Development to low attainment learners. The implications for instructional practice and for educational theory and research are discussed and a number of recommendations are made.

Keywords: Problem-Based Learning, Software Development, Computer Programming, Curriculum, Attainment, Programming Self-Efficacy, Motivation, Approaches to Studying, Teaching, Learning.

Declaration

This thesis is my own work and has not been offered previously in candidature at this or any other university.

Statement of Copyright

Dedications

For my wife Mairéad Creed a truly exceptional woman and my children James, Fionn, and Saoirse.

In memory of my late Mother Brenda to whom I owe everything.

Acknowledgements

I wish to thank Professor Steve Higgins for his invaluable comments on the penultimate draft of the thesis and my supervisor Dr Julie Rattray for her support, and for taking a genuine interest in my studies and welfare. Finally and above all I'd like to thank Mairead Creed for all her substantial help and direction without which I would not have been able to complete this Ed.D.

Chapter 1 - Introduction and Rationale

> "How can we know the dancer from the dance?"
> (From the poem 'Among School Children' by William B. Yeats, 1928).

1.1. Introduction

The production of defect-free quality software is essential for the correct operation of many critical systems such as auto-pilot systems, nuclear power station control systems and intensive care systems. The demand for software is growing and it has become ubiquitous, controlling devices as diverse as mobile phones and washing machines. However, there are many problems with the production of software, in particular that it is often poorly written and faulty. More time is spent fixing errors in existing software than writing new code. The economic cost of software failure is counted in billions: in the U.S. alone, software bugs cost the economy an estimated $59.5 billion annually (Newman, 2002). There are many causes of software failure and key among them are the deficits in the education and training its creators received. Software is not a mass-produced product: it is handmade, crafted by individuals. Most of these individuals are educated as Software Developers in universities and other higher education institutes and they require mastery of a diverse range of skills to become competent programmers (Lohr, 2001).

Some educationalists suggest that using a Problem-Based Learning (PBL) approach to teach Software Development may improve the education learners receive. Newman (2004a, p. 5) states that Problem-Based Learning (PBL) "represents a major development and change in educational practice that continues to have a large impact across subjects and disciplines worldwide. PBL is promoted by professional and funding bodies [such as Ireland's Higher Education Authority's Strategic Initiatives Programme] as an appropriate strategy for education and increasingly as a method of choice". This is a view shared by many other researchers (Barrett, Mac Labhrainn & Fallon, 2005). Newman (2004a, p. 5) also states that "[t]he claims made for PBL would, if substantiated, represent an important improvement in outcomes from higher education. Thus it is of considerable importance that questions about what forms of PBL produce which outcomes for which students in what circumstances are rigorously investigated".

This study attempts to evaluate the impact on learner outcomes of a Hybrid PBL approach used in the teaching of an introductory Software Development (computer programming) module at an Irish third-level institution. This institution will be referred to as Anon College throughout the thesis.

1.2. Context of the Study

Ireland is one of the biggest producers of software in the world (Enterprise Ireland, 2004), and the Irish Government views software production as an environmentally friendly industry that is central to economic development and prosperity (Kawasaki & Williams, 2008). Career prospects for graduates are good and jobs in the software industry are well paid; therefore many Governmental and industry bodies consider it vital that there is a supply of Computing graduates to meet the growing demand for software and to address the shortage of software developers (Expert Group on Future Skills Needs, 2008). However, the low take up by school leavers of offers of places on tertiary level Computing courses and the high dropout and failure rates on those courses is a cause for particular concern (Radio Telefís Éireann, 2005; Skelly, 2006). In recent years, the number of second level students choosing Computing at third level generally has been falling (Donnelly, 2008; Donnelly & Walshe, 2008; Radio Telefís Éireann, 2005; Skelly, 2006). This has led to the entry into first year Computing at Anon College of low attainment learners. This in turn has exacerbated the problem of poor student retention in first year, with the Software Development module having particularly high failure rates.

At Anon College it was considered that if a new way of teaching the Software Development module was introduced, the high failure rates in that subject could be redressed and first year retention rates ultimately improved. This study examines whether PBL can help to address the retention problem, and whether the PBL approach provides any other benefits to learners.

1.3. Research Problem

It is well accepted within the computer science community that first year students find Software Development difficult (Dijkstra, 1989; Jackson, 2003; Jenkins, 2002). Failure rates are high and learner retention is low (Bennedsen & Caspersen, 2007). Many learners have low intrinsic motivation (Mamone, 1992). Many students show high reproduction orientation in their approaches to studying (Jenkins, 2001). Novices' programming self-efficacy levels are low (Wiedenbeck, LaBelle & Kain, 2004); and improvements need to be made in the way that Software Development is taught (Fincher, 1999b; Fincher *et al.*, 2005; Jenkins, 2002). Recent educational research may help provide some solutions to these problems. A number of research papers have identified that Software Development (Java programming) is a Threshold Concept in Computing (Boustedt *et al.*, 2007; Eckerdal *et al.*, 2006). The literature suggests that Problem-Based Learning can improve the teaching of difficult concepts (Ayres, 2002; Hmelo-Silver, 2004; O'Kelly, 2005) and bring about improvements in learner behaviour (Dolmans & Schmidt, 2006; Richardson, 2005; Schmidt, Loyens, van Gog & Paas, 2007). This study examines PBL classes to see if they can improve first year learners' acquisition of Threshold Concepts in Computing and modify their behaviours. If improvements could be made in learners' acquisition of the key Threshold Concepts in Computing then this would be of great benefit to learners and the wider Computing community.

1.3.1. Aims and Objectives of the Research

The overall objective was to investigate whether PBL is a more effective teaching method for Software Development than the traditional approach, both in terms of student attainment, approaches to learning and motivation. Hence the aim of the research was to determine whether using a Problem-Based Learning approach instead of conventional lectures:

1. improves learner attainment in the subject;
2. improves learner motivation to learn the subject;
3. improves learner Software Development self-efficacy;
4. changes learners' approaches to studying the subject (towards a deep approach);

5. changes learners' preferences for different types of teaching (towards teaching that supports understanding).

1.3.2. The Significance of this Study and Potential Impact of the Research

Most studies of PBL have been based on high-attainment learners, particularly in the field of medicine (Albanese & Mitchell, 1993; Colliver, 2000). There have been few studies of low-attainment learners and few studies in the field of Software Development. As far as this author is aware, this research is the first of its kind in Ireland to investigate the effectiveness of a PBL model for first-year students with low attainment status in an Irish college. While there have been a number of excellent Irish PBL case studies (e.g. Barrett *et al.*, 2005), none has provided a rigorous quantitative statistical analysis of students' attainment.

Dolmans, De Grave, Wolfhagen, and van der Vleuten (2005, p. 739) state that "[r]esearch is needed [to provide] a clearer understanding of how PBL does or does not work and under which circumstances [and to provide] us with guidelines on how to deal with problems encountered in PBL practice". They go on to call for research that "makes use of mixed methods, triangulates multiple sources and types of data and does not rely on a single method […], or a single source of data". This study attempts to meet both of these demands. It seeks to link educational theory to practice by using a mixed methods approach, by triangulating multiple sources and types of data, and by investigating the effectiveness or otherwise of PBL in facilitating the acquisition of key Threshold Concepts in computer programming. It is hoped that findings from this study will assist educational researchers and practitioners alike by providing empirical evidence of the impact of PBL on a range of outcomes and by providing a set of practical recommendations and guidelines for curriculum developers and teaching practitioners that will help to improve the experiences and behaviours of learners on introductory programming courses.

1.4. Research Questions

This thesis asks five main Research questions and tests five related hypotheses as outlined in Table 1-1 below:

Table 1-1: Initial Research Questions

Research questions	Hypotheses
(1.a) What are the effects of using a PBL model on learner attainment in exams on a first year programming module? (1.b) What are the effects of using a PBL model on learner attainment in continuous assessment on a first year programming module?	(1.a) Learners in the PBL group will score higher in exams than those in the control group. (1.b) Learners in the PBL group will score higher in continuous assessment than those in the control group.
(2) What are the effects of using a PBL model on learner self-regulation?	(2) Learners who complete the PBL course will have a higher degree of intrinsic motivation than those in the control group.
(3) What are the effects of using a PBL model on learners' programming self-efficacy?	(3) Learners in the PBL group will show a higher degree of programming self-efficacy than those in the control group.
(4) What are the effects of using a PBL model on students' approaches to learning and on general learner engagement?	(4) Learners in the PBL group will show higher scores on meaning orientation and lower scores on reproduction orientation than those in the control group.
(5) What are the effects of using a PBL model on learner preferences for different types of course and teaching?	(5) Learners in the PBL group will show a greater preference for courses and teaching that supports deep learning (as opposed to surface learning) than those in the control group.

Hypotheses 1a and 1b were tested over four cohorts of learners, using statistical analysis of variance of learner attainment data. Hypothesis 2 was tested over two cohorts of learners, using a statistical analysis of learner responses on the Learning Self-Regulation Questionnaire (SRQ-L) (Williams & Deci, 2007a). Hypothesis 3 was tested over two cohorts of learners, using a statistical analysis of learner responses on the Programming Self-Efficacy instrument (PSE) (Ramalingam & Wiedenbeck, 1998). Hypotheses 4 & 5 were measured over one cohort of learners, using a statistical analysis of learner responses on parts B and C of the Approaches and Study Skills Inventory for Students (ASSIST) (Entwistle, 1997; Entwistle, McCune & Tait, 2006)[1]. Learner motivation, engagement and approaches to studying were also examined over four cohorts of learners using qualitative techniques such as observations, field notes and interviews.

In addition to the five hypotheses, the study allowed some tentative conclusions to be drawn about a number of related issues. These concerned what higher education subjects may lend themselves to using Problem-Based Learning (e.g. Computer Science versus English); what aspects of the Computing curriculum may be most suitable for applying a Problem-Based Learning approach; whether PBL classes can improve first year learners' acquisition of Threshold Concepts in Computing; when in a learner's college lifetime a Problem-Based Learning approach might be most effective (e.g. with first-years or with final year learners); and which sets of learners may benefit most from using a Problem-Based Learning approach (low attainment or high attainment)?

1.5. Ethics

The study was conducted ethically from a professional, academic and moral standpoint. Pole and Morrison (2003) make the point that researchers need "to recognise that ethical issues will permeate every stage in the research process". From the outset of this research, ethical issues were considered and the research methodology was shaped to incorporate these concerns. To protect participants' rights and welfare, the study was bound by the Republic of Ireland's "Data Protection (Amendment) Act (2003)", which contains strict rules about how data must be stored, who can access it, and how it can be processed (Clark, 1996).

[1] The Approaches and Study Skills Inventory for Students (ASSIST) can be downloaded from the Enhancing Learning and Teaching project website at http://www.etl.tla.ed.ac.uk/publications.html#measurement

The research builds upon a pilot study done by this author in 2006, which received Durham University Ethics Committee approval. The full study received Ethics Committee approval from the author's own college in 2007 and from Durham University School of Education Ethics Committee in 2008. The research design was informed by the research ethical guidelines issued by the Sociological Association of Ireland (SAI, 2008) and the British Educational Research Association (BERA, 2004): these guidelines state in particular that before any surveys, observations or interviews are completed, the participants will receive a consent form outlining the purpose of the research, guaranteeing their anonymity, and specifying that their participation/non-participation will not be discussed with their instructors or otherwise affect their standing in the college. Appendices A1, A2 and A3 contain the Ethics approval forms. Appendix B contains the consent forms for participants, an audit trail and procedures for management of data collection.

1.6. Structure of the Thesis

This thesis starts with a review of the literature on a number of interrelated areas: the pedagogy of Software Development, Threshold Concepts, Problem-Based Learning student approaches to learning, learner motivation and self-efficacy. Following the literature review, in chapter three, details of the hybrid PBL model used at Anon College and its implementation are described. The research methodologies used are set out. In chapters four and five the quantitative and qualitative findings are presented. Finally, in chapter 6 a discussion of the findings, including the implications for instructional practice and for educational theory and research, is undertaken.

Chapter 2 - Review of Related Literature

2.1. Introduction

This literature review provides the theoretical foundation for the present study and has five major sections. The first is a review of the literature on the pedagogy of Software Development, focusing in particular on the problems associated with teaching computer programming to novices. The second is an examination of Threshold Concepts, both as a way of characterising particular concepts that are troublesome for learners and as a framework for examining the wider Computing curriculum. It includes a review of Threshold Concepts in Computing. Consideration is given to how Threshold Concepts might be used to organise and focus the educational process and the relationship between Threshold Concepts and PBL is examined. The third section provides a critical review of the literature on the effectiveness of PBL, including an examination of the literature on approaches to learning in a PBL context. The chapter continues with a discussion of two key theories of motivation: goal theory and self determination theory, which were selected to provide a framework for understanding some of the psychological factors that underpin and explain learner behaviour. The chapter ends with a consideration of how these different areas of literature converge in the context of using PBL to teach Software Development, and from this discussion a number of research questions emerge.

2.2. The Literature on the Pedagogy of Software Development

This section provides an overview of the research into the learning and teaching of computer programming, identifying several significant issues. It focuses in particular on novice programmers, exploring the difficult nature of the tasks they are asked to master, the nature of the knowledge they must understand, the strategies they need to apply that knowledge, the mental models they must build, their capabilities and typical problems, and their characteristic behaviours.

This is followed by an exploration of the different types of novices, their motivations and possible predictors of their success at programming courses. The levels of achievement of students on introductory programming courses are examined. Factors relating to teaching and course design are explored; a number of studies on the use of Problem-Based Learning to teach programming are scrutinized. And finally, issues relating to the types of programming languages taught are discussed.

2.2.1. Overview and Scope

Studies of programming generally divide into two main categories. The first is those that focus on experienced or professional programmers, often working in teams, and how they develop large and complex commercial software projects effectively (Boehm, 1981; Brooks, 1995; Humphrey, 1999). The second category of studies focuses on novice programmers and the initial development of an individual's programming skills, in which learning is addressed from a psychological and educational perspective. This second category is the focus of this review. Also, while some comparisons between procedural and object-oriented programming languages will be made, the main focus will be on object-oriented languages such as Java. Other programming paradigms such as functional or logic programming will not be covered, as these paradigms are seldom taught to novices, are often not taught on general Computing courses, and are rarely used in commercial Software Development.

Weinberg (1971) and Sackman (1970) have identified programming as an area of psychological interest. Weinberg (1971) was one of the first researchers to address programming as an individual and team effort, focusing on programming as a people oriented rather than a technologically focused task. Hoc, Green, Samurcay, & Gilmore (1990) in their book continue this focus on people, deploying more recent developments in psychology to study the behaviour of programmers. Soloway and Spohrer's (1989) book explicitly focuses on novice programmers and the types of programming errors that they make. Robins *et al.* (2003) and Winslow (1996) have also conducted extensive reviews of the literature relating to the educational study of programming and identified several areas of research including programming knowledge versus programming strategies; the ability to comprehend versus the ability to generate code; the differential learning effects of different programming languages; the programming behaviour of expert versus novice programmers; and the characteristics of effective novices. Using these sources and others, these research areas and their findings are summarised, critiqued and the implications for practice discussed in the following sections.

2.2.2. Introduction

Both in Ireland and internationally there is evidence of high failure and dropout rates and low retention rates in introductory programming courses at tertiary level (Bennedsen & Caspersen, 2007), particularly among first year students (Jackson, 2003). Computer Science courses have the highest university dropout rates in the UK, with one in 10 undergraduates not continuing into a second year of study (Williams, 2007). These figures are in stark contrast to medicine and dentistry, which have the highest retention rate, at 98 per cent (UK National Audit Office, 2007). PBL is well established in the teaching of medicine and dentistry, but not in Software Development. This suggests a possible link between the use of PBL and learner retention rates that needs to be investigated further. Exactly why Software Development should pose these difficulties for learners is an active subject of cross-disciplinary research involving Psychology, Education and Computer Science researchers (Hoc *et al.*, 1990; Khalife, 2006). The research has focused on the learning process in introductory programming courses, and considerable work has been done to identify the difficulties encountered by learners (Barros, Estevens, Dias, Pais, & Soeiro, 2003; Connolly, Murphy, & Moore, 2008; Fincher *et al.*, 2005; Simon *et al.*, 2006).

Students approach Computing degrees with a variety of motivations (Jenkins, 2001). However, almost all students are motivated to succeed. Jenkins (2002, p. 54) makes the point succinctly, saying that “they do not fail on purpose”. Therefore the difficulty may lie in the types of learning tasks novices are asked to perform, or in the teaching methods employed on introductory programming courses. Since the main interest of this thesis lies in novices and the early stages of learning, novices are now examined in detail, focusing on the difficulty of the task they must master, their mental models, their behaviours, and their capabilities.

2.2.3. The Difficulty of Learning to Program

Learning to program is a difficult task (Dijkstra, 1989; du Boulay, 1989; Jackson, 2003; Jenkins, 2002) and many lecturers find it a very difficult skill to teach (van

Roy & Haridi, 2004). Programming is a new subject for the majority of students who take first-year programming courses, and this might be part of the problem. Dijkstra (1989), in his classic essay on the teaching of computer programming, titled “On the Cruelty of Really Teaching Computing Science”, argues that programming is a "radical novelty" that the existing higher education learning system cannot cope with. According to Dijkstra (1989, p. 1398), at the heart of the problem is that “radical novelties are so disturbing that they tend to be suppressed or ignored, to the extent that even the possibility of their existence in general is more often denied than admitted”. Two particular features of computer programming are, as Perkins *et al.* (1988) say, both "problem-solving intensive" and "precision intensive". To be precise, programming requires both a significant amount of effort in several skill areas to produce a very modest return, and the modest success that can be achieved by a novice programmer requires a very high level of precision, and certainly a much higher level than most other academic subjects (Dijkstra, 1989; Jenkins, 2002). Dijkstra (1989, p. 1399) notes that the "smallest possible perturbation" in a program, for example, “changes of a single bit - can have the most drastic consequences” (*ibid*, p. 1399), rendering a program totally worthless. A single missing semi- colon in a thousand lines of code can be, as Jenkins (2002, p. 56) puts it, “the difference between glorious success and ignominious failure…This is precision indeed”.

Outlining what is involved in learning to program, du Boulay (1989) describes five overlapping domains that are each potential sources of difficulty and which must be mastered. These are: (1) general orientation, what programs are for and what can be done with them; (2) the notional machine, a model of the computer as it relates to executing programs; (3) notation, the syntax and semantics of a particular programming language; (4) structures, that is, the design schema and plans; and (5) pragmatics, which are skills such as testing and debugging. Du Boulay goes on to clarify the problems faced by learners:

> None of these issues are entirely separable from the others, and much of the ‘shock’ [. . .] of the first few encounters between the learner and the system are compounded by the student’s attempt to deal with all these different kinds of difficulty at once.
>
> Du Boulay (*ibid*, p. 284)

Rogalski and Samurçay sum up the task as follows:

> Acquiring and developing knowledge about programming is a highly complex process. It involves a variety of cognitive activities, and mental representations related to program design, program understanding, modifying, debugging (and documenting). Even at the level of computer literacy, it requires construction of conceptual knowledge, and the structuring of basic operations (such as loops, conditional statements, etc.) into schemas and plans. [And] it requires developing strategies flexible enough to derive benefits from programming aids (programming environment, programming methods).
>
> (Rogalski & Samurçay, 1990, p. 170)

Green (1990, p. 117) suggests that programming is best regarded not as a “transcription from an internally held representation”, but as an exploratory process where programs are created “opportunistically and incrementally”. This view is supported by Visser (1990) and by Davies (1993, p. 265), who argues that “emerging models of programming behaviour suggest an incremental problem-solving process where strategy is determined by localized problem-solving episodes and frequent problem re-evaluation”.

2.2.4. Programming Knowledge and Strategies

This section focuses on the nature of the programming knowledge novices must master and the characteristics of the strategies they must employ to utilise that knowledge. The majority of studies of programming have focused on the content and structure of programming knowledge, particularly upon the declarative aspects of programmers' knowledge (R. E. Brooks, 1990; Détienne & Soloway, 1990; Guindon, 1990; Rist, 1990; Robertson & Yu, 1990; Visser, 1990). Davies (1993, p. 237) states that “this literature has sought to describe the nature of programming knowledge structures and their organization”. Robins *et al.* (2003, p. 140) support this, saying that “[t]ypical introductory programming textbooks devote most of their content to presenting knowledge about a particular language […], and in our experience typical introductory programming courses are also ‘knowledge driven’’’. However, Davies (1993, p. 237) points out “that one major limitation of many of these knowledge-based theories is that they often fail to consider the way in which knowledge is used or applied”.

One kind of knowledge representation identified as fundamental is the schema or plan (Abelson & Sussman, 1996). Ormerod (1990) states that “[a] schema [...] consists of a set of propositions that are organised by their semantic content”. This view of a schema as structured pieces of related knowledge is common. Nonetheless, there is considerable flexibility and overlap in the interpretation of the terms schema and plan. Rist (1995) claims that “[t]here is considerable evidence in the empirical study of programming that the plan is the basic cognitive chunk used in program design and understanding. Exactly what is meant by a program plan, however, has varied considerably between authors”. Robins *et al.* (2003, p. 140) say the term ‘‘plan is often used to emphasize an ‘action oriented’ rather that [*sic*] static interpretation. In other words, the term ‘schema’ implies a ‘program as text’ perspective, while the term ‘plan’ implies a ‘programming as activity’ perspective”. This view of plans and schemas is supported by other researchers (Rogalski & Samurçay, 1990).

Davies (1993) contends that the strategic aspects of programming skills are less well represented in the literature. He suggests that research is needed to determine “the relationship between the development of structured representations of programming knowledge and the adoption of specific forms of strategy” (*ibid*, p. 238), and he identifies as significant strategies relating to the general problem domain, the specific programming task, the programming language and the programming tools used. Davies (1993, p. 237) states that this area of research has been “directed towards an analysis of the strategies commonly employed by programmers in the generation and comprehension of programs”.

2.2.4.1. Strategies and Models of Program Comprehension and Program Generation

There are six major models of program comprehension: the Letovsky (1986) model; the Shneiderman and Mayer (1979) model; the Brooks (1983) model; Soloway, Adelson and Ehrlich's (1988) top-down model; Pennington's (1987a, 1987b) bottom-up model; and the integrated meta-model of von Mayrhauser and Vans (1995b). While these general models can be used to promote a complete understanding of a piece of code, the literature suggests that it is doubtful that programmers rely on any one of these models and related strategies exclusively, rather, they subconsciously adopt one of these to be their predominant strategy, based on their knowledge of the domain under study (Shaft & Vessey, 1998; von Mayrhauser, Vans & Howe, 1997), and switch between strategies as cues become available to them (von Mayrhauser & Vans, 1995b). Von Mayrhauser and Vans (1995a) identify a number of open research questions that relate to the scalability of existing experimental results due to the small programs used, and the validity and credibility of results which are based on experimental procedures. Also Wiedenbeck *et al.* (1999) note that the comprehension models used by novice programmers can be influenced by different task requirements, for example, whether they are coding using an object-oriented or procedural programming language.

Of the models mentioned, the Brooks (1983) model has the strongest support. Davies (1993) reviewed a range of studies that support Brooks' model. The Brooks model is set in the context of various knowledge domains such as the original problem domain, which is transformed and represented as values and structures in intermediate domains, and finally instantiated and coded in the data structures and algorithms of a program in the programming domain. The same set of domains has been identified by Pennington (1987a, 1987b), based on the text comprehension model developed by van Dijk and Kintsch (1983). Brooks (1983) claims that his model is able to account for observed variation in comprehension performance arising from such factors as the nature of the problem domain, variations in the program text, the effects of different comprehension tasks and the effects of individual differences.

Rist (1995) presents a comprehensive model of program generation. Knowledge is represented using nodes in a programmer's internal memory, including working, episodic, semantic, and procedural memory, or recorded externally in written notes, books, internet, the program specification, and the code itself. A node encodes an "action" that may range from a line of code to one or more code routines of arbitrary size. A program is built by starting with a search cue and retrieving any matching nodes from memory. Nodes can contain cues, so cues within the newly linked node are then expanded and linked in the same way. Linked systems of code that produce a specific output are called plans, and common and useful plans are assumed to be stored by experts in schema-like knowledge structures (Rist, 1986a, 1986b, 1989, 1990). Using these underlying knowledge representations, a number of different code generation strategies can be implemented, including both expert and novice strategies. Rist (1995) remarks that a realistic code generation process will involve "the interaction between a search strategy and opportunistic design, plan creation and retrieval, working memory limitations, and the structure of the specification and the program" (*ibid*, p. 508).

Studies and models of comprehension are more numerous than studies and models of generation. A possible reason for this is, as Robins *et al.* (2003, p. 144) say, "because comprehension is a more constrained task and … is therefore easier to interpret and describe". Robins *et al.* (2003, p. 144) go on to say that "clearly [comprehension and generation] are related, not least because during generation the development, debugging (and in the long term maintenance) of code necessarily involves reviewing and understanding it. Although we might therefore expect that these abilities will always be highly correlated, the situation may in fact be more complex". This view is supported by Winslow (1996), who in his review of psychological studies of programming pedagogy states that "studies have shown that there is very little correspondence between the ability to write a program and the ability to read one. Both need to be taught along with some basic test and debugging strategies" (*ibid*., p. 21).

2.2.5. Mental Models and Processes

A mental model is a person's internal model of a system's properties and behaviour (Johnson-Laird, 1983). The use of a mental model makes it possible to predict the system's responses to various actions and thus makes it possible for an individual to select the best possible action (Kieras & Bovair, 1984). The consequence of this is that an incorrect mental model can lead to incorrect actions. So a faulty mental model of how various computer programming constructs work will cause problems when learners try to write programmes.

Winslow (1996, p. 21) stresses that mental "[m]odels are crucial to building understanding. Models of control, data structures and data representation, program design and problem domain are all important. If the instructor omits them, the students will make up their own models of dubious quality". Programs are written for a purpose, with respect to some problem, or specification. Clearly an understanding and associated mental model of this problem domain must precede any attempt to write an appropriate program (Brooks, 1983, 1999; Davies, 1993; Rist, 1995; Spohrer, Soloway & Pope, 1989). This suggests the need to use relevant problems directly in the teaching of programming languages, possibly imbedded in the Problem-Based Learning teaching method (Deek, Kimmel & McHugh, 1998).

Ben-Ari (2001) argues that the lack of mental models plays an important part in why students find it difficult to learn how to program. His argument is that, having no previous models to build on, programmers are forced to construct their own mental models from scratch. Wiedenbeck and Ramalingam (1999) investigated how novice programmers' mental models of their programs depended on whether a procedural or an object-oriented language was used. Similarly, Yehezkel *et al.* (2005) describe the importance of forming a mental model of a computer system in order to understand it, suggesting that visualization tools can enable the construction of a viable mental model.

Other important mental models can be identified. Many studies have noted the central role played by a model of (an abstraction of) the computer, often called a 'notional machine' (Cañas, Bajo & Gonzalvo, 1994; du Boulay, 1989; du Boulay, O'Shea & Monk, 1989; Hoc & Nguyen-Xuan, 1990; Mayer, 1989; Mendelsohn, Green & Brna, 1990). Du Boulay *et al.* (1989, p. 431) say that "the notional machine [is] an idealized, conceptual computer whose properties are implied by the constructs in the programming language employed". According to Robins *et al.* (2003, p. 149), "the notional machine is defined with respect to the language is an important point". For example, the notional machine underlying an object-oriented language like Java is very different from the machine underlying a logic programming language like Prolog. Robins *et al.* (*ibid*, p. 149) go on to state that "[t]he purpose of the notional machine is to provide a foundation for understanding the behaviour of running programs". Du Boulay *et al.* (1989) also add the requirement that the 'notional machine' be visible to the learner and simple in its construction and workings. Mayer (1989) showed that students supplied with a notional machine model were better at solving some kinds of problem than students without the model. Du Boulay explains the purpose of the 'notional machine' as follows:

> [T]o present the beginner with some model or description of the machine she or he is learning to operate via the given programming language. It is then possible to relate some of the troublesome hidden side-effects to events happening in the model, as it is these hidden, and visually unmarked, actions which often cause problems for beginners. However, inventing a consistent story that describes events at the right level of detail is not easy.
>
> (du Boulay, 1989, pp. 297-298).

The programmer must also develop a model of the written static programming code, and the dynamic executing program it will become. Du Boulay (1989, p. 285), states that a "running program is a kind of mechanism and it takes quite a long time to learn the relation between a program on the page and the mechanism it describes". Building the model is complicated by the fact that there are many different ways of viewing a program, such as linear order, control flow, data flow, and object based structure (Rist, 1995).

Robins *et al.* (2003, p. 150) add that "[c]omplicating this picture still further […] is the distinction between the model of the program as it was intended, and the model of the program as it actually is. Designs can be incorrect, unpredicted interactions can occur, bugs happen. Consequently, programmers are frequently faced with the need to understand a program that is running in an unexpected way". Perkins *et al.* (1989) suggest that for a learner to be able to build a model of the program and predict its behaviour, the learner must be able to mentally trace the flow of code by "taking the computer's point of view" Robins *et al.* (2003, p. 150) say that "[t]he process of building such a model (which itself supposes models of both the features of the language and the behaviour of the machine) is a central part of program comprehension, and of the planning, testing and debugging involved in program generation".

2.2.6. The Programming Capabilities and Behaviours of Novice Programmers

The distinction between a novice and an expert may become clearer by asking the question, what makes an expert programmer? Winslow (1996) reviewed a number of psychological studies of programming pedagogy that were undertaken between 1974 and 1994 and reported that it takes 10 years to turn a novice into an expert programmer, a view that is now generally accepted. Dreyfus *et al.* (2000) suggest five stages in this process: novice, advanced beginner, competence, proficiency, and expert. There are many studies of expert programmers, although Robins *et al.* (2003, p. 139) point out that “some are based on graduate students who are probably only competent or proficient”. Most studies of experts focus on the sophisticated knowledge representations and problem solving strategies that they employ (Détienne, 1990; Gilmore, 1990; Pennington, 1987b; Visser & Hoc, 1990). Von Mayrhauser and Vans (1994) reviewed a number of studies, most notably Guindon (1990), and state that experts have efficiently organised and specialised programming knowledge schemas that have associated testing and debugging strategies (Linn & Dalbey, 1989). Experts employ both general problem solving strategies and specialised strategies and are flexible in their approach to program comprehension and construction. This was demonstrated by Widowski and Eyferth (1986) who showed that in program comprehension experts demonstrate highly flexible strategies and are better able to recognise and respond to novel situations.

Novices do not have these strengths, and studies of novices have concluded that, as Robins *et al.* (2003, p. 140) state, they “[are] limited to surface and superficially organised knowledge, lack detailed mental models, fail to apply relevant knowledge, and approach programming ‘line by line’ rather than using meaningful program ‘chunks’ or structures.” A number of studies collected in Soloway and Spohrer (1989) and other studies reviewed by Winslow (1996) outline, as Robins *et al.* (2003, p. 140) put it, “deficits in novices’ understanding of various specific programming language constructs (such as variables, loops, arrays and recursion), shortcomings in their planning and testing of code, [and] how prior knowledge can be a source of errors”. Novices are also, as Wiedenbeck *et al.* state (1999, p. 278), “very local and concrete in their comprehension of programs”.

Robins *et al.* (2003, p. 151) state that “in contrast to experts, novices spend very little time planning. Novices also spend little time testing code, and tend to attempt small ‘local’ fixes rather than significantly reformulating programs (Linn & Dalbey, 1989). They are often poor at tracing the flow of control through multiple lines of code (Perkins *et al.*, 1989). Robins *et al.* (2003, p. 151) state that “[n]ovices can have a poor grasp of the basic sequential nature of program execution”. Du Boulay (1989, p. 294) adds that “[w]hat sometimes gets forgotten is that each instruction operates in the environment created by the previous instructions". Kurland, Pea, Clement and Mawby (1989) found that even after two years of study, novices' programming knowledge tends to be context specific rather than general. Some of these failings relate to aspects of knowledge, while others relate to strategies.

Novices' understanding and use of specific features of programming languages are also problematic. Programme variables cause particular problems with assignment and initialisation (du Boulay, 1989; Samurcay, 1989). Spohrer *et al.* (1989) found that errors associated with loops and conditionals were much more common than those associated with input, output, initialisation, update, syntax/block structure, and overall planning. That novices have particular difficulty with loops has been identified by a number of other researchers (Rogalski & Samurçay, 1990; Soloway, Bonar & Ehrlich, 1989). Du Boulay (1989) states that 'for' loops are problematic because novices often fail to understand that 'behind the scenes' the loop control variable is being updated. Du Boulay (*ibid*, p. 295) points out that "[t]his is another example of the ubiquitous problem of hidden, internal changes causing problems". Another language feature that novices find difficult is arrays, with errors such as confusing an array subscript with the value stored being common (du Boulay, 1989; Rogalski & Samurçay, 1990). Rogalski and Samurçay (1990) suggest that this difficultly might be caused by the array data structure itself rather than issues relating to processing array elements. Novices also have more difficulty understanding issues relating to flow of control than other kinds of processing (Rogalski & Samurçay, 1990).

While specific language related features cause problems, the main difficulty for novices is program design and planning. Robins *et al.* (2003, p. 153) state that "[t]he underlying cause of the problems faced by novices is their lack of […] programming specific knowledge and strategies […and] this lack manifests itself primarily as problems with basic planning and design". Spohrer and Soloway (1989) conducted a study of programming errors made by first year university students, and identified two "common perceptions" of errors:

> Our empirical study leads us to argue that (1) yes, a few bug types account for a large percentage of program bugs, and (2) no, misconceptions about language constructs do not seem to be as

> widespread or as troublesome as is generally believed. Rather, many bugs arise as a result of *plan composition problems* – difficulties in putting the pieces of the program together [. . .] and not as a result of *construct-based problems*, which are misconceptions about language constructs
>
> (Soloway & Spohrer, 1989, p. 401).

Spohrer *et al.* (1989) found that novices mix up plans and often omit part of the plan. This suggests that Robins *et al.* (2003, p. 153) are correct when they say that "basic program planning rather than specific language features is the main source of difficulty". This finding is supported by Winslow in his 1996 review of studies:

> [A] large number of studies conclude[d] that novice programmers know the syntax and semantics of individual statements, but they do not know how to combine these features into valid programs. Even when they know how to solve the problems by hand, they have trouble translating the hand solution into an equivalent computer program.
>
> (Winslow, 1996, p. 17).

Winslow suggests that the problem with novices is their lack of ability to create a program rather than any general lack in understanding the required underlying problem solving. Robins *et al.* (2003, p. 154) state that "the most basic manifestation of novices' lack of relevant knowledge and strategies is evident in problems with focal design". Rist (1995, p. 537) defines focus design as "when a problem is decomposed into the simplest and most basic action and object that defines the focus of the solution, and then the rest of the solution is built around the focus. Essentially, the focus is where you break out of theory into action, out of the abstract into the concrete level of design".

2.2.7. Different Kinds of Novice Programmers and Indicators of Success

Connolly *et al.* (2008) point out that for some Computing students, learning programming is intimidating, giving rise to a lack of confidence and anxiety. Barros *et al.* (2003) suggest that the high variability of students' backgrounds typically found in introductory programming courses creates additional difficulties in fostering motivation. Robins *et al.* (2003, p. 155) add that "[a] group of novices learning to program will typically contain a huge range of backgrounds, abilities, and levels of motivation, and also typically result in a huge range of unsuccessful to successful outcomes". This assertion suggests a need to identify what factors might be good indicators of success. A study by Evans and Simkin (1989) has shown that no demographic factor is a strong predictor of success in programming, and while there are a number of commercially developed programming aptitude tests which claim to test aptitude for computer programming, the evidence for their effectiveness is inconclusive (Mazlack, 1980). Alexander *et al*. (2003) carried out an analysis of students' success during the early part of their study of programming at university and found nothing in entry qualifications that indicated which students will be successful in the study of programming. On the other hand, Moran and Crowley (1979) analysed the relationship between entry qualifications and attainment of learners on courses at an Irish tertiary level colleges and found a tipping point in entry qualifications, below which learners do not make good progress and above which prior attainment is not an indicator of success.

Wiedenbeck *et al.* (2004) claim that the ability to form mental models is a predictor for Software Development course outcome. Nonetheless, Bishop-Clark (1995) found no clear trends emerging from a review of studies of the effects of cognitive style and personality on programming, saying that "the work relating cognitive styles and personality traits to computer programming has been both scattered and difficult to interpret" (*ibid*, p. 257). The author goes on to say that:

> [c]ognitive styles and personality traits […] have failed to consistently explain individual differences in achievement. In the majority of these studies, computer programming has been measured as a single activity. Computer programming has been described as an activity having separate and distinct phases: problem representation, program design, coding, and debugging. It may be that certain cognitive styles and personality dimensions affect some phases but not others.
>
> (*ibid*, p. 241)

Rountree *et al.* (2002) conducted a study of first year students on a computer programming course at the University of Otago, examining factors such as background, intended major and expected workload, and found that the most reliable predictor of success was the grade that the students themselves expected to achieve. The authors say (*ibid*, p. 124) that "[w]e believe that there is a […] problem in learning to program: many people would like to have the skill, but find the mental attitude required to gain it is hard to sustain. Our results suggest that a positive attitude is the most important factor”. It is estimated that between 25 and 80 percent of students choose not to major in Computing due to the difficulty they face in learning a programming language (Carter & Jenkins, 2002).

Rountree *et al.* (2002) also reported that students who feel confident about their learning perform best. Two similar studies at the University of Glasgow support these findings (Black, 2003; Roddan, 2002). Black (2003) found that students’ self-estimates correlate with exam performance, reporting a reasonably strong significant positive correlation of (0.586), and that a second significant factor was the level of a student’s academic integration into the Computing course. Black (2003, p. 14) reports that 36% of the variance in class test results was explained by academic integration, however social integration did not predict a significant amount of the variance. These results are in line with Tinto’s (1975) theory of integration. Nonetheless, caution must be exercised when interpreting these results as the questionnaire items used were completely exploratory and had not been used previously to test levels of academic and social integration.

Good indicators of success include measures of general intelligence, which have been shown to relate to success at learning to program (Mayer, 1989; Mayer, Dyck & Vilberg, 1989; Nickerson, 1982). Other than measures of intelligence, the best indicators of success appear to be self-predicted success, attitude, keenness and general academic motivation (Black, 2003; Roddan, 2002; Rountree *et al.*, 2002). However, this does not distinguish computer programming from other disciplines, and given the large effect sizes of these factors, they may mask more subtle, discipline-specific, indicators.

Cantwell-Wilson and Shrock (2001) undertook a wide-ranging study to determine factors that promote success in an introductory Computer Science course. Examining 105 participants, the authors explored twelve possible predictive factors including mathematical background, attribution for success/failure (luck, effort, difficulty of task, and ability), domain specific self-efficacy, encouragement, comfort level in the course, work style preference, previous programming experience, previous non-programming computer experience, and gender. The study revealed three predictive factors in the following order of importance: firstly, 'comfort level' was based on students' perceptions of course/programming difficulty and level of anxiety; secondly mathematical background; and thirdly, 'attributions' which were based on students' beliefs about their reasons for success or failure. Comfort level was found to be the most significant positive predictor of success, with mathematical background second, and attribution of success to luck, which correlated negatively with success, was the third in order of significance. That mathematical background is a good predictor of success has been found by other researchers (Konvalina, Wileman & Stephens, 1983). The Cantwell-Wilson and Shrock (2001) study also found some minor factors relating to different types of previous computer experiences. Previous formal training in programming had a positive influence and computer game playing had a negative influence on class grade. Previous experience was also investigated by Jenkins (2002) who found that students who have had some training in programming before beginning programming courses have a higher probability of success, and this finding is supported by other studies (Hagan & Markham, 2000).

Fincher *et al.* (2005; 2006) describe a multi-national, multi-institutional study that investigated predictors of success in a first programming course. Participants were drawn from eleven institutions, and four different diagnostic tasks were used in the study: a spatial visualisation task (a standard paper folding test); a behavioural task used to assess ability to design and sketch a simple map; a second behavioural task used to assess the ability to articulate a search strategy; and an attitudinal task focusing on approaches to learning and studying. The authors reported a significant relationship between novice programmers' map-drawing styles (landmark, route or survey) and success in a first programming course at tertiary level. The authors also reported a significant correlation between high scores and increasing measures of richness of articulation of a search strategy. The results indicate that a deep approach to learning was positively correlated with high scores for the course, while a surface approach was negatively correlated. However, there was only a small positive correlation between scores in the spatial visualisation (paper folding) task and programming marks. Fincher *et al.* (2005, p. 45) suggest that "components of 'IQ' other than spatial skills may account for most of the effect of IQ on programming success". A strong point of the Fincher *et al.* (2005; 2006) study is the large number of participants, with 177 participants from eleven institutions in three countries. The main limitations of the study arise from the use of multiple experimenters, and include issues with respect to the consistency of the application of the study protocol, and the consistency of coding, transcription and analysis.

Different kinds of characteristic behaviour are evident when observing novices in the process of writing programs. Perkins *et al.* (1989) distinguish between three main kinds of behaviour, what they call "stoppers", "movers", and "tinkerers". Stoppers, when confronted with a problem or a lack of a clear direction to proceed, simply stop. Perkins *et al.* (1989, p. 265) say that "[t]hey appear to abandon all hope of solving the problem on their own". Movers on the other hand are students who keep trying, experimenting, and modifying their code. Movers can use feedback about errors effectively, and have the potential to solve the current problem and progress. Perkins *et al.* (1989) call a form of excessive movers "tinkerers". These are students who are not able to trace their program, and make a large number of random changes to their code. Tinkerers, like stoppers, have little effective chance of progressing. Perkins *et al.* (1989) also found that students' attitudes to mistakes are an important factor in their progress, with those who are frustrated by their mistakes or have a negative emotional reaction to making errors, likely to become stoppers.

2.2.8. Motivation in Programming

A number of studies have examined the importance of motivation in programming, with some researchers focussing on the reasons why students choose to study programming and how students' motivations change over the time they spend on the course (Jenkins, 2001; Mamone, 1992). Curzon and Rix (1998), found that the major motivation when students start programming courses is to become a professional programmer but this is not the case when the course has advanced. Although programming continues to be regarded as useful, it is seen as a secondary skill later on. Indeed, being able to program appears not to be the ultimate objective of some students taking programming courses, with Curzon and Rix (1998, p. 62), stating that many "students appear not to ultimately expect to become programmers or directly use their programming skills". Low levels of intrinsic motivation and high levels of extrinsic motivation have been identified in programming courses. Mamone

(1992), in a three year study of 126 students on programming courses at two New York universities, found that 22% of the first year introductory programming class were studying programming because they were interested in it, with the remaining 78% studying it because of career prospects and because it was a requirement. Unfortunately, none of these studies correlated the reasons for studying programming with performance on the course.

Some researchers have considered the relationship between student motivation and impressions of Computing subjects. Mitchell, Sheard, and Markham (2000) found that students who have a strong motivation to study programming have a more positive perception of the subject, of the amount of practical work involved and of their final grades. Other researchers have focused on aspects of programming and technologies that can be used as motivators. Tharp (1981) suggested that programming exercises should be improved to make them more motivating. Feldgen and Clua (2003) found programming examples based on the internet and computer game programming to be more motivating than classical mathematical or business-based programming examples on first programming courses. Lawrence (2004) reports that the use of competitive games and competitive programming, where students develop and improve their code throughout an assignment by competing in a tournament against instructor-defined code and the code of other students, increases student motivation. Some researchers have suggested redesigning introductory programming courses specifically to improve students' experiences and to improve retention (Mahmoud, Dobosiewicz & Swayne, 2004). Hadjerrouit (1998b) suggests that Java should be used as a first programming language due to its perceived ability to improve learner motivation because of the high levels of pay for Java programmers.

In recent years, a number of studies have examined the role of comfort-level in programming. Irani (2004), drawing on surveys, interviews, and five years of enrolment data at Stanford University, suggests that while female and male students report similar levels of comfort in using computers, women assessed their peers on the course as being more comfortable with computers than they were, while men assessed themselves, on average, as slightly more comfortable than their peers. Thomas *et al.* (2003) studied students on an introductory programming course that replaced all individual assignments with paired assignments. Programming confidence levels were found to be important in students participating in pair programming activities, most notably in that students who were very confident did not enjoy the experience of pair programming as much as other students, and that students produced their best work when placed in pairs with students of similar confidence levels. Researchers have also examined the relationship between students' expectations of, and experiences on, an introductory Computing module. Wiedenbeck *et al.* (2004) found that a positive relationship has recently been identified between students' self-efficacy for programming and performance. Bergin and Reilly (2005, 2006) carried out a multi-institutional multivariate study of Irish first year students and found a link between comfort-level and introductory programming performance.

2.2.9. The Teaching and Learning of Novice Programmers

Teaching standards clearly influence the outcomes of courses that teach programming (Linn & Dalbey, 1989). Linn and Dalbey (1989) propose a set of "cognitive accomplishments" that should arise from the ideal computer programming instruction. This starts with the features of the language being taught, followed by design skills, including plans, and skills of planning, testing and reformulating code. The final element of the set of cognitive accomplishments includes problem-solving skills, and knowledge and strategies that are abstracted from the specific language being taught that can be applied to new languages and situations. Robins *et al.* (2003, p. 155) say that this set of "accomplishments forms a good summary of what could be meant by deep learning in introductory programming".

An observation that recurs with regularity in the literature is that the average student does not make much progress in an introductory programming course. Linn and Dalbey (1989) suggest that few students get beyond the 'features of the language' accomplishment, and conclude that "the majority of students made very limited progress in programming" (*ibid*, p. 74). This view is supported by Kurland *et al.* (1989) who studied students who had completed two years of programming instruction and found that "many students had only a rudimentary understanding of programming". Winslow (1996, p. 21) adds that "study after study has shown that [students] do not understand [even] basic loops". Soloway *et al.* (1982) studied students who had completed a single semester programming course, and found that 38% could not write a loop to calculate the average of a set of numbers. McCracken *et al.* (2001) conducted a multi-national, multi-institutional study of the programming skills of first-year students on Computer Science courses, and found that "many students do not know how to program at the conclusion of their introductory courses". Others agree: Fincher *et al.* (2005, p. 2) say they "believe that learning to program is problematic, and that the results achieved by students do not correlate well with their other academic results. Understanding of this phenomenon is patchy and poorly integrated, but it does seem clear that there are many influences at play". There is clearly room for improvement in the way students are taught programming. Jenkins (2002, p. 53) makes the point strongly, saying that "[a]t the moment the way in which programming is taught and learned is fundamentally broken".

2.2.10. Course Design and Teaching Methods

Brooks (1990) points out that the programming strategies that novices use strongly impact on the quality of final program that is produced. Yet most introductory programming courses are conventionally structured with lectures and practical laboratory work, and, as Robins *et al.* (2003, p. 157) state, are based on a "conventional curriculum focused largely on knowledge – particularly relating to the features of the language being taught and how to use them". This may be due to the fact that, as Robins *et al.* (2003, p. 157) state, "strategies themselves cannot (in most cases) be deduced from the final form of the program. Finished example programs are rich sources of information about the language which can be presented, analysed and discussed. The strategies that created those programs, however, are much harder to make explicit". Soloway and Spohrer (1989, p. 412) add that "students are not given sufficient instruction in how to 'put the pieces together.' [There is a need to focus] explicitly on specific strategies for carrying out the coordination and integration of the goals and plans that underlie program code". Given the observations regarding the limited progress made by novices in introductory courses, Robins *et al.* (2003, p. 157) call for introductory courses that are realistic in their expectations. Winslow (1996, p. 21) echoes this call, saying that "[g]ood pedagogy requires the instructor to keep initial facts, models and rules simple, and only expand and refine them as the student gains experience".

A number of studies show that students who are encouraged to actively engage with and explore programming related information perform better at problem solving (Hoc & Nguyen-Xuan, 1990; Mayer, 1989). Robins *et al.* (2003, p. 157) say that "laboratory based programming [problems] have some pedagogically useful features. Each one can form a 'case based' problem solving session. The feedback supplied by compilers and other tools is immediate, consistent, and (ideally) detailed and informative. The reinforcement and encouragement derived from creating a working program can be very powerful". Linn and Dalbey (1989) make the point that students using laboratory based programming problems can work and learn on their own and at their own pace, and "programming can be a rich source of problem-solving experience" (*ibid*, p. 78). Nevertheless, problems need to be carefully selected and based on clear programming principles (Kurland *et al.*, 1989).

Working collaboratively on programming problems in groups has been shown to be beneficial, particularly for weaker students (van Gorp & Grissom, 2001). Paired programming, where two students code together, working on the same problem, has been demonstrated to make students more self-sufficient (Williams, Wiebe, Yang, Ferzli & Miller, 2002). Wills *et al.* (1999) have shown that peer learning is beneficial for novices on introductory programming courses.

There is some evidence that teaching schemas to novices rather than waiting for schemas to emerge from examples and experience, improves skills transfer (Robins, 1996). While supporting the idea of teaching schemas, Perkins *et al.* (1989) also suggest that constructivist methods may be more generally effective:

> Instruction designed to foster bootstrap learning but not providing an explicit schematic repertoire might produce competent and flexible programmers, and might yield the broad cognitive ripple effects some advocates of programming instruction have hoped for.
>
> (Perkins *et al.*, 1989, p. 277).

Anderson (1976, 1993, 1996) developed a cognitive architecture model called ACT-R (Adaptive Control of Thought - Rational), and used this architecture to explore learning and knowledge consolidation. He suggests that abstract representations of knowledge (such as program code) cannot be learned directly and must be learned by the learner practising the operations on which the representations are based. This suggests that problem solving should be central to any programming course (Fincher, 1999b). Nonetheless, Robins *et al.* (2003, p. 160) say that "problem solving is necessary, but not sufficient, for programming. The main difficulty faced by novices is expressing problem solutions as programs. Thus the coverage of language features and how to use and combine them must remain an important focus". This view is supported by other researchers (Rist, 1995; Winslow, 1996).

Deek *et al.* (1998) describe a first year Computer Science course at the New Jersey Institute of Technology which was based on a Problem-Based Learning model, and where programming language features were introduced only in the context of the students' solutions to specific problems. Deek *et al.* (1998, p. 319) reported outstanding results, stating that the "final grades for the class receiving the alternative [problem-based] methodologies is skewed towards the higher grades (71% of the students received a grade of 'A,' 'B+,' or 'B.') But the class of students using the traditional approach received grades skewed towards the lower end of the scale (Approximately 56% received a grade of 'D,' 'F,' an Incomplete, or a Withdrawal.)". However, there are methodological issues affecting this study, particularly relating to the lack of controls for other non-PBL influences.

Kay *et al.* (2000) describe a foundation Computer Science course where Problem-Based Learning was implemented. The authors report exceptional success, for example, an increase in mean examination marks from 63% in the last non-PBL year to 91% in the second full PBL year. Kay *et al.* (*ibid*) also describe feedback from six students (2 from the PBL group and 4 from the non-PBL group), noting that the PBL students found "learning programming a positive experience", although one student in the PBL group "had extremely negative memories" of PBL (*ibid*, p. 121). Kay *et al.* (*ibid*) go on to describe a three year longitudinal follow-up of PBL students using self-report questionnaires, noting that students were satisfied with the PBL course. However, while the Kay *et al.* (*ibid*) study provides a detailed description of the implementation of PBL in a Computing context, there are a number of major methodological weakness in the study, including the small sample size (16) of the long term follow up, and the lack of controls for other non-PBL influences, such as students' prior attainment, student general intelligence levels, the teacher effect, and the introduction of a different programming language.

2.2.11. Programming Languages Used to Teach Programming

Many researchers have called for the use of simple, specially-designed programming languages for teaching such as Logo (du Boulay, 1989; Jenkins, 2002). However, they are seldom used, with the vast majority of courses using standard workplace languages such as Java or C++ (Robins *et al.*, 2003). While early studies in particular explored particular kinds of programming language structure or notation (Sheil, 1981), during the mid 1990s there was a focus on exploring issues relating to the object-oriented programming paradigm, in contrast to the procedural paradigm. Robins *et al.* (2003, p. 145) say that "in general such studies should be seen in the context that there is not likely to be any universally 'best' programming notation". Nevertheless a given notation may assist in the comprehension of certain kinds of information by highlighting it in some way in the program code (Gilmore & Green, 1984). Traditionally, Software Development courses have focused on the teaching of procedural computer programming languages. Over the last 15 years, nearly all Software Development courses have moved to teaching object-oriented programming languages, with the most widely taught programming language being the Java programming language. Java is also used extensively within the Software Development industry (Sun Microsystems, 2008). Therefore, in many ways the analysis of the differences between paradigms is now redundant, as nearly all Computing programming courses now use the object-oriented paradigm, reflecting its near total dominance in industrial and commercial Software Development, due to its expressive power which allows the development of new types of computer systems, particularly web based systems (Meyer, 1997). Nonetheless discussion is necessary to place the object-oriented paradigm in context and for a consideration of its characteristics.

While the object-oriented paradigm may be more powerful, many claims were also made that programming using the object-oriented approach would be easier than using the procedural approach (Meyer, 1997; Rosson & Alpert, 1990). However, the literature does not support this view. Détienne (1997) reviews a number of studies and shows that identifying objects is not an easy process, that objects identified in the problem domain are not necessarily useful in the program domain, that the mapping between domains is not straightforward, and that novices need to construct a model of the procedural aspects of a solution in order to properly design objects and classes. Similarly, Rist (1995, p. 555) describes the relationship between program design plans and objects as "orthogonal, because one plan can use many objects and one object can take part in many plans". Rist (1996, p. 39) suggests that object-oriented programming is not different, "it is more", because object-oriented design adds the overheads of class structure to a procedural system.

It seems that object-oriented programming might be particularly difficult for novices. Wiedenbeck *et al.* (1999) studied students' comprehension of procedural and object-oriented programs in their second semester of study at university, finding (*ibid*, p. 276) that "the distributed nature of control flow and function in a[n object-oriented] program may make it more difficult for novices to form a mental representation of the function and control flow of a[n object-oriented] program than of a corresponding procedural program". This suggests that when teaching the object-oriented paradigm, particular attention should be paid to control flow and data flow (Wiedenbeck & Ramalingam, 1999), and some researchers advocate the use of visualisation tools to aid comprehension (Baecker, 1998; Cooper, Dann & Pausch, 2003).

One new area of research is the identification and teaching of detailed reusable object-oriented program schemas called design patterns. Patterns are solutions to particular classes of programming problems (Gamma, Helm, Johnson & Vlissides, 1995). Some researchers have suggested teaching special pedagogical patterns, using pattern languages such as Seminars (Sharp, Manns & Eckstein, 2003; The Pedagogical Patterns Project, 2001). However, these have been subject to detailed examination and problems have been identified with them (Fincher, 1999a). Fincher and Utting (2002, p. 200) make the point that pedagogical patterns "miss some of the requirements: they are either so abstracted from the domain (of tertiary Computer Science education), and therefore generic, that they lack insight; or they are so tightly coupled to specific instances of practice that they are not transferable." Others are in favour of patterns, suggesting that they allow students to adapt simpler strategies to new and more complex problems (Proulx, 2000; Reed, 1998).

2.2.12. Summary

Computing courses have high failure and dropout rates (McAllister & Alexander, 2003). Programming has been shown to be a difficult task (Dijkstra, 1989). It requires the application of complex knowledge and associated strategies. The literature shows a clear distinction between the nature of the programming knowledge novices must master and the characteristics of the strategies they must employ to utilise that knowledge. Most introductory programming courses concentrate on teaching programming knowledge but not on the strategies needed to use this knowledge. Furthermore, there is little correspondence between the ability to write a program and the ability to read one; both need to be taught to novices. A number of complex mental models need to be constructed by novices if they are to learn to program effectively. Ensuring the correct construction of these mental models is crucial in allowing novices to build an understanding of programming. The literature shows that the main problem for novices is program design and planning. The strategies that they employ appear to distinguish effective from ineffective novices.

For some Computing students, learning programming is intimidating, giving rise to anxiety and a lack of confidence. Other than measures of general intelligence, novices' programming self-efficacy is the most accurate predictor of success at programming. Cognitive styles and personality traits do not impact success at programming. Low levels of intrinsic motivation and high levels of extrinsic motivation have been identified in programming courses.

A review of the literature shows that the results achieved by students in programming do not correlate well with their other academic results. There is clear evidence that there is room for improvement in the way students are taught programming. Brooks (1990) points out that the programming strategies that novices use strongly impacts on the quality of final program that is produced. Yet most introductory programming courses are conventionally structured with lectures and practical laboratory work. A number of researchers suggest that constructivist methods may be more effective; in particular Problem-Based Learning has been reported to produce better outcomes (Deek *et al.*, 1998; Kay *et al.*, 2000). A number of studies show that students who are encouraged to actively engage with and explore programming related information perform better at problem solving, and working collaboratively on programming problems in groups has been shown to be beneficial, particularly for weaker students (Mayer, 1989; van Gorp & Grissom, 2001; Wills *et al.*, 1999). A number of researchers have called for the use of simple teaching programming languages to improve outcomes, though most programming courses use fully functional industrial standard object-oriented programming languages.

This section has shown that the Software Development curriculum is viewed as one of the most challenging to teach and one of the most difficult and troublesome for learners. In the next section on Threshold Concepts, one possible framework is examined that may help explain why learners find computer programming so troublesome. A general discussion of Threshold Concepts will be followed by an examination of Threshold Concepts in the context of Computing.

2.3. Threshold Concepts

Meyer and Land (2005) have proposed using the term Threshold Concepts to characterise particular concepts whose mastery is necessary to make progress in a discipline, that transform the way a student looks at a discipline, but are also places in the curriculum where students get stuck, unable to make progress until they become unstuck.

Threshold Concepts are a subset of core concepts in a discipline. Core concepts are building blocks that must be understood. As well as being core concepts, Threshold Concepts have additional properties. As Meyer and Land state:

> [a] Threshold Concept can be considered as akin to a portal, opening up a new and previously inaccessible way of thinking about something. It represents a transformed way of understanding, or interpreting, or viewing something without which the learner cannot progress. As a consequence of comprehending a Threshold Concept there may thus be a transformed internal view of the subject matter. […] This transformation may be sudden or it may be protracted over a considerable period, with the transition to understanding proving troublesome. […] Such a transformed view […] may represent how people 'think' in a particular discipline, or how they perceive, apprehend, or experience particular phenomena within that discipline

(Meyer & Land, 2006, p. 3).

Meyer and Land (*ibid*, p. 7) define Threshold Concepts as:

1. Transformative: they change in a significant way a student's perception of a subject.

2. Irreversible: the change in perspective occasioned by acquisition of a Threshold Concept is unlikely to be forgotten or can be unlearned only by considerable effort.
   Meyer and Land (*ibid*, p. 7) suggest that "this can account for the difficulty experienced by expert practitioners looking back across thresholds they have personally long since crossed and attempting to understand (from their own transformed perspective) the difficulties faced from (untransformed) students' perspectives".

3. Integrative: they expose the previously hidden interrelatedness of something and tie together concepts in ways that were previously unknown to the student.

4. Often boundary markers: they indicate the limits of a conceptual area or the discipline itself. Students who have mastered these Threshold Concepts have, at least in part, crossed over from being outsiders to belonging to the field they are studying.

5. Potentially troublesome for students: they can be conceptually difficult for students. Perkins (1999) has defined troublesome knowledge as that which appears counter-intuitive, alien, or incoherent.

Meyer and Land (2006, p. 9) make the point that "[g]iven the centrality of such [threshold] concepts within sequences of learning and curricular structures, their troublesomeness for students assumes significant pedagogical importance". Therefore, it is worth examining why Threshold Concepts should be so troublesome for learners.

2.3.1. Types of Troublesome Knowledge

Threshold Concepts are closely tied to the constructivist tradition. Indeed, Meyer and Land's use of the term 'troublesome' follows from Perkins' (1999) discussion of the challenges that constructivists must face. Perkins (2006) suggests five types of troublesome knowledge, which he classifies as ritual knowledge, inert knowledge, conceptually difficult knowledge, tacit knowledge, and foreign knowledge. Perkins goes on to suggest that constructivist teaching practices, such as problem-based learning, can help students master these troublesome areas.

Ritual knowledge has a routine and rather meaningless character (Perkins, 1992). It feels like part of a social or individual ritual. Gardner (1993) suggests that a number of misconceptions in science are a consequence of ritual knowledge. A constructivist response to the problem of ritual knowledge strives to make the knowledge more meaningful, for example, Solomon (1998) outlines how this could be done in the teaching of mathematics.

Inert knowledge, as Perkins (2006, p. 37) claims, "sits in the mind's attic, dusted off only when specifically needed", Much research has been done on how knowledge and skill acquired in one context for one purpose impacts performance in other contexts for other purposes (Haskell, 2001). There is a long history of research into the transfer of learning that shows that transfer where the initial learning and target applications differ occurs only partially and sporadically. Indeed as McKeough, Lupart and Marini (1995) point out in the preface of their book Teaching for Transfer:

> Transfer of learning is universally accepted as the ultimate aim of teaching. However, achieving this goal is one of teaching's most formidable problems. Researchers have been more successful in showing how people fail to transfer learning than they have been in producing it, and teachers and employers alike bemoan students' inability to use what they have learned.
>
> (McKeough *et al.*, 1995, p. vii)

Eylon and Linn (1988) have observed of science education that students deal with apparent contradictions between subjects by keeping their knowledge isolated. This might explain why students often fail to transfer knowledge from one module to another. However, conditions of learning that foster good initial mastery, diverse practice, and mindful abstraction can enhance transfer substantially (Bransford & Schwartz, 1999). This solution has been shown to be particularly true of computer programming instruction (Salomon & Perkins, 1989). Strategies and approaches that students bring to learning are also important: Bereiter and Scardamalia (1985, p. 66) say that "efforts to solve the inert knowledge problem may fail if they deal only with how knowledge is presented to students and what they are asked to do with respect to that knowledge. Unless direct attention is given to the coping strategies [learners] bring to knowledge use tasks, those strategies may defeat instructional intentions". Perkins (2006, p. 38) suggests that one constructivist solution to the knowledge transfer problem is to "engage students in problem-based learning, where they acquire the target concepts while addressing some medium-scale problem". This view is strongly supported by other researchers (Boud & Feletti, 1998; Savery & Duffy, 1995).

Another problem is learners' misconceptions. Research into the misconceptions of learners has been influential in the formation of constructivist theory (Smith, diSessa & Roschelle, 1994). As Eckerdal *et al.* (2006, p. 105) state, "[m]isconceptions naturally occur as students modify and extend their knowledge frameworks to learn new topics. For example, an individual's previous understanding can lead to misconceptions when familiar terms are used in unfamiliar contexts", such as in a computer programming context. Bonar and Soloway (1985, p. 133) say that "many programming bugs can be explained by novices inappropriately using their knowledge of step-by-step procedural specifications in natural language". They illustrate this by using the *while* statement to demonstrate the problem of linguistic transfer. In common language the *while* can imply continual testing of the condition (e.g., "hold your breath while underwater"). In programming loops the time of the test is limited, it occurs only once on each iteration. Students who interpret the test as continual have a misconception. The overloading of language, mathematical symbols and previous programming experience, have all also been shown to cause misconceptions for novice programmers (Clancy, 2004).

In any field of study students at some stage have to deal with **conceptually difficult** knowledge. Some researchers suggest that this is a particular problem in the study of mathematics, science and computer programming at higher levels (Boustedt *et al.*, 2007; Perkins & Simmons, 1988). Perkins (2006, p. 39) suggests that there is a need to “engage students with qualitative problems rather than with solely quantitative ones, as qualitative problems lead students to confront the character of the phenomenon rather than just to master computational routines”. Another possible solution suggested by Gentner and Stevens (1983) is to introduce learners to imagistic mental models or to invite them to invent their own.

There is evidence that in the context of computer programming, having an incorrect mental model can cause problems. In a study of first year students on a university Java programming course using tape-recorded interviews, Fleury (2000) found that as students familiarize themselves with new topics, their partial knowledge leads them to develop their own rules. Unfortunately, if the knowledge is incomplete, these self-constructed rules may result in false assumptions and misconceptions, which once established are difficult to change. In a study of students on an object-oriented programming course using Smalltalk at the Open University, Holland, Griffiths and Woodman (1997) observed that these premature generalizations are then used to filter and distort new information, often compounding the misconception. McCracken, Newstetter, and Chastine (1999) conducted a descriptive study of 290 first year students in a technological institute and found that in order to repair such misconceptions significant effort is required, and that this involves a radical reordering of the concepts taught.

While there are some similarities between Threshold Concepts and mental models - both, for example, can be transformational - there are major differences. Threshold Concepts are troublesome while some mental models can be easily learnt, for example modelling the computer screen as a desktop (Eckerdal *et al.*, 2006). Threshold Concepts are accepted concepts within a discipline, while mental models are subjective and individual (Johnson-Laird, 1983).

It has been observed that students in Computing often display ritualised routines rather than genuine enquiry and problem solving (Sproull, Kiesler & Zubrow, 1984). This is also true of students in mathematics and science. Perkins (2006, p. 42) suggests that “authentic problem solving and Problem-Based Learning that foreground the games of the discipline are constructivist practices that can help”.

Perkins (2006, p. 40) argues that "much of the knowledge we rely upon everyday in both commonplace and professional activities is **tacit**; we act upon it but are only peripherally aware or entirely unconscious of it", and he goes on to say that constructivist approaches to teaching can make tacit knowledge clear.

Perkins (2006, p. 39) characterises **foreign** or alien knowledge as that which "comes from a perspective that conflicts with our own". Many students consider the process of creating a set of precise instructions in code, submitting these instructions to a machine, and then having the machine accept or reject the instructions an alien one. As du Boulay (1989, p. 278) points out, "[t]he notion of the system making sense of the program according to its own very rigid rules is a crucial idea for the learner to grasp". Indeed, as novices know how they intend a given piece of code to be interpreted, they have a tendency to assume that the computer will interpret it in the same way, and are surprised when it does not (Soloway & Spohrer, 1989).

It could in fact be argued that in Computing courses, not only do the concepts and knowledge appear alien to learners, but the whole culture of the course is strange to many students. Sproull, Kiesler, and Zubrow (1984) carried out a study at Carnegie-Mellon University to measure levels of learner alienation in different college courses. This empirical study compared the experiences of 250 novice programmers in their freshman year on a Computer Science course to the novices' other first year courses (English, Social Science, and Mathematics) by using a fixed-response questionnaire. The results showed much higher levels of alienation in the Computer Science course than in other courses. Sproull, Kiesler, and Zubrow (*ibid*, p. 2) point out the need to address "the social, organizational, and cultural context within which encounters with Computing occur", and add that "introducing novices to Computing is more than simply providing a machine and teaching a set of skills for using it. It is also introducing a new culture. Novices learn cultural lessons as well as technical ones. And the nature of those cultural lessons does much to determine novices' attitudes toward Computing and their willingness to pursue it further". Sackrowitz and Parelius (1996), in a study of first year Computing students at two major universities, found strong evidence that women find Computing an even more alien environment than men, and that this places women at a disadvantage in introductory Computer Science classes.

Perkins and Martin (1986) classify some of the knowledge that Computing novices have as "fragile". This is characterised by the novice appearing at first not to understand. Nonetheless, the required knowledge may have been learned and can be uncovered by the tutor providing judicial hints. Perkins and Martin (1986) suggest that fragile knowledge may be either inert and unused or misplaced and used inappropriately. Strategies can also be fragile, with novices failing to trace code even when showing an understanding of the technique (Davies, 1993; Gilmore, 1990).

2.3.2. Liminality

The above categories of troublesome knowledge are not the only ones, and they are not mutually exclusive. Perkins (2006, p. 41) states that "[c]oncepts as categoriser set the stage for a more elaborate function. Associated with clusters of concepts are activity systems or conceptual games that animate them". Perkins (*ibid*, p. 41) adds that "[a]lthough some of what is troublesome about knowledge squarely concerns the categorical function of concepts, much concerns the larger conceptual games around them [and] difficulty with particular disciplinary concepts may derive from difficulty with the underlying episteme" (*ibid*, p. 43). The solution he suggests is that educators make the rules of the epistemic game explicit. The value of making things explicit, principally in problem solving and mathematics, is supported by other researchers (Schoenfeld, 1979, 1980; Schoenfeld & Herrmann, 1982).

Meyer and Land (2006, p. 16) state that "difficulty in understanding Threshold Concepts may leave the learner in a state of *liminality* (Latin *limen* – 'threshold'), a suspended state in which understanding approximates to a kind of mimicry or lack of authenticity". As they point out: "central to the acquisition of Threshold Concepts is a consideration of what it might mean to be 'in the threshold'." (*ibid*, p. 22). Meyer and Land (2005), drawing directly on the work of Van Gennep (2004) and Turner (1995), develop the argument that acquiring a Threshold Concept may be likened in some disciplines to 'a rite of passage'. They propose a number of reasons for the rite of passage analogy, including that the condition of liminality may be transformative in function, with participating individuals acquiring new knowledge and subsequently a new status and identity within the community. This sense of a rite of passage is echoed in the words used to describe their experiences by students who have completed computer programming courses (Sproull *et al.*, 1984). Meyer and

Land (2006, p. 24) state that “the transformation can be protracted over periods of time, and involve oscillations between states, often with temporary regression to earlier states. This regression may be viewed as a form of compensatory mimicry”, and they point out that “in student learning terms mimicry, it seems, may involve both attempts at understanding, and perhaps not merely an intention to reproduce information in a given form”. Rountree and Rountree (2009, p. 140) point out that “there is a significant emotional reaction to dealing with liminality, and that such reaction is normal and should be managed rather than ignored or dismissed”. Palmer (2001) adds that crossing the threshold might be distressing and leave learners with a sense of loss.

Interestingly, Meyer and Land (2006, p. 29) suggest that one possible solution to moving learners out of liminality or what Ellsworth (1997) calls “stuck places”, is to use simplified representations of authentic concepts in a form that novices can engage with. These representations are proxies for the underlying Threshold Concepts, and retain the correct episteme, but without troublesome definitions. However, as Reimann and Jackson (2003) illustrate in the discipline of economics, the procedure of formulating proxies is difficult and likely to involve a process of trial and error.

Savin-Baden (2000) calls these “stuck places” disjunction and suggests that disjunction can be both enabling and disabling in terms of its impact on learners. She suggests that disjunction can occur when a learner encounters a Threshold Concept but has not yet mastered it fully. She goes on to suggest that staff and students use various strategies to try to deal with disjunction, which include “retreating from the difficulty and opting out of any further learning, using strategies to avoid it, temporising and waiting for an event or stimulus that will help them to move on or engaging with it directly in an attempt to relieve their discomfort” (Savin-Baden, 2006, p. 161).

Evidence that a Problem-Based Learning approach might help with disjunction is provided by Savin-Baden (*ibid*), who argues that although disjunction occurs in many forms and diverse ways in different disciplines, it seems to be particularly evident in curricula where Problem-Based Learning has been implemented. She suggests that this is because:

> Problem-Based Learning programmes prompt students to critique and contest knowledge early on in the curriculum and thus they encounter knowledge as being troublesome earlier than students in more traditional programmes. [However] it might also be that Problem-Based Learning encourages students to shift away from linear and fact-finding problem solving. Instead they move towards forms of problem management that demand the use of procedural and personal knowledge as students are asked to engage with strategy or moral dilemma problems. Thus it might be that disjunction is not only a form of troublesome knowledge but also a 'space' or 'position' reached through the realisation that the knowledge is troublesome. Disjunction might therefore be seen as a 'troublesome learning space' that emerges when forms of active learning (such as problem-based learning) are used that prompt students to engage with procedural and personal knowledge.
>
> (*ibid*, p. 178)

Indeed, Savin-Baden (*ibid*) describes Problem-Based Learning itself as a Threshold Concept that is difficult to grasp as it challenges both staff and learners to see learning and knowledge in a new way.

To help students develop an understanding of a troublesome concept, a number of studies e.g. Colby, Ehrlich, Beaumont, & Stephens (2003); Klem and Connell (2004) point to the need for teachers to support active student engagement with, and manipulation of, the conceptual material. This has also been shown to be true in the teaching of computer programming, with the addition of multimedia visualisation proving to be most useful in helping understanding (Bergin *et al.*, 1996; Razmov & Anderson, 2006).

Efklides (2006, p. 48) emphasises the role of metacognition in the learning process, "both directly by activating control processes and indirectly by influencing the self-regulation process that determines whether the student will get engaged in Threshold Concepts or not". She also claims (2003, p. 1) that "[m]etacognitive experiences serve the monitoring and control of the learning process and at the same time provide an intrinsic context within which learning processes take place". Section 2.4 of this literature review will discuss further the role of self-regulation in the process of learning.

Threshold Concepts can be used to evaluate teaching strategies. Meyer and Land (2006, p. 16) state that wherever Threshold Concepts are present they “constitute an obvious, and perhaps neglected, focus for evaluating teaching strategies and learning outcomes”. However, it is worth considering Meyer and Land’s (2006, p. 16) question ‘whose threshold concepts?’ as it is possible to interpret Threshold Concepts as part of a “colonising view of the curriculum”, tools used to cement power relations within curricula. Expanding this point, Meyer and Land (2005, p. 375) state that “[F]rom the learner's perspective there is an unwelcome power relation deemed to be in operation in which one academic tribe is seen imperialistically to be colonising the discursive space of other tribes”. Nonetheless, in Computer Science, like other science subjects, there is an assumption that what is taught is empiric physical knowledge and that educators can safely assume that students must and should internalise these Threshold Concepts in order to progress in their discipline. The assumption is made that the thresholds and the knowledge are essentially politically or culturally neutral in nature (Palmer, 2001).

2.3.3. Criticisms of Threshold Concepts

Although the Threshold Concepts model is fashionable and gaining popularity, not all researchers are convinced of its usefulness. Rowbottom (2007) puts forward several criticisms of Threshold Concepts, all of which could apply to Threshold Concepts in Computer Science. His main criticism is “that ‘threshold concepts’ as defined by Meyer and Land [(2006)] are unidentifiable even in principle”, adding “that several authors understand ‘threshold concepts’ in different and incompatible ways”. Rowbottom is supported in this view by Rountree & Rountree (2009, p. 142) who in their discussion of Threshold Concepts in Computing state that “[t]he features attributed to threshold concepts are insufficiently precise to distinguish them from any other concept”, adding that “any concept you care to mention might be a threshold concept, even though it has none of the features [of Threshold Concepts described by Meyer & Land], and any concept that has all of the features may not in fact be a threshold concept [and without a clear definition of Threshold concepts], it is not logically feasible even to use empirical research to support or refute a claim that something is or is not a threshold concept”.

Rowbottom's (2007, p. 264) second criticism is that there are at least three standard definitions of concepts and it is not clear which is meant by the 'concept' in Threshold Concepts. Firstly, in Cognitive Science, a concept is seen as a mental model functionally equivalent to symbols or words. Secondly, in Philosophy, "concepts are abilities. More carefully, any given concept is supposed to be reducible to a peculiar set of abilities" (*ibid*). Thirdly, also in Philosophy, concepts can be seen as abstract entities of thought associated with names. Rowbottom (*ibid*) argues that Meyer & Land have not clearly articulated which of the "three standard accounts of concepts" they are referring to when they talk about concepts.

Rowbottom's (2007, p. 267) third criticism is that "being transformative is arguably an extrinsic property, rather than an intrinsic one. Specifically, what is transformative for Mr. A need not be so for Mrs. B, because this depends on the conceptual scheme (or system of concepts) initially possessed by each".

Rowbottom (2007, p. 263) goes on to argue that these shortcomings raise a barrier to empirical research into Threshold Concepts, questioning "how is it possible to test for concepts, rather than abilities? [and] how can we tell if there is more than one possible conceptual route to the same ability?"

These criticisms suggest that before we can apply Threshold Concepts to our teaching we must overcome some difficulties and answer important questions relating to the operationalisation of the idea of Threshold Concepts:

> [W]hat precisely are threshold concepts? Can we identify them? Can we agree on which concepts are threshold concepts and which are not? Can we validate them? If threshold concepts do exist, and can be identified and agreed upon, then how would they alter what we teach, how we teach, and how we assess? Do threshold concepts represent anything new or unexpected?
>
> (Rountree & Rountree, 2009, p. 139).

Nonetheless, Rountree & Rountree (2009, p. 142) argue that these "caveats are not necessarily enough to dismiss the Threshold Concepts model, nor the possibility of identifying good candidates for threshold concept instances. Even though our definitions of Threshold Concepts may not be perfectly precise, we can defeasibly posit their existence, and agree upon their most distinctive features, until such time as we find evidence to suggest that we should retract our assertion. Imprecise definitions are insufficient evidence for retraction”. The next section looks in detail at Threshold Concepts in Computing.

2.3.4. Threshold Concepts in Computing

The Computing field has evolved from its origins in Computer Science, and new Computing-related disciplines have emerged, such as software engineering, information technology, information systems and computer engineering (Association for Computing Machinery, 2008). Many universities and colleges also have general Computing degree courses which cover aspects of all five strands. It is important to point out at this stage that the focus of the research for this thesis is on a general Computing curriculum, rather than on any of the five strands.

As Threshold Concepts are a subset of the core concepts in a given discipline, a list of the core concepts in Computing would be a good basis for any identification of the Threshold Concepts in Computing. The concepts covered vary from course to course, but a review of the literature shows some common themes. There are two main approaches to designing the Computing curriculum: firstly what has been termed the ‘fundamental ideas’ approach, and secondly the breadth-first approach.

Schwill (1994, 1997) has proposed a set of ‘fundamental ideas’ that are central in Computing and he suggests that the Computing curriculum be arranged around them. In this, Schwill follows from the work of Bruner (1960), who proposed that science teaching should be organized around the structure of science, as expressed by its fundamental ideas. Schwill (1994, 1997) suggests that these ‘fundamental ideas’ should be taught throughout the curriculum, and when new concepts are presented, they are related to the appropriate fundamental ideas that the students already know, thus providing context. In addition, relating new concepts to these ideas should further develop the ideas, so in the learning process, the learner gradually gains a greater understanding of these fundamental ideas. There is some obvious overlap between fundamental ideas and Threshold Concepts. Both are integrative, and both include topics that should be understood by any competent Computing professional. However fundamental ideas are not likely to be transformative, in that they are gradually developed from common-sense understanding of everyday phenomena, and they are clearly not boundary markers, as these ideas have everyday out-of-discipline meanings.

Eckerdal *et al.* (2006, pp. 105-106) say that Threshold Concepts and fundamental ideas:

> seem to be orthogonal. Threshold Concepts are based on transformative events, while fundamental ideas are based on long-term development. It seems likely that any given Threshold Concept could be described in terms of the related fundamental ideas, and that there are Threshold Concepts that appear at points in the development of a given fundamental idea. Threshold Concepts identify the discontinuities in a student's development, while fundamental ideas identify different ongoing threads in this process which may or not have such discontinuities.
>
> (Eckerdal *et al.*, 2006, pp. 105-106)

The breadth-first approach to the Computing curriculum is well defined. The Computing Curricula (2001) project, which was a joint undertaking of two international organisations, the Computer Society of the Institute for Electrical and Electronic Engineers and the Association for Computing Machinery, set out to develop curricular guidelines for all undergraduate programmes in Computing. The project recommended a set of breadth-first guidelines which includes the following concepts and topics: decision trees, number representation, patterns in programming, divide-and-conquer, recursion, the Church-Turing thesis, the von Neumann architecture, time complexity, intractability, types and values, object-orientation (including classes and objects, design, encapsulation, inheritance, and polymorphism), program correctness, iteration, recursion, conceptual and formal models, levels of abstraction, reuse, and tradeoffs. These concepts and topics have also been identified as core topics by other researchers (Biermann, 1997; Brookshear, 2007; Denning, 2004; Schneider & Gersting, 2006; Zendler & Spannagel, 2008). While these concepts are well established core concepts in Computer Science, they are probably not all Threshold Concepts, as many of the concepts on the breadth-first courses may not be transformative. Which, if any, of them qualify as Threshold Concepts is a question for ongoing empirical investigation (Boustedt *et al.*, 2007; Eckerdal *et al.*, 2006; Rountree & Rountree, 2009).

Davies (2006) points out that the identification of Threshold Concepts may be difficult due to their being ‘taken for granted" within a subject, and therefore rarely ‘made explicit’. He goes on to suggest two methods for recognising the Threshold Concepts within a discipline. The first approach suggests that Threshold Concepts might be identified by examining the different ways in which two disciplines analyse the same situation. The second approach focuses on the differing ways in which novices and experts in the field analyse the same problem or group of problems. Rountree & Rountree (2009, p. 142) argue that this “is empirically very convenient for educators in a given field, as they have the best opportunities to conduct research on their own students. Consequently, most work on identifying Threshold Concepts within disciplines has focused on this approach”. Rountree & Rountree go on to say that the “clear disadvantage [of this approach] is that there is no equivalence between novice/expert comparisons and expert/expert comparisons” (*ibid*). Most substantial work on identifying Threshold Concepts in Computer Science has used this second approach, examining the responses of students in Computer Science to questions about what they found troublesome while studying. An example of this approach is a study by Boustedt, *et al* (2007) who conducted in-depth interviews with both final year Computer Science students nearing graduation and their lecturers, and identified object-oriented programming as a Threshold Concept in Computer Science. However, the research shows that there are difficulties in articulating the granularity of Threshold Concepts. For example, while both lecturers and students referred to object-orientation as a threshold concept, Boustedt *et al* (2007) note that this is almost certainly too broad a term, given that their interviews reflected that the ‘stuck places’ were more at the level of polymorphism or object cooperation. Two subcomponents of object-oriented programming have been suggested as Threshold Concepts in a number of studies. These are levels of abstraction and object-orientation (Eckerdal *et al.*, 2006). These are areas of the Computing curriculum that are part of both the breadth-first and fundamental ideas approaches. Both of these concepts would be included in any object-oriented programming course and both are mainly covered as part of introductory Java programming courses.

2.3.4.1. Levels of Abstraction

Abstraction is a core concept in Computing and it has been widely studied. Détienne (1997), in a review of empirical research on object-oriented programming, found that the ability to deal with and move between many different levels of abstraction is central to gaining the ability to design and write object-oriented programming code. Using a cognitive task analysis taxonomy regarding abstraction and inheritance, Or-Bach and Lavy (2004) came to the same conclusion and further determined that abstraction is a higher order cognitive skill that is difficult for students to conceptualize. These findings are supported by Rehder *et al.* (1995), who show a clear distinction in the way novices and expert object-oriented developers handle different levels of abstraction and point out that abstraction is a key skill that students need to acquire to be able to successfully design software. In addition, one of the major stumbling blocks for learners is the abstraction of the problem to be solved from the exercise description (McCracken *et al.*, 2001). Hoc and Nguyen-Xuan (1990) showed that certain kinds of abstractions can lead to errors in the use of conditional tests.

A number of researchers have identified that many students have an inadequate appreciation of the concept of abstraction (Détienne, 1997; Haberman & Averbuch, 2002; Sooriamurthi, 2001). One of the main difficulties, particularly for novices, is an inability to distinguish between the declarative and procedural aspects of a solution (Sooriamurthi, 2001). This distinction between the declarative and procedural aspects of code has also been identified by Haberman and Averbuch (2002) as a barrier to the understanding of recursion, which is a central concept in Computer Science and has been identified as a very difficult concept for beginners to learn (Anazi & Uesato, 1982; Levy & Lapidot, 2000; Wiedenbeck, 1988). Although constructivist approaches to teaching and dramatization have been shown to be helpful (Ben-Ari, 2001; Ben-Ari & Reich, 1997), Velázquez-Iturbide (2000) suggests that the difficulty in learning recursion does not come from the recursion concept itself, but from its interaction with other mechanisms of imperative programming such as abstraction. Interestingly, in the context of pharmacy education, Fisher (1994) has shown that Problem-Based Learning helps students to develop or reformulate declarative and procedural knowledge in such a way that students' cognitive strategies are enhanced.

The mental models of recursion that students develop have been identified (Bhuiyan, Greer & McCalla, 1994; Dicheva & Close, 1996; Kahney, 1983). Kahney (1989) showed that users have a variety of (mostly incorrect) approximate models of recursion. Similarly, Kessler and Anderson (1989) found that novices were more successful at writing recursive functions after learning about iterative functions, but not vice versa. Constructivist approaches to teaching programming have been shown to help repair misconceptions that can develop in students' mental models of recursion (Gotschi, Sanders & Galpin, 2003).

Détienne (1997) points out that while an object can be thought of as an abstract data type, in object-oriented programming it is also appropriate to consider the abstraction inherent in object orientation as a behavioural abstraction. This understanding is seen in advanced object-oriented designers, but not in novices (Rehder *et al.*, 1995).

Box and Whitelaw (2000) argue from a constructivist perspective that abstraction helps partially explain the difficulty of learning object-oriented programming, saying that abstraction is the most difficult step a student has to face when learning object-oriented programming. Significant parts of this abstraction are the decisions as to which entities are to be grouped together and which attributes are to be ignored or parameterized (Eckerdal *et al.*, 2006).

Hadjerrouit (1998a) describes a pedagogical framework motivated by constructivist learning principles for integrating the Java object-oriented programming language into the undergraduate curriculum. When discussing students' learning of Java, she states the need for students to understand abstraction, saying that "...to understand Java concepts properly, problem solving should begin at the conceptual level, not at the code level where programming becomes the main issue. Furthermore, substantial attention should be devoted to the meta-level process required to develop solutions" (*ibid*, p. 107).

2.3.4.2. Object Orientation

Object orientation is another core concept in Computing taught to most first-year students on Computing courses, and it is considered essential that all Computing students understand it. Like abstraction, the teaching of object orientation, particularly to novices, has been widely studied. The literature suggests that object orientation possibly satisfies the requirements of a Threshold Concept. There is much evidence in the literature that students find basic object-orientation troublesome to learn and it is widely acknowledged that object-oriented programming is difficult to teach (Kölling, 1999).

Eckerdal and Thuné (2005) interviewed first year students who had just finished their first programming course on their understanding of the concepts of object and class. Many students stated that they found the concepts troublesome to learn despite expending great effort to understand them. Likewise Ragonis and Ben-Ari (2002) identified that students on a first programming course in Java had great difficulty understanding the creation of an object by a constructor method. Fleury (2001, p. 191), through audio-taped student interviews, examined novice Java students' conceptions of object-oriented programming, particularly their comprehension of encapsulation and reuse of code. One result she reports is that many students find methods with parameters difficult to learn. Also many students are annoyed by the "jumping around" necessary when reading programs with multiple classes. Détienne (1997) summarises some problems that are specific to novices learning object-oriented languages, including a tendency to think that instance objects are created automatically, and misconceptions about how inheritance structures operate. Holland *et al.* (1997) claim that misconceptions of object-oriented concepts can be hard to shift later, pointing out that such misconceptions can act as barriers through which all later teaching on the subject may be inadvertently filtered and distorted.

There is evidence that object orientation is transformative. In addition, many learners describe their experiences of learning to program as transformative in their understanding of the wider Computing curriculum. Indeed Luker (1994, p. 58) goes so far as to argue that learning the object-oriented paradigm "requires nothing less than a complete change of world view". Eckerdal (2004) carried out a phenomenographic study at Uppsala University of fourteen first year students' understanding of the concepts object and class after sitting a semester long Java programming course. She reports that some students, who had used other programming paradigms before, could use the concepts object and class in a way that made programming more efficient. She found that most students had problems separating the concepts object and class. Luker (1994) suggests using encapsulation to tie together the concepts object and class, and suggests that object orientation integrates these concepts.

The connection between fundamental programming concepts in Java, such as objects, and the understanding of the programming paradigm itself is stressed by Hadjerrouit (1998a) who writes that "[i]t is critical to understand that Java is not only a programming language, but that it is also an [object-oriented] paradigm with a set of fundamental concepts that can be used to explore a wide range of problems that was previously beyond the reach of Computing". Later she points out the need to view the "Java [programming language] as a Computing paradigm organised around a set of fundamental concepts". Eckerdal (2004, p. 33) states that "[u]nderstanding central concepts within object-oriented programming is fundamental, and is closely related to understanding the object-oriented paradigm itself". She goes on to add that "[a] rich understanding of the concepts object and class includes an understanding that classes and objects are models of real world phenomena" (*ibid*, p. 33).

Abstraction and object-orientation are two possible concepts that may be Threshold Concepts; certainly evidence exists that they have the appropriate characteristics. Both abstraction and object-orientation are taught through the medium of the Java programming language; therefore, as these concepts may be thresholds that students must cross and are places where many students find difficulty, if the teaching of object-oriented (Java) programming can be improved, students can be helped over the threshold, thus ensuring that they continue to make progress in the Computing discipline.

This section has shown that constructivist approaches, and PBL in particular, may help learners with the disjunction caused by Threshold Concepts (Savin-Baden, 2000). This, coupled with the evidence from Section 2.1 that PBL has improved outcomes on programming courses, suggests that PBL would be a good instructional choice for the teaching of programming. The next section examines PBL in more detail.

2.4. Problem-Based Learning Literature

The theoretical basis for Problem-Based Learning has roots in a number of learning theories: social constructivism theories where social interaction plays a fundamental role in the development of any higher cognition (Phillips, 2000; Vygotsky, 1978); experiential learning theories where learning is seen as a cyclic process with the starting point of the learning process being the learners' own experiences (Dewey, 1998; Kolb, 1984); and person-centred learning theories that emphasize the communal and interactive nature of learning and allow the learner to control the learning process, with the teacher taking the role of facilitator who fosters the learning process (Rogers, 1969).

2.4.1. What is PBL?

PBL emerged from the work of Barrows in 1963, and was first implemented in medical education in McMaster University in Canada in 1964. Since its inception, PBL has been adopted in many institutions worldwide and has been implemented in many different ways in diverse contexts, which makes PBL difficult to define exactly.

According to Barrows and Tamblyn:

> Problem-Based Learning is the learning that results from the process of working toward the understanding or resolution of a problem. The problem is encountered first in the learning process!
>
> (Barrows & Tamblyn, 1980, p. 1)

A useful definition of PBL that is cited on many websites is that:

> PBL is both a curriculum and a process. The curriculum consists of carefully selected and designed problems that demand from the learner acquisition of critical knowledge, problem solving proficiency, self-directed learning strategies, and team participation skills. The process replicates the commonly used systemic approach to resolving problems or meeting challenges that are encountered in life and career.

(Maricopa Community Colleges Center for Learning and Instruction, 2001) [2]

PBL is part of the shift from the teaching paradigm to the learning paradigm (Barr & Tagg, 1995). The focus is on what the students are learning rather than what the teacher is teaching. PBL is a teaching method that can be used in many formats, such as small-group tutorials, problem-based lectures, large-group case method discussion, and problem-based laboratories (Kaufman, 1995). However, it is used most commonly in small groups with a facilitator/tutor. The essence of the PBL method involves three steps: confronting the problem; engaging in independent study; and returning to the problem (Wilkerson & Feletti, 1989).

Boud (1985) outlined broad characteristics of PBL that move beyond any single prescriptive definition. Barrows (1986) developed a taxonomy of PBL methods. However, since then there have been a myriad number of PBL and hybrid PBL implementations. Nonetheless, several researchers have made attempts to define and provide guidelines for the implementation of a 'true' PBL instructional model (Savery & Duffy, 1995; Woods, 1996), while Boud and Feletti (1998) have provided a list of the practices considered characteristic of the philosophy, strategies and tactics of Problem-Based Learning. Even so PBL can mean quite different things to different people (Thomas, 2000). Maudsley (1999) claims that the widespread adoption of the PBL instructional approach in different disciplines, at different stages of learning, and in different content domains has produced some misapplications and misconceptions of PBL. This suggests a need, as Richardson (2005, p. 51), points out, to "develop an authoritative classification of the different ways that PBL has been implemented". This would allow future research to identify the key characteristics that differentiate successful from unsuccessful PBL. Even given this lack of classification, Richardson (2005, p. 51), goes on to state that "the evidence suggests that the implementation of PBL can bring about measurable changes in students' performance that are of theoretical and practical importance. Nevertheless, all recent reviews of the effectiveness of PBL have shown that the observed effect

[2] This definition, which is cited on many websites, has been attributed by some to Barrows & Kelson; however, Professor John T. E. Richardson reports correspondence from Barrows stating that he didn't formulate this definition. The author would like to thank Professor Richardson for his help on this matter.

sizes are extremely heterogeneous. […] So the answer to the question ‘Does PBL work?’ is ‘it depends’”. Factors that impact on the effectiveness of PBL will now be examined in more detail.

2.4.2. The Effectiveness of PBL

The research literature on the value of PBL analyses the advantages and disadvantages of the PBL method. While some aspects of PBL are considered highly effective, the effectiveness of other aspects is disputed (Albanese, 2000; Albanese & Mitchell, 1993; Berkson, 1993; Butler, Inman & Lobb, 2005; Newman, 2004a; Norman & Schmidt, 1992; Schmidt, Henny & de Vries, 1992; Vernon & Blake, 1993).

The literature has focused on five principal areas which are discussed below:

- attitudes;
- cost;
- basic knowledge;
- team working skills;
- stress and enthusiasm.

Examples of positive changes in students’ attitudes include those occurring when the University of Southern California introduced a new PBL approach to the teaching of introductory accounting (Pincus, 1995), and when PBL was introduced in medical and managerial education (Bernstein, Tipping, Bercovitz & Skinner, 1995; Bridges & Hallinger, 1991; Schmidt *et al.*, 1992). From these studies, it would appear that PBL students are more positively disposed to their course than non-PBL students. The effects of a more positive attitude were reflected in a greater number of students enrolling on PBL courses, a higher level of interest by students in their major course of study, positive feedback from employers and lecturers, including non-PBL lecturers (Pincus, 1995); lower dropout rates (Bridges & Hallinger, 1991; Pincus, 1995) and favourable comments by students about their PBL course (Bernstein *et al.*, 1995). Schmidt, Henny & de Vries (1992, p. 197) conclude that "Problem-Based curricula do appear to provide a friendlier and more inviting educational climate."

Finucane *et al.* (1998, p. 447) say that “[m]ost students enjoy the active participation which PBL fosters and consider the process to be relevant, stimulating and even fun". This view is supported by others (Albanese & Mitchell, 1993; Des Marchais, 1993), while lecturers tend to enjoy the increased student contact (Albanese & Mitchell, 1993). According to both students and lecturers, the removal of traditional barriers between teacher and learner leads to a more positive learning environment (Blight, 1995).

A number of studies (Duch, Groh & Allen, 2001; Hmelo-Silver, 2004; Torp & Sage, 2002) describe the methods used in PBL and claim that PBL enhances a range of learning skills, including the ability to think critically, to analyze and solve complex, real-world problems, to find, evaluate, and use appropriate learning resources, to work cooperatively in teams, to demonstrate effective communication skills, and to use content knowledge and intellectual skills to become continual self-directed learners who reflect on what they learned and the effectiveness of the strategies they employed.

PBL appears to foster self-motivation in learners (Barrows & Tamblyn, 1980; Blumberg & Michael, 1992; Norman & Schmidt, 1992; Shin, Haynes & Johnson, 1993), which may help medical graduates to become life-long learners (Donner & Bickley, 1993; Headrick, Kaufman, Stillman, Wilkerson & Wigton, 1994; Shin *et al.*, 1993). But "while there is both theoretical support and anecdotal evidence that PBL enhances motivation and helps in the development of interpersonal skills, these effects have never been proven". (Finucane *et al.*, 1998, p. 446). Berkson (1993) also supports the view that these effects have not been proven due to the lack of evidence-based studies.

One major criticism of PBL is that it is expensive to implement and places a strain on both staff time and on physical resources. The physical resources required for PBL include well-equipped classrooms and access to high-quality computer and library facilities. PBL curriculum development and training for lecturers and students entail ongoing costs over several years, while staff workload may see a significant increase. Cost considerations must therefore be taken into account in deciding whether to implement a PBL module.

It has been estimated that the staff workload increased by 30% at the University of Sherbrooke in Canada when PBL was introduced (Des Marchais, 1993). Finucane, Johnson and Prideaux (1998) state that class size is the major factor determining staff workload in PBL courses. Compared with the cost of conventional courses, the relative cost of PBL rises as class size increases. A point of equilibrium between the cost of PBL and conventional courses appears to be reached with class sizes of around 40 or 50 (Berkson, 1993; Donner & Bickley, 1993). According to Albanese and Mitchell (1993), PBL may not be economically viable for courses with more than 100 students. Advances in Computing and telecommunications technology may, however, make it viable to introduce PBL to larger classes.

Another major problem with PBL compared with traditional learning is that it is more difficult to cover the curriculum in the same number of hours. Albanese and Mitchell (1993) found that only 80% curriculum coverage can be attained in the same number of contact hours. Albanese and Mitchell (1993) also found shortfalls in students' knowledge following PBL courses compared with students enrolled on traditional courses. This shortfall in knowledge is supported by the findings of other studies (Baca, Mennin, Kaufman & Moore-West, 1990; Eisenstaedt, Barry & Glanz, 1990).

Newman (2004a) carried out a meta-analysis of 91 studies reviewing the effectiveness of PBL. Of these 91 studies, only 12 provided acceptable data and were included in the analysis. Newman (2004a) showed a negative mean effect size of (-0.3) for students' acquisition of knowledge. A negative effect size of (ES = -0.22) on knowledge was also reported by Dochy *et al.* in their meta-analysis of the effects of PBL (2003, p. 548). However Dochy *et al.* state "that students in PBL remember more of the acquired knowledge" (*ibid*, p. 543), and they go on to suggest that a possible explanation for this is the emphasis placed on elaboration in PBL (Schmidt, 1990), as elaboration promotes the recall of declarative knowledge (Gagné, 1978; Wittrock, 1989). As Dochy *et al.* say, "[a]lthough the students in PBL would have slightly less knowledge…, their knowledge has been elaborated more and consequently they have better recall of that knowledge." (2003, p. 543)

This finding is supported by Albanese and Mitchell (1993) who found that while PBL students performed worse than their peers on traditional courses in examinations immediately following the PBL module, there was no difference between the marks obtained by the two groups in later tests given three months later and again two years later. Based on this finding, Albanese and Mitchell suggest that PBL learning may become more deeply rooted than traditional lecture-based learning. It should be noted that lecturer enthusiasm may have a positive impact on the effectiveness of PBL (Marsh, 1987), and the introduction of PBL in less fertile educational environments may be more problematic (Finucane *et al.*, 1998). An examination of the effect of PBL on skills reveals a different picture, with Dochy *et al.* reporting a positive effect size of 0.46 for PBL (2003, p. 548). This distinction between knowledge and skills highlighted by Dochy *et al.* (2003) is used in this study because it provides a framework for a more in depth analysis of the effectiveness of PBL than simply analysing the combined overall module grade.

PBL has been implemented in environments varying in scope from one single course (Lewis & Tamblyn, 1987) up to an entire curriculum (Kaufman *et al.*, 1989). As Dochy *et al.* state, "while the impact of PBL as a curriculum is certainly going to be more profound, a single course can offer a more controlled environment to examine the specific effects of PBL". This view is shared by other researchers (Albanese & Mitchell, 1993; Schmidt, 1990) and is the approach taken in this study.

As most PBL is done in small groups, PBL students tend to prefer cooperative learning and teamwork (Bernstein *et al.*, 1995). Albanese and Mitchell (1993) found that medical graduates who had been taught using PBL preferred to work in group practices rather than on their own, pointing possibly to a need to collaborate within their profession and a difficulty in making independent diagnoses.

Berkson (1993) argues that PBL can initially be stressful for students and lecturers since most students have previously been taught in traditional learning situations. Unlike conventional learning, PBL does not place boundaries on students' learning, so it may provide little direction to students about how to achieve their learning goals. According to Finucane, Johnson and Prideaux (1998), students may become fearful that their learning strategies are wrong, and PBL lecturers should address these fears in PBL tutorials by helping students to master the necessary skills. Unfortunately, however, some PBL lecturers are uncomfortable with their role as facilitators and dislike working with small groups. Other lecturers also resent the fact that PBL is unduly demanding of their time (Finucane *et al.*, 1998).

Finally, it should be noted that most of the accounts of PBL have come from higher-level colleges where PBL has been introduced as part of a major reform of the curriculum, with much enthusiasm and investment in the process. It is therefore likely that this enthusiasm itself positively impacts on PBL effectiveness (Roethlisberger & Dickson, 1939; Rosenthal & Jacobson, 1992), and it may be difficult to differentiate enthusiasm for the new curriculum from real gains in student learning. Lecturer enthusiasm may also have a positive impact on the effectiveness of PBL (Marsh, 1987), and the introduction of PBL in less fertile educational environments may be more problematic (Finucane *et al.*, 1998).

Kirschner, Sweller, and Clark (2006) suggest that minimally guided instructional approaches, such as PBL, are less effective and efficient for novices than guided instructional approaches because they ignore the structures that constitute human cognitive architecture, and specifically that PBL is in conflict with the architecture commonly used by cognitive load theory (Sweller & Sweller, 2006). In other words, as Kirschner *et al.* (2007, p. 116) say: “novices should not be presented with material in a manner that unnecessarily requires them to search for a solution with its attendant heavy working memory load rather than being presented with a solution”.

When Dolmans and Schmidt (2006) researched the cognitive and motivational effects of small group tutorials in PBL, they found that “studies focusing on the motivational effects of PBL demonstrate that group discussion positively influences students' intrinsic interest in the subject matter under discussion” (p. 321). Others disagree: Groves (2005), in her study of medical students, found evidence that questions previous conclusions that PBL curricula foster a deep approach to learning, and suggests that other factors such as work load may be greater determinants of learning approach than curriculum type. Taken together, these findings emphasise the context-dependent nature of learning approach as well as the importance of assessment as a driver of student learning.

Norman and Schmidt (1992, p. 557) carried out a critical review of the PBL literature and they concluded “that (1) there is no evidence that PBL curricula result in any improvement in general, content-free problem-solving skills; (2) learning in a PBL format may initially reduce levels of learning but may foster, over periods up to several years, increased retention of knowledge; (3) some preliminary evidence suggests that PBL curricula may enhance both transfer of concepts to new problems and integration of basic science concepts into clinical problems; (4) PBL enhances intrinsic interest in the subject matter; and (5) PBL appears to enhance self-directed learning skills, and this enhancement may be maintained". A positive link between PBL and intrinsic interest was also found in a study based on biology students in the Netherlands (De Volder, Schmidt, Moust & De Grave, 1986). The same study noted that the increased interest shown did not result in improved learner grades. However, Sweller, Kirschner and Clark (2007) reemphasize the importance of randomized, controlled experimental tests of competing instructional procedures and point out a number of methodological flaws in the experiments carried out by De Volder *et al.*

These findings appear to point to a link between PBL as an instructional method and increased intrinsic motivation, possibly through the promotion of mastery goals over performance goals. However, researchers do not concur on the value of PBL as a learning approach, and further research is required to determine whether the apparent enhanced motivation observed with PBL compared to traditional learning can be applied across student groups and disciplines, including low attainment learners. From the research, it would appear that PBL would be better employed in later academic years and not with first year college learners, but this assertion also requires further investigation to determine whether it holds across student groups and disciplines.

It has been suggested that the PBL environment, where students work together as a team, focused on mastering a problem, promotes mastery goals (Barrett *et al.*, 2005; Dolmans & Schmidt, 2006), and hence increases learner motivation, but establishing a causal link requires more research.

2.4.3. PBL and Approaches to Learning

Two main approaches to studying have been identified: a deep approach based on mastery and understanding of the material; and a surface approach based on memorising of the course material for the purposes of assessment performance

(Laurillard, 1979; Marton, 1976; Ramsden, 1979). These approaches to learning have been investigated by a number of researchers and a number of instruments have been developed to measure them (Biggs, 1988, 1993; Entwistle & Waterston, 1988).

Richardson (2005, p. 43) notes that "the same students may exhibit different approaches to studying in different courses". Recent research has shown that student approaches are moderated by the academic demands, the quality of teaching, and the nature of assessment on different courses. Richardson (2005, pp. 43-44) clarifies that "[t]he choice of one approach to studying rather than another seems to depend upon the content, the context, and the demands of specific learning tasks".

There is some evidence that PBL enhances students' approaches to learning, their perceptions of the quality of their course, their conceptions of learning (Sadlo & Richardson, 2003), and their academic attainment (Deek *et al.*, 1998; Kay *et al.*, 2000). Sadlo (1997) conducted a study which compared 255 students' approaches to studying on occupational therapy courses in six different countries. Two of the courses used Problem-Based Learning teaching methods, two used a hybrid of problem-based and traditional methods, and two used traditional lecture-based teaching methods. Students in their second year were given a short version of the Approaches to Studying Inventory (ASI) devised by Richardson (1990). The students on the problem-based curricula scored higher on deep learning approach measures and lower on surface learning approach measures than students taking the traditional curricula, while students following the hybrid curricula obtained intermediate scores on both measures, implying that the quality of learning tends to increase with the extent to which a problem-based approach has been implemented by an institution (Sadlo & Richardson, 2003). These results suggest that, as Richardson (2005, p. 45) puts it, "the use of PBL *can* bring about desirable changes in students' approaches to studying" and more generally suggest:

> that changes in the design and delivery of particular courses affect how students tackle those courses, and in particular that desirable approaches to studying could be promoted by appropriate course design, teaching methods and modes of assessment.
>
> (Sadlo & Richardson, 2003, p. 254)

This general view is supported by other studies. Gibbs (1992) highlights ten studies in which students moved away from a superficial reproducing approach to studying towards one involving them in a search for understanding through the introduction of new teaching, learning and assessment methods. Nonetheless, a number of studies have found PBL to be largely ineffective in inducing desirable approaches to studying. Indeed some argue that PBL is generally ineffective (Sweller *et al.*, 2006; Sweller *et al.*, 2007), especially for novices, and that discovery learning techniques have failed in the teaching of computer programming (Mayer, 2004). Richardson (2005, p. 45) asks "why is this the case?" suggesting that "[o]ne possibility is that the effects of contextual factors are mediated by students' perceptions of their academic environment. Consequently, PBL and other teaching interventions will not be effective unless they also bring about changes in the students' perceptions".

Sadlo (1997) examined students' perceptions of the academic quality of their courses using the Course Experience Questionnaire (CEQ) devised by Ramsden (1991). The results show, as Richardson (2005, p. 45) points out, that "the use of PBL *can* enhance students' perceptions of their programme in terms of the teaching, the assessment, their own independence and the acquisition of generic skills, but probably not in terms of the clarity of goals and standards or their workload". However, Sadlo and Richardson (2003, p. 268) add that "students at problem-based schools might judge their programs more favourably because they have adopted more congenial ways of studying". Nonetheless, on traditional courses students' approaches to studying still vary, even when taking into account variations in their perceptions of their course. This may be because, as Richardson (2005, p. 46) puts it, "students may adopt one approach rather than another depending on their conceptions of learning and their conceptions of themselves as learners".

Marton (1976) suggests that students who take a deep approach to learning take on an active role and see learning as something that they themselves do, whereas students who adopt a surface approach take a passive role and see learning as something that happens to them. Building on the work of Säljö (1979), Van Rossum and Taylor (1987) and Morgan *et al.* (1981), Marton *et al.* (1993) say students have a limited number of conceptions of learning, and summarises them as follows: increasing one's knowledge; memorising and reproducing; applying; understanding;

seeing something in a different way; changing as a person. These conceptions represent a development hierarchy through which students proceed during the course of their studies. This seems to suggest that PBL may be better suited to later stages of learning rather than with novices.

Savin-Baden (2000) suggests that students' conceptions of learning and their conceptions of themselves as learners are a key factor in any attempt to implement Problem-Based Learning effectively. Claims are made for PBL that it promotes improvements in students' conceptions of learning to a greater extent than traditional curricula, thereby increasing their potential to be lifelong learners (Savin-Baden, 2000). However, Richardson (2005, p. 49) points to the need for "more systematic research both to confirm the role of PBL in encouraging the development of more sophisticated conceptions of learning and also to identify the conditions under which this does and does not occur". Richardson (*ibid*, p. 49) adds that "it can also be argued that PBL actually presupposes more sophisticated conceptions of learning on the part of the students, and this might explain why some students have difficulty adapting to PBL". This might be an explanation for the high dropout rates reported in the Newman meta-analysis on the effectiveness of PBL (Newman, 2004a; Utley, 2004), and suggests the need for further research into students' conceptions before they commence a Problem-Based Learning course.

2.4.4. PBL from the Teacher's Perspective

Richardson (2005, p. 54) points out that "PBL presupposes a student-centred and learning-orientated conception of teaching on the part of the teacher". However, the evidence suggests that it is difficult to change teachers' conceptions of teaching (Trigwell & Prosser, 1996). Richardson (2005, p. 54) suggests that "[t]his might explain why some teachers have difficulty adapting to PBL or accepting it as an approach. In particular, teachers who have a teacher-centred conception of teaching through experience with a subject-based curriculum may well have considerable difficulty adapting to a problem-based curriculum". Contextual factors can frustrate teachers' intended approaches to teaching (Gibbs, 1992). In particular, as Richardson (2005, p. 54) points out, "situational factors will tend to undermine attempts to implement PBL in contexts where subject-based curricula are well established". Students can demand a more didactic approach from teachers (Newman, 2004b, p.

131), while Richardson (2005, p. 54) states that "staff who hold traditional, teacher-focused conceptions of teaching will raise issues about standards and coverage of the curriculum".

2.5. Learner Motivation and Self-Efficacy

Given the evidence that PBL improves learners' motivation and self-efficacy, two theories of motivation will now be examined that may help to explain observed learner behaviours and clarify the reasons behind any increase in learner motivation seen on PBL courses. This discussion also includes an overview of the literature on self-efficacy to provide the theoretical groundwork for exploring the relationship between PBL and learners' programming self-efficacy.

According to Elliot and McGregor (2001, p. 501), "[o]ver the last two decades, a majority of the theoretical and empirical work conducted in the achievement literature has used an achievement goal perspective". Pintrich (2000a, p. 92) states that "current achievement goal constructs address the issue of the purpose or reason students are pursuing an achievement task as well as the standards or criteria they construct to evaluate their competence or success on the task". The achievement goal construct was developed in the 1970s and 1980s by Carol Ames, Carol Dweck, Marty Maeher and John Nicholls, and the work of Dweck and Nicholls has been particularly influential (Ames, 1984; Ames & Archer, 1988; Dweck, 1975, 1986; Dweck & Reppucci, 1973; Maeher, 1983; Nicholls, 1976, 1978, 1979, 1980).

In a series of studies, Dweck *et al.* found that children of similar ability differed in their responses to failure (Diener & Dweck, 1978, 1980; Dweck, 1975; Dweck & Reppucci, 1973). Some pupils persisted whilst other pupils chose to move on to another task. Dweck *et al.* (1986; 1983) sought to explain this by suggesting that in achievement settings, individuals operate with one of two goals. Firstly, to try and understand as much as possible about the task they are attempting, which Dweck called learning or mastery goals, or secondly to perform well and outdo their peers, which Dweck called performance goals.

Brophy (2004, p. 91) suggests that because performance goals emphasise maintaining a good impression of oneself, they should be labelled "ego-protective goals" or "ability display goals". Therefore, it is not just about achieving a certain level of performance, it is about how that performance is perceived by others.

According to Nicholls (1976, 1978, 1980), children do not distinguish between ability and effort until about the age of 12, when they start to construe ability as a fixed capacity. From this perspective, high ability is inferred when one outperforms others while expending equal effort, or performs the same as others while expending less effort. Although they use different terminology, the work of Nicholls and Dweck has many similar features. Indeed, Ames and Archer (1987, 1988) proposed that the theories on the achievement goal concept by Dweck, Nicholls and others (Ames, 1984; Covington, 1984; Maeher, 1983) had a common base and justified the adoption of a common mastery/performance dichotomy terminology. Ames and Archer (1988) assessed students' classroom goal perceptions, mastery or performance, and linked these perceptions to students' learning strategies, task choices, attitudes, and attributions, and they examined how different combinations of mastery and performance perceptions correlated with these processes and outcomes.

Maehr and Midgley (1991) postulated the need to promote mastery goals over performance goals. Mastery goals were associated with positive learning behaviours, e.g. attributing failure to insufficient effort, using failure information diagnostically, and sustained persistence in the face of failure. Performance goals, when accompanied by high confidence, also led to the mastery pattern. However, when performance goals were accompanied by low confidence, they were associated with negative learning behaviours, such as helplessness upon failure, attributing failure to lack of ability, negative expectancies, and avoidance of subsequent challenges (Diener & Dweck, 1978, 1980; Dweck, 1975). A belief that ability is a stable entity was posited to lead to performance goal adoption whereas a belief that ability is malleable was posited to lead to mastery goal adoption (Dweck & Leggett, 1988). While both Nicholls and Dweck view achievement goals as applicable to situation specific as well as dispositional levels of analysis, they also identified some limitations of dispositional constructs.

Nicholls (1979) stated his belief in equal motivational opportunities for all, and that from this standpoint alone mastery/task involvement should be championed over performance/ego involvement, as only mastery involvement provides equal opportunity for all. In this tradition Ames (1990, 1992) designed the TARGET (Tasks, Authority, Recognition, Grouping, Evaluation, Time) intervention framework, which aimed to create classroom environments that would enhance mastery goal adoption and minimize performance goal adoption in students.

The view that the effects of performance goals were entirely negative was challenged by a number of researchers, particularly Harackiewicz (Butler, 1992; Harackiewicz & Elliot, 1993). It was shown that performance goals had a positive effect or no effect in certain types of achievement contexts (Koestner, Zuckerman & Koestner, 1987), and for individuals with certain types of personality dispositions (Harackiewicz & Elliot, 1993; Harackiewicz & Sansone, 1991). A number of studies indicated that a high mastery and high performance goal combination was linked to the best processes and outcomes (Bouffard, Boisvert, Vezeau & Larouche, 1995; Wentzel, 1993), while others supported the high mastery and low performance goal combination (Meece & Holt, 1993; Pintrich & Garcia, 1991).

Elliot and Harackiewicz (1996) suggested adding a third motivation, the motivation to avoid performing badly, an avoidance goal. Further work suggested that the goals divided into approach and avoidance orientations: consequently a distinction can be made between performance-approach, performance-avoidance, and mastery-approach (Elliot, 1999; Elliot & Church, 1997; Elliot & Harackiewicz, 1996). Mastery-avoidance was identified at a later stage and will be discussed later in this review.

A number of research studies tested the trichotomous framework, as it was termed, and it was shown that goals could be measured and separated by factor analysis, and that each type of goal had different predictive utility (Elliot & Church, 1997; Middleton & Midgley, 1997). Additional empirical work showed the usefulness of the framework (Elliot & McGregor, 1999) and that an individual's perceived confidence could be used to predict selection of achievement goals (Lopez, 1999).

Using the trichotomous achievement goal framework, Church, Elliot and Gable (2001) carried out two studies to examine the classroom impact of learners' goal choices. Using the Patterns of Adaptive Learning Styles questionnaire (PALS)

(Midgley *et al.*, 2000), pupils were asked to rate their perceptions of their classrooms as either performance-focused or mastery-focused. They found that performance goals lead to shallow learning styles, an increased level of cheating and less willingness to seek help, while mastery goals lead to deep learning styles and a more positive attitude to school (Church *et al.*, 2001). These findings were also supported by other studies (Elliot & McGregor, 2001, p. 515). However, when, in attempting to identify why some students excel in their college classes and develop an interest in an academic discipline, Harackiewicz *et al.* (2000) examined both the short-term and long-term consequences of students' achievement goals in an introductory psychology course, they found that mastery goals positively predicted subsequent interest in the course, but not course grades, and performance goals positively predicted grades, but not interest. The finding that performance goals have been linked to positive outcomes in terms of graded academic performance is supported by other studies (Barron & Harackiewicz, 2001; Bouffard, Vezeau & Bordeleau, 1998; Elliot & Church, 1997; Elliot & McGregor, 2001; Harackiewicz, Barron, Carter, Lehto & Elliot, 1997; Harackiewicz *et al.*, 2000; Skaalvik, 1997; Wolters, Yu & Pintrich, 1996).

This phenomenon might be explained by suggesting that in the context of a competitive environment with an emphasis on attaining grades, adopting a mastery approach may help students understand material but it may also result in them spending too much time studying non-specific or non-examinable material. Therefore, adopting a mastery approach, given existing higher education assessment methods and limited study time, may be maladaptive when it comes to getting high marks.

Elliot and McGregor (2001, p. 501) suggest that “competence is at the conceptual core of the achievement goal construct”, and that competence can be defined and evaluated according to “whether one has acquired understanding or mastered the task (absolute), improved one’s performance or developed one’s knowledge or skills (intrapersonal), or performed better than others (normative)”. This view is also supported by Dweck (2005).

Elliot and McGregor (2001, p. 501) further state that “competence can be valenced in that it is either constructed in terms of a positive, desirable possibility (i.e. success) or a negative undesirable possibility (i.e. failure)”. These approach and avoidance tendencies are present in infancy, and are ubiquitous across situations (Elliot & Covington, 2001). Rawsthorne and Elliot (1999) carried out a meta-analysis and observed that in general, measures classified as performance-approach tended to produce a positive set of outcomes and processes, while those classified as performance avoidance produced a negative set of outcomes and processes.

Harackiewicz *et al.* (2002a) suggest that goal theory needs to be revised in three ways: to confirm the separation of approach and avoidance, and this is generally accepted (Elliot & Harackiewicz, 1996; Pintrich, 2000c); to identify the ways in which performance-approach goals can combine with mastery goals to promote optimal motivation (Barron & Harackiewicz, 2001; Pintrich, 2000b); and most controversially, that the positive potential of performance-approach goals relative to mastery goals be recognised (Harackiewicz, Barron & Elliot, 1998; Harackiewicz, Barron, Tauer & Elliot, 2002b).

On the concept that individuals can hold multiple goals, McGregor and Elliot (2002, p. 393) state that: “[r]ecent work on multiple goals indicates that mastery and performance-approach goals are not necessarily incompatible, and that the combination of mastery goals and performance-approach goals may indeed be optimal for some outcomes”.

The value of a mastery goal orientation in promoting adaptive patterns of cognition, effect and behaviour, is generally recognised, as are the maladaptive patterns of learning associated with performance avoidance. The role of performance approach goals is more controversial: Midgley, Middleton and Kaplan (2001) argue that while some studies (Bouffard *et al.*, 1995; Elliot & Church, 1997) have found performance approach goals to have positive outcomes, these outcomes only arise when mastery goals are also present, and the positive outcomes may indeed be ones that only measure surface level learning, such as high scores on multiple choice tests, memorization and rote learning for exams.

Harackiewicz *et al.* (2002a) disagree, challenging Midgley *et al*'s methodology, suggesting that in the studies they reviewed they mixed up both performance-approach goal measures and general unspecified performance goal measures. Harackiewicz *et al.* (2002a) also disagree with Midgley *et al*'s view of the outcomes of research studies, claiming that they have overstated the maladaptive outcomes of performance-approach goals and that outcomes for performance-approach goals are more consistent than Midgley *et al.* suggest. A point to note is that Harackiewicz *et al*'s (2002a) review focused on college level studies and their findings may not be generalisable to younger children.

Harackiewicz *et al.* (2002a) state that while performance-approach goals have been shown to be unrelated to some adaptive variables, such as deep processing (Elliot, McGregor & Gable, 1999; Harackiewicz *et al.*, 2000) and intrinsic motivation (Church *et al.*, 2001; Elliot & Church, 1997; Harackiewicz *et al.*, 2000), they nonetheless can promote positively related adaptive variables such as task value (Church *et al.*, 2001), academic self-concept (Skaalvik, 1997), effort expenditure (Elliot *et al.*, 1999; Lopez, 1999) and performance attainment (Barron & Harackiewicz, 2001; Bouffard *et al.*, 1998; Elliot & Church, 1997; Elliot & McGregor, 2001; Harackiewicz *et al.*, 1997; Harackiewicz *et al.*, 2000; Skaalvik, 1997; Wolters *et al.*, 1996).

Kaplan and Middleton's (2002) response to Harackiewicz *et al.* (2002a) helps illuminate some philosophical differences between the two groups of researchers, and indeed educationalists in general. Kaplan and Middleton (2002, p. 647) state that "the discrepancy between our agreement over research findings and our disagreements about the meaning of these findings…may be grounded in somewhat different ideologies concerning social science" and that the view that "the simplistic statement that 'mastery goals are always good and performance goals are always bad' (Harackiewicz *et al.*, 2002a, p. 643) is not an inherent underlying assumption of achievement goal theory. Rather, it is a value that is based on the type of success that one believes should be emphasized in the achievement context." Nicholls sees this not as a question of theory but as an ethical question (Nicholls, 1989). Others agree: Urdan (1997, p. 136) suggests that the positive associations of performance goals with outcomes "may be due more to the way schools are than the way they could be.

Task [mastery] goals represent a hope that all students, not just those who think they are more able than others or those that enjoy beating others, can become actively involved in school and be motivated to learn for the sake of learning."

Kaplan and Middleton (2002) make the point that from an equality perspective, "instead of interpreting the finding that performance-approach goals contribute to achievement whereas mastery goals contribute to interest as indicating that the most desirable motivational orientation is high performance approach–high mastery, one might question the educational characteristics of a context in which a focus on mastering and understanding the material does not contribute to a higher grade."

Returning to whether performance-approach goals are adaptive, more recent research paints a more complex picture. Brophy (2005, p. 167) states that "evidence is emerging that students disposed toward performance-approach goal orientations in the present are at risk for shifting to performance-avoidance goal orientations in the future and that students' responses to performance-approach goal scales are more reflective of their past achievement histories in the domain than of motivational states likely to exert forward effects on subsequent achievement". Brophy suggests that the term performance goals be phased out in favour of output goals to minimise the social comparison connotations carried by the term performance goals.

Bråten *et al.* (2004, p. 232) suggest that "another possibility [for] the mixed pattern of findings for performance goals results from a failure to consider a moderating influence of self-efficacy beliefs on performance goal effects". However, when they tested this hypothesis they found no (*ibid*, p. 241) "significant interaction of self-efficacy with performance-approach goals on self-regulatory strategies", but "found evidence that perceived self-efficacy moderated the relation between performance-avoidance goals and reported use of self regulatory strategies for business administration students but not for student teachers". These mixed results may be explained by the fact that the sample of business students only included high achievers and that the business students were immersed in a more competitive grade-focused learning environment than the student teachers, suggesting that achievement goals are context specific. The nature of the interaction between performance avoidance and self-efficacy was unexpected, with "a negative effect of increased performance-avoidance goal orientation for students with high self-efficacy and a positive effect of increased performance-avoidance goal orientation for students with low self-efficacy".

The trichotomous achievement goal framework was modified when Elliot and McGregor (2001) suggested there was also a fourth goal construct, mastery-avoidance. Elliot (2005, p. 61) defines this as where individuals "focus on avoiding self-referential or task-referential incompetence". This construct was tested by Elliot and McGregor (2001) as part of a 2 X 2 achievement goal framework in three studies conducted on undergraduate students, and they (*ibid*, p. 501) found "distinct empirical profiles for each of the achievement goal" constructs.

Further research studies tested the 2 X 2 framework and the mastery-avoidance goal construct, showed them to be viable, and identified the antecedents and consequences of the mastery-avoidance goal to be similar to performance-approach goals rather than mastery-approach goals (Elliot & Reis, 2003; Finney, Pieper & Barron, 2004; Karabenick, 2003, 2004).

Other researchers (Ford, 1992) have adopted a broader perspective on goals and motivation, arguing that there are many different kinds of goals that individuals can have in achievement settings and that other goals may have equal or greater importance than mastery and performance goals. Some of the other suggested goals include social goals that focus on building friendships (Wentzel, 1989); intimacy goals that focus on maintaining close friendships; social-responsibility goals where the focus is on meeting social obligations; status goals where inviduals want to be admired by their peer group (Urdan & Maeher, 1995); extrinsic goals where individuals are seeking a reward (Brophy, 2004, p. 100; Maeher, 1983; Pintrich & Garcia, 1991); and work-avoidant goals where the focus is on trying to get away with as little work as possible (Elliot, 1999; Nolen, 1988). Elliot and Thrash (2001, p. 150) suggest it is best to conceptualize work avoidance goals as objectives that individuals have in achievement settings when they do not have an achievement goal of any type.

There are issues around the limitations of the survey and experimental methods used in the goal theory research to be considered: for example, when interviewed, students do not spontaneously mention the types of goals that appear on goal questionnaires (Brophy, 2005; Urdan & Mestas, 2006). There may also be issues about how generally applicable results are, given that so much of the research has been done with undergraduate psychology students or in laboratory settings. These issues raise important questions about the external validity of such studies, and highlight the need for careful consideration when extrapolating to the classroom findings derived from experimental research. As Urdan and Mestas (2006, p. 364) suggest: "achievement goals may be more complex and multidimensional than often depicted in research, and this complexity warrants further examination".

There are empirical links between achievement goal theory and self-determination theory, suggesting that the controlling features of performance orientation undermine autonomy and foster an external locus of causality, whereas mastery orientation facilitates autonomy of behaviour (Brunel, 1999; Ntoumanis, 2001).

Self-determination theory is based on an organismic-dialectical meta-theory (Deci & Ryan, 1985). It is organismic in the sense that individuals have a natural and innate tendency to seek out challenges and be curious, and dialectic in the sense that "[t]his natural human tendency does not operate automatically, however, but instead requires ongoing nutriments and supports from the social environment in order to function effectively" (Williams & Deci, 2007b).

Self-determination theory suggests that people are fundamentally motivated if they perceive themselves as the origin of their actions, and not a pawn being manipulated by somebody or something else (Ryan & Grolnick, 1986). Deci and Ryan (2000, p. 231) define autonomy as "volition - the organismic desire to self-organise experience and behaviour …". Whenever a person's perception of autonomy changes, their locus-of-causality changes and as soon as they perceive an external reason for engaging in an activity, they also change their locus-of-causality. People's levels of perceived autonomy or perceived choice strongly determine the type of motivation they have.

Deci and Ryan (1985) proposed a self-determination continuum to describe motivational variables with different degrees of self-determination. From higher to lower self-determination, these are: intrinsic motivation, extrinsic motivation (integrated regulation, identified regulation, introjected regulation and external regulation) and amotivation. Intrinsically motivated behaviour has the highest self-determination, occurs without the incentive of external rewards and is undertaken out of interest in the activity itself rather than the outcomes of the activity.

Extrinsic motivation refers to activities that are carried out as a means to an end and not for their own sake (Deci & Ryan, 1985). Extrinsic motivation comprises four dimensions. The first, integrated regulation, represents the most self-determined form of the internalization process. It refers to behaviours where individuals may be doing something for extrinsic reasons but they fully agree with those reasons. The second dimension, identified regulation, describes behaviours that are highly valued and performed out of choice but the individual does not fully enjoy doing the activity. Both integrated and identified regulation represent self-determined forms of behaviour but they are still extrinsic because individuals perform them to achieve personal goals and not for their inherent appeal (Deci & Ryan, 2000).

The third dimension of extrinsic motivation is introjected regulation, which refers to behaviours that individuals perform to achieve social recognition, out of pride or to avoid feelings of guilt. The fourth dimension, external regulation, describes behaviours only regulated through external means, such as rewards or constraints. Both introjected regulation and external regulation are considered to be controlling or low self-determined types of motivation. Finally, amotivation, or a lack of intrinsic or extrinsic motivation, is evident when individuals perceive no contingencies between their actions and the end result, and question whether they should still be involved in a particular activity. It is viewed as a non-self-determined type of motivation.

A learner being intrinsically motivated has many benefits and leads to greater persistence, enjoyment and interest (Deci, 1971), greater conceptual understanding (Grolnick & Ryan, 1987), and a more positive attitude to school (Miserandino, 1996). Being extrinsically motivated has negative affects and leads to anxiety (Ryan & Connell, 1989) and an increased probability of dropping out of school (Vallerand & Bissonnette, 1992). However, this view has been challenged as simplistic, with some researchers arguing that moderate amounts of both intrinsic and extrinsic motivation are optimal for learning.

Self-determination theory suggests that rather than increasing intrinsic motivation, efforts must be made to try and reduce extrinsic motivation and make learning as autonomy- supportive as possible. The literature postulates that this can be achieved by teachers spending more time listening and less time giving directives, asking the students what they want, answering students' questions specifically, promoting the value of education, and providing a clear rationale for tasks (Reeve, Bolt & Cai, 1999, p. 546; Urdan & Turner, 2005, p. 304).

A number of studies suggest that the PBL teaching method promotes perceived autonomy and self-determination (Butler, 1999; van Grinsven & Tillema, 2006), which in turn can have a positive effect on students' motivation (Deci & Ryan, 1985; Hidi & Harackiewicz, 2000) Goal theory and self-determination theory are two complementary theories of motivation. They share an emphasis on promoting feelings of autonomy and providing a non-competitive learning environment.

More evidence of a link between PBL and learner motivation can be seen in studies on the introduction of PBL in medicine, accountancy and managerial education (Bernstein *et al.*, 1995; Bridges & Hallinger, 1991; Pincus, 1995; Schmidt *et al.*, 1992). These studies show positive changes in student attitudes and motivation compared to non-PBL students. The positive changes include a greater number of students enrolling on PBL courses, a higher level of interest by students in their major course of study, positive feedback from employers and lecturers, including non-PBL lecturers (Pincus, 1995), lower dropout rates (Bridges & Hallinger, 1991; Pincus, 1995), and favourable comments by students about their PBL course (Bernstein *et al.*, 1995). Schmidt, Henny and de Vries (1992, p. 198) conclude that "Problem-Based curricula do appear to provide a friendlier and more inviting educational climate." According to both students and lecturers, the removal of traditional barriers between teacher and learner leads to a more positive learning environment (Blight, 1995).

Another factor in determining learner motivation is the role of students' self-efficacy beliefs. Bandura (1994) asserts that highly efficacious students see difficult tasks as challenges to be mastered, not threats to be avoided. In Computing, efficacy levels have been found to affect the type of interaction with computers, which in turn affects proficiency (Compeau & Higgins, 1995). This finding is supported by studies that have shown that self-efficacy is related to computer anxiety as well as learning performance and computer literacy (Beckers & Schmidt, 2001; Chou, 2001). An individual's computer self-efficacy and outcome expectations were found to be positively influenced by the encouragement of others in their work group (Compeau & Higgins, 1995). This suggests that PBL groups may promote increased computer self-efficacy.

Bergin and Reilly (2005) carried out a study at an Irish university on the role of motivation and comfort-level on a first-year object-oriented Java programming module. The module was taught using a Problem-Based Learning approach (O'Kelly *et al.*, 2004). Bergin and Reilly (2005, p. 293) found "that intrinsic motivation had a strong correlation with programming performance as did self-efficacy for learning and performance, [with] $r=0.512$, $p < 0.01$ and $r=0.567$, $p < 0.01$ respectively".

Dunlap (2005) examined how students' self-efficacy, as it relates to becoming Software Development professionals, changed while involved in a PBL environment. In a study of 31 undergraduate university Computer Science students on a 16-week course in Software Development during their final semester prior to graduation, Dunlap, using a self-efficacy scale as pre-and post-measures, and guided journal entries as process data, found that students increased their levels of self-efficacy. In explaining these findings, she suggests that specific instructional strategies used in PBL, namely the use of authentic problems of practice, collaboration and reflection, are the catalysts for students' improved self-efficacy.

Venkatesh and Davis (1996) suggest that a computer user with a high level of computer self-efficacy feels a stronger sense of control over the computer-based activities being performed. Self-determination theory would suggest that this increase in autonomy may lead to increased intrinsic motivation.

Another significant issue is students' attributions, what they perceive as the causes of success or failure (Schunk, 1991). Individuals can perceive success or failure as either independent of their own actions and thus externally controlled, or dependent on the way they behave and thus internally controlled. An attribution such as effort would most likely be considered controllable, whereas luck or task difficulty would be considered uncontrollable (Weiner, 1983). Whether students believe they have control over learning outcomes affects how much effort they expend in learning and how long they persist in their efforts. This 'locus of control' has been shown to be true for software developers, with self-esteem and locus of control having a direct relationship to perceived performance (Rasch & Tosi, 1992).

Overall these findings point to a link between PBL as an instructional method and increased intrinsic motivation, possibly through the promotion of greater student autonomy. However, researchers do not concur on the value of PBL as a learning approach, and further research is required to determine whether the apparent enhanced motivation observed with PBL compared to traditional learning can be applied across student groups and disciplines, including low attainment learners.

2.6. Summary

A review of the literature shows that Computing courses have high failure and dropout rates and there is clear evidence that learning to program is problematic. It has been shown that programming is a difficult task and the results achieved by students do not correlate well with their other academic results. Programming requires a diverse range of skills and the application of complex knowledge and associated strategies. The literature shows that the main problem for novices is program design and planning, not code syntax. The programming strategies that they employ appear to account for the distinction between effective and ineffective novices. However, most introductory programming courses are conventionally structured with lectures and practical laboratory work; they concentrate on teaching programming knowledge but not on the strategies needed to use this knowledge. There is evidence that there is room for improvement in the way students are taught programming. A number of researchers suggest that constructivist methods may be more generally effective, and PBL in particular has been shown to produce improvements in outcomes (Deek *et al.*, 1998; Kay *et al.*, 2000).

Threshold Concepts were examined as a framework that may help explain why learners find computer programming so troublesome. Two aspects of programming were identified that may be Threshold Concepts in Computing: object orientation and levels of abstraction, both of which are included in any object-oriented programming course. The literature suggests that constructivist approaches and PBL in particular can help learners with the disjunction caused by Threshold Concepts (Savin-Baden, 2000). This, coupled with the evidence that PBL has improved outcomes on programming courses, suggests that PBL would be a good instructional choice for the teaching of programming. The literature on PBL shows differential effects on learners' knowledge and skills acquisition. PBL can be effective in improving students' skills; however, its effect on knowledge has not been proven. The effect of PBL on skills acquisition may be particularly relevant to programming courses given that some researchers suggest that computer programming is more appropriately viewed not as a body of knowledge but rather as a skill or competence-based task (Robins *et al.*, 2003; van Roy & Haridi, 2004).

Low levels of intrinsic motivation and high levels of extrinsic motivation have been identified in programming courses. Studies have found that PBL enhances intrinsic interest in the subject matter (De Volder *et al.*, 1986; Norman & Schmidt, 1992), possibly because the PBL teaching method promotes perceived autonomy and self-determination (Blumberg & Michael, 1992; Butler, 1999; Shin *et al.*, 1993; van Grinsven & Tillema, 2006), which in turn can have a positive effect on students' motivation (Deci & Ryan, 1985; Hidi & Harackiewicz, 2000). It has been suggested that the PBL environment, where students work together as a team, focused on mastering a problem, promotes mastery goals (Barrett *et al.*, 2005; Dolmans & Schmidt, 2006), and hence increases learner motivation, but establishing a causal link requires more research. These findings appear to point to a link between PBL as an instructional method and increased intrinsic motivation, possibly through the promotion of mastery goals over performance goals.

Dunlap (2005) examined how students' self-efficacy levels, as they relate to becoming Software Development professionals, increased while involved in a PBL environment. In explaining these findings she suggests that specific instructional strategies used in PBL, namely the use of authentic problems of practice, collaboration and reflection, are the catalysts for students' improved self-efficacy. For some Computing students, learning programming is intimidating, giving rise to anxiety and a lack of confidence. Other than measures of general intelligence, novices' programming self-efficacy is the most accurate predictor of success at programming. Cognitive styles and personality traits do not impact on success at programming.

The literature suggests that the use of PBL as an instructional method to teach programming may help improve both learner motivation and self-efficacy. A number of studies show that students who are encouraged to actively engage with and explore programming-related information perform better at problem solving; and working collaboratively on programming problems in groups has been shown to be beneficial, particularly for weaker students (Mayer, 1989; van Gorp & Grissom, 2001; Wills *et al.*, 1999). It was demonstrated that PBL can bring about positive changes in the approaches to study that students employ (Sadlo & Richardson, 2003).

From the literature review a number of initial research questions emerge. These research questions will be examined in detail in the coming chapter in the context of the implementation of a hybrid PBL Java programming module for novices at an Irish higher education establishment. They are:

(1.a) What are the effects of using a PBL model on learner attainment in exams (measuring knowledge) on a first year programming module?

(1.b) What are the effects of using a PBL model on learner attainment in continuous assessment (measuring skills and strategies) on a first year programming module?

(2) What are the effects of using a PBL model on learner self-regulation?

(3) What are the effects of using a PBL model on learners' programming self-efficacy?

(4) What are the effects of using a PBL model on students' approaches to learning and on general learner engagement?

(5) What are the effects of using a PBL model on learner preferences for different types of course and teaching?

Chapter 3 - Context and Research Methodologies

3.1. Introduction

This chapter begins with a description of the context of the study and model of PBL used in Anon College, followed by a reiteration of the five research questions that determine the empirical design, the process and the selection and adaptation of appropriate measurement instruments. The variables measured and methods of analysis utilized are then illustrated, including a description of the experimental design. The instruments and measures used are examined and the different data collection techniques employed are described. The qualitative analysis used is outlined: this includes a review of the participant observations, field notes and interview procedures used. The validity and reliability of the different instruments employed is examined and the experimental controls are presented. Finally, some limitations of the analysis are outlined.

3.2. Context of the Introduction of PBL in Anon College

Following the introduction of a Hybrid PBL model to teach first year Software Development in the Computer Science Department of another Irish College (O'Kelly, 2005), it was decided to apply the same model at Anon College. This decision to implement PBL and to evaluate it using a controlled trial was not made by this researcher; rather it was made by the Department of Computing at Anon College. As was done in the other College, lecturers at Anon College were provided with PBL training to help initiate and develop the PBL programme and to assist them in adjusting to the role of facilitator/mentor/coach (Woods, 1996).

3.2.1. The Hybrid PBL Model Used at Anon College

First year Software Development is traditionally taught at Anon College using a combination of lectures, tutorials and labs (7 hours per week). Due to constraints in the course schedule, it was not possible to increase the total amount of time allocated to Software Development under the PBL model. The major differences between the hybrid PBL model used at Anon College and the pure PBL model are: the short duration of the problems, the continued inclusion of at least one lecture every week and the methods of assessment (which include traditional end term exams). Under the hybrid PBL model, the physical learning environment of classrooms and computer laboratories remained unchanged, but groups were allocated space to work in and resources such as whiteboards, markers and flipcharts. A tutor was also assigned to each group to help manage the PBL process.

Furthermore, all Computing students at Anon College have access to a virtual learning environment using Moodle, where course notes, interactive quizzes, past exam papers, discussion groups, attendance monitoring, etc., are used to support teaching. The PBL students also use Moodle to keep an online PBL journal. The journal was updated by the students on a week-by-week basis, and contained a record of their collaborative work.

3.2.2. Implementation of the Hybrid PBL Module

In addition to the induction programme run by the Department for all Computing students, an additional induction programme was run in the first week of semester for students taking the PBL Software Development module. This was designed specifically to introduce the students to each other and to the concept of teamwork. Students were given an introduction to PBL, the roles involved and the expectations for those roles, and were shown their designated workspaces. Every student in the class was also assigned to a formal group. Ellis and Dick (2000) argue that group size has a number of effects, including the degree of participation possible and the strength of bonds between members. Groups of 7-8 students were decided upon. Gender balance was difficult to achieve with approximately 90% of the class being male each year. Each group developed its own set of ground rules for behaviour and goal achievement, and these rules were reviewed regularly by the group. Each group worked together for the entire semester.

The problems used to teach the PBL module at Anon College were developed by O'Kelly (2005) and are based around specific Software Development learning outcomes. The problems created fall into three broad categories: firstly, extendable conceptual problems, that is, problems that ensure the students focus on core concepts of computer programming in order to solve a problem. These problems involve no programming but require that the students understand programming-related concepts. The problems also allow for increased levels of difficulty to be added to the problem once a solution is found to ensure that the problem sustains the students' interest. The second category of problems used is non-extendable conceptual problems which help students to understand programming-related concepts without performing any programming. This type of problem has just one solution and is not extendable. The third category of problems, programming problems, are typical computer programming problems that the group tries to solve collectively. This type of problem aids the weaker student as he/she gets to see how a stronger student solves a programming problem (O'Kelly *et al.*, 2004).

In selecting the problems to use in the PBL module, a number of factors were considered. Ellis *et al.* (1998) argue that first-year students who are making the transition from a teacher-centred school environment to a more self-directed university environment may need the comfort of a well-defined problem with considerable scaffolding. However, PBL advocates the use of 'messy ill-structured problems' (Mauffette, Kandlbinder & Soucisse, 2004). The problems chosen by the Anon College lecturers fell between these two poles and took into consideration the following needs: each problem should be engaging, engender multiple viable hypotheses, allow enquiry, represent real-world problems, sustain engagement, provide accessible resources for subsequent learning and be based on the current curriculum. The problems were discussed at a weekly meeting between tutors and any perceived difficulties were addressed.

While the amount of lecture time provided for Software Development remained unchanged under the PBL model, the structure of the lectures was altered. Waite *et al.* (2003) argue that if students sit passively while difficult concepts are explained and they are told what is important, they are not taking responsibility for their learning. The approach traditionally employed in lectures begins by introducing the syntax of a particular programming construct. This is demonstrated in isolation and later incorporated into a larger program that solves a particular problem. It has been found that students are able to understand the construct in isolation and recognise it in the sample programme but are unable to transfer this knowledge to their laboratory work.

The PBL approach used copied that used by O'Kelly (2005) and was informed by the work of Deek and Kimmel (1993), Woods (1996) and Waite *et al.* (2003). A problem is presented at the beginning of each PBL laboratory class: each group is asked to generate possible ideas to solve the problem. Then each group is asked to develop an algorithmic solution based on their agreed combined ideas. The scribe then writes this solution in pseudocode on the whiteboard assigned to the group. The tutor facilitates the group during this period of problem solving. The tutor then collaborates with all the students to solve the problem algorithmically with ideas generated from different groups of students. Once a solution to the problem is drafted, the tutor steps through the solution with the students, any difficulties are identified and rectified by the class and the step-through process begins again until such time as a viable solution is reached. At this point the translation of the algorithm to Java code occurs. During this process any programming concepts that students do not understand are flagged and covered later in tutorials. It is the responsibility of each group member to keep their own PBL journal. The PBL journal records the problem solving process and is updated after each PBL session.

The methods used to assess the students summatively remained unchanged under the PBL model (two in-laboratory practical assignments and a paper-based closed book end-term exam). This allowed for a feasible comparison with the previous year's results and with the results of the non-PBL Software Development learners in the same semester, in the knowledge that the only change instigated was the course delivery. Formative assessment of students' performance in the PBL tutorials was introduced.

3.2.3. Development Software Used and Virtual Learning Environment Support

IBM's Eclipse development platform is used in all laboratories at Anon College to run students' Java programmes (Eclipse Foundation, 2004). Eclipse is a fully functional professional development platform and it contains complex Software Development tools that are not required in an introductory programming course. To simplify the process of code creation, the DrJava development environment plug-in for Eclipse was used to allow students to create their programmes. DrJava is designed primarily for students, providing an intuitive interface and the ability to interactively evaluate Java code. It is available free of charge and was developed by the JavaPLT group at Rice University (JavaPLT Group, 2008).

The open-source e-learning platform Moodle was chosen as the VLE (Virtual Learning Environment) for all Computing courses at Anon College. On the first-year Software Development course the VLE is used for a number of purposes: dissemination of course material; formative self e-assessment and guided learning; module forums and learner blogs; learner activity tracking; virtual whiteboard; and mobile phone enquiry-based collaborative learning support. A study was undertaken using interviews and questionnaires of learners' experiences of the VLE and it was found that the majority of learners on the course agreed that the VLE provided useful support for their learning (Doody, O'Reilly, Cardiff & Magee, 2006).

3.3. Research Questions

From the literature review a number of research questions and related hypotheses emerged. These are outlined in Table 3-1 below:

Table 3-1: Research Questions

Research questions	Hypotheses
(1.a) What are the effects of using a PBL model on learner attainment in exams on a first year programming module? (1.b) What are the effects of using a PBL model on learner attainment in continuous assessment on a first year programming module?	(1.a) Learners in the PBL group will score higher in exams than those in the control group. (1.b) Learners in the PBL group will score higher in continuous assessment than those in the control group.
(2) What are the effects of using a PBL model on learner self-regulation?	(2) Learners who complete the PBL course will have a higher degree of intrinsic motivation than those in the control group.
(3) What are the effects of using a PBL model on learners' programming self-efficacy?	(3) Learners in the PBL group will show a higher degree of programming self-efficacy than those in the control group.
(4) What are the effects of using a PBL model on students' approaches to learning and on general learner engagement?	(4) Learners in the PBL group will show higher scores on meaning orientation and lower scores on reproduction orientation than those in the control group.
(5) What are the effects of using a PBL model on learner preferences for different types of course and teaching?	(5) Learners in the PBL group will show a greater preference for courses and teaching that supports deep learning (as opposed to surface learning) than those in the control group.

The different research questions and related hypotheses probe the impact of PBL on each of the different specified outcomes, namely attainment, motivation, programming self-efficacy, approaches to studying and preference for a deep approach to learning. These research questions and hypotheses are tested and explored using the methods of analysis outlined in subsequent sections of this chapter.

3.4. Participants

Participants in the study were drawn from four cohorts of first-year students who enrolled at Anon College for the academic years 2005/2006, 2006/2007, 2007/2008, and 2008/2009. In all, 398 first year students (294 Computing students and 104 Engineering students) took part in the study. Repeat students taking the module for a second time were excluded from the study, therefore, each year the cohort contained a different set of participants from the previous year.

Demographic details show a learner population profile with a male:female ratio of around 9:1, with all students speaking English as their first language, almost all of Irish nationality, all except one learner between 18 and 23 years of age, and the majority living in areas of Dublin suffering from socioeconomic disadvantage. In general, students in Ireland do not study programming in secondary school and the majority of students taking this module had recently completed second level education.

Ten lecturing staff and four tutors also took part in the study. The lecturing staff comprised five female and five male staff members aged between 28 and 60 years. Nine of these staff members are Irish nationals while one is British. All the lecturing staff are full-time tenured employees of the Department of Computing. The four tutors comprised one male and three female tutors aged between 23 and 26 years. All are post-graduate students at Anon College and all have similar qualifications in Computing. All the tutors come from outside the European Union, with one male from Mexico, two females from Russia, and one female from Turkey. The role of lecturers and tutors was strictly limited to their participation in the study: they were observed working in Software Development laboratories and some were interviewed about their PBL experiences. No staff members were involved in data collection.

3.4.1. Analysis of Learner Participants Background Questionnaires

To provide background and contextual information, a general questionnaire was given to all four cohorts of learners. The questionnaire was distributed at student induction and 284 out of 294 students completed it. The general questionnaire therefore includes students who later dropped out of the course. Learner self report responses were analysed statistically. The questionnaire included data on the following items: gender, socio-economic and family background (via home address), previous programming experience, previous non-programming computer experience, and encouragement by others to pursue Computing as a career. The full questionnaire is given in Appendix F2.

The results show that 270 out of 284 participants (95%) had no previous computer programming experience. The 14 participants who had previous programming experience were split evenly between the PBL and non-PBL groups. Most of those who had some programming experience gained it during their school years, while the rest had some self-learned programming experience. None had any professional programming experience. All the participants had some previous computer usage experience, mainly playing computer games (91%). Only 37 (13%) claimed to have been encouraged by others to pursue Computing as a career. 181 out of 284 (63.7%) participants come from areas of Dublin city classified as disadvantaged. These details are presented in Table 3-2: .

Table 3-2: Learner Background Questionnaires

Total Number of Learners	**Number with Previous Computer Programming Experience**	**Number with any Previous Computer Experience**	**Number Encouraged by Others to Pursue Computing as a Career.**	**Number with a Disadvantaged Socio-Economic or Family Background**
284 students	14 students	284 students	37 students	181 students

Both encouragement and previous programming experience seem to give some indication of future success in Software Development. The correlation coefficients were (r=0.69) for encouragement and (r=0.81) for previous programming experience. However, these results are based on very small sample sizes of (n=37) and (n=14) respectively.

3.5. Methods of Analysis

A mixed method design including both qualitative and quantitative measures was used in this study. A concurrent triangulation strategy was employed to add validity to the research findings (Creswell, 2003, p. 215). The approach is concurrent because both collection and analysis of quantitative and qualitative data was performed at the same phase of the study. It is also a triangulation strategy as qualitative and quantitative measures were used to confirm, cross-validate or corroborate findings.

A quantitative, controlled, experimental research design was used to empirically test each of the research questions. In addition, a qualitative approach based on grounded theory was used to further explore and scrutinize each research question. The qualitative approach aims to explore the feelings and experiences of learners and staff, and focuses on a number of contextual factors that might explain learner behaviour. This included an examination of how these various factors interrelate and the interplay between them. Using a mixed method approach provided a better picture of the phenomenon under study and both qualitative and quantitative approaches were seen as complementary (Alasuutari, 1995).

3.5.1 Methodology Procedure

Students in the study were split into a PBL group and a non-PBL control group. Each hypothesis was tested quantitatively over a number of cohorts using the instruments described in sections 3.6.2.1 to 3.6.2.4, which were given out before and after the teaching, and effect sizes for each hypothesis were calculated. In addition, information on learners' attendance and use of computer systems was taken from a number of databases and analysed statistically. Qualitative information on learners' backgrounds and PBL experiences was collected using questionnaires. Furthermore, interviews were carried out with learners and staff involved in the PBL group and detailed field notes were taken of observations of learner in-class behaviour.

3.6. Quantitative Methodology

Fraenkel and Wallen (2005) suggest that statistics can be viewed in a descriptive or inferential manner. In this study, descriptive statistics were used to illustrate features of the learner cohort such as composition and performance in assessments. Inferential statistics such as t-tests and analysis of variance were used to provide a deeper analysis of the data.

3.6.1. Experimental Design

Torgerson and Torgerson (2001) recommend that more educational studies use the experimental method, and that studies use the random assignment of learners to treatment and control groups. This study follows that recommendation and uses an experimental design, with random assignment of students to PBL classes, and a non-PBL control group as outlined in Tables 3.2 to 3.6. The unit of analysis is the individual learner.

Table 3-2: Pre-test/post-test - Hypotheses 1a and 1b (Learner attainment)

For each of 4 cohorts	**Group A**	**Group B**
Pre-test (start of semester 1)	Prior attainment (Leaving Certificate Points)	Prior attainment (Leaving Certificate Points)
Intervention	PBL Teaching	Non PBL Teaching
During Intervention	Programming assignments	Programming assignments
Post-test (end of semester 1)	Closed book exam	Closed book exam

Table 3-3: Pre-test/post-test - Hypothesis 2 (Learner Self-Regulation)

For each of 2 cohorts	**Group A**	**Group B**
Pre-test (start of semester 1)	Learning Self-Regulation Questionnaire (SRQ-L)	Learning Self-Regulation Questionnaire (SRQ-L)
Intervention	PBL Teaching	Non PBL Teaching
During Intervention	Observation of PBL classes, Learner attendance monitoring	Observation of PBL classes, Learner attendance monitoring
Post-test (end of semester 1)	Learning Self-Regulation Questionnaire (SRQ-L), Interviews	Learning Self-Regulation Questionnaire (SRQ-L), Interviews

Table 3-4: Pre-test/post-test - Hypothesis 3 (Programming Self-Efficacy)

For each of 2 cohorts	Group A	Group B
Pre-test (start of semester 1)	Programming Self-Efficacy instrument (PSE)	Programming Self-Efficacy instrument (PSE)
Intervention	PBL Teaching	Non PBL Teaching
During Intervention	Observation of PBL classes, Learner attendance monitoring	Observation of PBL classes, Learner attendance monitoring
Post-test (end of semester 1)	Programming Self-Efficacy instrument (PSE), Interviews	Programming Self-Efficacy instrument (PSE), Interviews

Table 3-5: Pre-test/post-test - Hypothesis 4 (Students' Approaches to Learning)

For 1 cohort	Group A	Group B
Pre-test (start of semester 1)	Approaches and Study Skills Inventory for Students (ASSIST)	Approaches and Study Skills Inventory for Students (ASSIST)
Intervention	PBL Teaching	Non PBL Teaching
During Intervention	Observation of PBL classes, Learner attendance monitoring	Observation of PBL classes, Learner attendance monitoring
Post-test (end of semester 1)	Approaches and Study Skills Inventory for Students (ASSIST), Interviews	Approaches and Study Skills Inventory for Students (ASSIST), Interviews

Table 3-6: Pre-test/post-test - Hypothesis 5 (Learner preferences for different types of course and teaching)

For 1 cohort	Group A	Group B
Pre-test (start of semester 1)	Approaches and Study Skills Inventory for Students (ASSIST)	Approaches and Study Skills Inventory for Students (ASSIST)
Intervention	PBL Teaching	Non PBL Teaching
During Intervention	Observation of PBL classes, Learner attendance monitoring	Observation of PBL classes, Learner attendance monitoring
Post-test (end of semester 1)	Approaches and Study Skills Inventory for Students (ASSIST), Interviews	Approaches and Study Skills Inventory for Students (ASSIST), Interviews

3.6.2. Instruments and Measures

A review of the literature identified a number of established instruments that could be used to help test the different hypotheses. The chosen instruments are discussed in the following sections.

3.6.2.1. Learner Attainment (Hypotheses 1a and 1b)

Due to the differential effects of PBL on knowledge and skills identified in the literature, it was necessary to distinguish between these when looking at attainment marks.

Learner attainment results comprise both continuous assessment and exam scores. End semester exams in Software Development are designed to test learning outcomes that reflect students' knowledge of the module, while continuous assessments are designed to test learning outcomes that reflect students' programming skills. Thus, students' attainment scores in end semester exams are taken to indicate basic ability in Software Development theory and knowledge, while attainment scores in continuous assessment are taken to indicate team working and programming skills.

The attainment data used to test Hypotheses 1a and 1b was collected over four cohorts of learners. The attainment data of all learners who attended the Software Development module was analysed using statistical techniques including residual gain analysis, t-tests and analysis of variance, and an effect size was identified. ANCOVA general linear modelling was applied to the attainment data to control for prior attainment.

An indicator of the success of the Hybrid PBL model is whether Group A's performance in Software Development is superior to Group B's. A number of statistical tests were carried out on the results to assess the effectiveness of the PBL module:

- Group A's attainment results (for semester 1) from both final exam and continuous assessments were compared against Group B's results.
- Group A's course entry points (achieved in the Irish Leaving Certificate or equivalent) were compared against Group B's course entry points.

From these comparisons it was possible to test the following Hypotheses:

(1.a) Learners in the PBL group will score higher in exams than those in the control group.

(1.b) Learners in the PBL group will score higher in continuous assessment than those in the control group.

3.6.2.2. Learner Self-Regulation (Hypothesis 2)

Participants' Learning Self-Regulation (Autonomous or Controlled Regulation) was measured over two cohorts of learners using a statistical analysis of learner responses on the Learning Self-Regulation Questionnaire (SRQ-L) (Williams & Deci, 2007a), which was given out to all participants in both groups at the start and end of semesters 1 and 2.

Two other possible questionnaires could have been used: these were the *Motivated Strategies for Learning Questionnaire (MSLQ)* (Pintrich, Smith, Garcia & McKeachie, 1991) or the *Academic Motivation Scale Questionnaire* (Vallerand *et al.*, 1992). Both alternatives were investigated, but the SRQ-L questionnaire was selected due to its validation and reported use with science subjects at college level (Black & Deci, 2000).

The Learning Self-Regulation Questionnaire (SRQ-L) investigates why people engage in learning-related behaviours, and was developed at the Department of Clinical and Social Sciences in Psychology at the University of Rochester by Williams and Deci (1996). The scale was later adapted slightly for use with university students learning organic chemistry[3] (Black & Deci, 2000). Williams and Deci (2007a) state that this "is essentially the same scale, although two items were dropped for the sake of brevity. The questionnaire can be adapted as needed to refer to the particular course or program being studied". The version used in the study at Anon College was adapted very slightly from this scale, the only change being the words 'organic chemistry' replaced by 'computer programming' to allow for use

[3] Both versions of the Learning Self-Regulation Questionnaire (SRQ-L) can be downloaded from the University of Rochester website at http://www.psych.rochester.edu/SDT/measures/selfreg_lrn.html

within the context of a Computing course. In addition the final two questions from the original questionnaire were retained. Both the fully adapted Learning Self-Regulation Questionnaire and the original questionnaire are provided in Appendix C.

Analyses can be done with the two separate subscales (Autonomous or Controlled Regulation), and a Relative Autonomy Index is formed by subtracting the controlled subscale score from the autonomous subscale score. Expanding on this, Williams and Deci (2007a) say that "the responses that are provided are either controlled (i.e., external or introjected regulation) or autonomous (identified regulation or intrinsic motivation). Because the scale was designed to have just the two 'super' categories of regulation, there was no attempt to have the same number of items from each regulatory style (e.g., identified and intrinsic), and there was no psychometric work done on the individual regulatory styles. The validation was done only at the level of the two 'super' categories." All the Self-Regulation Questionnaires are well validated, with details of their validation described by Ryan and Connell (1989). In particular, the Learning Self-Regulation Questionnaire has been validated in a number of studies set in the higher education context (Black & Deci, 2000), with Williams and Deci (2007a) reporting "alpha reliabilities [of] approximately 0.75 for controlled regulation and 0.80 for autonomous regulation" for the Learning Self-Regulation Questionnaire 'Chemistry' (SRQ-L).

An indicator of the success of the Hybrid PBL model is whether Group A's Computer intrinsic motivation increased at a greater rate than Group B's. A number of statistical tests were carried out on the results to assess changes in learners' intrinsic motivation due to attending the PBL module:

- Group A's Learning Self-Regulation results at the start and finish of semester 1 were compared against Group B's results;
- Any change in Group A's Learning Self-Regulation during semester 1 was compared against Group B's results.

From these comparisons it was possible to test the following hypothesis:

(2) Learners who complete the PBL course will have a higher degree of intrinsic motivation than those in the control group.

3.6.2.3. Programming Self-Efficacy (Hypothesis 3)

‘Programming’ rather than ‘Computer’ Self-Efficacy was investigated in this study as it is more appropriate to a study on the teaching of programming. Also, as the participants have chosen a specialised Computing degree rather than a general Science degree, they probably already have high computer efficacy but might not have high efficacy about how to programme computers.

Participants’ Programming Self-Efficacy was measured over two cohorts of learners using a statistical analysis of learner responses on the Computer Programming Self-Efficacy instrument (PSE) (Ramalingam & Wiedenbeck, 1998), which was given out to all participants in both groups at the start and end of semesters 1 and 2. The full Computer Programming Self-Efficacy instrument is provided in Appendix D.

The Computer Programming Self-Efficacy Scale was developed by Ramalingam and Wiedenbeck (1998) for use with object-oriented programming languages. The PSE has been used in a number of studies in higher education on learners of the Java programming language (Askar & Davenport, 2009; Bergin & Reilly, 2005), and in studies of introductory programming courses (Cantwell-Wilson & Shrock, 2001; Ramalingam & Wiedenbeck, 1998). The PSE is a well-validated and reliable instrument, with Cronbach’s alphas of (.89) and (.98) reported by Bergin and Reilly (2005), and Cantwell-Wilson and Shrock (2001) respectively.

The Computer Programming Self-Efficacy Scale consists of thirty-three items that ask students to judge their capabilities in a wide range of programming tasks and situations. Responses are answered on a 7-point Likert-type scale (1=not at all confident, 7=absolutely confident). Computer Programming Self-Efficacy is measured as a continuous variable, which is the summation of the choices made on the scale. The maximum score achievable is 231. An indicator of the success of the Hybrid PBL model is whether Group A’s Computer Programming Self-Efficacy increased at a greater rate than Group B’s. A number of statistical tests were carried out on the results to assess changes in learners’ Self-Efficacy due to attending the PBL module:

- Group A’s Computer Programming Self-Efficacy results at the start and finish of semester 1 were compared against Group B’s results.

- Any changes in Group A’s Computer Programming Self-Efficacy during semester 1 was compared against Group B’s results.

From these comparisons it was possible to test the following hypothesis:

(3) Learners in the PBL group will show a higher degree of programming self-efficacy than those in the control group.

3.6.2.4. Students’ Approaches to Learning and Learner Preferences (Hypotheses 4 & 5)

Students’ approaches to studying and learner preferences were measured over one cohort of learners, using a statistical analysis of learner responses on parts B and C of the Approaches and Study Skills Inventory for Students (ASSIST)[4], which was given out to all participants in both groups at the start and end of semesters 1 and 2. The ASSIST instrument used in this study is provided in Appendix E.

ASSIST was developed by Entwistle (1997) at the Centre for Research on Learning and Instruction in the University of Edinburgh in 1997. ASSIST is based on the Revised Approaches to Studying Inventory (RASI) developed by Entwistle and Tait (1994) which in turn was based on the Approaches to Studying Inventory (ASI) developed by Ramsden (1979).

The theoretical basis for current research into students’ approaches to studying comes from research undertaken by Marton and Saljo (1976, 1997) on approaches to learning, Entwistle and Ramsden's (1983) research on approaches to studying, combined with the work of Biggs (1979, 1987) on learning outcomes. In particular, the ASSIST inventory is conceptually based on research studies by Briggs (1993) and Richardson (2000). Entwistle and McCune (2004) provide a detailed description of the conceptual bases of the ASSIST inventory, while Entwistle, McCune, and Tait (2006) provide details of its usage, validity and reliability.

[4] The Approaches and Study Skills Inventory for Students (ASSIST) can be downloaded from the Enhancing Learning and Teaching project website at http://www.etl.tla.ed.ac.uk/publications.html#measurement

Entwistle *et al.* (2006, p. 1) state that “[ASSIST] identifies the tendencies of students to adopt deep, surface and strategic approaches to learning and studying. The inventory uses a Likert technique for measuring attitudes which involves asking students to rate the extent of their agreement on a five-point scale with a series of related items that cover the aspects of a specific construct. Summing these responses across items produces a scale score for each construct”. ASSIST has been widely used, is well validated, and has had its reliability well tested (Entwistle, Tait & McCune, 2000; Long, 2003; Tait & Entwistle, 1996). This is also true of its predecessor the RASI (Duff, 1997).

The first section of ASSIST contains items relating to conceptions of learning, and this section was not used in this study. Section B is based on the Approaches to Studying Inventory (ASI) which was developed in the University of Lancaster in the late 1970s (Entwistle & Ramsden, 1983). Section B contains 52 items and produces scores on Deep, Surface and Strategic Approaches to learning. The ‘alertness to assessment’ scale was omitted from this study because it is not suitable for use with first-year students early on in their course. The final section (C) invites students to indicate their preferences for different kinds of teaching.

An indicator of the success of the Hybrid PBL model is whether learners in Group A show higher scores on meaning orientation and lower scores on reproduction orientation and a greater preference for teaching that supports deep learning than learners in Group B. A number of statistical tests were carried out on the results to measure and assess changes in learners’ learning orientation and preferences due to attending the PBL module:

- Group A’s approaches to study and their preferences scores at the start and finish of semester 1 were compared against Group B’s results;
- Any change in Group A’s approaches to study and preference scores during semester 1 was compared against Group B’s results.

From these comparisons it was possible to test the following Hypotheses:

(4) Learners in the PBL group will show higher scores on meaning orientation and lower scores on reproduction orientation than those in the control group.

(5) Learners in the PBL group will show a greater preference for courses and teaching that support deep learning (as opposed to surface learning) than those in the control group.

3.6.2.5. The Adaptation of Instruments

As has been seen in sections 3.6.2.2, 3.6.2.3 and 3.6.2.4, the quantitative instruments (Learning Self-Regulation Questionnaire, Programming Self-Efficacy Instrument and the Approaches and Study Skills Inventory for Students) used in this study have been employed in numerous studies, are well validated, and have had their reliability well tested (Black & Deci, 2000; Entwistle *et al.*, 2000; Long, 2003; Ramalingam & Wiedenbeck, 1998; Tait & Entwistle, 1996; Williams & Deci, 1996, 2007a). For convenience, the Learning Self-Regulation Questionnaire (SRQ-L), the Computer Programming Self-Efficacy Scale and parts B and C of the Approaches and Study Skills Inventory for Students were combined into one long multipart questionnaire for distribution to students. A small-scale pilot study was carried out before the main process of data collection was started. As a result of this study some minor modifications were made to the instruments.

3.6.3. Controls

A well-controlled research design is an essential feature of the experimental method (Cohen, Manion & Morrison, 2000, p. 211). The controls used in this study are outlined in the following sections.

3.6.3.1. Random Allocation of Learners to Treatment and Control Group

That participants are divided into a treatment group and a non-treatment control group is one of the most important controls in an experiment (Bell, 1993, pp. 11-12). This was the approach taken in this study.

First year full-time Computing students at Anon College are randomly split into two groups for Software Development (Java programming). Software Development is taught over two 15 week semesters. Due to resource issues, for the 2005/2006, 2006/2007, 2007/2008 and 2008/2009 academic years, in semester 1 half of the class (Group A) were taught using PBL, while the other half (Group B) were taught using a traditional approach. In semester 2, the groups switched over, i.e. Group B was taught using a PBL approach, while Group A reverted to the traditional approach.

The random allocation of learners to the treatment groups (Groups A and B) helps to maximise the probability that they do not differ in any systematic way. This situation, where two groups of students are taught the same subject using different instructional approaches, afforded a unique opportunity to gauge the effectiveness of the PBL approach in semester 1. Only semester 1 exam and continuous assessment results were used to gauge the effectiveness of the PBL approach in terms of attainment (Hypotheses 1a and 1b).

3.6.3.2. Control for Prior Attainment

When testing the impact of PBL on attainment scores (Hypotheses 1a and 1b), course entry points (achieved in the Irish Leaving Certificate or equivalent) were used to control for prior attainment. This was done for each of the four cohorts. Overall Leaving Certificate entry points have been shown to be a fair indicator of success at Computing, with a correlation coefficient of (r = .66) (Moran & Crowley, 1979).

3.6.3.3. Control for Teacher Effects

The same staff member acted as overall coordinator for the module for the duration of the study. The same four lecturers (two male and two female) delivered the module in all four years, with two lecturers assigned to each group. This allowed for the control of teacher effects. Over the duration of the study an additional five lecturing staff and four tutors provided support in computer laboratories. However, in any given year both groups had the same staff and tutors.

3.6.3.4. Control for Types of Assessment

The same methods of summative assessment were used for all four cohorts of learners. Within each cohort, identical marking schemes and assessments were used for both groups (two in-laboratory practical assignments and a paper-based closed book end-term exam). All learners had their final grade point averages calculated in the same way. This allowed for a feasible comparison between groups and with attainment results from previous non-PBL years as the only change instigated was the teaching method.

3.6.3.5. Control for Physical Teaching Environment

The physical learning environment of classrooms and computer laboratories, and the time allocation and combination of lectures, tutorials and laboratories (7 hours in total per week), was the same for both groups. The same computer hardware and software was used by both groups.

3.6.3.6. Control for Statistical Assumptions

In cases where statistical tests assumed a normal distribution of data, the Kolmogorov-Smirnov normality test was carried out to ensure normality. For testing normality the Kolmogorov-Smirnov test is less powerful than the Shapiro-Wilk test or Anderson-Darling test but it was considered sufficient. In all cases equality of variances between groups was tested using F-tests. Given the characteristics of the data, the more robust tests for variances, such as Levene's test, Bartlett's test, or the Brown-Forsythe test were considered unnecessary.

3.7. Effect Sizes

In this study effect size measures were used as they separate the magnitude of an effect from whether or not it is statistically significant and provide a measure that is independent from the instruments and procedures used (Richardson, 1996; Rosenthal, 1994). These properties allow effect size results from different studies to be combined together in a meta-analysis. Cohen (1988) provides a framework for comparing and measuring effect sizes, suggesting that an effect size of (0.2) is small, (0.5) is medium and (0.8) is large. Colliver (2000) suggests that when measuring the effectiveness of PBL, an effect size of 1.0 should be sought. However, Albanese (2000, p. 729) disagrees, saying that "[e]ffect sizes of 0.8-1.0 are an unreasonable expectation from PBL". Others agree: Richardson (2005), citing the work of Lipsey and Wilson (1993), suggests that an effect size of 0.50 would be a more reasonable choice. For the purposes of this study, an effect size of (0.5) was used as a benchmark measure of effectiveness.

3.8. Questionnaire Data

Two additional questionnaires were used in this study. The first questionnaire examined learners' Software Development PBL experiences and the second was a general learner background questionnaire. Learner self report responses were analysed statistically. The design of both questionnaires was refined by implementing an additional small scale pilot study before the main process of data collection began. The PBL questionnaire is provided in Appendix F1, and the background questionnaire in Appendix F2.

3.8.1. PBL Questionnaire

The PBL questionnaire was given to all four cohorts of learners in the PBL groups after completion of their PBL course. The questionnaire was adapted for use in a Software Development context from an instrument developed by Antepohl and Herzig (1999) for use with Medical Students. The questionnaire is well validated (Dolmans & Schmidt, 2000). It contains 32 questions/statements aimed at testing six areas influencing students' opinions and decisions on Software Development:

- PBL group work;
- the PBL method;
- student interest in Software Development;
- course objectives and content;
- the PBL tutor;
- teaching resources.

Learners were asked to indicate on a 5-point Likert scale whether they (1) totally disagreed, (2) disagreed, (3) were neutral, (4) agreed, or (5) totally agreed with each statement. At the end of the questionnaire learners were asked to add their own comments.

3.8.2. General Background Questionnaire

To provide background and contextual information, a general questionnaire was given to all four cohorts of learners. The questionnaire collected data on the following items: gender, socio-economic and family background (via home address), previous programming experience, previous non-programming computer experience, and encouragement by others to pursue Computing as a career.

3.9. Data Collection Procedures

All data was collected and stored in accordance with the ethical requirements outlined in section 1.5. In particular, before the questionnaires were handed out, their purpose was explained to the participants. They were told that the results would be included in a thesis. All the participants were guaranteed anonymity. In line with ethics requirements, all participants gave informed consent to being interviewed and recorded. At the start and end of each winter and spring semester, the questionnaires were distributed at a class lecture session. The questionnaires were completed during class by participants and collected by the researcher.

3.10. Database Information Mining

Over the four cohorts of learners detailed information on participants' prior educational attainment, level of class attendance and time spent 'logged on' was taken from a number of Anon College's databases and online systems (Virtual Learning Environment, Attendance Registers, Student Computer Network Audit Logs, and College Management Information System). In particular, participants' attendance at PBL classes was taken from the college attendance database and used to compare Group A's class attendance against Group B's class attendance in semester 1.

3.11. Qualitative Methodology

In addition to the measures outlined above qualitative techniques were used to provide a greater insight into the feelings, thinking and experiences of learners and staff. Participants' attitudes, motivation, stress and enthusiasm were qualitatively analysed using a variety of techniques including participant observation, field notes and both formal and informal interviews. This allowed the development of a deeper understanding and examination of all the hypotheses.

Qualitative research explores a social or human problem through an inquiry process that occurs in the natural setting where the researcher is an instrument of data collection (Creswell, 2003). Connelly and Clandinin (1990) point out that all qualitative observational research involves formulating a thoughtful and well-understood relationship between the researcher and research participants.

Patton (1990, p. 40) states that a "holistic approach assumes that the whole is greater than the sum of its parts." This research takes a holistic perspective, focusing on the experiences of the whole group, which were uncovered through observations and interviews of individuals as well as group measures.

Interviews and participant observations were used as data-collection instruments to measure feedback from staff and learners on their PBL experience. As outlined in section 3.11.1., eleven semi-structured interviews (five with staff and six with learners) were conducted. Also as set out in sections 3.11.2 and 3.11.3, field notes were made of classroom observations and informal conversations for both the PBL and non-PBL groups. The journals kept by all learners in the PBL groups were also examined. The results of these inquiries are presented in this section.

A grounded theory approach was employed for the analysis of the field notes and interview responses. Strauss and Corbin's (1990, p. 24) analysis method was used to develop "an inductively derived grounded theory about a phenomenon". The first stage involved the 'open-coding' of interview data, defined as "breaking down, examining, comparing, conceptualizing, and categorizing data" (Strauss & Corbin, 1990, p. 61). All transcripts from the interviewees were read for emerging commonalties and patterns. A line-by-line approach was taken to analyse each sentence and to separate data into categories relevant to staff and students' attitudes, motivation, stress and enthusiasm. Five main categories were identified, and are described in section 5.4.5. Category selection was modified during the coding process for a richer description of the phenomenon. The next step involved 'axial coding' in which connections were formed among the categories found in open coding, and some possible causal relationships were identified within a frame of relationships. Then 'selective coding' was done to see if one category could be identified as a core category, and to relate all other categories to that category.

Data analysis took place both during and after data collection, in line with Creswell's (1998) emphasis on a crisscross approach between data gathering and its analysis.

This allowed the patterns, commonalties and differences that emerged early in the collection process to be examined in further detail in later interviews and observations. To support this process, Minitab 15 software was used to help statistically analyse the questionnaire data, while NVivo 7 software was used to help analyse both the interview and observation data sets. NVivo assisted in classifying, sorting, and arranging information, as well as exploring trends and testing theories. It must be emphasised that this was not an easy option: the software was used to support the manual analysis and was not employed to provide a simple ‘automated’ set of categories.

3.11.1. Interviews

The interviews allowed an in-depth exploration of the values, feelings and beliefs of the lecturing staff, tutors and learners involved in the PBL module. Two female and four male learner participants, and two male and three female staff/tutor participants were selected at random and asked to take part in an interview. If a participant declined the request, another was selected at random. In all, eleven interviews (five with staff and six with learners) were conducted.

3.11.1.1. Interview Procedures

Interviews with learners and staff were conducted individually in an empty classroom in Anon College. Each interview lasted between 10 or 15 minutes. All interviews were carried out in English. The interviews were audio recorded and transcribed before data analysis was carried out.

3.11.1.2. Interview Protocol

Before each interview started, its purpose was explained to the participant, who was told that the interview transcripts would be included in a thesis. Participants were guaranteed anonymity and were assured that if individual portions or full transcripts were published, their real names would be substituted by a pseudonym.

In line with ethics requirements, all participants gave informed consent to being interviewed and recorded. Most questions were open-ended and designed to explore the perspectives of the interviewee. However, a few directive questions were also embedded to test responses. To ensure that all key areas would be investigated, the same set of interview questions was used in all interviews. The set of interview questions for staff was adapted from Maudsley (2002). Maudsley (2002) designed her questions to explore how PBL tutors conceptualised their students' integrated learning agenda.

All participants were allowed to freely answer the pre-set questions in the interview. No limits were placed on their answers and they could discuss any area they wanted to. After all the pre-set questions had been asked, participants were asked if they had any questions or comments, or if they would like to add anything, which allowed all participants to speak their mind freely. In addition to audio-recording the interviews field notes were made after each interview of how the interview went. This included observations of how certain new avenues of interest opened up and of the mood of the participant: whether they were talkative or reserved, cooperative or resistant, nervous or relaxed, etc. About a week after the interview, the transcript of the interview was shown to the participant to verify that it was accurate. The set of interview questions for staff is given in Appendix G, while the set of interview questions for students is given in Appendix H. The entire set of transcripts is contained in Appendix J.

3.11.2. Conversations

In addition to the interviews, a large number of informal conversations were held with staff and learners. These mainly involved small talk although there were some longer discussions. The conversations helped the researcher explore the feelings and experiences of the participants. Field notes of conversations were made in the research diary.

3.11.3. Participant Observations

Burns (1999, p. 80) describes observation as the process "of taking regular and conscious notice of classroom actions and occurrences which are particularly relevant to the issues or topics being investigated". This enables researchers to gain personal insights into classroom activities and helps to build a deeper understanding that can help provide a framework to support possible answers to research questions. A great strength of observation is that it allows the researcher to reflect on actual occurrences in the classroom, and as Burns (1999, pp. 81-82) points out, it builds new "perspectives […] on familiar situations [and …] allows us to see in a relatively unobtrusive way what it is that people actually do compared with what they say they do".

In this study, the researcher is a staff member in the Department under study and therefore had a position in the group before taking on the role of observer. In this sense, the researcher could be viewed as an observing-participant. However, the researcher did not teach or take part in any of the classes observed and had never taught any of the participants, and in that sense was a neutral-observer in all Computing classes. The researcher's role was to look and listen and record group interactions and behaviours as objectively as possible for later analysis (Eisner, 1993).

At all times the observer was aware that his presence might influence the behaviour of participants. To guard against Hawthorne effects, the observer was as unobtrusive as possible, and account was taken of the influence the act of observing the participants (both staff and learners) was having on their behaviour. In addition, the observer did not take part in any classroom activities and he tried to be aware of any presumptions he held that might influence the findings.

Each week of the 15 week semester the researcher spent half an hour in PBL and half an hour in non-PBL Software Development laboratories observing participants' behaviour. In addition, the researcher observed five minutes of participant social interaction both before and after laboratory classes. This was done for each of the four cohorts of learners over four academic years, resulting in a total of 80 hours of participant observation. The full observation schedule is given in Appendix I.3.

Notes were taken of learners' class arrival and departure times, their body language and discussions, and their time spent 'on task' on programming problems. After each laboratory session detailed field notes were written up. These direct, first-hand observations of daily behaviour provided a high face validity of data and an understanding of group behaviour.

3.11.4. Field Notes and Diary

Wallace (1998, p. 59) states that "field-notes […] are terms used to describe what has happened during a lesson, and may be written up during the lesson or shortly after”. Burns (1999, p. 87) advises that field notes should be “relatively informal” and taken “at suitable intervals during the lesson through ‘jottings’ or stream-of-behaviour records made on the spot as the lessons proceed”. Burns (*ibid*) goes on to suggest that the advantage of this technique is that issues that are central to the classroom investigation can be clarified and emerging classroom patterns can be identified.

As outlined earlier, detailed field notes were taken of all activities. Notes were dated and organised in categories. Supplementing the field notes, a reflective research diary was kept to allow contemplative analysis of the time-line of observations and interviews, and to link with the time-line of the quantitative analysis. This allowed for later analysis and the identification of issues and areas that needed to be researched further.

3.11.5. Student PBL Journals

All PBL students keep a journal of their classroom activities, which they update on a week-by-week basis, and which contains a record of their collaborative work including details of the problems worked on and coded solutions. However, unfortunately the PBL journals are not reflective of and do not record students’ feelings or experiences of the learning process. Even given this shortcoming, the journals do provide a measure of student engagement and interest in problem-solving. A random sample of these journals was taken each year and analysed. Ten journals were analysed in the first year and five journals in each subsequent year. In total twenty five journals were analysed. The researcher read the journals a number of times to help eliminate any errors and reduce any possible misunderstandings.

3.11.6. The Reliability and Validity of the Quantitative Data

According to Creswell (2003), research findings must be reliable, valid and objective. In the following sections, issues of reliability and validity will be discussed in relation to the study at Anon College.

‘Reliability’ refers to the dependability, consistency, and stability of the research findings and ensures that the results are stable over time and across research methods (Hammersley, 1993). Three key types of reliability are: Equivalency Reliability, Stability Reliability and Internal Consistency (Colorado State University, 2009).

Equivalency reliability is the extent to which two items measure identical concepts at an identical level of difficulty. Equivalency reliability is determined by relating two sets of test scores to one another to highlight the degree of relationship or association. In quantitative studies and particularly in experimental studies, a correlation coefficient, statistically referred to as *r*, is used to show the strength of the correlation between a dependent variable (the subject under study), and one or more independent variables, which are manipulated to determine effects on the dependent variable. An important consideration is that equivalency reliability is concerned with correlational, not causal, relationships (Creswell, 2003).

Stability reliability (sometimes called test-retest reliability) is the agreement of measuring instruments over time. To determine stability, a measure or test is repeated on the same subjects at a future date. Results are compared and correlated with the initial test to give a measure of stability. Internal consistency is the extent to which tests or procedures assess the same characteristic, skill or quality. It is a measure of the precision between the observers or of the measuring instruments used in a study. This type of reliability often helps researchers interpret data and predict the value of scores and the limits of the relationship among variables (Colorado State University, 2009).

For each of the four years of the study, the end of semester exams used to measure attainment were designed to be of equivalent difficulty and to measure student knowledge of the required learning outcomes. Exams are reviewed by external examiners to ensure adherence to these requirements. Learning outcomes are specified in the Software Development syllabus and they did not change during the course of the study. Correlation coefficients are reported for all suitable statistical tests.

There are a number of types of validity that must be considered, in particular both internal and external (Miles & Huberman, 1984). An experiment is internally valid to the extent that it shows a cause-effect relationship between the independent and dependent variables, while external validity is “the extent to which causal propositions hold true in other settings” (Seale, 1999, p. 40). The conclusions drawn through the interpretation of the results of data analysis should be objective, that is, they should be based on the facts of the findings derived from actual data and not on one's own subjective or emotional values. This ensures that the researcher has no conflict of interests (Eisner, 1993).

Another important type of validity is construct validity. Construct validity refers to the degree to which inferences can legitimately be made from the operationalisations used in a study to the theoretical constructs on which those operationalisations were based. In other words, construct validity seeks agreement between a theoretical concept and a specific measuring device or procedure. For example, when researchers measure 'programming self-efficacy', is that what they are really measuring? Carmines and Zeller (1979, p. 23) suggest that to determine if a piece of research has construct validity, three steps should be followed. "First, the theoretical relationships must be specified. Second, the empirical relationships between the measures of the concepts must be examined. Third, the empirical evidence must be interpreted in terms of how it clarifies the construct validity of the particular measure being tested". As discussed in section 3.6.2, all the instruments used in this study have been used in many other studies and are well validated.

Another type of validity is face validity. Face validity is concerned with how a measure or procedure appears. Does it seem like a reasonable way to gain the information the researchers are attempting to obtain? As outlined in section 3.11.3., the participant observations helped to provide face validity as they are direct, first-hand observations of daily behaviour.

3.11.7. The Trustworthiness of the Qualitative Data

Mertens (1998) suggests that validity and reliability in qualitative research is based upon the trustworthiness of the data. Lincoln and Guba (1985) discuss four constructs against which the trustworthiness of a study can be evaluated: credibility; transferability; dependability and confirmability. These criteria are explicitly offered as an alternative to more traditional quantitatively-oriented criteria. They suggest that credibility should be used in place of internal validity, transferability in place of external validity, dependability in place of reliability, dependability and confirmability in place of objectivity. Lincoln and Guba (*ibid*) suggest that their four criteria better reflect the underlying assumptions involved in much qualitative research, and in this they are supported by other researchers (Talbot, 1995).

Credibility depends on how accurately the subject is identified and described. To be credible it must be established that the results of qualitative research are credible or believable from the perspective of the participant in the research.

Transferability refers to the degree to which the results of qualitative research can be generalized or transferred to other contexts or settings. Transferability is noted to be impossible from the stance of external validity, but is greatly assisted by providing the greatest possible range of information, and thick descriptive data. Lincoln and Guba (1985) suggest that thick description is a way of achieving a type of external validity, as by describing a phenomenon in sufficient detail, one can begin to evaluate the extent to which the conclusions drawn are transferable to other times, settings, situations, and people. Thick description was first described by Geertz (1977) who applied it in ethnography. It refers to the detailed account of field experiences in which the researcher makes explicit the patterns of cultural and social relationships and puts them in context (Holloway, 1997). From a qualitative perspective, transferability is primarily the responsibility of the one doing the generalizing. The qualitative researcher can enhance transferability by doing a thorough job of describing the research context and the assumptions that were central to the research. The person who wishes to 'transfer' the results to a different context is then responsible for making the judgment of how sensible the transfer is. Therefore, the applicability of one set of findings to another setting rests more with the later researcher making the transfer than the original researcher.

The idea of dependability, on the other hand, emphasizes the need for the researcher to account for the ever-changing context within which research occurs. Lincoln and Guba (1985) point out that dependability is difficult to predict in a changing social world. The researcher is responsible for describing the changes that occur in the setting and how these changes affected the way the researcher approached the study. To help establish dependability, the researcher attempts to account for changing conditions in the phenomenon chosen for study as well as changes in the design created by an increasingly refined understanding of the setting.

Qualitative research tends to assume that each researcher brings a unique perspective to the study (Patton, 1990). Confirmability refers to the degree to which the results could be confirmed or corroborated by others. McLean *et al* (1997) advocate that "qualitative confirmation provides a way of reporting the researcher's thought processes, helps promote a constructivist approach to qualitative research and answers the demand for increased rigor". This demand for rigour is made by a number of researchers, for example, Miles and Huberman (1984, p. 21) state that "we lack a body of clearly-defined methods for drawing valid meaning from qualitative data. We need methods that are practical, communicable, and not self deluding; scientific in the positivist's sense of the word, and aimed toward interpretive understanding in the best sense of that term". To help ensure that there was internal agreement between the investigator's interpretations and the actual evidence, a number of strategies for enhancing confirmability were employed.

Mays and Pope (1995) suggest that if the researcher can "create an account of method and data which can stand independently", then that will help improve rigour and establish the 'trustworthiness' of the qualitative findings (Guba & Lincoln, 1989). To this end, a transparent and open research process was employed in this study.

Talbot (1995) suggests a number of steps to improve the credibility of findings, including that the researcher remains in the field over a long period of time, using triangulation, and having participants review the researcher's interpretations and conclusions. Lincoln and Guba (1985, p. 328) also list a number of techniques that help establish the trustworthiness of a study. These include persistent observation, triangulation (sources, methods, and investigators) and negative case analysis. All these techniques were used in this study.

In this study the researcher carried out observations over a four year period and the summaries of interviews were discussed with participants. In addition, triangulation (Hammersley, 1998) was used to check the validity of the observational, interview and questionnaire data by ensuring that all data sets confirmed and supported each other. The type of triangulation used in this study was 'methods triangulation', which involved checking out the consistency of findings generated by different data collection methods (Denzin, 2006; Patton, 1999).

Lincoln and Guba (1985) describe deviant case analysis as a process of refining an analysis until it can explain or account for a majority of cases. Analysis of deviant cases may revise, broaden and confirm the patterns emerging from data analysis. Thus, during the data analysis the researcher actively undertook a deviant/negative case analysis. This involved searching for and discussing elements of the data that did not support or appeared to contradict patterns or explanations that were emerging from the data analysis.

Lincoln and Guba (1985, p. 328) suggest that researchers should provide an "audit trail" to help establish trustworthiness. An audit trail is a transparent description of the research steps taken from the start of a research project to the development and reporting of findings. These are records that are kept regarding what was done in an investigation. Malterud (2001, p. 486) underscores the need for researchers to provide a detailed report of the analytical steps taken in a study when she writes: "Declaring that qualitative analysis was done, or stating that categories emerged when the material had been read by one or more persons, is not sufficient to explain how and why patterns were noticed. [...] the reader needs to know the principles and choices underlying pattern recognition and category foundation". To this end, once data collection was complete an audit trail was conducted to examine the data collection and analysis procedures. The researcher documented the procedures for checking and rechecking the data throughout the study and reported any potential bias or distortion.

3.12. Limitations of the Analysis

The unique occasion of a random allocation of learners to the PBL and non-PBL groups on the first year Software Development course at Anon College allowed the use of a randomised experimental design. Nonetheless, the sampling frame used in the study at Anon College was an opportunity sample. However, the likelihood of obtaining a biased sample was reduced because data was taken from the total population of learners on the course. No participants declined to take part in the study and therefore problems relating to self-selected samples were prevented. Nonetheless, the study would have been be greatly strengthened if it had been carried out across a number of different higher education institutions in different countries.

The analysis could be strengthened through the use of a larger sample size, and a better gender balance within groups. The lack of female participants could impact the transferability of findings to domains where females predominate. Although hypothesis 1 was measured over four cohorts of learners, hypotheses 2 and 3 were measured over only two cohorts, and hypotheses 4 and 5 were measured over only one cohort. The analysis of additional cohorts would improve the dependability of the findings.

Another limitation of the analysis is that there was only one researcher. Having additional researchers would increase the trustworthiness of the data by allowing the cross-checking of qualitative findings by other researchers. However, issues of inter-rater Reliability would have to be addressed in the study design if there were more than one researcher.

A point worthy of note is that the groups were not totally statistically independent, as both groups of learners are located in the same college and Computing students mix freely between groups outside of class time. Therefore there is the possibility of learners mixing socially and discussing Software Development topics.

3.13. Conclusions

The study design employed makes it possible to integrate in detail the five research questions identified in the literature review. The next two chapters will present the quantitative and qualitative findings.

Chapter 4 - Quantitative Analysis

In this chapter the results of the quantitative analysis of the data relating to Hypotheses 1 to 5 are presented. However, the quantitative results provide an incomplete view of the findings, in the absence of the qualitative results. For this reason, an elucidation and interpretation of the results is not undertaken until chapter 6. A summary of the results is given in the tables which are presented throughout this chapter.

4.1. Population

Over the four cohorts of students there were 51 dropouts from the course. Data from these students are not included in the study. Overall 40 of the 51 students who dropped out of the module either enrolled and never attended or dropped out within the first two weeks, and therefore their omission from the analysis did not bias the results.

As can be seen in Table 4-1: Learner Population, in semester 1 2005/2006 63 students (3 female and 60 male) enrolled on the module. 49 students completed the module: three female (6%) and 46 male (94%). They were split randomly between Group A (24 Students) and Group B (25 Students).

In semester 1 2006/2007 74 students (6 female and 68 male) enrolled on the module. 58 students completed the module: six female (11%) and 52 male (89%). They were split randomly between Group A (29 Students) and Group B (29 Students).

In semester 1 2007/2008 76 students (10 female and 66 male) enrolled on the module. 64 students completed the module: ten female (11%) and 52 male (89%). They were split randomly between Group A (32 Students) and Group B (32 Students).

In semester 1 2008/2009 81 students (12 female and 69 male) enrolled on the module. 72 students completed the module: 12 female (17%) and 61 male (83%). They were split randomly between Group A (37 Students) and Group B (35 Students).

Table 4-1: Learner Population

Semester	Enrolment	Drop out	Completed	Group A (# Students)	Group A (Mean Entry points)	Group B (# Students)	Group B (Mean Entry points)
2005/06	63 students (3 female and 60 male)	14 students (all male) (A:7, B:7)	49 students completed the module: three female (6%) and 46 male (94%).	24	269	25	248
2006/07	74 students (6 female and 68 male)	16 students (all male) (A:9, B:7)	58 students completed the module: six female (11%) and 52 male (89%).	29	248	29	245
2007/08	76 students (10 female and 66 male)	12 students (all male) (A:5, B:7)	64 students completed the module: 10 female (16%) and 54 male (84%).	32	255	32	258
2008/09	81 students (12 female and 69 male)	9 students (all male) (A:6, B:3)	72 students completed the module: 12 female (17%) and 60 male (83%).	37	259	35	265
2005/08	294 students (31 female and 263 male)	51 students (A:27, B:24)	243 students completed the module: 31 female (13%) and 212 male (87%).	122	257	121	255

4.2. Analysis of Learner Attainment Scores (Hypotheses 1a and 1b)

Detailed results of the statistical tests carried out on the attainment data are presented in Appendix K, while an overview is provided below.

Attainment marks in Software Development are divided into two components: final exam and continuous assessment. Each learner's overall grade is determined by combining both components in equal proportion. As well as analysing this combined grade, each component was analysed separately. This allowed an investigation into how a hybrid PBL teaching method influences learners' skills, as measured by continuous assessment mark, and knowledge, as measured by exam mark.

As described in section 3.6.2.1., a number of statistical tests were carried out on the attainment scores of all four cohorts to assess the effectiveness of the PBL module:

- Group A's attainment results (for semester 1) from both final exam and continuous assessments were compared against Group B's results.
- Group A's course entry points (achieved in the Irish Leaving Certificate or equivalent) were compared against Group B's course entry points.

The above tests are sufficient to test Hypotheses 1a and 1b. However, for completeness, the following additional tests were carried out on the attainment scores of the first cohort 2005/2006):

- Group A's attainment results (for semester 1) from both final exam and continuous assessments were compared against historical (semester 1) attainment data for the Software Development module.
- First year engineering students were used as an additional control group. They also take the Software Development module but do not use PBL. The Engineering students have very similar course entry points to the Computing students (achieved in the Irish Leaving Certificate or equivalent), and both the Computing and Engineering class have very similar demographics.

Table 4-2: Group A (Exam Results) vs. Group B (Exam results) (Hypothesis 1a)

Year	Group A Mean (Rounded)	Group B Mean (Rounded)	F-test (Rounded)	T-test (equal variance)	Effect Size
2005/06	57	51	P=0.0756 (variances are equal)	P=0.1593 (not significant)	0.28
2006/07	51	52	P=0.0636 (variances are equal)	P=0.4211 (not significant)	-0.05
2007/08	50	52	P=0.0806 (variances are equal)	P=0.3012 (not significant)	-0.13
2008/09	48	54	P=0.4088 (variances are equal)	P=0.0933 (not significant)	-0.31
2005/09	50.91	52.35	P=0.1970 (variances are equal)	P=0.2836 (not significant)	-0.07

Table 4-3: Group A (CA Results) vs. Group B (CA results) (Hypothesis 1b)

Year	Group A Mean (Rounded)	Group B Mean (Rounded)	F-test	t-test (equal variance)	Effect Size
2005/06	67	52	P=0.2001 (variances are equal)	P=0.0072 (significant @ .05)	0.71
2006/07	63	58	P=0.4882 (variances are equal)	P=0.1647 (not significant)	0.25
2007/08	64	61	P=0.3411 (variances are equal)	P=0.2800 (not significant)	0.14
2008/09	64	54	P=0.4078 (variances are equal)	P=0.0114 (significant @ .05)	0.54
2005/09	64.28	56.37	P=0.4776 (variances are equal)	P=0.0009 (significant @ .05)	0.40

4.2.1. Analysis of Exam Attainment Scores (Hypothesis 1a)

As can be seen in Table 4-2, when Group A's exam marks were compared with Group B's for each of the four academic years 2005/06, 2006/07, 2007/08 and 2008/09, there were some small changes in the mean scores but t-tests show that these differences are not statistically significant at the .05 level. While an effect size of 0.28 can be seen in the first year of PBL introduction, a trend of increasing negative effects on knowledge is noticeable in the following three years (-0.05, -0.13 and -0.31 respectively). When the data for the four years were tested as a single population, no evidence was found that the hybrid PBL module made a positive difference to students' knowledge as measured by exams, with an overall effect size of (ES = -0.07). Indeed, over the four years of the study, PBL had a small negative effect on knowledge acquisition.

4.2.2. Analysis of Continuous Assessment Scores (Hypothesis 1b)

Table 4-3 shows the comparison of Group A's continuous assessment marks against Group B's for the four academic years 2005/06, 2006/07, 2007/08 and 2008/09. An examination of skills as measured by continuous assessment marks reveals a different picture. Here a larger difference in the means is noticeable, particularly in the first and last year of the study, where the differences are statistically significant. The difference in continuous assessment marks over the four years of the study is also statistically significant. This significance was shown by both simple t-tests and by further more rigorous tests on Groups A and B through the performance of residual gain analysis, which compares like with like explicitly by taking out the effect of a student's ability at entry (measured by Leaving Certificate points or equivalent). These more rigorous tests show that the differences in means between Group A and Group B in 2005/06, and 2008/09 as well as the overall marks are significant at the .05 level.

Confirmation of the finding that the differences in means were significant was provided by creating an ANCOVA general linear model (GLM) using the overall mark and group, and then making the Leaving Certificate (LC) points the covariate, which produces a p-value that also takes the control (LC points) into account explicitly.

Group A's course entry points (achieved in the Irish Leaving Certificate or equivalent) were compared against Group B's course entry points to determine whether there was any difference in ability between the two groups. Overall Leaving Certificate performance has been shown to be a reasonable predictor of Software Development performance with a correlation coefficient of (r = .66) (Moran & Crowley, 1979). Over the four years of the study Leaving Certificate points had a correlation coefficient of (r = .602) with combined exam and continuous assessment scores. However, the regression analysis for the 05/06 data showed that Leaving Certificate points had a correlation coefficient of (r = .3).

Over the four years of the study the mean Leaving Certificate entry points for Group A were 257 while Group B's were 255, and t-tests showed that that these differences in the mean are not statistically significant at the .05 level. The biggest difference in mean entry points between the groups was in 2005/06. In that year the mean entry points for Group A were 269 while Group B's were 248. However, again t-tests show that that these differences in the mean are not statistically significant at the .05 level.

An examination of the effect of PBL on skills reveals larger effect sizes than those for knowledge. Indeed, there is a very large effect size of 0.71 in the first year of PBL, with effects of 0.25, 0.14 and 0.54 in the following three years. When the data for the four years were tested as a single population, evidence was found that the hybrid PBL produced a significant improvement in learners' skills as measured by continuous assessment, with an overall effect size of (ES = 0.40).

4.2.3. Additional Analysis of Attainment Data

In the year 2005/ 2006 some additional tests were carried out on the attainment data. A comparison of the PBL Group (Group A) against historical (semester 1) attainment data for the Software Development module, shows that the mean increased from 55 to 62 when compared against the 04/05 intake and from 57 to 62 when compared against the 03/04 intake. However, t-tests show that these effect sizes are not statistically significant at the .05 level.

When compared against the non-PBL Engineering Group, overall performance showed an improvement with the mean score increasing from 53 to 62. Simple t-tests show that this change is significant at the .05 level. However, performing more rigorous tests on Group A and the Engineering Group through the performance of residual gain analysis showed that between Group A and the Engineering Group the changes in mean were not significant at the .05 level.

The effect sizes reported here on knowledge and skills are in line with those reported in a number of other studies (Dochy *et al.*, 2003). A point worthy of note is that the groups are not totally independent, as Computing students mix freely between groups and with Engineering students outside of class time.

4.3. Analysis of Learner Self-Regulation (Hypothesis 2)

As described in section 3.6.2.2., participants' Learning Self-Regulation (Autonomous or Controlled Regulation and their Relative Autonomy Index) was measured over two cohorts of learners, using a statistical analysis of learner responses on the Learning Self-Regulation Questionnaire (SRQ-L), which was given out to all participants in both groups at the start and end of semesters 1 and 2.

A number of statistical tests were carried out on the results to assess changes in learners' intrinsic motivation due to attending the PBL module:

- Group A's Relative Autonomy Index at the start and end of semester 1 were compared against Group B's results.
- Group A's and Group B's Relative Autonomy Index at the start and end of Semester 1 were compared separately using t-tests.
- Any change in Group A's Relative Autonomy Index during semester 1 was compared against Group B's results.

Detailed results of the statistical tests carried out on the Relative Autonomy Index data are presented in Appendix L, while an overview is provided below.

In total 136 students took the SRQ-L, 64 in 2007/08 and 72 in 2008/09. As shown in Table 4-4, when Group A's Relative Autonomy Index scores were compared with Group B's at the start of semester one for each of the two academic years 2007/08 and 2008/09, there were some small changes in the mean scores but t-tests show that these differences were not statistically significant at the .05 level.

When both groups' Relative Autonomy Index mean scores were compared at the end of semester one, after the teaching intervention, a slightly different picture emerges. The overall mean Relative Autonomy Index scores for the PBL group show a small increase from 11.08 to 13.98, while the overall mean scores for the non-PBL group show a slight decrease from 12.38 to 11.73. Statistical analysis using t-tests show that these differences are not statistically significant at the .05 level. However, when the data for the two years were tested as a single population, some evidence was found that the hybrid PBL model produced a slight improvement in learners' relative autonomy with an overall effect size of (ES = 0.23). Nonetheless, given the statistical results it cannot be said that learners who complete the PBL course will have a higher degree of intrinsic motivation than those in the control group.

Table 4-4: Learner Self-Regulation (Hypothesis 2)

	Group A (PBL) Relative Autonomy Index Scores			**Group B (Non-PBL) Relative Autonomy Index Scores**		
Two Cohorts	Mean (Rounded) (Start of Semester 1 Before PBL Teaching)	Mean (Rounded) (End of Semester 1 After PBL Teaching)	Mean Difference between Start and End of Semester Scores.	Mean (Rounded) (Start of Semester 1 Before Non PBL Teaching)	Mean (Rounded) (End of Semester 1 After Non PBL Teaching)	Mean Difference between Start and End of Semester Scores.
2007/09	11.08	13.98	2.9	12.38	11.73	-0.65

Table 4-5: Learner Programming Self-Efficacy (Hypothesis 3)

	Group A (PBL) Programming Self-Efficacy Scores			**Group B (Non-PBL) Programming Self-Efficacy Scores**		
Cohort	Mean (Rounded) (Start of Semester 1 Before PBL Teaching)	Mean (Rounded) (End of Semester 1 After PBL Teaching)	Mean Difference between Start and End of Semester Scores.	Mean (Rounded) (Start of Semester 1 Before Non PBL Teaching)	Mean (Rounded) (End of Semester 1 After Non PBL Teaching)	Mean Difference between Start and End of Semester Scores.
2007/08	132	181	49	132	153	21
2008/09	116	179	63	119	138	19
2007/09	123	180	57	125	145	20

4.4. Analysis of Learner Programming Self-Efficacy (Hypothesis 3)

As described in section 3.6.2.3., participants' Programming Self-Efficacy was measured over two cohorts of learners using a statistical analysis of learner responses on the Computer Programming Self-Efficacy Instrument (PSE) which was given out to all participants in both groups at the start and end of semesters 1 and 2. A number of statistical tests were carried out on the PSE results to assess changes in learners Self-Efficacy due to attending the PBL module:

- Group A's Computer Programming Self-Efficacy results at the start and end of Semester 1 were compared against Group B's results.
- Group A's and Group B's Computer Programming Self-Efficacy results at the start and end of Semester 1 were compared separately using t-tests.
- Any changes in Group A's Computer Programming Self-Efficacy during Semester 1 was compared against Group B's results.

Detailed results of the statistical tests carried out on the Self-Efficacy data are presented in Appendix L, while an overview is provided below.

In total 136 students took the PSE: 64 in 2007/08 and 72 in 2008/09. As can be seen in Table 4-5, when Group A's PSE scores were compared with Group B's at the start of semester one for each of the two academic years 2007/08 and 2008/09, there were some small differences in the mean scores but t-tests show that these differences are not statistically significant at the .05 level.

When both groups' PSE mean scores were compared at the end of semester one, after the teaching intervention, much wider differences in the mean scores can be seen. In 2007/08 and 2008/2009 the mean PSE scores for the PBL group were 181 and 179 respectively, while the mean scores for the non-PBL group were 153 and 138. Statistical analysis using t-tests shows that these differences are statistically significant at the .05 level. In 2007/08 the effect size was (ES = 1.53), and in 2008/09 the effect size was (ES = 1.92) Furthermore, when the data for the two years were tested as a single population, evidence was found that the hybrid PBL model produced a significant improvement in learners' self-efficacy with an overall effect size of (ES = 1.70). Therefore it can be said that learners who complete the PBL course will have a higher degree of programming self-efficacy than those in the control group.

4.5. Analysis of Students' Approaches to Learning (Hypothesis 4)

As described in section 3.6.2.4., participants' approaches to studying were measured over one cohort of learners, using a statistical analysis of learner responses on part B of the Approaches and Study Skills Inventory for Students (ASSIST), which was given out to all participants in both groups at the start and end of Semesters 1 and 2. Section B contains 52 items and produces scores on Deep, Surface and Strategic Approaches to learning.

An indicator of the success of the Hybrid PBL model is whether learners in Group A show higher scores on meaning orientation and lower scores on reproduction orientation than learners in Group B. A number of statistical tests were carried out on the results to measure and assess changes in learners' learning orientation due to attending the PBL module:

- Group A's approaches to study results at the start and end of semester 1 were compared against Group B's results.
- Group A's and Group B's Approaches to Study results at the start and end of Semester 1 were compared separately using t-tests.
- Any change in Group A's approaches to study scores during semester 1 was compared against Group B's results.

Detailed results of the statistical tests carried out on the approaches to learning data are presented in Appendix L, while an overview is provided below.

In total 72 students from the 2008/09 cohort took part B of the Approaches and Study Skills Inventory for Students (ASSIST). As can be seen in Table 4-6, when Group A's ASSIST mean scores for deep, strategic and surface apathetic approaches were compared with Group B's at the start of semester one, there were some small differences in the mean scores, but t-tests show that these differences were not statistically significant at the .05 level. However, when both groups' ASSIST mean scores were compared at the end of semester one, after the teaching intervention, much wider differences in the mean scores could be observed. The mean ASSIST scores for deep, strategic and surface apathetic approaches for the PBL group were 58.89, 66.06 and 40.10 respectively, while those for the non PBL group were 54.59, 71.01 and 47.63. Statistical analysis using t-tests show that these differences are statistically significant at the .05 level.

Furthermore, evidence was found that the hybrid PBL model led to an improvement in learners' meaning orientation, with an overall effect size of (ES = 0.35) on deep approaches to learning and a reduction in reproduction orientation, with an effect size of (-0.75) on surface apathetic approach. A small negative effect was also seen on the strategic approach with an effect size of (ES = -0.41). From these findings it can be said that learners in the PBL group will show higher scores on meaning orientation and lower scores on reproduction orientation than those in the control group.

Table 4-6: Students' Approaches to Learning: Deep, Strategic and Surface Apathetic Approach Scores (Hypothesis 4)

	Group A (PBL) Students' Approaches to Learning Scores			Group B (Non-PBL) Students' Approaches to Learning Scores		
Cohort	Mean (Rounded) (Start of Semester 1 Before PBL Teaching)	Mean (Rounded) (End of Semester 1 After PBL Teaching)	Mean Difference between Start and End of Semester Scores.	Mean (Rounded) (Start of Semester 1 Before Non PBL Teaching)	Mean (Rounded) (End of Semester 1 After Non PBL Teaching)	Mean Difference between Start and End of Semester Scores.
Deep Approach 2008/09	57.13	58.89	1.76	57.20	54.59	-2.61
Strategic Approach 2008/09	71.0	66.06	-4.94	69.87	71.01	1.14
Surface Apathetic Approach 2008/09	46.20	40.10	-6.10	46.79	47.63	0.84

4.6. Analysis of Learner preferences (Hypothesis 5)

As described in section 3.6.2.4., participants' preferences for different types of teaching were measured over one cohort of learners, using a statistical analysis of learner responses on part C of the Approaches and Study Skills Inventory for Students (ASSIST), which was given out to all participants in both groups at the start and end of semesters 1 and 2.

Section (C) invites students to indicate their preferences for different kinds of teaching. The scales are 'Supporting Understanding' which indicates a deep approach and 'Transforming Information' which indicates a surface approach.

An indicator of the success of the Hybrid PBL model is whether learners in Group A show a greater preference for teaching that supports deep learning than learners in Group B. A number of statistical tests were carried out on the results to measure and assess changes in learners' preferences due to attending the PBL module:

- Group A's preferences for different types of teaching preference scores at the start and end of Semester 1 were compared against Group B's preference scores.
- Group A's and Group B's Preferences for Different Types of Teaching scores at the start and end of Semester 1 were compared separately using t-tests.
- Any change in Group A's preference scores during Semester 1 was compared against Group B's scores.

Detailed results of the statistical tests carried out on the learner preference data are presented in Appendix L, while an overview is provided below.

In total 72 students from the 2008/09 cohort took part B of the Approaches and Study Skills Inventory for Students (ASSIST). As can be seen in Table 4-7, when Group A's ASSIST mean scores for Supporting Understanding and Transforming Information were compared with Group B's at the start of semester one, there were some small differences in the mean scores but t-tests show that these differences were not statistically significant at the .05 level. However, when both groups'

ASSIST mean scores were compared at the end of semester one, after the teaching intervention, much wider differences could be seen in the mean scores.

The mean ASSIST scores at the end of semester for Supporting Understanding and Transforming Information for the PBL group were 15.42 and 15.80 respectively, while the mean scores for the non-PBL group were 14.22 and 17.82. Statistical analysis using t-tests showed that these differences are statistically significant at the .05 level.

Evidence was found that the hybrid PBL model led to an improvement in learners' preference for Supporting Understanding approaches to teaching, with an overall effect size of (ES = 0.36), and a reduction in learners' preference for Transforming Information approaches to teaching, with an effect size of (-0.63). These results suggest that learners in the PBL group will show a greater preference for courses and teaching that support deep learning (as opposed to surface learning) than those in the control group.

Table 4-7: Learner preferences, Supporting Understanding and Transforming Information (Hypothesis 5)

	Group A (PBL) Learner preferences Scores			**Group B (Non-PBL) Learner preferences Scores**		
Cohort	Mean (Rounded) (Start of Semester 1 Before PBL Teaching)	Mean (Rounded) (End of Semester 1 After PBL Teaching)	Mean Difference between Start and End of Semester Scores.	Mean (Rounded) (Start of Semester 1 Before Non PBL Teaching)	Mean (Rounded) (End of Semester 1 After Non PBL Teaching)	Mean Difference between Start and End of Semester Scores.
Supporting Understanding 2008/09	13.97	15.42	1.45	14.54	14.22	-0.32
Transforming Information 2008/09	16.93	15.80	-1.13	17.63	17.82	0.19

4.7. Summary

This chapter outlined a randomised controlled experiment that took place over four cohorts of learners. The experiment tested five hypotheses: around whether using a Problem-Based Learning approach instead of conventional lectures in the subject of Software Development:

1. improves learner attainment in the subject;
2. improves learner motivation to learn the subject;
3. improves learner Software Development self-efficacy;
4. changes learners' approaches to studying the subject (towards a deep approach);
5. changes learners' preferences for different types of teaching (towards teaching that supports understanding).

The results show that using a Problem-Based Learning approach instead of conventional lectures brought about statistically significant:

- improvements in learner attainment in continuous assessment (ES = 0.40).;
- improvements in learner Software Development self-efficacy (ES = 1.70);
- changes in learners' approaches to studying the subject (towards a deep approach) (ES = 0.35);
- changes in learners' preferences for different types of teaching (towards teaching that supports understanding) (ES = 0.36).

However, the results showed no statistically significant effects on:

- improvements in learner attainment in exams (ES = -0.07);
- improvements in learner motivation to learn the subject (ES = 0.23).

Chapter 5 will outline the second part of the study where additional data was collected and analysed and where a qualitative study was also undertaken. A full discussion of the findings from Chapter 5 will be undertaken in Chapter 6.

Chapter 5 - Additional Data Collection and Qualitative Findings

5.1. Introduction

This chapter has three sections. The first examines the classroom activities of all learners participating in the study based on the log files stored on a number of databases. Secondly an analysis of students' responses to two questionnaires is undertaken. Finally, the results of a qualitative study based on observations and interviews are analysed. A full discussion of the findings is undertaken in Chapter 6.

5.2. Database Information Mining

Over the four years of the study, detailed information was taken from a number of Anon College's databases and online systems (Virtual Learning Environment, Attendance Registers, Student Computer Network Audit Logs, and College Management Information System). This data allowed an analysis of student attendance and participation. A summary of the results is given in Table 5-1.

Table 5-1: Database Information Mining (4 cohorts)

05/06, 06/07, 07/08, 08/09 Grouped Data on:	Group A Mean (Rounded)	Group B Mean (Rounded)
Class attendance	61 out of 78 classes	45 out of 78 classes
VLE usage (Weekly, theoretical maximum possible time 10080 minutes)	115 minutes	156 minutes
Time logged onto PC during labs (Weekly, maximum possible time 360 minutes)	49 minutes	322 minutes

5.2.1. Analysis of Learner Class Attendance

An analysis of the student attendance registers showed that on average each semester the PBL group attended 61 out of 78 classes, while the non-PBL group attended 45 out of 78 classes. Therefore the PBL group attended on average approximately 20% more classes. This large difference was seen in all four years of the study. An important point to make in relation to student attendance at class is that at Anon College all students receive a European Social Fund Grant, the value of which is based on their rate of attendance.

5.2.2. Analysis of Virtual Learning Environment Learner Logs

The Moodle Virtual Learning Environment (VLE) logs record when and for how long learners access Software Development course notes (slides, tutorial sheets, examples, etc.). Analysis of the VLE logs showed that the PBL Group spent less time accessing course material than the non-PBL group. This difference may be due to the fact that the PBL group do their problem-solving as a group without course notes away from the PC, while in contrast the non-PBL group work at their PC with course notes when working on problems.

5.2.3. Analysis of Computer Network Logs

The Computer Network logs record when students log on to and log off their PC. Analysis of the Network logs taken of Software Development Laboratories shows that the PBL Group spent only about 15% of the time the non-PBL group spent logged on to their PC. This large difference is again likely to be due to the fact that the PBL group do their problem-solving as a group on the white board away from the PC, while in contrast the non-PBL group work individually at their PC on problems. Also, only logs of Software Development laboratories were analysed. Students often spend time working at the PC on Software Development problems outside of class contact hours.

5.3. Analysis of Questionnaire Responses

As outlined in section 3.8., two questionnaires were given to learners, firstly a background questionnaire to all students and secondly a PBL questionnaire to the PBL groups. This section details the results of these questionnaires. The questionnaire responses of the four cohorts were analysed as a single population.

5.3.1. Analysis of Learner PBL Questionnaires

The PBL Questionnaires were given to all students in Group A (PBL Group) who completed the module for the four academic years 2005/06, 2006/07, 2007/08 and 2008/09. In total 122 students were given the questionnaire with 106 students completing it. The PBL questionnaires are contained in Appendix F1 and students' answers are given in full in Appendix M. A summary is provided in the following sections.

The questionnaire focuses on six areas influencing students' opinions and decisions on Software Development:

- PBL Group work;
- the PBL method;
- student interest in Software Development;
- course objectives and content;
- the PBL tutor;
- teaching resources.

5.3.1.1. PBL Group Work

Students had broadly positive opinions of the work done in the PBL group. 69.8% (74 out of 106) felt that the tutorial group discussions were an important stimulus for their Software Development learning activities, while 73.6% (78 out of 106) considered that the learning issues generated in the group tutorials were the most important starting point for their learning activities. 63.2% of the students (67 out of 106) said they did not study independently of the learning issues generated by the PBL group tutorials, which reinforces the view that the learning issues generated in the group tutorials were an important starting point for students' learning activities. Most students (79.2%, 84 out of 106) felt that they learned something in the PBL tutorials that improved their Software Development skills. However, 46.2% (49 out of 106) of students did not feel that the PBL tutorials improved their communications skills. 63 out of 106 students (59.4%) said they would recommend PBL tutorials to other students.

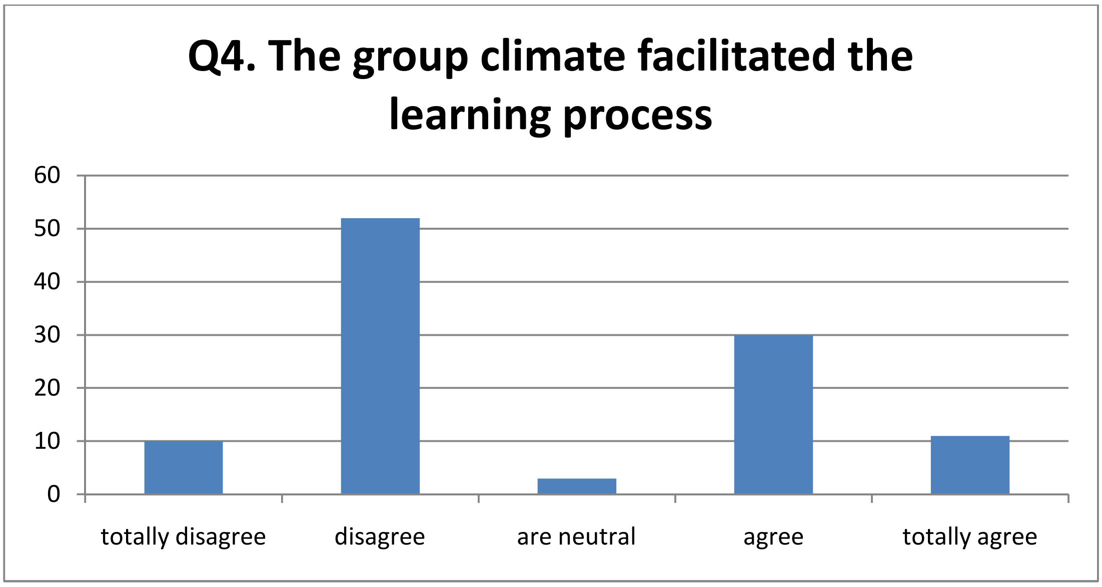


Figure 5-1: Q4: The group climate facilitated the learning process

As seen from Figure 5-1, one interesting finding was that 62 out of 106 students (58.5%) felt that the group climate did not facilitate the learning process, a result that held true for all four years of the study. This was a surprising result given the students' other answers about the PBL groups, and it was investigated further in the interviews with students and in the classroom observations.

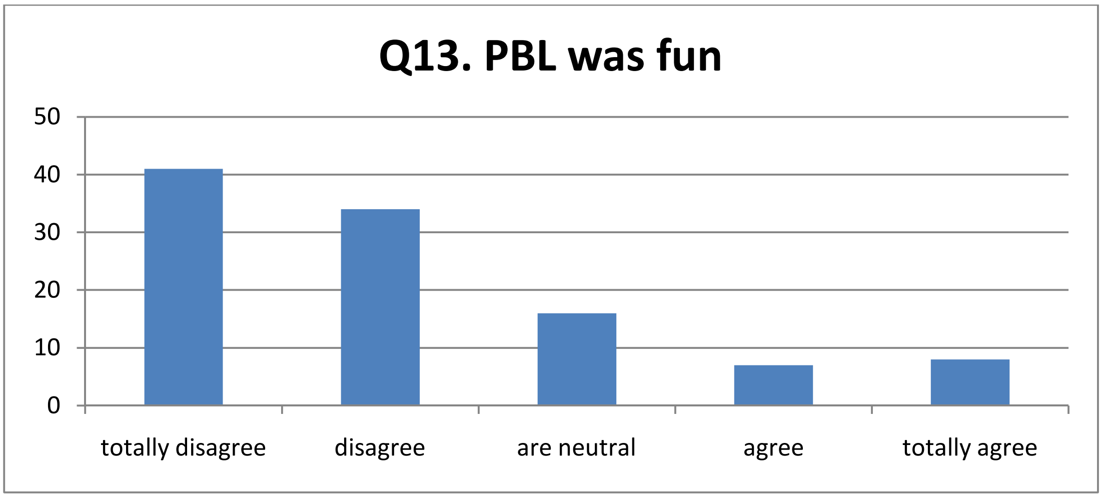


Figure 5-2: Q13: PBL was fun

5.3.1.2. The PBL Method

100 out of 106 students (94.3%) were open to the PBL method before the tutorials but were neutral when asked whether, if they had had the possibility to choose before the course, they would have opted for the PBL-course or the lecture-based course.

This may reflect a lack of information about PBL at the start of the course. However, at the end of the PBL course, 93.4% (99 out of 106) felt they were well informed about the PBL method.

When asked whether, after their experience of the course, they would now opt for the PBL-course or the lecture-based course if they had to choose again, 62.2% (66 out of 106) chose the PBL course. However, as seen in Figure 5-2., 75 out of 106 students (70.6%) did not consider that PBL was fun, or that the PBL classes motivated them to use additional learning resources, although 64.1% (68 out of 106) did consider PBL to be an effective way of learning for themselves. The finding that students did not find PBL fun but that they did consider PBL to be an effective way of learning was investigated further in the interviews with students and in the classroom observations.

5.3.1.3. Student Interest in Software Development

80 out of 106 students (75.5%) were interested in Software Development and considered it to be an important part of their study of Computing. However, other than class attendance, the mean amount of additional learning time (per student) invested each week in Software Development was just under two hours. This would be considered less than adequate by academic staff within the Computing Department and was investigated further in interviews with students. However it should be noted that in informal conversations many students said they were 'too busy' working in paid employment to 'study hard'.

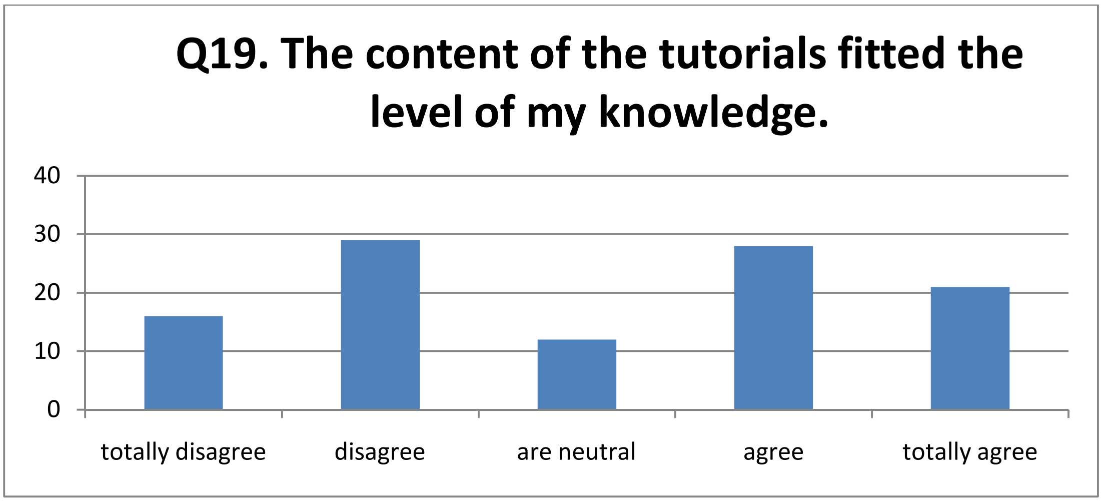


Figure 5-3: Q19: The content of the tutorials fitted the level of my knowledge

5.3.1.4. Course Objectives and Content

Slightly over half of the students (52.8%, 56 out of 106) felt that the topics covered during PBL classes stimulated their interest in Software Development. Nonetheless, as shown in Figure 5-3., 42.45% of students (45 out of 106) did not feel that the content of the tutorials fitted their level of knowledge.

59.4% of the students (63 out of 106) agreed that the problems used in the PBL classes illustrated Software Development concepts. However, students were ambiguous about whether the learning issues generated in the PBL classes were tuned to the subject matter to be tested.

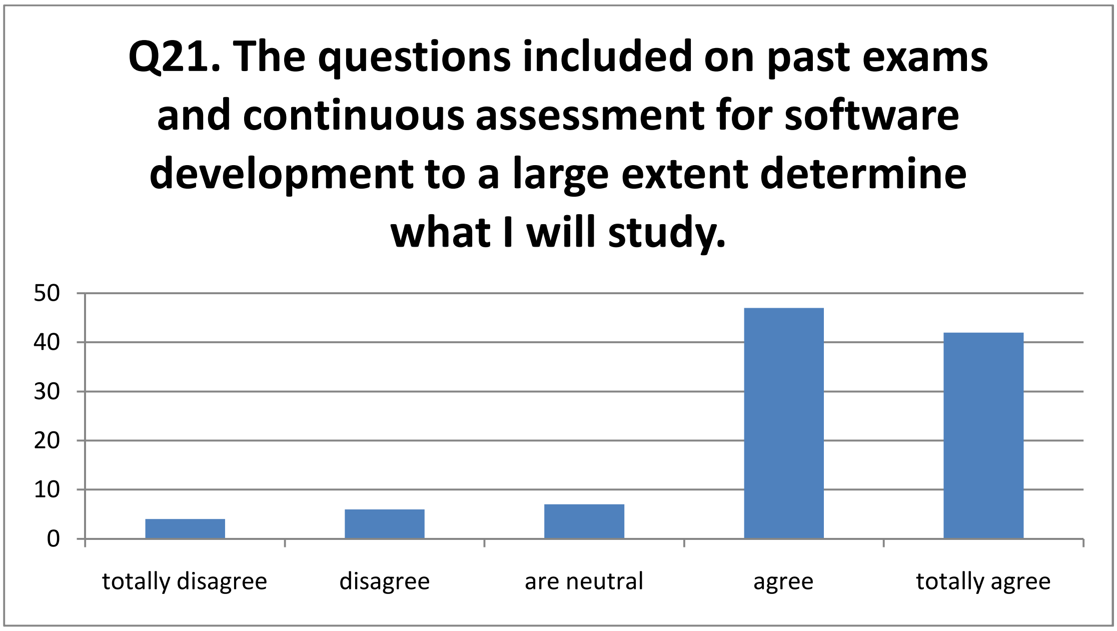


Figure 5-4: Q21: The questions included on past exams and continuous assessments for Software Development to a large extent determine what I will study

As shown in Figure 5-4., 89 out of 106 students (84%) agreed that the questions included on past exams and continuous assessment for Software Development to a large extent determined what they would study. This strategic approach to learning was to a degree limited by the fact that a majority of students did not consult the course objectives set out in the Software Development syllabus, either at the start or end of the Software Development course, to check whether they had covered all the subject matter they were expected to cover in Software Development (84%, 89 out of 106 at the start and 63.2%, 67 out of 106 at the end).

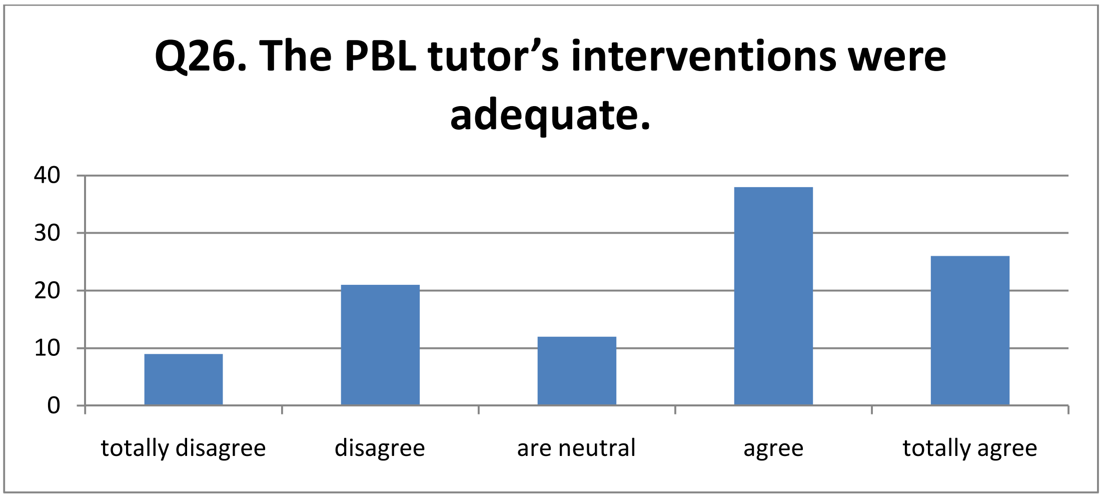


Figure 5-5: Q26: The PBL tutor's interventions were adequate

5.3.1.5. The PBL Tutor

72.6% of students (77 out of 106) felt that the PBL tutor had steered the group strongly. But surprisingly, as seen in Figure 5-5., 60.4% (64 out of 106) felt that the PBL tutor's interventions were inadequate. 74.5% (79 out of 106) felt their PBL tutor was enthusiastic about PBL, though 52.8% (56 out of 106) did not feel that the tutor stimulated students to make use of different sources of information and 40.6% (43 out of 106) did not consider that in general the tutor stimulated their Software Development learning activities. These findings were further explored in the interviews.

5.3.1.6. Teaching Resources

Overall students were very happy with the teaching resources used in the teaching of Software Development. 95 out of 106 (89.6%) students were content with the classrooms; laboratories and computer equipment used in the Software Development course. 92 out of 106 (86.8%) students were satisfied that the Moodle e-learning environment supported their learning activities, while only 11 wanted more timetabled PBL Software Development classes.

5.4. Qualitative Findings

The Qualitative findings are presented in the following sections and will be discussed in chapter 6.

5.4.1. Analysis of Classroom Observations

Each week of the 15 week semester the researcher spent half an hour in PBL and half an hour in non-PBL Software Development laboratories observing participants' behaviour. This was done for each of four cohorts of learners over four academic years, resulting in a total of 80 hours of participant observation. The full observation schedule is given in Appendix I.3.

No observable difference between groups was noted in relation to students' arrival and departure times at class. However, there was a difference in the numbers of students attending each class, with the PBL students having a better attendance. This finding was supported by the analysis of Learner Class Attendance data (see Section 5.2.1).

There was a striking difference between how the PBL and non-PBL groups spent their classroom time. The non-PBL group spent much more time 'logged-on' to their computers, while the PBL groups spent more time developing solution plans and schemas on paper.

Students' in-class behaviour showed a difference between groups. A large proportion of the non-PBL groups' time was spent 'off task' engaged in entertainment-related computer activities such as emailing friends, playing computer games, surfing the web, watching Youtube videos and interacting with their social networks (Bebo, Myspace, etc.). At times the non-PBL groups would spend up to 60% of class time on these activities. Consequently students really enjoyed their time in the computer lab although they got little useful Software Development work done. The non-PBL groups showed little intra group tension, with students chatting and laughing about their social activities.

The PBL groups only spent about 10% of class time 'off-task', usually chatting to other members of their tutorial group about their social life. The rest of the time was spent 'on task' discussing Software Development problems. The vast majority of these conversations were good natured but sometimes (in approximately one lab in every five) intra-group tension and conflict was apparent, and on a limited number of occasions the tension escalated into open conflict between group members. This conflict was always non-physical, involving shouting, heated arguments and name calling. The conflict sometimes led to an unsuccessful laboratory session where the time wasted arguing resulted in the problem not being solved during the allocated time. Successful sessions seemed to rely most crucially on balanced discussions between the students and careful preparation of the session by the tutors/lecturers.

PBL groups that worked well had a majority of the following characteristics: group members participated in the discussion of the PBL problems; the atmosphere in the group was relaxed; group members had done some preparation for the tutorial sessions; and group members attended nearly all PBL classes. Problem PBL groups had a majority of the following characteristics: group members did not participate in the discussions of the PBL problems, with some members making no contribution at all, consequently, the programming problem was solved only by one or two conscientious group members; the atmosphere in the group varied from pleasant to distant and tired; and some members were dominant and had very strong opinions, which they failed to convey or make comprehensible to other members.

It was also noted that the PBL groups sought less direct assistance from tutors/lecturers than the non-PBL groups. Tutors in the non-PBL groups spent a lot of time answering basic programming syntax questions and were asked few questions about alternative solution schemas. In the PBL groups this was completely reversed, with the focus on schema construction.

In the PBL groups it became clear that students had many misconceptions of, or an incomplete understanding of, the problems under discussion. Tutors/lecturers would make notes of these deficiencies and then cover material aimed at rectifying the problems at the end of the tutorial session. In the course of exploring a problem, the members of the PBL group inevitably discovered areas in which their collective knowledge was deficient. Staff would hope that, having recognised such a deficiency, students would study the topic further outside of the tutorial session, and then bring back new knowledge to the tutorial group during the next class. However, observations of the PBL groups showed that students seldom did this.

The Software Development problems used in laboratories aimed to help students master basic object orientation and abstraction. The same set of programming problems was used in both the PBL and non-PBL groups. However, the way in which the groups tackled the search for solutions to these programming problems was strikingly different. Students in the non-PBL group worked individually on problems seated at their computers. Students' efforts focused on writing Java code and then trying to compile this code into a running programme. Generally they had numerous (usually between 10 and 30) syntax errors in their solutions, and their focus would turn to removing the syntax errors in their code. They would then ask basic syntax-related questions of their tutors. This process was quite demoralising for students as even after fixing a large number of syntax problems and getting their code to run, they would find that the solution schema they had used did not produce the correct answer. They would then try to modify the schema, in the process introducing new syntax errors into their code and the whole process would start again. Sometime students would become disheartened and would stop working on their Software Development problems and start to surf the web instead. Tutors also found this process irritating as they spent so much time answering basic code syntax-related questions rather than more advanced programme construction schema questions. They constantly had to exhort students to work on their lab sheet instead of surfing the web. Sometimes tutors would become bored and disengaged from the students: so much so that they would surf the web or answer emails themselves. Overall the learning effort was individual and mainly focused on programme syntax, with trial and error attempts to develop a correct programme schema.

Students in the PBL groups worked in teams based around a white board and away from the computers, which would usually be switched off. Students' efforts focused on describing the correct algorithm or schema to solve the programming problem. The process would usually involve one or two of the more 'talented' group members suggesting the steps towards a solution. The talented members would explain their choices to the less talented group members, some of whom would withdraw from the group discussions; indeed, many seemed not to take an active part in the search for solutions at all. Sometimes the more talented members could not agree a set of steps towards a solution. When this happened they would argue about the different approaches until they either decided on one approach or asked the tutor for the 'correct' answers. Tutors would then try to get the whole group to consider the issues and see if they could approach the problem from a different perspective; seldom did tutors ‘impose’ a solution. This process would continue until the group felt they had a correct and working solution schema. Then they would have the tutor examine their solution for flaws and sometimes this would lead to a revision of the proposed solution. Students then recorded their solution schemas in their PBL journals. Overall it appeared that the learning effort was team-based and focused primarily on developing a correct programme schema, rather than on code syntax.

Observation of the students revealed that most students in the PBL groups constructed new contextualized knowledge through the process of problem-solving and through their search for solutions to the problems. However, while most students in the PBL groups took ownership of the problems, a minority did not take an active part in problem-solving. This was also the case in the non-PBL groups where some students seemed to be helpless in the face of difficult problems. The non-participation of some students was investigated further during the interviews with learners.

These direct, first-hand observations of in-class behaviour provided a high face validity of data and an understanding of group behaviour. The observed tensions within the PBL groups were investigated further in interviews with staff and students.

5.4.2. Analysis of Students’ PBL Journals

All PBL students are required to keep a journal of their classroom activities. A random sample of these journals was taken each year over four cohorts and analysed. Ten journals were analysed in the first year and five journals in each subsequent year. In total twenty-five journals were analysed. These journals are not reflective of and do not record students’ feelings about, or experiences of, the learning process. Nonetheless, they do provide a measure of student engagement and interest in problem-solving. One thing that became clear from looking at the journals was that the vast majority of students kept their journals up to date and had clearly recorded many aspects of their problem solving work. For example, details of different programming problems, initial attempts at solutions, and fully completed efficient and effective solutions were recorded. These problems ranged from simple introductory problems to advanced coding problems using complex programming structures such as two dimensional arrays. The journals clearly showed students involved in the process of problem solving, outlining a progressive series of possible solution schemas, incrementally working towards an optimal solution. An analysis of the plans and schemas developed by students showed that work on abstraction and object orientation were key components of the solutions developed. A key point identified was that abstraction and object orientation were made explicit within the solution process undertaken by the PBL students.

5.4.3. Analysis of Informal Conversations

The researcher had a number of informal conversations about their experiences of Software Development with students from both the PBL and non-PBL groups in each of the four academic years of the study. The researcher also had informal conversations with staff about their opinions of PBL.

One interesting finding was that a number of students felt that the group climate did not facilitate the learning process. This point was raised mainly by the female students. The main problem, they suggested, was that some students always wanted to lead or take over the group, and when others resisted, arguments ensued. Two female students stated that this problem was caused by 'non-nationals', in particular male Asian and African students. When the author discussed this group interaction problem with staff, they pointed out that it was a problem in a very small number of groups and that a number of steps had been taken to address this problem, such as tutors assigning and rotating roles (scribe, etc.) between all members of the PBL group. When the author asked staff members if this was a particular problem for groups containing non-national students, none of them said the problem was related to nationality or ethnicity, although they did suggest it was sometime caused by male students trying to impress other group members. These different possible causes of conflict within PBL groups were investigated further in the interviews with staff and students.

5.4.4. Analysis of Interviews

In the fourth year of the study eleven semi-structured interviews (five with staff and six with learners) were conducted. Each interview lasted between 10 and 15 minutes.

Interviews were carried out over a two week period at the end of semester one. After the interviews, detailed field notes were made of how the interviews proceeded.

Transcription of tapes was carried out after the interviews were complete. The process of transcribing digital audio inevitably meant the loss of some data from the original interviews as it involved translating verbal and non-verbal material into the rules of written language (Cohen *et al.*, 2000). Miles and Huberman (1984) note that transcripts are unavoidably selective so the outcome is material that has undergone not only reduction, but also a transformation and a form of interpretation. To minimise any errors, great care was taken in the making of transcripts, and notes were taken during the interviews regarding body language, voice inflection, mood, interruptions and facial expressions. Transcriptions are presented in Appendix J.

Four male and two female learner participants, and two male and three female staff/tutor participants were selected at random and asked to take part in an interview. All names used are pseudonyms. It should be noted that all findings based on the interview data are tentative due to the small number of interviews conducted and their short duration.

5.4.4.1. Staff Interviews

The set of interview questions used for staff was adapted from Maudsley (2002) and are listed in Appendix G. Eleven questions were asked of staff, exploring how they conceptualised their students' learning. In addition, questions were asked that explored some of the observations noted in section 5.4.1. The names of all staff members who taught on the PBL module were put in a hat and five names drawn at random. If a staff member refused to be interviewed, another name was drawn from the hat. Five staff members were interviewed (Catherine, Stuart, Mary, David, and Natasha). Catherine, Stuart, Mary and David are all in their late thirties/early forties and are Irish full-time members of academic staff, while Natasha is in her early twenties and is a foreign postgraduate student working part-time as a tutor. Stuart was the overall PBL coordinator and was in charge of the PBL module. All five staff members had also taught in the non-PBL Software Development laboratories. This allowed them make comparisons with students' behaviour in non-PBL Software Development laboratories.

The results of the staff interviews showed that some staff found their PBL experience rewarding and most had a high level of enthusiasm, but some also reported an increased workload. Some staff felt that while management was supportive of PBL, the extra workload involved was not recognised. There was also some tension between the PBL coordinator and other staff. Stuart felt that the other staff members were not doing their fair share of the preparatory work, while some of the other staff thought that Stuart did not delegate work to them and wanted all the credit for undertaking PBL.

Staff reported an improvement in students' motivation levels and better student engagement with the PBL course. They also felt that students' critical reasoning skills had improved vis-à-vis non-PBL, particularly in relation to their ability to create programme schemas and plans. In addition, staff felt that learners took more ownership of the Software Development problems and asked fewer 'basic' questions of tutors. The following excerpt from the interview with Catherine gives a flavour both of the satisfaction tutors gained from seeing the PBL students' progress, and of the tensions between staff teaching PBL:

> ***Interviewer: "For PBL then, what do you see as its main advantage…..main disadvantage?"***
>
> ***Catherine: "Well...I think the good thing is seeing how involved with the problems the students get. And after they have tried their initial solutions, working with the students is enjoyable, much better than in the ordinary labs."***
>
> ***Interviewer: "Why is that?"***
>
> ***Catherine: "Because they are actually trying to construct a solution that works rather than just stopping every time they have a problem and asking for help, so their labs are much better."***
>
> ***Interviewer: "Any disadvantages?"***

> ***Catherine: "Yes, there is a lot of extra work preparing for a lab."***
>
> ***Interviewer: "Why is that?"***
>
> ***Catherine: "Well in the ordinary labs you are never stuck for a solution as the questions are so simple. But in the PBL labs you come up against complex problems that you have to figure out on the spot. But look the worst thing is having to deal with X [other staff member]. [Segment of interview omitted[5]]***

It may be that the observed tension between staff was a result of personal animosity between individuals. It is certainly the case that teaching Software Development using PBL requires greater interaction between the staff and this may exacerbate existing tensions or bring them to the fore.

Staff felt that the VLE technologies used in Anon College had a positive impact on the PBL tutorials. However, they noted that some students tended to use the internet to access coded solutions and get immediate answers to questions. Staff also questioned whether the process of accessing information from the internet reduced their students' depth of understanding of Software Development concepts. Concerns were raised as to the reliability and validity of the code accessed on the internet. It should be noted that this was not just a problem in the PBL group: it was even more the case for the non-PBL group due to the fact that they spent more time logged on to their computers.

When asked during the interviews to account for the time learners spent off task in the non-PBL labs, staff stated that they felt management would not support them if they insisted on students concentrating solely on Software Development problems. Staff also felt under pressure to 'entertain' their students.

One interesting point made by both Stuart and Catherine during the interviews was that students working in the non-PBL group remained stuck dealing with syntax

[5] See note at start of Appendix J.1.

issues, rather than mastering the concepts of abstraction and object orientation. They both noted that the PBL students seemed to better master these concepts than the non-PBL students. Other staff members also noted that the PBL students did not get stuck at the syntax level. The following excerpts give a flavour of these issues. Here is what Catherine said:

> ***Interviewer: "Have you noticed any change in learner behaviour in the PBL lab classes as opposed to the traditional lab environment?"***
>
> ***Catherine: "In the PBL labs the students actually keep working on their problems and are not surfing the web and using Bebo. You don't have to be constantly asking them to stop messing."***
>
> ***Interviewer: "Have you noticed any change in how students go about problem-solving?"***
>
> ***Catherine: "Yes well it is quite different. Em, firstly the PBL students are in a group and away from the machines. They are trying to get the steps in their algorithm right. Whereas the others are always stuck on syntax."***
>
> ***Interviewer: "In your opinion, have the students' problem-solving skills improved more in the PBL lab classes as opposed to the traditional lab environment?"***
>
> ***Catherine: "Yes I think they have. They have better critical reasoning skills."***
>
> ***Interviewer: "Can you expand on that?"***
>
> ***Catherine: "I mean they can have a higher level focus on the problem and they work out their overall problem-solving approach before they start to code. They have a plan of what they want to do. Also they understand the underlying concepts***

> ***better. They know what is meant by abstraction for example. They know what you mean when you talk about an object-oriented programme. They know what a class is."***

Here is what Stuart had to say:

> ***Interviewer: "Have you noticed any change in learner behaviour in the PBL lab classes as opposed to the traditional lab environment?"***
>
> ***Stuart: "Em, my labs are much more professional. The students love the problems and love working together. (Laughs) When I bring in the whiteboard they fight over the pens. They really enjoy the labs. They only thing is I'm so busy I never get a break, I have to get around to all the groups. Yesterday I missed lunch, but I don't care, it's fun."***
>
> ***Interviewer: "Have you noticed any change in how students go about problem-solving?"***
>
> ***Stuart: "My students are much better team workers and they are better problem solvers because I make them work on their problems at a high level, working out the correct set of steps in pseudo code before they go near Java."***

The interview David also suggests that PBL helps develop learners' problem-solving skills:

> ***David: "I suppose the main advantage would be the group work, building a common solution, building on suggestions of other students and trying to see the positives and negatives for each suggestion and then building that into a solution without as I say getting bogged down in low level details of how it would be solved at a code level or whatever. So it's just focusing on the problem as a whole."***

> ***Interviewer: "Have you noticed any change in learner behaviour in the PBL lab classes as opposed to the traditional lab environment?"***
>
> ***David: " […] they're not really thinking about the larger problems in the low-level traditional labs. They're more stuck with syntax errors when they're starting programming. So they're bogged down in that rather than thinking about how they would solve the problem and worried about viewing bits of code that might do what they want rather than thinking about it in a logical step-by-step what do I need to do, not what's this bit of code that I might need to use. So it kind of focuses their thinking a little bit more."***

David also raised the issue of tensions in the PBL groups which as will be seen later, was also an issue for some of the student interviewees.

> ***David: "Sometimes you might get a loud mouth or someone who's a bit more vocal than the others or may have some prior knowledge or just natural ability and it's their way or no way and that can be a little bit unfortunate … you need to get everybody contributing and not just agreeing with the leader. I haven't really seen too much of people being shouted down but there have been. It's more … somebody might say something stupid and they get a little bit of slagging[6] for it and they don't contribute anything more after that. So you have to try and kind of guide that in the right way."***

Stuart and Catherine suggested during interviews that PBL had fostered in students not just a greater interest in Software Development in first year but also in Computing in general and that this effect lasts into the following years of their study.

[6] To slag (Hiberno-English slang): to mock or to criticize someone in an unpleasant way

During the interviews some staff members acknowledged that some students were not happy working in a group, especially the weaker less confident ones. The following excerpts from the interviews with Mary and Natasha demonstrate this point:

> ***Mary: "One of the main disadvantages that I've found in my work was to do with the way we structured the groups. I found that the groups were far too big, that a lot of people got lost within the group … there's some very extrovert people and very quiet people and it didn't really suit some people. It suited some, and others it didn't, so you're losing some people in that setting."***
>
> ***Natasha: "The disadvantage is that the students who are weak, like, they don't really need to work in the PBL because the stronger students do everything for the group…"***

Catherine noted that a disadvantage of PBL was that:

> ***"only some students do the work, the rest are just too weak to work out the problems so they just tag along. Also most of the students don't know where to start, they have no idea.***

David on the other hand, considered that the PBL environment actually helped some less confident students, as the following extract shows:

> ***David: I think they enjoyed it. Certainly in the latter stages they seemed to get a decent amount out of it especially the group work. Actually some of them might be a bit reluctant say in a lecture to speak up but in a small group setting may be more inclined to volunteer their opinion.***

5.4.4.2. Learner Interviews

Six learners from the 08/09 cohort were interviewed (Sarah, Ahmed, Nichole, Paul, William and Darren). Sarah, Nichole, Paul, William and Darren are all Irish full-time students, while Ahmed is a foreign student studying in Ireland. All are between the ages of nineteen and twenty-one. The set of interview questions for students is given in Appendix H and the transcriptions of the interviews are provided in Appendix J.

The interviews with learners supported the opinion of staff that there were differences in students' comfort levels when involved in teamwork. The stronger or at least more confident students were happy to contribute to the group, while the weaker, or less confident, students were inclined not to contribute. The following excerpt from the interview with William demonstrates this point:

> ***Interviewer: Did you enjoy the PBL classes?***
>
> ***William: "No. Not at all."***
>
> ***Interviewer: "Why was that?"***
>
> ***William: "Cos like you're sitting there and if you didn't know anything and then the lecturer comes over and starts talking to you and you feel under pressure and it's like ooooh I don't know anything and everyone else is just standing around and they're doing everything and you just haven't got a clue what's going on so it's not really very nice just sitting there like that."***
>
> ***[…]***
>
> ***Interviewer:" Do you think the PBL group environment facilitates the learning process?"***
>
> ***William: "No. I didn't learn much in it."***
>
> ***Interviewer: "Why did you think that was? Was it as you were saying that you felt a bit lost?"***
>
> ***William: "Yeah, they knew everything and if you were lost and behind and all that and they were flying ahead of you so you don't know and they're moving on. You don't really know what***

> ***they're on about like cos you're like you're falling behind. You're unsure." […]***
>
> ***Interviewer: "Did you feel isolated or uncomfortable in your PBL Group?"***
>
> ***William:" Yeah, I did, because if you don't know what you're doing you just feel, you know, you're just sitting there."***

Four of the six interviewees gave positive answers when asked if working together with other students in the PBL groups helped them make friends. PBL seems to have facilitated students in developing a peer group support network. The following excerpt from an interview with Darren was indicative of the general opinion:

> ***Interviewer: "Did working together with other students in the PBL groups help you make friends?"***
>
> ***Darren: "Oh yes, you get to meet all the other people in the class and as you're working together you get to really know them. You wouldn't be friends with them all but you get to know people and you'll know who to ask if you get a problem."***

These results show that the PBL model used in Anon College may provide a good transition for students to a third-level environment.

In their interviews, Sarah and Ahmed noted that the stated role of the tutors outlined in the induction at the start of the PBL course diverged from the actual situation that emerged. The students were given to believe that they would have a choice of topics to cover and that they could set some of their own learning objectives, depending on their prior knowledge. This element of choice, they claimed, did not transpire; in fact the tutors decided on the problem topics and set all the learning objectives. For example:

> ***Interviewer: "Did you feel well informed about the PBL method?"***

> ***Sarah: "Yes, but I thought we would be able to pick some of the problem areas ourselves, but the tutors always set the problems. We had no say in choosing them."***
>
> ***Ahmed: "Yes I did, but I think the tutors didn't. You see I did Java before coming here and I wanted to pick problems on arrays or methods but they didn't allow that. I had to do the same problems as everyone else."***

Learner control is referred to in many of the definitions of student-centred learning (Boud & Feletti, 1998), and is an important goal of the implementation of PBL at Anon College. The finding from this study suggests that there is a role for further PBL research to examine how the power relationships between staff and learners in the learning process should best be developed.

PBL questionnaire responses showed that a majority of the students (64 out of 106) felt that the PBL tutor's interventions were inadequate. When this was explored further in the interviews it transpired that some of the students interviewed found the method troublesome. For example, during her interview Nichole said that PBL was '*too much effort*' and thought the tutors were '*a bit lazy'* or '*didn't know how to solve some of the problems'*. This comment suggests that some learners were challenged by having to try to solve the problems '*on their own*' and that they expected the tutor to provide them with the '*right answer*'. Such feelings might explain why some students said during interview that they did not like the course structure and would have preferred to work independently or in smaller groups. For example, William stated that:

> ***William: "Classes should be smaller. [……] I think you need a lecturer beside you. Like if there was a lecturer like there'd be two between four or five groups and that's not enough. I think you need more lecturers. Like I think you need to look at that and there should be less people within the class rather than having so many groups and the lecturer not being able to help you that much like."***

> ***Interviewer: "Ok, so you'd like to see the lecturers help a bit more."***
>
> ***William: Yeah, and less students in the PBL groups. We had eight in our group sometimes and that's too much. You're not going to learn much.***

Nichole went on to say that she did not feel she had actively participated in the PBL sessions.

> ***Interviewer: "Why didn't you take an active part in the sessions?"***
>
> ***Nichole: "I couldn't be bothered. I'm a bit lazy in the mornings. I'll ask someone to go over it with me later on, and then I'll know the solutions for the exams. I don't like talking in the big group; if you make a mistake, they will always say you're wrong. Not everyone like, but some of them will."***

Nichole's comment suggests that the PBL group environment did not suit some students. This view is supported by results from the PBL questionnaires, with a majority of students stating that the group climate did not facilitate the learning process. The following excerpts from interviews with William and Paul illustrate the fact that some students did not learn in the groups:

> ***Interviewer: "How did you find the distribution of the work between group members?"***
>
> ***William: Yeah, it was ok but there was always certain people. Like, half of them would know what they were doing and the other half was just sitting there not knowing what was going on so the people that knew what they were doing would communicate between themselves and we kind of just sat back and you may say the odd thing whether it was right or wrong***

> ***but people who knew what they were doing they just moved ahead really. We were falling behind still.***
>
> ***Paul: "The stronger ones took on more but everyone contributed in my personal group. I saw other groups where it was desperate, where two people did everything but in mine personally it was brilliant."***

Given the very low numbers of females in the classes it is possible that they felt isolated and uncomfortable in a mainly male atmosphere. However, during their interviews, both female participants stated that they were not isolated. Rather, they did not like arguments and some male members of the group did not contribute to the problem solving, instead they tried to '*act like jokers*'. These issues were explored further in the interviews. The following excerpts from the interviews with Nichole and Sarah give a flavour of the difficulties. Firstly Nichole:

> ***Interviewer: "Do you think the PBL group environment facilitates the learning process?"***
>
> ***Nichole: "Yes in a way. It's good to be in the group because you get help solving the problems, but the guys are always trying to show off who knows the most, and impress the tutors. I couldn't be bothered with all that stuff."***
>
> ***Interviewer: "Did you feel isolated or uncomfortable in your PBL Group"?***
>
> ***Nichole: "No not at all, working in a group was good, but the problems were hard but I made friends and stuff.***

When asked the same question Sarah had a different perspective:

> ***Interviewer: "Do you think the PBL group environment facilitates the learning process?"***

> ***Sarah: "Yeah, but not everyone does the work. Some of the guys just sit back and let others do all the work. They don't do any preparation or nothing, but they still get credit when we get the right answer. It's not fair, the tutors don't do anything about it."***
>
> ***Interviewer: "Did you mention it to the tutors?"***
>
> ***Sarah: "No, I'm not going to get into a fight over it. Like, it doesn't bother me that much. And some of the guys are me mates anyway, so I'm not going to cause trouble. You know what I mean? Anyway the tutors are paid to do the job, aren't they? They should notice and do something about it."***
>
> ***Interviewer: "Did you feel isolated or uncomfortable in your PBL Group"?***
>
> ***Sarah: "No that's not what I'm saying. I wasn't isolated or uncomfortable, I just didn't like it that I had to do more work than some of the others."***

The PBL questionnaire results showed that many of the students did not feel the PBL sessions were fun. When the interviewees were asked about this they gave the following replies:

> ***Interviewer: "Do you enjoy the PBL classes?"***
>
> ***William: "No. Not at all."***
>
> ***Nichole: "No not really, they are ok, but they're a bit boring. You can't do anything but talk about the problems. That's not fun, that's work. I'd rather be doing something else."***
>
> ***Sarah: "Yes, they are enjoyable. Some of the problems are very difficult and you feel great when you get them sorted out."***

> ***Darren: "Well, they get you involved and are a good way of learning, but I can think of better things to be doing."***
>
> ***Ahmed: "I know a lot about Java and I enjoy showing the others how to solve the problems."***
>
> ***Paul: "I would say overall yes. At first more so than at the end.***
>
> ***Interviewer: "Why was that?"***
>
> ***Paul: "At the end I think maybe I got bored with it. Like, it starts off at the start and you're interested. Half way through you think it's the best thing that ever happened and by the end you're just frustrated with it."***
>
> ***Interviewer: "What's the cause of that frustration?"***
>
> ***Paul: "I don't know, maybe just doing the same thing again and again."***

Some students found the PBL journal useful and were proud of their work. For example, when asked about the journal Sarah said:

> ***Interviewer: "Do you think keeping a PBL journal is useful?"***
>
> ***Sarah: "Yeah, you can use it to study the answers and it shows you how much you have learnt. Yeah, the journal is a good idea."***

The PBL questionnaire results show that only two students did not consider Software Development to be important within the frame of their studies. This view that knowing how to program is important is reflected in most of the answers given during the interviews that contain references to the Software Development course being a 'rite of passage' that had to be undertaken in order to become a true computer scientist. The literature suggests that when confronted with difficult Threshold Concepts learners view them as a rite of passage (Meyer & Land, 2005;

Turner, 1995; van Gennep, 2004). The following experts from student interviews illustrate the importance most interviewees attach to Software Development.

> ***Interviewer: "Do you consider Software Development to be an important subject?"***
>
> ***Nichole: "Yeah, it is a very difficult subject, but if you can get through it, you have made it, because it is the subject people fail."***
>
> ***Sarah: "Yes of course. If you can't program you can't do anything. You need to know how to program to be able to handle other subjects. […] Once you know how to program then you feel like you have made it."***
>
> ***Darren: "Yes, so many people fail it, it's scary but you just have to get through it. You'll never get a job if you can't program."***
>
> ***Paul: "Yeah"***
>
> ***Ahmed: "Yes, the most important subject. If you can't program you'll never make much money. Some of the others don't realize that, that's why they fail. It is easy once you work at it. You have to do all the lab problems and study the notes. You can't learn it from a book, you just have to do the labs."***
>
> ***William: "It is if you want to go down that road of doing games and all that and if you don't it's not of use at all but it wouldn't be of use to me now because I don't want anything to do with it. I hate it."***

When asked how much time they spent studying Software Development outside of class contact hours, most of the students said that they had to go to work at their part-time jobs, and that this left little or no time for study. This was the case for all subjects, not just Software Development.

The interviews show that most students see value in doing individual work. The following excerpts demonstrate the general opinion:

> ***Interviewer: "What advice would you give to other students who are having problems with Java programming?"***
>
> ***Darren: "I'd tell them that they need to work on their own."***
>
> ***Sarah: "They need to sit down and think about what they're doing, try the problems for themselves."***
>
> ***Paul: (Laughs). "Ask for help."***
>
> ***William: "Listen from the very beginning and take out loads of books, and study. And take out books and all. And ask loads of questions as well."***
>
> ***Ahmed: "Sit down and work it out on paper first and so you really understand what is happening."***

5.4.5. Categories and Themes Identified

An attempt was made to identify categories and themes from the work. Open coding was undertaken which involved scrutinising, line by line, the interview transcripts, PBL questionnaires, PBL journals and field notes of observations. This process resulted in the identification of five categories of issues of relevance to the PBL staff and students, as follows:

- Learner engagement with problem solving;
- The difficulty of learning to program;
- Managing intra-group relationships;
- Managing tutor-student relationships;
- The troublesome nature of PBL for staff and students.

Axial coding was then undertaken in an attempt to make connections between the categories by examining the data in context and examining causal relationships/conditions. This process points to “expressions of learner behaviour in the PBL classroom” as the main theme.

5.5. Summary

The findings from the experiment outlined in Chapter 4 as well as the findings from the questionnaires, database logs and qualitative study were analysed using a concurrent triangulation strategy as outlined in Chapter 3.5, and Chapter 6 provides a detailed discussion of these findings.

Chapter 6 - Discussion and Conclusions

“[T]he answer to the question ‘Does PBL work?’ is: it depends.”

(Richardson, 2005, p. 51).

6.1. Introduction

This chapter discusses the quantitative and qualitative findings presented in chapters 4 and 5, linking them to the work of other researchers in the field and to the literature on PBL, motivation and Computing. It looks at the implications of the research for educational theory and practice, and examines what future research could be undertaken in the area to improve educational outcomes for learners.

6.2. Discussion of Findings

This section discusses the findings around whether using a Problem-Based Learning approach instead of conventional lectures improves outcomes for learners. The main outcomes focused upon are learner attainment, learner motivation, learner Software Development self-efficacy; learners’ approaches to studying and learner preferences for different types of teaching. In addition, a discussion of whether PBL classes can improve first year learners’ acquisition of Threshold Concepts in Computing is undertaken. Questions relating to what parts of the Computing curriculum are most suitable for PBL, what types of learners are most suited to learning through PBL and at what stage in their college lifetime they should undertake PBL are also discussed.

Discussion of the findings has been postponed until this chapter, as delaying the interpretation of the experimental results, the qualitative findings and the data mining allows the results of each part to be cross referenced against each other and provides a fuller picture.

6.2.1. Attainment

The quantitative analysis of the data from the four cohorts shows in relation to knowledge acquisition a small negative difference in the performance of students taught using the PBL approach over those not using PBL, but this difference is not significant, with an effect size of (ES = -0.07). However, an examination of the effect of PBL on skills shows a significant increase and a larger effect size (ES = 0.40). The effect sizes reported here in relation to knowledge and skills are in line with those reported in Dochy *et al.* (2003) in their meta analysis on the effects of PBL. The finding that the PBL groups had a knowledge deficit compared to the non-PBL groups is supported by other studies (Albanese & Mitchell, 1993; Baca *et al.*, 1990; Eisenstaedt *et al.*, 1990). However, the emphasis placed on problem elaboration in PBL has been shown promote the recall of declarative knowledge (Gagné, 1978; Schmidt, 1990; Wittrock, 1989), and mastering declarative knowledge has been shown to be central to an understanding of programming (Brooks, 1990; Détienne & Soloway, 1990; Guindon, 1990; Rist, 1990; Robertson & Yu, 1990; Visser & Hoc, 1990). As Dochy *et al.* say "[a]lthough the students in PBL would have slightly less knowledge…, their knowledge has been elaborated more and consequently they have better recall of that knowledge." (2003, p. 543). Albanese and Mitchell (1993) also support this view.

The causes of the observed deterioration in the positive effects of PBL on knowledge in the second, third, and fourth years, and skills in the second and third years of its implementation are unknown. One possible contributing factor is that when PBL was first introduced in the Computing Department, staff were particularly enthusiastic and devoted a great deal of effort to its organisation and delivery. In the following years the enthusiasm lessened, mainly due to the high workload involved in supporting the PBL classes. Marsh (1987) reports that this is a common occurrence when PBL is introduced. Also, it should be noted that course entry points decreased over the time period analysed in this study and students in the earlier years had a higher ability level (gauged by Leaving Certificate points or equivalent). However, this applied to both groups equally, and was taken into account in the linear model used in the comparison of PBL with historical non-PBL attainment data.

A point worthy of note is that the continuous assessment results of the non-PBL students improved considerably over the first three years under scrutiny, a phenomenon which merits further investigation. However, the continuous assessment results of the PBL group remained better than those of the non-PBL group throughout. It is also interesting to note that dropouts were spread evenly between both the PBL and non-PBL group, a finding which is contrary to what

Newman (2004a, p. 151) observed in his meta-analysis of PBL, where dropout rates were much higher in the PBL groups. However, at Anon College all students receive a grant, the value of which is based on their rate of attendance, and this monetary incentive may serve to reduce dropout rates.

This study found support for the hypothesis that using a PBL approach in the teaching of first year Software Development will improve students' performance in continuous assessment that tests skills, but not in final exams that test knowledge. This is perhaps because the PBL group spent more time 'on task' working on Software Development problems. Another likely reason for the differences in programming skill levels as measured by continuous assessment grades is that in the non-PBL group the learning effort was mainly focused on programming strategies based on code syntax and a trial and error attempt to develop a correct program schema, while in the PBL group the learning effort was mainly focused on developing programming strategies based on a correct program schema, and not on code syntax. This view finds support in the literature: for example, creating correct schemas has been shown to be a central element in program design (Davies, 1993; Ormerod, 1990). Brooks (1990) points out that the programming strategies that novices use strongly impact on the quality of final program that is produced, and Winslow in his review of studies (1996) states that:

> [A] large number of studies conclude[d] that novice programmers know the syntax and semantics of individual statements, but they do not know how to combine these features into valid programs (Winslow, 1996, p. 17).

This view that novices' main difficulties lie with schema composition problems and not programming language construct-based problems is supported by many other researchers (du Boulay, 1989; Linn & Dalbey, 1989; Perkins *et al.*, 1989; Robins *et al.*, 2003; Soloway & Spohrer, 1989). Thus would appear that using a PBL approach focusing on producing a correct program schema rather than on code syntax provides better training for novice programmers.

6.2.2. Motivation

Some evidence was found that the hybrid PBL model brought about a slight improvement in learners' relative autonomy with an overall effect size of (ES =

0.23). Nonetheless, given that the results are not statistically significant, it cannot be said that learners who complete the PBL course will have a higher degree of intrinsic motivation than those in the control group. This was an unexpected result in view of the research that suggests that the PBL teaching method promotes perceived autonomy and self-determination (Butler, 1999; De Volder *et al.*, 1986; van Grinsven & Tillema, 2006), which in turn can have a positive effect on students' motivation (Deci & Ryan, 1985; Hidi & Harackiewicz, 2000). Furthermore, studies on the introduction of PBL in medicine, accountancy and managerial education show positive changes in student attitudes and motivation compared to non-PBL students (Bernstein *et al.*, 1995; Bridges & Hallinger, 1991; Pincus, 1995; Schmidt *et al.*, 1992). However, one major difference between those studies and the study at Anon College is that the participants in the former were high attainment learners. This suggests a need for more research into the possible motivational benefits of PBL for low attainment learners which will be discussed in section 6.2.9. In addition, given that research has shown low levels of intrinsic motivation and high levels of extrinsic motivation to be attributes of learners on programming courses (Mamone, 1992), research is needed to examine if learners on certain Computing courses are less intrinsically motivated than learners on high status courses like medicine.

6.2.3. Software Development Self-Efficacy

Evidence was found that the hybrid PBL model brought about a significant improvement in learners' programming self-efficacy with an overall effect size of (ES = 1.70). Therefore it can be said that learners who complete the PBL course will have a higher degree of programming self-efficacy than those in the control group. This result was expected given the research that shows a link between programming self-efficacy and PBL (Bergin & Reilly, 2005; Dunlap, 2005), and programming self-efficacy and improved performance in skills (Wiedenbeck *et al.*, 2004). To explain this finding, it might be the case that the specific instructional strategies used in PBL, namely the use of authentic problems of practice, collaboration and reflection, increase student engagement and are therefore the catalysts for students' improved self-efficacy (Hendry, Frommer & Walker, 1999).

The effect size in this study was larger than that reported by Bergin and Reilly (2005) in a study at an Irish university on the role of comfort-level (including programming self-efficacy) on a first-year object-oriented Java programming module taught using a Problem-Based Learning approach. This divergence in findings might be partially explained by the difference in prior attainment of the participants. Given the low prior attainment of learners in the study at Anon College, it is possible that they had greater scope for improvement in programming self-efficacy.

As exam attainment results were similar for both PBL and non-PBL groups, improved exam attainment grades can be ruled out as an explanation for the increased learner self-efficacy on the PBL module. However, the improvements in continuous assessment results, which are given out to students during the semester, may have a role to play in the increase in learner programming self-efficacy.

6.2.4. Approaches to Studying

When compared against the non-PBL group there was evidence that the hybrid PBL model led to an improvement in learners' meaning orientation, with an overall effect size of (ES = 0.35) on deep approaches to learning, and a reduction in reproduction orientation with an effect size of (-0.75) on surface apathetic approach. A small negative effect was also seen on the strategic approach, with an effect size of (ES = -0.41). From these findings it can be said that learners in the PBL group will show higher scores on meaning orientation and lower scores on reproduction orientation than those in the control group. This result was expected and is in line with the results of studies of paramedical and medical students (Newble & Clarke, 1986; Sadlo, 1997). It supports the claim by Sadlo and Richardson (2003, p. 267) that "students who are taking programs with a problem-based curriculum appear to have approaches to studying that are more desirable than those of students taking programs with a subject-based curriculum in the sense that they are more compatible with the stated aims of most programs of study in higher education". However, Groves (2005) disagrees that PBL curricula foster a deep approach to learning, and suggests that other factors such as workload may be greater determinants of learning approach than curriculum type. Taken together, these findings emphasise the context-dependent nature of learning approaches as well as the importance of assessment as a driver of student learning.

Although the experimental findings show a significant effect on approaches to studying, one issue which may require further investigation is that 89 out of 106 students in the PBL group stated that the questions included on past exams to a large extent determined what they would study. This suggests that, while less so than the control group, the PBL learners are still extrinsically motivated by performance in exams, and have a strategic or surface-learning approach to learning. However, as only the PBL group was asked this question, their responses cannot be compared against those of the non-PBL group.

As stated earlier, there was no statistically significant difference between the PBL and non-PBL teaching approaches in terms of exam attainment marks. This finding is in line with the research which shows that adopting a deep learning approach alone may not be the most optimal for achieving high grades (Barron & Harackiewicz, 2001; Bouffard *et al.*, 1998; Elliot & Church, 1997; Elliot & McGregor, 2001; Harackiewicz *et al.*, 1997; Harackiewicz *et al.*, 2000; Skaalvik, 1997; Wolters *et al.*, 1996).

6.2.5. Preferences for Different Types of Teaching

Evidence was also found that the hybrid PBL model led to an increase in learners' preference for Supporting Understanding approaches to teaching with an overall effect size of (ES = 0.36) and a reduction in learners' preference for Transforming Information approaches to teaching with an effect size of (-0.63). These results suggest that learners in the PBL group will show a greater preference for courses and teaching that support deep learning (as opposed to surface learning) than those in the control group. These findings are in line with results from other studies that show evidence that PBL enhances students' approaches to learning and improves their perception of the quality of their course (Sadlo, 1997; Sadlo & Richardson, 2003).

6.2.6. Acquisition of Threshold Concepts in Computing

Threshold Concepts were examined in section 2.3 as a framework that may help explain why learners find computer programming so troublesome. A review of the literature identified two aspects of programming that may constitute Threshold Concepts in Computing: object-orientation and levels of abstraction (Eckerdal *et al.*, 2006). These concepts are certainly concepts that students find troublesome to master (Eckerdal & Thuné, 2005; Fleury, 2001; Or-Bach & Lavy, 2004; Ragonis & Ben-Ari, 2002; Rehder *et al.*, 1995). Misconceptions of object-oriented concepts and abstraction can be hard to shift later, and such misconceptions can act as barriers through which all later teaching on the subject may be inadvertently filtered and distorted (Hoc & Nguyen-Xuan, 1990; Holland *et al.*, 1997). There is evidence that the concepts of object-orientation and abstraction are transformative (Eckerdal, 2004) "requir[ing] nothing less than a complete change of world view" (Luker, 1994, p. 58). From the PBL questionnaires, observations and interviews it was clear that students saw their Software Development course as a rite of passage that had to be undertaken in order to become a Computing professional. The literature suggests that this can be a view that learners take of difficult Threshold Concepts (Meyer & Land, 2005; Turner, 1995; van Gennep, 2004).

Student responses to the PBL questionnaires show that a majority of students thought that the learning issues generated in the group tutorials were the most important starting point for their learning activities and that the problems used in the PBL classes illustrated Software Development concepts. The same set of programming problems was used in both the PBL and non-PBL groups. These problems aim to help students master basic programming constructs, object orientation and the use of different levels of abstraction in Software Development. The analysis of the PBL journals shows that students worked on the problems of abstraction and object orientation in a detailed, thorough and thoughtful way, with a focus on schema development. In contrast, interviews with staff suggest that the non-PBL group working on the same set of problems remained stuck dealing with syntax issues, rather than mastering the Threshold Concepts of abstraction and object orientation.

This finding suggests that the PBL method may be better than conventional lectures and tutorials at helping students to master Threshold Concepts in Computing. Coupled with the evidence that PBL has improved outcomes on programming courses, including the skills element of the course at Anon College, this indicates that PBL may be a good instructional choice for the teaching of programming. This in turn suggests that the use of PBL to teach novice learners may help to increase student retention. Such a view is supported by the literature which postulates that constructivist approaches, and PBL in particular, can help learners overcome the disjunction caused by Threshold Concepts (Savin-Baden, 2000). This may be because Computer Programming requires learners to master complex conceptual knowledge, and any misunderstandings at the conceptual level will directly affect learners' skill levels (Bonar & Soloway, 1985; Clancy, 2004). The PBL tutorials cause students to engage with and attempt to solve complex programming problems earlier in the curriculum than the traditional approach (Savin-Baden, 2006). Therefore any misconceptions students have are confronted immediately they attempt to solve the PBL problems. Tutors can then identify the misconceptions and intervene to correct them, thus preventing the misconception from becoming ingrained. Learning problems are identified and dealt with earlier than in the traditional approach. Tutors can help this process by selecting PBL tutorial problems that focus learners' efforts on concepts that they find troublesome. In addition, the group work aspect of PBL may be beneficial because it allows learners to articulate the underlying concepts they are trying to master.

6.2.7. The Computing Curriculum and Other Subjects

Although Mayer (2004) argues that discovery learning techniques have failed in the teaching of computer programming, this study suggests that PBL may be more effective than traditional methods in producing improved outcomes on programming courses. Its effectiveness is that it teaches strategies rather than concentrating on syntax, which in turn may render it effective in helping master the Threshold Concepts of abstraction and object-orientation. In addition, PBL may be a more effective teaching methodology on other courses on the wider Computing curriculum where learning outcomes require similar skills to programming courses. Examples are subjects such as networking, databases, systems analysis and software engineering that require students to apply the same complex design and diagnostic skills as are needed in Software Development. This view is supported by reports in the literature of the successful application of PBL to network design (Linge & Parsons, 2006), databases (Connolly & Begg, 2006), systems analysis (Bentley, Lowry & Sandy, 1999) and software engineering (Kay *et al.*, 2000). Outside of the Computing discipline there are many other higher education subjects where improving learners design and diagnostics skills would lead to better outcomes, such as in Civil, Electrical, and Mechanical Engineering.

6.2.8. At What Stage to Apply PBL

Kirschner, Sweller, and Clark (2006) suggest that PBL is less effective and efficient for novices than guided instructional approaches because it is in conflict with the architecture commonly used by cognitive load theory (Sweller & Sweller, 2006). However, this study found positive effects on first year learners and discovered that first year students were open to trying the PBL method. This suggests that PBL should be introduced at the beginning of a degree course, when students are more open to new ideas rather than towards the end. This view is supported by a study of the introduction of PBL in the final year of an Electronic Engineering degree, where Mitchell *et al.* (2005) found that the students had difficulty changing and were very uncertain when faced with anything that required initiative. This suggests that the introduction of PBL is a profound change to teaching and learning and that, if changes so profound are to succeed, they must be based on evaluated experience and good theory.

6.2.9. Types of Learners

A number of studies have shown PBL to be effective in improving some learning outcomes for higher attainment learners in higher education, mainly in the fields of business and medical related studies (Bernstein *et al.*, 1995; Bridges & Hallinger, 1991; Pincus, 1995; Schmidt *et al.*, 1992). This study has shown PBL to have beneficial effects for a group of learners, many of whom would be classed as low attainment in a higher education context. However, this study has also highlighted that many weaker students dislike the group work associated with PBL. Nonetheless, a number of studies have shown that working collaboratively on programming problems in groups is beneficial for weaker students (Mayer, 1989; van Gorp & Grissom, 2001; Wills *et al.*, 1999). These findings suggest that PBL can bring about desirable changes in learning outcomes on courses attended by learners with either low or high attainment. Nonetheless, given that the effect size on motivation measured in this study was not statistically significant, it is possible that the positive effects of PBL on the motivation of high attainment learners observed in other studies (Blumberg & Michael, 1992; Norman & Schmidt, 1992; Shin *et al.*, 1993) may not hold for low attainment learners. More research is needed to determine if this is true and if so, why.

6.2.10. Discussion of Other Findings

Feedback from interviews with learners suggests that the PBL model used in Anon College may provide a good transition for students to a third-level environment by helping them get to know the other students in their class. It also facilitates students in developing peer group support networks that help to remove the feelings of isolation commonly experienced by first-year students. This view is supported by the results of a study on the introduction of PBL in Computer Science in another Irish college (O'Kelly, 2005). However, due to the lack of a comparative study, it is impossible to determine whether the PBL approach is more successful than other student induction programmes using different methods (Edward, 2001).

Nearly all the students felt the classes helped them make friends, and 66 out of 106 said they would like to take another PBL module. 68 out of 106 students considered PBL to be an effective way of learning for themselves. 56 out of 106 students felt that the PBL classes stimulated their interest in Software Development. However, some students said during interview that they did not like the course structure and some students said that they did not feel that they had actively participated in the PBL sessions. A group size of 7-8 may be too large and allow some members to avoid working on the problems. Tutors need to be aware of these difficulties and provide independent work in the laboratories and closely monitor the division of work within PBL groups. Successful tutorials seemed to rely most crucially on balanced discussions between the students and careful preparation for the session.

Learner responses on the PBL questionnaire and learner and staff interviews showed that the use of virtual learning environments such as Moodle was found to be helpful to students and useful in managing the students' learning. This result was as expected and is supported by the literature (O'Neill, Singh & O'Donoghue, 2004).

There were observable differences in the rate of attendance at the PBL and non-PBL classes, with the PBL groups having a higher attendance. This was evident both from observation and from an analysis of Learner Class Attendance logs. There was also a striking difference in how students spent their classroom time. The non-PBL group spent much longer ‘logged-on’ to their computers, while the PBL groups spent more time developing solution plans and schemas on paper. This observation was supported by an analysis of network activity logs.

Students’ in-class behaviour showed a striking difference between groups. A large proportion of the non-PBL groups’ time was spent ‘off task’ while the PBL group was much more focused on Software Development problem solving. This raises issues about how staff manage activities in the non-PBL laboratories. The non-PBL group enjoyed their time in the computer lab although they got little useful Software Development work done. It was also noted that the PBL groups asked for less direct assistance from tutors than the non-PBL groups. Tutors in the non-PBL groups spent a large amount of time answering basic programming syntax questions and were asked few questions about alternative solution schemas. In the PBL groups this was completely reversed, with the focus on schema construction. This finding is in line with studies of PBL that show it enhances learners’ ability to analyse and solve problems (Duch *et al.*, 2001; Hmelo-Silver, 2004; Torp & Sage, 2002).

Observations of the PBL labs showed that they were in general active learning environments where there was a dynamic interplay of questioning, explanation, argumentation, design of programme schemas, communication of ideas and findings, and collaboration. However, questionnaire responses showed that students did not spend much time outside of class revising software topics or problems. This suggests that students do not reflect on their learning activities outside of class time. The lack of teamwork observed in the non-PBL group and the high level of teamwork observed in the PBL group is supported by the body of research showing that PBL students tend to prefer cooperative learning and teamwork (Bernstein *et al.*, 1995). However, the point needs to be made that staff discourage teamwork in the non-PBL laboratories because of fears of students colluding on individual assignments. The observations of the PBL labs as active learning environments support the finding that learners in the PBL group will show higher scores on meaning orientation and lower scores on reproduction orientation than those in the control group, and that they will also show a greater preference for courses and teaching that support deep learning (as opposed to surface learning) than those in the non-PBL group. However, the observation that the PBL learners did little further work outside class time shows that these effects are limited.

The analysis of the VLE logs showed that the PBL Group spent less time accessing course material than the non-PBL group. This difference may be due not to any less engagement with the Software Development course but rather to the fact that the PBL group undertook their problem-solving as a group without course notes, away from the PC, while in contrast the non-PBL group worked at their PC with course notes when working on problems. This view is supported by the observation that the PBL students carefully recorded their learning in their PBL journals.

The non-PBL groups showed little inter group tension, with students chatting and laughing about their social activities. The PBL groups displayed some limited intra group tension and arguments, and a number of students felt that the group climate did not facilitate the learning process. This point was raised particularly by the female students. Given the very low numbers of females in the classes it is possible that they felt isolated and uncomfortable in a mainly male atmosphere. However, in interviews, both female participants stated that this was not the case. The issues they raised were that some male members of the group did not contribute to the problem-solving and that the females did not like engaging in arguments about group activities. Other studies have also identified issues of an unfair distribution of work in PBL groups (Kinnunen & Malmi, 2005; Woods, Hall, Eyles, Hrymak & Duncan-Hewitt, 1996), and strategies need to be identified to address this problem.

PBL groups that worked efficiently had focused discussions about programming problems: their conversations did not lapse into irrelevant topics. Another aspect of efficient groups was that members gave each other positive encouragement and this fostered an increase in positive contributions from group members. In some inefficient groups members made ill-mannered comments to each other and this caused a decrease in positive contributions from group members. Similar findings have been found in other studies of group interactions (Postmes, Tanis & de Wit, 2001; Wheelan & Williams, 2003). Inefficient PBL groups also had members who were very dominant due to their previous knowledge or their personality. Tutors need to be aware of this problem and can help other students to cope with dominating students in constructive ways. Other studies have also identified the problem of dominant group members and they provide guidance for tutors in addressing this problem (Benbow & McMahon, 2001; Woods, 1996).

PBL group tensions have also been noted by Kinnunen & Malmi (2005) who conducted a study of PBL in an introductory programming course in Finland. In their findings they stress the need for tutor intervention to prevent conflict between PBL group members. One possible cause of the PBL group tensions is that students find PBL troublesome (Savin-Baden, 2000), possibly because they can become worried that their strategies are wrong (Finucane *et al.*, 1998), leading to a wish for more tutor intervention and demands for a more didactic approach from teachers (Newman, 2004b, p. 131). This view is supported in this study by the fact that a large number of PBL students stated that the PBL tutor needed to steer the group more strongly. Nonetheless, both staff and students said PBL helped to remove barriers between staff and students, and this has been shown to lead to a better learning environment (Blight, 1995). The study at Anon College highlights the fact that the learner/learner and learner/tutor relationships need careful monitoring due to their importance in influencing students' learning and performance. Tutors need to be aware that some learners who find the programming problems difficult may seek solutions from tutors and peers without understanding the key concepts that underlie the solutions, and in these cases there is a danger that these learners may plagiarise the work of others (Irons & Alexander, 2004).

Staff believed that PBL helps to develop students' verbal and written communications skills and their ability to work in teams. However, a majority of students did not feel that PBL helped their communication skills. Students and staff alike believed that PBL helps to develop students' critical thinking skills, but this opinion was not given in comparison to the traditional teaching method. Whether these beliefs are true remains unproven as this study found no difference in knowledge between the PBL and non-PBL groups. Nonetheless, as seen from the continuous assessment results, the PBL group were able to apply their knowledge better than the non-PBL group.

It should be noted that in this study learner participants had very homogeneous backgrounds. However, other than previous programming experience, students' backgrounds were found not to be a factor in their success, with no demographic factor or personality trait a strong indicator of success. This is in line with the findings of other studies (Bishop-Clark, 1995; Evans & Simkin, 1989).

6.3. Implications for Instructional Practice and for Educational Theory and Research

The magnitude of the effects on skills, programming self-efficacy, approaches to learning and preferences for different types of teaching identified in the study at Anon College implies that the findings are of both theoretical and practical importance.

There is evidence of high failure and dropout rates and low retention rates in introductory programming courses at tertiary level (Bennedsen & Caspersen, 2007), particularly among first year students (Jackson, 2003). Computer Science courses have the highest university dropout rates in the UK (Williams, 2007). Given that almost all students starting Computing degrees are motivated to succeed (Jenkins, 2001, 2002), it is important to examine why outcomes are so poor. Learning to program is a difficult task (Dijkstra, 1989; du Boulay, 1989; Jenkins, 2002) and achieving mastery can take a long time, about 10 years of constant effort (Winslow, 1996). Therefore it is hardly surprising, as Connolly *et al.* (2008) point out, that for many Computing students, learning programming is intimidating.

Many multi-national, multi-institutional studies conclude that the average first year student does not make much progress on an introductory programming course (Fincher *et al.*, 2005; Kurland *et al.*, 1989; Linn & Dalbey, 1989; McCracken *et al.*, 2001; Soloway *et al.*, 1982; Winslow, 1996). These studies show that there is clearly room for improvement in the way students are taught programming. Jenkins (2002, p. 53) makes the point strongly, saying that "[a]t the moment the way in which programming is taught and learned is fundamentally broken". Given that novices make limited progress in introductory programming courses, the literature calls for introductory courses that are realistic in their expectations (Robins *et al.*, 2003; Winslow, 1996). Some researchers have suggested redesigning introductory programming courses specifically to improve students experiences and to improve retention (Mahmoud *et al.*, 2004). How might the design and delivery of novice computer programming courses be improved? The literature shows that teaching standards clearly influence the outcomes of courses that teach programming (Linn & Dalbey, 1989) and that the main problem for novices is program design and planning (Winslow, 1996).

The programming strategies that they employ appear to account for the distinction between effective and ineffective novices (Brooks, 1990; Robins *et al.*, 2003). However, most introductory programming courses are conventionally structured with lectures and practical laboratory work; they concentrate on teaching programming knowledge but not on the strategies needed to use this knowledge. This may be due to the fact that, as Robins *et al.* (2003, p. 157) state, "strategies themselves cannot (in most cases) be deduced from the final form of the program". PBL may be able to help in this regard, and this study has shown that students' PBL journals contain finished example programs which are rich sources of information that can be presented, analysed and discussed. However, as Robins *et al.* (2003, p. 157) add, "[t]he strategies that created those programs, however, are much harder to make explicit." Soloway & Spohrer (1989, p. 412) suggest that "students are not given sufficient instruction in how to 'put the pieces together.' [There is a need to focus] explicitly on specific strategies for carrying out the coordination and integration of the goals and plans that underlie program code". Again PBL can help in this case: this study has shown that PBL students focus explicitly on strategies that 'put the pieces together'.

PBL can help to bring about other desirable changes. Sadlo (1997) conducted a study which suggested that the quality of learning tends to improve with the extent to which a problem-based approach has been implemented by an institution (Sadlo & Richardson, 2003). These results suggest that, as Richardson (2005, p. 45) puts it, "the use of PBL *can* bring about desirable changes in students' approaches to studying" and more generally suggests:

> that changes in the design and delivery of particular courses affect how students tackle those courses, and in particular that desirable

> approaches to studying could be promoted by appropriate course design, teaching methods and modes of assessment
>
> (Sadlo & Richardson, 2003, p. 254).

This general view is supported by other studies (Deek *et al.*, 1998; Gibbs, 1992; Kay *et al.*, 2000).

The results of the present study point towards an improved learning environment and the increased adoption of a deep approach to learning. This is perhaps partly due to the creation of a learning space where mastery goals are promoted. However, the introduction of some group-based work on non-PBL courses could possibly of itself promote mastery over performance learning goals.

Alexander and Murphy (2000, p. 44) make the point that as issues of motivation become part of instructional practice there will be "questions about the way in which instructional practice may need to be formed or transformed to energize these positive motivation forces." For example, "will teachers view these [motivational] constructs as unalterable traits that only serve to sort and categorize learners or to rationalize their current educational progress or the lack thereof? Or will these teachers see these constructs as motivational dimensions that are susceptible to instructional intervention?" And if so, what instructional strategies are most likely to bring about optimal motivation? For instance, "should teachers specifically aim their efforts at altering a particular motivation construct (e.g., self-efficacy or individual interest) or the conditions that might give rise to it (e.g., academic success or domain-specific knowledge)? With regard to these various constructs, what configuration of achievement motivations should be expected in highly successful students and how should these profiles transform over the course of students' educational careers?" (*ibid*, p. 44). Further research is needed to provide answers to these questions and to identify how PBL may need to be modified to create additional constructive motivation forces.

Studies (Lieberman & Remedios, 2007) have shown that as learners progress through college, they become more concerned with grades and are substantially less likely to want to master their subjects than first year learners. This raises questions as to whether pressure for grades undermines course enjoyment and, if so, what could be done to counteract this effect. It is possible that the adoption of teaching methodologies such as PBL that lead to learners adopting deeper approaches to learning might lead to improvements in students' motivation that would counteract the negative effects of exams on motivation.

The area of PBL implementation costs is not addressed in this study. However, the literature shows that the implementation costs and staff workload of PBL are directly related to class size (Finucane *et al.*, 1998), and PBL may not be economically viable with more than 100 students (Albanese & Mitchell, 1993). The study at Anon College highlights the need for a high level of academic management support for PBL and for the provision of additional teaching resources. Several of the staff involved in the delivery of the PBL module reported a greatly increased work load, even higher than the 30% increase suggested by Des Marchais (1993). If PBL were to be rolled out across all courses in Anon College, then significant industrial relations issues would need to be addressed. In fact, PBL in Computing will not be sustainable in the long term without the provision of additional resources.

The transition from lecturer to PBL facilitator may be stressful for staff (Berkson, 1993). It involves an increased workload, requires some prior PBL training, and necessitates management and peer support. There is also a major difficulty in finding suitable PBL problems in Software Development (O'Kelly *et al.*, 2004), as problems need to be carefully selected and based on clear programming principles (Kurland *et al.*, 1989). The creation of a facilitator/tutor support network and a database of suitable PBL problems would go some way towards solving these problems (O'Kelly, 2005).

The findings of this study give rise to some practical recommendations to improve the workings of PBL groups. Rules need to be put in place that encourage all students to actively engage with the problems. For example, the marking schemes for PBL programming problems should require students to be active participants before they receive marks. The PBL problems given out at the start of the course should include problem-solving tasks, as even at an early stage in their course students are capable of a conceptual analysis of the problem domain, and many can even sketch out a draft solution before they know many programming concepts or related syntax.

This focuses learner effort on problem-solving rather than code syntax. The composition of the PBL group should be as homogeneous as possible with all members having the same level of previous knowledge and skills, due to the observation that when some group members have more programming knowledge than others, weaker group members are inclined to ask them to provide the 'correct' answers rather than attempting to work out the solutions for themselves together with the group. There is a danger that these weaker students may become passive members of the group (Kinnunen & Malmi, 2005).

6.4. Suggestions for Further Research

Given that PBL has been implemented in many diverse contexts, in different disciplines, and at different stages of learning, there is a need, as Richardson (2005, p. 51) points out, "for an authoritative classification of the different ways PBL has been implemented".

The finding from the Anon College study that PBL led to a significant improvement in learners' Software Development skills, but had no effect on learners' knowledge as measured by exams, suggests that the type of knowledge tested in exams is not what ought to be tested on a practical-based course like Software Development: indeed it may be the case that Software Development should be tested through continuous assessment alone. On the other hand, if the knowledge tested by exams is essential knowledge, this suggests that research is needed to see how PBL can be adapted to help ensure better knowledge acquisition.

The study at Anon College took learners' mean prior attainment into account when measuring the effect of PBL on attainment. Given that individual learners had a wide range of prior attainment, further analysis could answer questions such as whether PBL was more successful with the higher prior attainment learners than with the lower prior attainment learners or vice versa. Taking this approach further, it would be interesting to examine the nature of the measured effects of PBL on attainment, self-efficacy, approaches to studying and preference for types of teaching at the individual level. The observed effects are non-linear, that is to say, they are mediated differently by the attributes of individual students. Therefore one could, for example, seek answers to questions such as what characteristics of learners would improve the effect of PBL on self-efficacy and would the same characteristic mediate differently other outcomes, like attainment or approaches to study, and if it did, then the exact relationship between the characteristic and the outcome could be studied.

In the course of informal conversations and interviews, some staff at Anon College suggested that PBL had fostered in students not just a greater interest in Software Development in first year but also in Computing in general and that this effect lasted into the following years of their study. Further research is needed to test this hypothesis. A longitudinal study that followed individual students through all four years of their study could answer questions about the impact of using PBL in year one on learners' longer term academic motivation and whether using PBL beyond year one could help to address the reduced course enjoyment and grade attainment pressures in later academic years.

The study at Anon College highlighted issues for PBL group composition, particularly in relation to gender, ethnicity, and dominance by some group members. Cases were identified where group members were uncomfortable with the group interactions and where there were tussles for leadership and dominance, and cases where female group members were uncomfortable with the group interactions and perceived male dominance of groups. Group members also reported tensions between different ethnic minority groups within PBL groups. However, it should be noted that in this study there were very small numbers of female or ethnic minority participants. While the study of gender and ethnicity issues within PBL groups was not the focus of this study, the findings suggest the need for further research in this area.

Many learners on the PBL course were frustrated and found the method troublesome, with some even stating that they thought the tutors were "lazy" or "didn't know how to solve the problems". Learners were challenged by having to try and solve the problems "on their own" and without the tutor providing them with the "right answer". This suggests the need for research to find better ways of dealing with the kinds of anxiety, self-doubt and frustration that learning can evoke and to provide better support for learners both in managing the transition from traditional approaches to a PBL-based approach and in facilitating better communication between tutors and learners.

It was stated by some learners in interviews that there was a difference between the roles of tutors outlined during induction at the start of the PBL course and the realities of their practice as tutors. Students wished for more control over their learning. This element of control is referred to in many of the definitions of student-centred learning (Boud & Feletti, 1998) and suggests that there is a role in further research in examining how the power relationships between staff and learners in the PBL learning process should best be developed.

Dysfunctional PBL groups and tensions between members have been noted in a number of studies (De Grave, Dolmans & van der Vleuten, 2001; Hendry, Ryan & Harris, 2003; Hitchcock & Anderson, 1997). The role of the tutor is critical in addressing the problem of dysfunctional PBL groups. Tutors who are too directive or too passive hinder the learning process (De Grave *et al.*, 2001; Hendry *et al.*, 2003), and research has shown that a tutor's performance is partly situation-specific and partly dependent on contextual circumstances (Schmidt, 1994). More research is needed to identify the role of the PBL tutor in different contextual situations and to provide guidance on how tutors might best guide dysfunctional PBL groups.

Further research is also needed to determine the causes of the tensions between students and tutors, between staff, and between students involved in the PBL groups. A possible reason may be that students' conceptions of learning and their conceptions of themselves as learners are a key factor in any attempt to implement Problem-Based Learning effectively (Savin-Baden, 2000). Claims are made for PBL that it promotes improvements in students' conceptions of learning to a greater extent than traditional curricula (*ibid*). However, Richardson (2005, p. 49) states that "it can also be argued that PBL actually presupposes more sophisticated conceptions of learning on the part of the students, and this might explain why some students have difficulty adapting to PBL". He goes on to point out that "PBL presupposes a student-centred and learning-orientated conception of teaching on the part of the teacher [and t]his might explain why some teachers have difficulty adapting to PBL or accepting it as an approach" (*ibid*, p. 54). This hypothesis suggests the need for further research into “whether students with a reproductive conception of learning and teachers with a teacher-centred conception of teaching have difficulty adapting to problem-based curricula, and which are the key characteristics determining the effectiveness of problem-based curricula” (*ibid*, p. 42).

Further exploration of the relationship between Threshold Concepts and learning difficulties is needed along with further research to develop a systematic method of identifying Threshold Concepts. Research is also needed to identify the full set of Threshold Concepts in Software Development and the wider Computer Science curriculum and to identify and develop innovative teaching strategies that help students better master Threshold Concepts in all disciplines.

Rountree & Rountree (2009, p. 142) suggest that “examining the different ways in which practitioners in Computer Science, Information Systems, Mathematics, Physics, Electrical Engineering and Linguistics tackle similar problems may produce excellent candidates for Threshold Concepts in each discipline, and opens up a research question concerning whether Threshold Concepts are shared between disciplines (and thus whether there is a hierarchy of Threshold Concepts), and whether Threshold Concepts mutate as they cross between disciplines”.

Research is also needed into how software applications such as messaging programs, virtual learning environments, Web 2.0, and social networks can be better integrated into the classroom. The computerised activity logs of such applications can provide a rich alternative source of data about learners. Research packages and tools need to be developed to allow researchers to access and analyse this data with ease, and more research is needed into how these data sources could be mined for a greater insight into learner behaviour. In harnessing these sources, privacy and ethical issues will need to be addressed to protect learners and teaching staff from unwarranted surveillance.

Davies (1993, p. 238) contends that research is needed to determine “the relationship between the development of structured representations of programming knowledge and the adoption of specific forms of strategy” and he identifies as significant strategies relating to the general problem domain, the specific programming task, the programming language and the programming tools used.

Von Mayrhauser and Vans (1995a) identify a number of open research questions in the area of program comprehension and generation that relate to the scalability of existing experimental results due to the small programs used, and the validity and credibility of results which are based on experimental procedures.

The study at Anon College supports the call by many researchers for the use of simple, specially-designed programming languages for teaching such as Logo (du Boulay, 1989; Jenkins, 2002). However, the pressure on programming course designers to ensure compliance with the norms of the software industry means that the vast majority of courses use standard workplace languages such as Java or C++ (Robins *et al.*, 2003). More research is needed into the possible benefits or disadvantages of teaching detailed reusable object-oriented program schemas called design patterns, particularly teaching special pedagogical patterns, using pattern languages (Sharp *et al.*, 2003; The Pedagogical Patterns Project, 2001), that allow students to adapt simpler known strategies to new and more complex problems (Proulx, 2000; Reed, 1998).

It seems that object-oriented programming might be particularly difficult for novices. Some researchers, including Wiedenbeck *et al.* (1999), suggest the use of visualisation tools to aid comprehension (Baecker, 1998; Cooper *et al.*, 2003). However, more research is needed to identify the pedagogical requirements so that these tools can be applied effectively in different teaching contexts (Gomez-Albarran, 2005; Rößling & Naps, 2002; Smith & Escott, 2006).

Most of the research findings reported in the literature relate to mainstream learners, often in the field of psychology, while the motivational requirements and the impact of learning environments on the motivation of other learners, such as those with disabilities or special educational needs, are not focused upon. Another factor that was apparent from the literary review done for this study was that the vast majority of the research reviewed was conducted by American, British or Australian researchers studying American, British or Australian students. Furthermore, almost all of the literature represents a Western philosophical orientation. This raises questions as to whether the conclusions and implications that educators draw from the literature can be applied across a broader cultural population. Such questions can only be answered through programmes of cross-cultural motivation studies.

6.5. Limitations of the Study

As far as this author can ascertain, this research is the first of its kind in Ireland which focuses on investigating the effectiveness of a PBL strategy for first-year students with low prior attainment status in an Irish college. Due to the fact that these efforts constitute the first research in response to the needs of students of this level, there are some limitations of this study that must be taken into account before reaching any generalisations.

First, the learners in this study were mainly low attainment learners and the findings may not be more generally applicable to contexts involving high attainment learners. Second, the groups are not totally statistically independent, as Computing students mix freely between groups and with engineering students outside of class time. Third, most of the learners in the Anon College study are grant-aided in that they are paid for attending classes. This may skew attendance and retention rates and lessen the general applicability of the findings. Fourth, some of the findings in this study are based on learner responses on self-report questionnaires. However, a number of steps were taken, as outlined in chapter 4, to ensure validity. Fifth, learner participants in this study were very homogeneous: there was a small number of female and ethnic minority participants, and the needs of students with disabilities and special educational needs were not focused upon. Finally, it should be noted that the sample frame used in this study, Anon College, constituted an opportunity sample, and that the finding cannot therefore be safely generalized to higher education as a whole. In addition this study was undertaken in only one tertiary level college. A multi-national, multi-institutional study would provide more generalisable findings and overcome some of the possible shortcomings of using an opportunity sample.

6.6. Conclusion

Although it cannot be said that learners who complete the PBL course will have a higher degree of intrinsic motivation than those in the control group, the comparisons between groups provide support for the hypotheses that first year Software Development students taught using a PBL approach will: have a higher degree of programming self-efficacy than those in the control group; show higher scores on meaning orientation and lower scores on reproduction orientation than those in the control group; show a greater preference for courses and teaching that support deep learning (as opposed to surface learning) than those in the control group; and perform better in continuous assessment that test skills but not in final exams that test knowledge.

The improvement in skills is perhaps because in the non-PBL group the learning effort was mainly focused on programming strategies focused on code syntax and a trial and error attempt to develop a correct programme schema, while in the PBL group the learning effort was mainly focused on developing programming strategies based on a correct programme schema, and not on code syntax.

The study at Anon College provides evidence that the PBL model assists students in problem abstraction, problem definition and problem refinement. Interviews with staff suggest that the non-PBL group working on the same set of problems remained stuck dealing with syntax issues, rather than mastering the concepts of abstraction and object orientation. Thus it is likely that the students taught using the PBL method will develop greater mastery of the concepts of object orientation and abstraction. This suggests that the PBL method is better at helping students master Threshold Concepts in Computing, which in turn suggests that the use of PBL to teach novice learners may help to improve student retention. And better student retention is the ultimate aim of the introduction of PBL in the first year Software Development course at Anon College.

Appendices

Appendix A - Ethics

A.1. Ethics Form Durham 2006

FORM EC2

(revised December 2004)

UNIVERSITY OF DURHAM

ETHICS ADVISORY COMMITTEE

APPLICATION FORM FOR RESEARCH ETHICS APPROVAL

OF WORK WITH HUMAN PARTICIPANTS

Introduction:

All University work with human volunteers must be assessed for ethics approval, whether it is in teaching, undergraduate or taught postgraduate project work or research. Applications should normally be submitted two months before the intended project start date.

Normally, Departmental Ethics Committees consider applications from undergraduates and taught postgraduates and from academic staff for teaching projects. Ethics approval for research projects, by research postgraduates or staff must be sought from either the University Ethics Advisory Committee or an NHS Local or Multi-Centre Research Ethics Committee.

Here, and in any country where it is intended to undertake work involving patients, tissue sampling, invasive procedures, or any clinical trial, full prior permission **must** be sought and obtained from the NHS Research Ethics Committee (www.corec.org.uk) or its authorised equivalent . Certain work with babies and children must also be referred to an NHS Local or Multi-Centre Research Ethics Committee. The researcher or academic supervisor must check as early as possible with the Insurance Officer, Claire Robinson (Claire.robinson@durham.ac.uk) that full insurance cover is in place and should forward to the Committee's Secretary a copy of the application form to (and later decision letter from) the NHS' MREC or LREC or equivalent.

Please use this form for research work and project work. Both need signed approval from the Head of your Department/School and, where established, the Chairman of your Department/School's Ethics Committee, before submission to the Ethics Advisory Committee.

You should also enclose a copy of the consent form you will be asking participants to sign and the information sheet (written in layperson's language) you will give to participants, and - where applicable - parents and teachers. (The term "participant" is used to cover any volunteers involved in the project, with the exclusion of the researcher and his/her supervisor.) An example consent form is included at the end of this form, and this should be followed as closely as possible.

You are recommended to provide participants with a separate information sheet, rather than combining the information sheet and consent form into one, in order that participants can take the information sheet away with them.

Please send the signed EC2 application form to the Secretary of the Ethics Advisory Committee (Katrina Tomlin, School of Education, telephone: ext 48402, e-mail: k.m.tomlin@durham.ac.uk*).*

Returned applications must be either typed or word-processed. It would assist members if you could also forward your form to the Secretary as an e-mail attachment - it is understood that this additional copy would be unsigned.

NB Please consult with the Research and Economic Development Support Services and the Home Office Website before planning any work involving animals.

SECTION A **INVESTIGATOR**:

1. NAME, QUALIFICATIONS, POST HELD STUDENT (course) /ACADEMIC STAFF/OTHER:

James Doody, M.Sc., Student (Ed.D.) / Lecturer

2. E-MAIL ADDRESS, DEPARTMENT, CONTACT ADDRESS and CONTACT TELEPHONE NUMBER

3. PRINCIPAL INVESTIGATOR

James Doody

4, PRINCIPAL INVESTIGATOR'S E-MAIL ADDRESS, DEPARTMENT, CONTACT ADDRESS and CONTACT TELEPHONE NUMBER

5. RESEARCH SUPERVISOR OR ACADEMIC-IN-CHARGE (TEACHING):

Prof. Peter Tymms

6. RESEARCH SUPERVISOR/ACADEMIC-IN-CHARGE’S E-MAIL ADDRESS, UNIVERSITY DEPARTMENT, CONTACT ADDRESS AND TELEPHONE NUMBER

p.b.tymms@durham.ac.uk Director CEM in the School of Education, +44 (0) 191 33 48413

7. LIST ALL CO-WORKERS, THEIR: STATUS, EMPLOYER (AND DEPARTMENT), AND RESEARCH EXPERIENCE:

None

8. INTENDED LOCATION/S FOR THE STUDY

Anon College, Ireland.

9. CONSENT: Please give details of any other consents applied for and/or obtained from: NHS Local Research Committees in this country, or their equivalent overseas, for medical/clinical projects etc., and attach copies of any relevant application forms submitted and decision letter/s received.

N/A

SECTION B **DESCRIPTION OF WORK**

10. TITLE OF PROJECT:

An evaluation of the effectiveness of using a Problem-Based Learning approach in the teaching of the Java programming language to 1st year third level students.

11. PROPOSED ROUTES OF PUBLICATION (for students, this may be by dissertation; for staff: an indication of the type of publication envisaged)

EdD. Assignment, Thesis, possible conference paper.

12. ABSTRACT:

Abstract

This study will evaluate the effectiveness of using a Problem-Based Learning (PBL) approach in the teaching of the Java programming language to 1st year third level students. Effectiveness will be measured solely by quantifying any change in students' attainment marks attributable to using PBL in the module. Other – qualitative - non-attainment improvements possibly attributable to PBL, such as any improvement in the learning environment and students' enjoyment of the subject, will be measured by means of questionnaires and the results presented, but a detailed analysis of these factors will be outside the scope of this study.

Detailed overview

PBL has been introduced (as outlined below) in the teaching of the Software Development module to 1st year Computing students at ITT Dublin for the 2005/06 academic year.

First year full-time Computing students at ITT Dublin are randomly split into two groups for Software Development (Java programming). Software Development is taught over two semesters. This year, due to resource issues, in Semester 1 Group A were taught using a PBL approach, while Group B were taught using a traditional approach (lectures and tutorials). Both groups were taught by the same lecturer and used the same computer equipment and laboratory space. At the time of writing, Semester 1 has finished and Semester 2 has yet to start. In Semester 2, the Groups will be switched over, i.e. Group B will be taught using a PBL approach, while Group A will revert to the traditional approach.

13. AIMS and OBJECTIVES: Please state the Research Question, including, where appropriate, the hypothesis to be tested.

Research question:

Is problem based learning (PBL) appropriate in the teaching of object-oriented programming languages (java) to first year college students?

Hypothesis: Using a PBL approach will improve students' performance in both final exam and continuous assessment.

14. EXPERT INDEPENDENT REVIEW: Please state who has conducted an expert independent review of your proposed project, and his/her verdict. (For a student, this will be your research supervisor; for staff, the review may be by another member of your department.)

Prof. Peter Tymms

Questionnaires will be used to obtain qualitative data (students' attitudes towards Problem Based Learning) Quantitative data (examination and assessment results) will be used to analyse the value added by using PBL. All data will be analysed using the SPSS statistical package, version 12.0.1 for windows.

The following statistical tests will be carried out:

- Group A's attainment results (for both semesters) from both final exam and continuous assessments will be compared against Groups B's results.
- Both groups' (A & B) attainment results (for both semesters) from both final exam and continuous assessments will be compared to the students' course entry points (achieved in the Irish Leaving Certificate or equivalent).
- The attainment results for the whole of 2005/06 will also be compared against historical attainment data for the Software development module, to identify any historical trend in the underlying data.
- First year engineering students will be used as a control group. They also take the Software Development module but do not use PBL. The Engineering students have very similar course entry points to the Computing students (achieved in the Irish Leaving Certificate or equivalent).

From these comparisons we should be able to determine whether there is a statistically significant relationship between the benefit (or otherwise) of using PBL and a student's ability (as measured by Irish Leaving Certificate points or equivalent).

This will allow us to test the following Hypothesis:

That using a PBL approach in the teaching of first year Software development will improve students' performance in both final exam and continuous assessment.

16. IS THIS PROJECT TO BE A DECEPTION STUDY? NO

(**If the response is YES, then please contact REDSS for advice and guidelines on how to proceed.)**

17. PARTICIPANTS:

(a) Who are they (eg students, colleagues,…)? Anon College Students.

(b) If students: course, year, size of groups, % of students involved

Bachelor Degree of Science (Computing) - Year 1 – approx 80 students - 100% involved, and Bachelor Degree of Engineering - Year 1 – approx 40 students. 100%

(Names of students may be required subsequently)

(c) How many participants are to be recruited? Approx 120

(d) Selection (eg age, sex)? Age range between 18-21, 80% male, 20% female

(e) How are the participants to be recruited?

Anon College, first year students taking the module Software Development

(f) Is there any link with the investigator (supervisor, tutor, etc.)? No

(g) Are any participants likely to be pregnant, or would pregnant women be excluded?

Possibly, and no.

(h) How are the participants to be involved in the study?

Their attainment statistics (their software development exam and assignment results) will be examined and they will be asked to complete questionnaires.

18. TESTS – QUESTIONNAIRES/OTHER

Questionnaires made up of about 16 questions

19. ARE SUBSTANCES TO BE GIVEN TO PARTICIPANTS? NO

If YES - complete Appendix A

20. ARE SAMPLES TO BE TAKEN FROM PARTICIPANTS? NO

If YES - complete Appendix A

21. ARE OTHER PROCEDURES TO BE APPLIED i.e. A QUESTIONNAIRE OR OTHER TOOL? YES / NO

If YES - complete Appendix A, including a copy of your questionnaire.

22. DETAILS OF DRUGS AND MATERIALS TO BE USED *(name of compound and dosage where appropriate - full details to be given in* ***Appendix (A)*** with details of NHS LREC/equivalent consent sought and obtained*)*

None

23. CONTROLS (needed?). If so, how many, who are they, how recruited/selected?

Bachelor Degree of Engineering, year 1 students will be used as a control group.

24. RISKS AND HAZARDS

Has a full risk assessment been carried out? **YES/NO**

Further details: Health and Safety Office at http://www.dur.ac.uk/healthandsafety/NewManualIndex.htm

What risks to participants are present?	PROBABILITY	SERIOUSNESS
None		

State precautions to minimise each risk

N/A

25. DEGREE OF STRESS EXPECTED

None

26. DISCOMFORT, INCONVENIENCE OR DANGER

What discomfort, danger or interference with normal activities will be suffered by the participant?

None

State precautions to minimise them:

N/A

27. STATE SPECIAL ARRANGEMENTS FOR INDEMNIFICATION IN THE EVENT OF INJURY AND NONE-NEGLIGENT HARM TO THE PARTICIPANTS

None

28. BENEFIT

Please state what benefit to society or individuals should arise from the work:

An improvement in the teaching methodology used to teach 1st year computing students an object oriented programming language, and if PBL is shown to enhance attainment it is assumed that it is beneficial to students and it can be extended to the teaching of other subjects and other courses.

29. STATISTICS

Has statistical advice been sought on study design?

YES X NO NOT APPLICABLE

If YES, from whom? If NO, give reasons

Prof. Tymms

30. SAMPLE SIZE

Please describe the statistical/other rationale for the sample size/number of participants to be used in this study and how the study size will yield meaningful research results.

I won’t be taking a sample; I will be using data from the whole population of 1st year computing students. For the qualitative questionnaires I will be asking all the students to complete it, but cannot guarantee that this will happen.

31. CONSENT

(a) Who will explain the investigation to the participant?

The investigator, James Doody.

(b) Will written explanation be given to the participant as a summary of the project written in layman's language? Please attach a copy to your form, or advise on why one is not to be used. Where schoolchildren/minors are involved, there should also be an information sheet directed at the teachers, parents/guardians.

Yes.

(c) Will written consent be obtained? This is the normal expectation, therefore if your response is that you do not intend to obtain written consent, please explain in detail

Yes.

(d) How and where will consent be recorded? Where schoolchildren/minors/persons with a mental incapacity are involved, there should be full details of your procedures for ensuring informed written consent would be given before participation commences.

Consent will be recorded by the investigator and stored in ITT Dublin. All Participants will be over 18 years of age.

Please attach copies of any participant explanation leaflets and written consent forms (*it is advised that the University's consent form is used).*

32. CONFIDENTIALITY

(a) Please indicate what steps will be taken to safeguard the anonymity and confidentiality of the participant's records, and confirm that the requirements of the Data Protection Acts will be complied with.

Participants will be identified only by a sequence number. Individual students' attainment data will be kept in ITT Dublin in accordance with the Irish Data Protection Act. Individual students will not be identifiable from the results published.

(b) If you are intending to make tape recordings or video recordings of participants please answer the following questions:

(i) Will tape or video recordings and any written transcriptions from these be destroyed at the end of the project? YES / NO

(ii) If NO, what further use do you intend to make of the recordings and what arrangements will be made for their secure storage?

N/A

(iii) Will consent be requested for this future use? YES/NO

If your response is "no", please give reasons:

N/A

33. PROJECT DURATION

(i) When do you hope to commence the project?

March 2006

(ii) When will the project finish and how long will it take to complete?

August 2009

34. FOLLOW-UP ACTION

(i) Please confirm that at the project's conclusion, all participants who have contributed to the project will receive a summary written in layman's language of the project and its results

Yes

(ii) If your response to the above is "no", please provide an explanation.

35. FUNDING

Please state the source of funding for the work

No funding

36. OTHER

Are you, or a collaborator, proposing to undertake any other related work which might involve any species of animal?

No

SECTION C: NOTES

- Applications must normally be submitted **at least two months** before the expected start of the project.
- On receipt of this form, members of the Committee will normally be given at least 3 weeks to consider the application. At the end of the three weeks, members' queries will be forwarded to the applicant for a response. The Chairman will then require a further 7 days from receipt of the applicant's responses to determine the application (totalling at least 4 weeks per application).
- Major modifications in the course of the study should be resubmitted to the Ethics Advisory Committee for approval.
- You should submit a report at the close of the project on form EC3, available on the University's website, or on request from the REDSS Office.
- Adverse events of a serious or potentially serious nature should be notified directly to the University Health and Safety Adviser.

SIGNATURE OF INVESTIGATOR: DATE:

....

SIGNATURE OF SUPERVISOR/ACADEMIC TEAM LEADER: DATE:

....

DECLARATION BY HEAD OF SCHOOL OR DEPARTMENT:

I confirm that:

1. I have read and approved this application for consideration by the Ethics Advisory Committee and
2. The principal investigator and other key researchers have the necessary expertise and experience and have access to the resources needed to conduct the proposed research successfully and
3. The research proposal is worthwhile, of high scientific value and represents good value for money.

SIGNATURE OF HEAD OF DEPARTMENT/SCHOOL: DATE:

....

NAME IN BLOCK CAPITALS

WHEN COMPLETE, PLEASE RETURN THIS FORM TO THE SECRETARY

OF THE ETHICS ADVISORY COMMITTEE

Project Title:

Following approval by the Head of School/Department, this signed form, with attachments, accompanied by an electronic unsigned copy of the form and all attachments, should be forwarded to the Secretary of the Ethics Committee, Katrina Tomlin, School of Education, telephone: ext 48402, e-mail: k.m.tomlin@durham.ac.uk.

Approved / Not Approved by the Ethics Advisory Committee

.... Date: ....

PLEASE NOTE THAT THIS APPROVAL EXPIRES:

1. **WHERE THE PROJECT CONTINUES UNCHANGED, THREE YEARS AFTER THE DATE OF APPROVAL;**
2. **WHERE THERE IS ANY CHANGE TO THE PROJECT, FROM THE DATE OF THAT CHANGE;**
3. **WHERE THERE IS ANY CHANGE TO THE LEGISLATION/REGULATIONS AFFECTING THIS PROJECT, FROM THE DATE OF THAT CHANGE.**

FORM EC2 (cont/...)

University of Durham School of Education

Ethical Guidelines for Research[7]

The University of Durham School of Education believes that all educational research should be conducted within an ethic of respect for persons, knowledge, democratic values and quality of educational research.

Code of Conduct: Responsibility to students and participants

1. The *informed* consent of participants should always be gained prior to research and participants should, in particular, be informed about the aims, purposes and any likely consequences of the publication of findings.
2. Informants have a right to remain anonymous. Researchers are responsible for taking appropriate precautions to protect the confidentiality of both participants and data. Where data is kept on a computer the researchers will be responsible for ensuring that the requirements of the Data Protection Act are fulfilled.
3. In the case of interviews involving children below school leaving age, permission should be obtained from the school and if they so suggest, the parents.
4. When filming or recording, researchers should make it clear to research participants the purpose of the recording and to whom the recording will be communicated.
5. Researchers should not deceive or coerce their students into serving as research subjects or assistants. They should not represent a student's work as their own.
6. In planning, and in the conduct and reporting of research, researchers should act in ways that ensure that no participant is disadvantaged by their age, class, 'race', gender, sexual orientation, religious or political beliefs or disability.
7. Sexual and racial harassment are recognised as abuses of power. Researchers have a duty to refrain from, and actively oppose such behaviour. Researchers should not use the inequalities of power which characterise the tutor – student, researcher – respondent relationship to obtain personal, sexual, economic or professional advantages.
8. Where a tutor/supervisor enters into an intimate or sexual relationship with their student, the emotional involvement, whether reciprocal or otherwise, is liable to compromise evaluation and assessment. Particular dangers arise in post-graduate supervision where the relationship between student and supervisor is necessarily one-to-one, protracted and supportive. In any such cases it is the tutor/supervisors responsibility to ensure that an alternative tutor/supervisor is found for the student.

Responsibility to the research profession

Educational researchers should;

1. Avoid fabrication, falsification or misrepresentation of evidence, data, findings or conclusions.
2. In case study and evaluative research, actively seek and include data and evidence provided by all relevant stakeholders.
3. Report their findings to all relevant stakeholders and avoid selective communication of findings.

[7] We are grateful to the British Sociological Association and to the British Educational Research Association who have granted us permission to adapt and abridge their documents; Guidelines for Good Professional Conduct (1991) and Ethical Guidelines for Funded Research (1992) , in the compilation of these guidelines.

4. Report research conceptions, procedures, results, and analysis accurately and in sufficient detail for other researchers to understand and interpret them.
5. Never *knowingly*, omit reference to any relevant work by others.

Responsibility to research assistants/partners

1. All employees should be properly informed of the terms and conditions of their employment. Care should be taken not to underpay part-time staff or to use them or secretarial staff for duties for which they are not being paid.
2. All those involved in research should be aware of the intellectual property rights with respect to the data collected or to which they have access. The general principle of academic freedom means that freedom to analyse and publish the results of research should only be limited in exceptional circumstances.
3. Researchers should never present other people's work as their own, nor hold up the publication of work by others so that their own gets precedence.
4. Researchers should acknowledge fully all of those who contributed to their research and publications.
5. Attribution and ordering of authorship and acknowledgements should accurately reflect the contributions of all main participants in both research and writing processes, including students.
6. Material quoted verbatim from the writing of others must be clearly identified and referenced to the author.

Relationship with Funding Agencies

The University of Durham School of Education code of practice governs ethical principles and establishes appropriate standards of academic freedom, including the right to disseminate research findings. While this code should be observed within all research it is particularly important in respect to contract research. The code should be honoured by researchers in the negotiation of contractual arrangements put forward by funding agencies, and in the carrying out of these obligations once they have been agreed.

1. The aims and sponsorship of research should always be made explicit by researchers.
2. Researchers should not agree to conduct research that conflicts with academic freedom, nor any other principles included in these guidelines. They should not agree to any undue or questionable influence by government or other funding agency in the conduct, analysis or reporting of research.
3. Academic staff should not engage in contract research without agreement by the institution and the institution will not compel academic staff to engage in any particular contract research.

JSB10/97

Research: Ethics 25 02 00

A.2. Ethics Form Durham 2008

Durham University

School of Education

Research Ethics and Data Protection Monitoring Form

Research involving humans by all academic and related Staff and Students in the Department is subject to the standards set out in the Department Code of Practice on Research Ethics. The Sub-Committee will assess the research against the British Educational Research Association's *Revised Ethical Guidelines for Educational Research* (2004).

It is a requirement that prior to the commencement of all research that this form be completed and submitted to the Department's Research Ethics and Data Protection Sub-Committee. The Committee will be responsible for issuing certification that the research meets acceptable ethical standards and will, if necessary, require changes to the research methodology or reporting strategy.

A copy of the research proposal which details methods and reporting strategies must be attached and should be no longer than two typed A4 pages. In addition you should also attach any information and consent form (written in layperson's language) you plan to use. An example of a consent form is included at the end of the code of practice.

Please send the signed application form and proposal to the Secretary of the Ethics Advisory Committee (Sheena Smith, School of Education, tel. (0191) 334 8403, e-mail: Sheena.Smith@Durham.ac.uk). Returned applications must be either typed or word-processed and it would assist members if you could forward your form, once signed, to the Secretary as an e-mail attachment

Name: James Doody

Course: Ed.D.

Contact e-mail address:

Supervisor: Dr. Julie Rattray **Second Supervisor:** Professor Steve Higgins

Title of research project: A longitudinal evaluation of the impact of a Problem-based learning approach to the teaching of Software Development in higher education

Questionnaire

		YES	NO	
1.	Does your research involve living human subjects?	Yes		IF NOT, GO TO DECLARATION AT END
2.	Does your research involve only the analysis of large, secondary and anonymised datasets?		No	IF YES, GO TO DECLARATION AT END
3a	Will you give your informants a written summary of your research and its uses?	Yes		If NO, please provide further details and go to 3b
3b	Will you give your informants a verbal summary of your research and its uses?	Yes		If NO, please provide further details
3c	Will you ask your informants to sign a consent form?	Yes		If NO, please provide further details
4.	Does your research involve covert surveillance (for example, participant observation)?		No	If YES, please provide further details.
5a	Will your information *automatically* be anonymised in your research?		No	If NO, please provide further details and go to 5b
5b	IF NO Will you explicitly give *all* your informants the right to remain anonymous?	Yes		If NO, why not?
6.	Will monitoring devices be used openly and only with the permission of informants?	Yes		If NO, why not?
7.	Will your informants be provided with a summary of your research	Yes		If NO, why not?

	findings?			
8.	Will your research be available to informants and the general public without authorities restrictions placed by sponsoring authorities?	Yes		If NO, please provide further details
9.	Have you considered the implications of your research intervention on your informants?	Yes		Please provide full details
10.	Are there any other ethical issues arising from your research?		No	If YES, please provide further details.

Further details

Q4: There will be observations and notes taken of learners “on task” behaviours but no covert surveillance.

Q5: All information/data collected will be made anonymous by me, student numbers will be used and false names will be used in published transcripts etc.

Continuation sheet YES/NO (delete as applicable)

Declaration

I have read the Department’s Code of Practice on Research Ethics and believe that my research complies fully with its precepts. I will not deviate from the methodology or reporting strategy without further permission from the Department’s Research Ethics Committee.

Signed …………………………………….. Date: …………………………

SUBMISSIONS WITHOUT A COPY OF THE RESEARCH PROPOSAL WILL NOT BE CONSIDERED.

A.3. Ethics Forms Anon College 2007

Application for Ethical Clearance for a Research Project Involving Human Participants **RE_2 Form**

To be completed by staff proposing to submit an application to conduct research involving human participants and human biological samples. The signed original and an electronic copy of the completed form should be returned to the Secretary of the Research Ethics Board.

Research must not commence until written approval has been received from the Research Ethics Board.

Guidelines to applicants submitting applications for ethical clearance are given in the SOP entitled "Procedures for Submitting an Application for Ethics Clearance for Research Projects".

Please check that all supplementary information is attached to your application (in both hard and soft copy).

	ATTACHED		NOT APPLICABLE
Information on existing protocols/best practice to be followed in the proposed research	X		
Ethical Approval from Other Committees Form	X		
Bibliography/Reference Section	X		
Participant recruitment advertisement			X
Informed Consent form(s)	X		
Case report forms/diary cards/questionnaires to be used	X draft	□ final	
Interview Schedule	X draft	□ final	
Hazard Assessment Form			X
Use of Drug/Medical Device Additional Information Form			X
Use of Ionising Radiation Additional Information Form			X
Use of GMO Form			X
Curriculum Vitae of principal researchers(s) and collaborators indicating expertise in the research area proposed	X		

SECTION 1: APPLICANT DETAILS

1.1 General Information

PROJECT TITLE	An evaluation of the effectiveness of using a Problem-Based Learning approach in the teaching of the Java programming language to 1st year third level students.			
THIS PROJECT IS:	X	Staff Research Project		Consultancy Project

(tick as many as apply)		Contract Research Project		Clinical Trial
		Funded Research		Consultancy
		Student Research Project *NO*		Other

		Masters		Taught postgraduate
	X	PhD		Undergraduate

Project Start Date:	**May 2008**	**Project End Date:**	**Oct 2009**

1.2 Investigator Contact Details

PRINCIPAL INVESTIGATOR(S):

TITLE	*SURNAME*	*FIRST NAME*	*POSITION & ROLE IN RESEARCH*	*PHONE*	*FAX*	*EMAIL*
Mr	Doody	James	Investigator			

OTHER INVESTIGATORS:

TITLE	*SURNAME*	*FIRST NAME*	*POSITION & ROLE IN RESEARCH*	*PHONE*	*FAX*	*EMAIL*
Dr	Rattray	Julie	Supervisor			julie.rattray@durham.ac.uk
Prof.	Higgins	Steven	Supervisor			s.e.higgins@durham.ac.uk

DEPARTMENT	Computing
SCHOOL	Science & Computing
RESEARCH CENTRE	

WILL THE RESEARCH BE UNDERTAKEN ON-SITE AT THE INSTITUTE OF TECHNOLOGY TALLAGHT?

X	YES		NO	*(If NO, give details of off-campus location.)*

IS THIS PROTOCOL BEING SUBMITTED TO ANOTHER ETHICS COMMITTEE, OR HAS IT BEEN PREVIOUSLY SUBMITTED TO AN ETHICS COMMITTEE?*)*

X	YES		NO	*(If YES, please complete the **Ethical Approval from Other Committees Form** and provide letter of approval, detail on decision received etc.)*

SECTION 2: DETAILS OF RESEARCH STUDY

2.1 PROJECT OUTLINE - LAY DESCRIPTION

The Project aims to determine if Problem Based Learning (PBL) improves the teaching of Software Development to first year learners. Participants will be required to fill out questionnaires detailing their experiences of PBL in Software Development and some participants will be interviewed about their experience of PBL

2.2 AIMS OF AND JUSTIFICATION FOR THE RESEARCH

Aims and objectives of the research:

To discover if using a Problem-Based Learning approach instead of conventional lectures improves learners':

1. attainment in the subject;
2. motivation to learn the subject;
3. enjoyment of the subject.

2.3 PROPOSED METHOD

Participants' time commitment will be about 30 minutes for the questionnaires, and 30 for interviews.

The following data will be collected:

- Questionnaires and interviews will be used to obtain qualitative data on participants' attitudes towards Problem Based Learning.
- Participants' examination and assessment results will be used to help analyse the value added by using PBL.
- Participants' class attendance records will also be analysed for evidence of increased learn involvement.
- Field notes of participants' actions in Software Development labs will be made. These observations will be helpful in determining an increase in time spent "on task"

Data will be analysed using the SPSS and NVIVO statistical packages and a grounded theory approach will be employed for the analysis of the field notes and interview responses.

2.4 PARTICIPANT/SAMPLE PROFILE

Participants in this study will be drawn from approximately 60 first year students, 40 second and third year students, 14 lecturing staff and 2 postgraduates in the Department of Computing. At the start of the academic year, all students will be informed of the study and their participation will be solicited. No participants will be under 18 years of age.

Demographic details will of the first year students not be available until the students enrol in September 2008. However based on the 2007 learner intake, whose population profile should be similar, we would expect a male:female ratio of around 9:1, with all students speaking English as their first language, almost all of Irish nationality, all between 18 and 20 years of age, and the majority from areas of Dublin suffering from socioeconomic disadvantage. The research design is informed by the British Educational Research Association ethical guidelines (BERA, 2004): in particular, before any surveys, observations or interviews are completed, the participants will receive a consent form outlining the purpose of the research, guaranteeing their anonymity, and specifying that their participation/non-participation will not be discussed with their instructors or otherwise affect their standing in the college. Confidentiality will be respected by limiting access to the audio taped interviews, interview transcripts and field notes to the author and supervisor. All respondents to questionnaires and the tapes and transcripts of interviews will be filed using a code number only. All data collected will be documented, stored on computer hard drives, kept confidential and kept in a secure place. Audio tapes, computer files, transcripts, and field notes will be destroyed after successful completion of the thesis. In addition to protect participants' rights and welfare the study will be bound by the Data Protection (Amendment) Act (2003), which contains strict rules about how data must be stored, who can access it, and how it can be processed (Clark, 1996)

2.5 PLEASE EXPLAIN WHEN, HOW, WHERE, AND TO WHOM RESULTS WILL BE DISSEMINATED, INCLUDING WHETHER PARTICIPANTS WILL BE PROVIDED WITH ANY INFORMATION AS TO THE FINDINGS OR OUTCOMES OF THE PROJECT?

Results will be presented at a seminar in Anon College that all participants can attend. Results of the research will be disseminated to the wider education research community conference papers, journal articles, and an Ed.D. Thesis.

2.6 OTHER APPROVALS REQUIRED Has permission to gain access to another location, organisation etc. been obtained?

	YES		NO	X	NOT APPLICABLE

(If YES, please specify from whom and attach a copy of approval letter. If NO, please explain when this will be obtained.)

2.7 HAS A SIMILAR PROPOSAL BEEN PREVIOUSLY APPROVED BY THE RESEARCH ETHICS BOARD?

	YES	X	NO

(If YES, please state both the REC Application Number and Project Title)

<u>SECTION 3:</u> PARTICIPANT SELECTION

What are the primary location(s) for data collection? Classroom and computer laboratories**.**			
Please specify the types of subjects involved in this study and indicate the number of each type:			
Type of Subject:	***Number***	***Type of Samples:***	***Number***
▪ healthy subjects	80	*– specify:*	

▪ in-patients		a)	
▪ clinic attendees		b)	
▪ minors		c)	

3.2 TO BE COMPLETED WHERE THE RESEARCH INVOLVES HUMAN SUBJECTS

With regard to subjects to be involved in the research:

How will subjects be recruited for the study? Participants in this study will be drawn from approximately 60 first year students, 40 second and third year students, 14 lecturing staff and 2 postgraduates in the Department of Computing.

▪ Is written consent to be obtained? *If YES, you must also complete* ***Section 4***	Yes	
▪ Are subjects under the age of 18 to be included? *If YES, you must also complete* ***Section 5***		No
▪ Will any payments be made to subjects? If YES give details:		No
▪ Is any proportion of this payment being paid by a commercially sponsored organisation and if so by whom?		No
▪ Are there potential risks within the project, if any, for the investigator, subjects, samples, the environment and/or participants? *If YES, you must complete* ***Section 6***		No

▪ If controls are to be included please state how they are to be selected:

NB. Names of Student Subjects receiving payment in commercially sponsored research must be notified to the Research Ethics Committee (attach list)

Students who did not attend PBL classes, may be used as controls.

Specify the number of subjects to be used in this project, the selection criteria and the exclusion criteria to be used: Number:

List your exclusion/inclusion criteria for participant selection:

<u>Inclusion criteria:</u>

Taking the Software Development module

<u>Exclusion criteria:</u>

Not taking the Software Development module or under 18 years of age

Specify whether any of the following procedures are involved: (*Delete yes or no as necessary*)

a) *Any invasive procedures*		No
b) *Physical contact*		No
c) *Any procedure that may cause mental distress*		No

Is a product such as pharmaceutical or devices to be administered to the participant? NO

Information on the sampling procedures involved in your study:

▪ Are samples to be taken? *If YES, indicate:*		No
a) *Types of sample to be taken:*		

b) Frequency of samples:		
c) Amount of sample:		
d) Is this part of the person's normal treatment?	Yes	

SECTION 4: PARTICIPANT CONSENT I

Informed consent is required for all human subject participants in the proposed research.

4.1 Will informed consent be obtained from the research participants?

YES	NO
X	

If yes, please give details of **who** will take consent and **how** it will be done.

(Please attach a copy of letter, consent form (if required) and information leaflet. See guidelines on how to prepare these documents in the Appendix 2 associated with the Institute Ethics Procedures and adapt examples accordingly to suit your study and participants)

Consent forms and information sheets will be given to participants by the principle researcher. The official record of consent will be held in hard copy with an original signature. Also each participant giving an interview will be given a transcript of the interview and, provided with an opportunity of deleting any wording that they may perceive as identifying them. The Consent forms and information sheet attached.

4.2 What is the time interval between giving information and seeking consent?

7 days will be allowed between giving information and seeking consent

4.3 Will the participants be from any of the following groups? (Put an x in the appropriate box)

	YES	**NO**
Children under 18 years of age		**No**
Adults with learning disabilities, if YES, please specify: There may possibly be participants who have dyslexia or dyspraxia	**Yes**	
Adults with communication difficulties		**No**
Adults who are unconscious or very ill		**No**
Adults who have a terminal illness		**No**

Adults with mental illness		**No**
Adults suffering from dementia		**No**
Prisoners		**No**
Young Offenders in custodial care		**No**
Those who could have been considered to have a particularly dependent relationship with the investigator, e.g. those in care homes, students	**Yes**	
People engaged in illegal activities (e.g. drug taking; illegal internet behaviour etc.). If YES please specify group:		**No**
Other groups who may be considered vulnerable (Please specify below)		**No**

4.4 If participants are to be recruited from any of the potentially vulnerable groups listed above, please give details of:

(a) the extra steps taken to ensure that participants from any of these vulnerable groups are as fully informed as possible about the nature of their involvement:
All participants will be informed of the aims and objectives of the research and that their participation is entirely voluntary.

(b) who will give consent:

All participants involved.

(c) how consent will be obtained (e.g. will it be verbal, written or visually indicated?):
Written consent will be obtained

(d) When consent will be obtained:
At the start of the study.

(e) The arrangements that have been made to inform those responsible for the care of the research participants of their own involvement in research:

Questions 4.5 and 4.6 to be completed for research involving human participants in biological or clinical trial studies

SECTION 6: POTENTIAL RISKS & RISK MANAGEMENT

6.1 ARE THE RISKS TO SUBJECTS AND/OR RESEARCHERS ASSOCIATED WITH YOUR PROJECT GREATER THAN THOSE ENCOUNTERED IN EVERYDAY LIFE?

	YES	X	NO

6.2 DOES THE RESEARCH INVOLVE?	**YES**	**NO**
use of a questionnaire? (if YES please attach copy)	X	
interviews (if YES please attach interview questions)	X	
observation of participants without their knowledge		X
participant observation	X	
audio- or video-taping interviewees or events	X	
access to personal and/or confidential data (including student, patient or client data) without the participant's specific consent		X
administration of any stimuli, tasks, investigations or procedures which may be experienced by participants as physically or mentally painful, stressful or unpleasant during or after the research process		X
performance of any acts which might diminish the self-esteem of participants or cause them to experience embarrassment, regret or depression		X
investigation of participants or direct contact with anyone involved in illegal activities		X
procedures that involve deception of participants		X
administration of any substance or agent to participant (if YES, please attach a **Hazard Assessment Form [Anon College RE_5 Form]** and a **Use of Drug/Medical Device Form [Anon College RE_6]**)		X
the use of a medical device on or in a human participant (if YES, please attach a **Use of Drug/Medical Device Additional Information Form [Anon College RE_6]**)		X
the use of genetically modified organisms (if YES, please attach a **Use of GMO Form [Anon College RE_7]**)		X
the use of ionising radiation on a human participant (if YES, please attach a **Use of Ionising Radiation Additional Information Form [Anon College RE_8]**)		X
use of non-treatment placebo control conditions		X
collection of body tissues or fluid samples		X
collection and/or testing of DNA samples		X
participation in a clinical trial		X

6.3 POTENTIAL RISKS TO PARTICIPANTS AND RISK MANAGEMENT PROCEDURES

Identify, as far as possible, all potential risks to participants (physical, psychological, social, legal or economic etc.), associated with the proposed research. Please explain what risk management procedures will be put in place.

There are no risks for participants.

6.4 ARE THERE LIKELY TO BE ANY BENEFITS (DIRECT OR INDIRECT) TO PARTICIPANTS FROM THIS RESEARCH?

	YES	X	NO	*(If YES, provide details.)*

6.5 ARE THERE ANY SPECIFIC RISKS TO RESEARCHERS? *(e.g. risk of infection or where research is undertaken at an off-campus location)*

	YES	X	NO	*(If YES, please describe.)*

6.6 ADVERSE/UNEXPECTED OUTCOMES

Please describe what measures you have, or will put in place, in the event that there are any unexpected outcomes or adverse effects to participants arising from involvement in the project.

All participants will have access to the Institute's extensive student and staff support systems.

6.7 MONITORING

Please explain how you propose to monitor the conduct of the project (especially where several people are involved in recruiting or interviewing, and administering procedures) to ensure that it conforms with the procedures set out in this application. In the case of student projects please give details of how the supervisor(s) will monitor the conduct of the project.

As well as two external supervisors, the principle investigator will monitor project on a daily basis.

6.8 SUPPORT FOR PARTICIPANTS

Depending on risks to participants you may need to consider having additional support for participants during and/or after the study. Consider whether your project would require additional support, e.g., external counseling available to participants. Please advise what support will be available.

All participants will have access to the Institute's extensive student and staff support systems.

SECTION 7: FUNDING & PAYMENT OF PARTICIPANTS

7.1 OUTLINE SOURCES OF FUNDING FOR THE STUDY IF APPLICABLE AND HOW YOU WILL MANAGE ANY POSSIBLE CONFLICT BETWEEN THOSE FUNDING THE STUDY AND THE AIMS AND RESULTS OF THE STUDY IF APPLICABLE?

N/A

7.2 DO YOU PROPOSE TO PROVIDE INCENTIVES AND/OR EXPENSES TO PARTICIPANTS?

	YES	X	NO

7.3 WILL A PAYMENT BE MADE TO RESEARCH PARTICIPANTS? No

SECTION 8: CONFIDENTIALITY/ANONYMITY

8.1 WILL THE IDENTITY OF THE PARTICIPANTS BE PROTECTED?

X	YES		NO	*(If NO, please explain)*

If you have answered YES to question 8.1, then please answer questions 8.2 to 8.8:

8.2 What steps will you take to protect the confidentiality of the following, during and after the study?

Participant identities: All respondents to questionnaires and the tapes and transcripts of interviews will be filed using a code number only.

Data collected and patient/client records:

Hardcopy records: All data collected will be documented, stored on computer hard drives, kept confidential and kept in a secure place.

8.3 Is there any potential confidentiality issue through identification of the study location?

No

8.4 If your data is to be held on computer, how will it be protected?

The computer is in an access controlled office, and both the computer and software files will be password protected.

8.5 What other person(s) other than the researcher/team as listed will have access to the data collected and what steps will be done to protect confidentiality?

No other persons will have access to the data

8.6 The Institute Data Protection Policy recommends secure retention of data for 5 years. If there is any reason to apply for variation from these guidelines, please give details and justify:

No

8.7 If identifiable data or material will be retained after the study is completed, is it stated on an informed consent form that this will be done and that material will not be used in future unrelated studies without further specific permission being obtained?

YES	NO	If No, please explain Why
X		

8.8 If the study involves audio taping interviews, you must allow the participant access to the transcript, if they so wish. This must be included in an Informed Consent Form and Information Leaflet (if these forms are being used). Will the participant be given access to a transcript of the audio tape interview?

YES	NO	N/A	IF NO, PLEASE EXPLAIN WHY
X			

SECTION 9: DATA/SAMPLE STORAGE, SECURITY & DISPOSAL

For the purpose of this section, "Data" includes that in a raw or processed state (e.g. interview audiotape, transcript or analysis). "Samples" include body fluids or tissue samples.

9.1 HOW WILL THE DATA/SAMPLES BE STORED? *(The REB recommends that all data be stored on campus)*

Stored at Anon College X

Stored at another site

(Please explain where and for what purpose)

9.2 WHO WILL HAVE ACCESS TO DATA/SAMPLES?

Access by named researchers only X

Access by people other than named researcher(s)

(Please explain who and for what purpose)

Other

(Please explain who and for what purpose)

9.3 IF DATA/SAMPLES ARE TO BE DISPOSED OF, PLEASE EXPLAIN <u>HOW</u>, <u>WHEN</u> AND <u>BY WHOM</u> THIS WILL BE DONE?

The researcher will disposed of all data other than exam and assessment results at the end of the study. This will be done by shredding questionnaires, interview transcripts, notes etc. Any digital video or audio recordings will be deleted. Exam and assessment results will be kept in line with the Institutes data protection policy.

SECTION 10: QUALIFICATIONS, EXPERIENCE & SKILLS OF PROPOSED RESEARCHERS

List the academic qualifications and outline the experience and skills relevant to this project that the researchers and any supporting staff have in carrying out the research and in dealing with any emergencies, unexpected outcomes, or contingencies that may arise. **No more than 200 words.**

The principle researchers has the following educational and professional qualifications

Dublin City University

B.Sc. in Computer Applications.	Oct. 1984 - Jun. 1988
M.Sc. in Computer Applications.	Oct. 1989 - Jun. 1991

The researcher has over 16 years experience working as a lecturer, including supervising four Masters level 9 postgraduate students.

SECTION 11: DECLARATION BY INVESTIGATORS & APPROVAL SIGNATURES

PRINCIPAL INVESTIGATOR DECLARATION

The information contained herein is, to the best of my knowledge and belief, accurate. I have read and agree to comply with the Institute's Code of Conduct for Researchers and Process and Procedures for Seeking Ethics Clearance for Research Projects. I have attempted to identify all risks related to the proposed research that may arise in conducting this research and acknowledge my obligations to and the rights of the participants. I am not aware of any other ethical issue not addressed within this form.

I and my co-investigators have the appropriate qualifications, experience and facilities to conduct the research set out in the attached application and to deal with any emergencies and contingencies related to the research that may arise.

I/We agree to abide by the decision of the Research Ethics Board.

Name of Principal Investigator(s): ____________________________

BLOCK CAPITALS

____

Signature(s):

Date: ____________________________

HEAD OF SCHOOL/DEPARTMENT APPROVAL

The Head of School/Department **must countersign** and date the application below:

I approve this study to be carried out under the auspices of my School/Department:

Name of Head of School/Department: ____________________________

Signature: ____________________________

Date: ____________________________

Appendix B - Participants' Consent Forms

Information Sheet and Informed consent form8

Purpose of the Study: As part of the requirements for a Doctorate in Education at Durham University I have to carry out a research study. The study is concerned with evaluating the effectiveness of using a Problem-Based Learning (PBL) approach in the teaching of the Java programming language to 1st year third level students.

What will the study involve? The study will involve filling in a couple of questionnaires which should take about 30 minutes. You may also be asked to give an interview about your experience of PBL (these interviews maybe videotaped).

Why have you been asked to take part? You have been asked because you were taught java in 1st year using PBL.

Do you have to take part? No, participation is totally voluntary. By signing the consent form you agree to take part in the study and allow your data to be kept and used in the study, however if you wish you have the option of withdrawing before the study commences (even if you have agreed to participate) or discontinuing after data collection has started, and you can ask to have any data (questionnaire responses etc) destroyed.

Will your participation in the study be kept confidential? Yes. I will ensure that no clues to your identity appear in the thesis. Any extracts from what you say that are quoted in the thesis will be entirely anonymous. Group data (e.g. average, mean, mode etc.) will be referred to by class group, but no individual will be named.

What will happen to the information which you give? The data will be kept confidential for the duration of the study. On completion of the thesis, data will be retained for a further five years in a secure environment and then destroyed.

What will happen to the results? The results will be presented in the thesis. They will be seen by my supervisor, and the external examiner. The thesis may be published in a library and read by future students. The study may be published in an academic journal.

What are the possible disadvantages of taking part? I don't envisage any negative consequences for you in taking part. It is possible that talking about your experience in PBL may cause some distress, but this is highly unlikely.

[8] The following draws extensively on a document produced by Dr R. Swain of UCC, and is used with permission. Copyright is vested in same and all rights therein remain with Dr Swain.

What if there is a problem? At the end of the interview or after filling in the questionnaires, I will discuss with you how you found the experience and how you are feeling. If you subsequently feel distressed, you should contact student support services, such as the nurse or student counsellor

Who has reviewed this study? Both Anon College and Durham University Research Ethics Committees' have reviewed the study and approval was given.

Any further queries? If you need any further information, you can contact me:

If you agree to take part in the study, please sign the consent form overleaf.

INFORMED CONSENT FORM

Project title: An evaluation of the effectiveness of using a Problem-Based Learning approach in the teaching of the Java programming language to 1st year students.		
Principal Investigators: James Doody		
BACKGROUND: Participants' will have to fill in questionnaires and take part in interviews, all data will be kept will be entirely anonymous.		
Participant Declaration: I (i.e. the participant): *Tick yes or no as appropriate*		
Have read or have had the information sheet read to me and that I understand the contents.	Yes	No
Have been given an opportunity to ask questions and am satisfied with answers.	Yes	No
Consent to take part in the study.	Yes	No
Understand that participation is voluntary and that I can withdraw at any time.	Yes	No
Understand that withdrawal will not affect my access to services or legal rights.	Yes	No
Consent to possible publication of results.	Yes	No
I (the participant) give my permission to: Use the data obtained from me in other future studies without the need for additional consent.	Yes	No
Researcher Declaration: I James Doody have *Tick yes or no as appropriate*		
Have explained the study to the participant	Yes	No

Have answered questions put to me by the participant about the research	Yes	No
Believe that the participant understands and is freely giving consent	Yes	No

Participant’s Statement:

I have read, or had read to me, this consent form. I have had the opportunity to ask questions and all my questions have been answered to my satisfaction. I freely and voluntarily agree to be part of this research study, though without prejudice to my legal and ethical rights. I understand I may withdraw from the study at any time. I have received a copy of this consent form.

Participant’s Name: **Contact Details:**

Participant Signature: ***Date:***

Date:

Researcher’s Statement:

I have explained the nature and purpose of this research study, the procedures to be undertaken and any risks that may be involved. I have offered to answer any questions and fully answered such questions. I believe that the participant understands my explanation and has freely given informed consent.

Signature:

Date:

Appendix C - Self-Regulation Questionnaires

C.1. Original Self-Regulation Questionnaire

Learning Questionnaire

The following questions relate to your reasons for participating in the interviewing class. Different people have different reasons for participating in such a class, and we want to know how true each of these reasons is for you. There are three groups of items, and those in each group pertain to the sentence that begins that group. Please indicate how true each reason is for you using the following scale:

1	2	3	4	5	6	7
not at all			somewhat			very
true			true			true

A. I will participate actively in the organ systems classes:

1. Because I feel like it's a good way to improve my skills and my understanding of patients.

2. Because others would think badly of me if I didn't.

3. Because learning to interview well is an important part of becoming a doctor.

4. Because I would feel bad about myself if I didn’t study this approach.

B. I am likely to follow my instructor's suggestions for interviewing:

5. Because I would get a good grade if I do what he/she suggests.

6. Because I believe my instructor's suggestions will help me interview effectively.

7. Because I want others to think that I am a good interviewer.

8. Because it's easier to do what I'm told than to think about it.

9. Because it's important to me to do well at this.

10. Because I would probably feel guilty if I didn't comply with my instructor's suggestions.

C. The reason that I will continue to broaden my interviewing skills is:

11. Because it's exciting to try new ways to work interpersonally with my patients.

12. Because I would feel proud if I did continued to improve at interviewing.

13. Because it's a challenge to really understand what the patient is experiencing.

14. Because it's interesting to use the interview to try to identify what disease the patient has.

C.2. Self-Regulation Questionnaire for Computing Students

Learning Self-Regulation Questionnaire (SRQ-L)

The following questions relate to your reasons for participating in the Software Development (java programming) classes. Different people have different reasons for participating in such a class, and we want to know how true each of these reasons is for you. There are three groups of items, and those in each group pertain to the sentence that begins that group. Please indicate how true each reason is for you using the following scale:

1	2	3	4	5	6	7
not at all true		somewhat true			very true	

A. **I will participate actively in the software development classes:**

1. Because I feel like it's a good way to improve my skills and my understanding of software development.

1	2	3	4	5	6	7
not at all true		somewhat true			very true	

2. Because others would think badly of me if I didn't.

1	2	3	4	5	6	7
not at all true		somewhat true			very true	

3. Because learning to programme well is an important part of becoming a Computing professional.

1	2	3	4	5	6	7

not at all true		somewhat true			very true	

4. Because I would feel bad about myself if I didn't participate actively in the software development classes.

1	2	3	4	5	6	7
not at all true		somewhat true			very true	

B. **I am likely to follow my lecturer/instructor's suggestions for studying software development:**

5. Because I would get a good grade if I do what he/she suggests.

1	2	3	4	5	6	7
not at all true		somewhat true			very true	

6. Because I believe my instructor's suggestions will help me programme effectively.

1	2	3	4	5	6	7
not at all true		somewhat true			very true	

7. Because I want others to think that I am a good programmer.

1	2	3	4	5	6	7
not at all true		somewhat true			very true	

8. Because it's easier to do what I'm told than to think about it.

1	2	3	4	5	6	7
not at all true		somewhat true			very true	

9. Because it's important to me to do well at this.

1	2	3	4	5	6	7
not at all true		somewhat true			very true	

10. Because I would probably feel guilty if I didn't comply with my instructor's suggestions.

1	2	3	4	5	6	7
not at all true		somewhat true			very true	

C. **The reason that I will continue to broaden my software development skills is**:

11. Because it's exciting to try new ways to solve software development problems.

1	2	3	4	5	6	7
not at all true		somewhat true			very true	

12. Because I would feel proud if I did continued to improve at programming.

1	2	3	4	5	6	7
not at all true		somewhat true			very true	

13. Because it's a challenge to really understand how to solve programming problems.

1	2	3	4	5	6	7
not at all true		somewhat true			very true	

14. Because it's interesting to develop programmes to solve problems.

1	2	3	4	5	6	7
not at all true		somewhat true			very true	

Appendix D - Self-Efficacy Questionnaire for Computing Students

Student Number: ----------------------　　　　　　　　**Class Group 1B**

Programming Self-Efficacy Scale

Rate your confidence in doing the following programming related tasks using a scale of 1 (not at all confident) to 7 (absolutely confident). **If a specific term or task is totally unfamiliar to you, please mark 1.**

Not confident at all 1	Mostly not confident 2	Slightly confident 3	50/50 4	Fairly confident 5	Mostly confident 6	Absolutely confident 7

1. I could write syntactically correct statements. _______
2. I could understand the language structure of and the usage of the reserved words. _______
3. I could write logically correct blocks of code. _______
4. I could write a program that displays a greeting message. _______
5. I could write a program that computes the average of three numbers. _______
6. I could write a program that computes the average of any given number of numbers. _______
7. I could use built-in functions that are available in various program libraries. _______
8. I could build my own libraries. _______
9. I could write a small program given a small problem that is familiar to me. _______
10. I could write a reasonably sized program that can solve a problem this is only vaguely familiar to me. _______

11. I could write a long and complex program to solve any given problem as long as the specifications are clearly defined. _______

12. I could organize and design my program in a modular manner. _______

13. I could understand the object-oriented paradigm. _______

14. I could identify the objects in the problem domain and could declare, define, and use them. _______

15. I could make use of a pre-written function, given a clearly labelled declaration of the function. _______

16. I could make use of a class that is already defined, given a clearly labelled declaration of the class. _______

17. I could debug (correct all the errors) in a long and complex program that I had written and make it work. _______

18. I could comprehend a long, complex multi-file program. _______

19. I could complete a programming project if someone showed me how to solve the problem first. _______

20. I could complete a programming project if I had only the language reference manual for help. _______

21. I could complete a programming project if I could call someone for help if I got stuck. _______

22. I could complete a programming project once someone else helped me get started. _______

23. I could complete a programming project if I had a lot of time to complete the program. _______

24. I could complete a programming project if I had just the built-in help facility for assistance. _______

25. While working on a programming project, if I got stuck at a point I could find ways of overcoming the problem. _______

26. I could come up with a suitable strategy for a given programming project in a short time. _______

27. I could manage my time efficiently if I had a pressing deadline on a programming project. _______

28. I could mentally trace through the execution of a long, complex multi-file program given to me. _______

29. I could rewrite lengthy and confusing portions of code to be more readable and clear. _______

30. I could find a way to concentrate on my program, even when there were many distractions around me. _______

31. I could find ways of motivating myself to program, even if the problem area was of no interest to me. _______

32. I could add features to a program in a modular manner that adhered to the style of the given program. _______

33. I could write a program that someone else could comprehend and add features to at a later date. _______

Appendix E - Approaches and Study Skills Inventory for Students (ASSIST) Questionnaire for Computing Students

Approaches and Study Skills Inventory for Students

(Short version)

This questionnaire has been designed to allow you to describe, in a systematic way, how you go about learning and studying. The technique involves asking you a substantial number of questions which overlap to some extent to provide good overall coverage of different ways of studying. Most of the items are based on comments made by other students. Please respond truthfully, so that your answers will **accurately** describe your **actual** ways of studying, and work your way through the questionnaire quite **quickly.**

Background information

Name or Identifier .... Age .... years Sex M / F

B. Approaches to studying

The next part of this questionnaire asks you to indicate your relative agreement or disagreement with comments
about studying again made by other students. Please work through the comments, giving your **immediate** response. In deciding your answers, think in terms of **this particular lecture course.** It is also
very important that you answer **all** the questions: check you have.

***5** means agree (√) **4** = agree somewhat (√**?**) **2** = disagree somewhat (**x?**) **1** = disagree (**x**).*
*Try not to use **3** = unsure (**??**), unless you really have to, or if it cannot apply to you or your course.*

√ √? ?? x? x

1. I manage to find conditions for studying which allow me to get on with my work easily. 5 4 3 2 1
2. When working on an assignment, I'm keeping in mind how best to impress the marker. 5 4 3 2 1
3. Often I find myself wondering whether the work I am doing here is really worthwhile. 5 4 3 2 1
4. I usually set out to understand for myself the meaning of what we have to learn. 5 4 3 2 1
5. I organise my study time carefully to make the best use of it. 5 4 3 2 1
6. I find I have to concentrate on just memorising a good deal of what I have to learn. 5 4 3 2 1
7. I go over the work I've done carefully to check the reasoning and that it makes sense. 5 4 3 2 1
8. Often I feel I'm drowning in the sheer amount of material we're having to cope with. 5 4 3 2 1
9. I look at the evidence carefully and try to reach my own conclusion about what I'm studying. 5 4 3 2 1
10. It's important for me to feel that I'm doing as well as I really can on the courses here. 5 4 3 2 1
11. I try to relate ideas I come across to those in other topics or other courses whenever possible. 5 4 3 2 1
12. I tend to read very little beyond what is actually required to pass. 5 4 3 2 1
13. Regularly I find myself thinking about ideas from lectures when I'm doing other things. 5 4 3 2 1
14. I think I'm quite systematic and organised when it comes to revising for exams. 5 4 3 2 1
15. I look carefully at tutors' comments on course work to see how to get higher marks next time. 5 4 3 2 1
16. There's not much of the work here that I find interesting or relevant. 5 4 3 2 1
17. When I read an article or book, I try to find out for myself exactly what the author means. 5 4 3 2 1
18. I'm pretty good at getting down to work whenever I need to. 5 4 3 2 1
19. Much of what I'm studying makes little sense: it's like unrelated bits and pieces. 5 4 3 2 1
20. I think about what I want to get out of this course to keep my studying well focused. 5 4 3 2 1
21. When I'm working on a new topic, I try to see in my own mind how all the ideas fit together. 5 4 3 2 1

22 I often worry about whether I'll ever be able to cope with the work properly. 5 4 3 2 1

23. Often I find myself questioning things I hear in lectures or read in books. 5 4 3 2 1
24. I feel that I'm getting on well, and this helps me put more effort into the work. 5 4 3 2 1
25. I concentrate on learning just those bits of information I have to know to pass. 5 4 3 2 1
26. I find that studying academic topics can be quite exciting at times. 5 4 3 2 1
27. I'm good at following up some of the reading suggested by lecturers or tutors. 5 4 3 2 1

28. I keep in mind who is going to mark an assignment and what they're likely to be looking for. 5 4 3 2 1
29. When I look back, I sometimes wonder why I ever decided to come here. 5 4 3 2 1
30. When I am reading, I stop from time to time to reflect on what I am trying to learn from it. 5 4 3 2 1
√ √? ?? x? x
31. I work steadily through the term or semester, rather than leave it all until the last minute. 5 4 3 2 1
32. I'm not really sure what's important in lectures so I try to get down all I can. 5 4 3 2 1
33. Ideas in course books or articles often set me off on long chains of thought of my own. 5 4 3 2 1
34. Before starting work on an assignment or exam question, I think first how best to tackle it. 5 4 3 2 1
35. I often seem to panic if I get behind with my work. 5 4 3 2 1
36. When I read, I examine the details carefully to see how they fit in with what's being said. 5 4 3 2 1
37. I put a lot of effort into studying because I'm determined to do well. 5 4 3 2 1
38. I gear my studying closely to just what seems to be required for assignments and exams. 5 4 3 2 1
39. Some of the ideas I come across on the course I find really gripping. 5 4 3 2 1
40. I usually plan out my week's work in advance, either on paper or in my head. 5 4 3 2 1
41. I keep an eye open for what lecturers seem to think is important and concentrate on that. 5 4 3 2 1
42. I'm not really interested in this course, but I have to take it for other reasons. 5 4 3 2 1
43. Before tackling a problem or assignment, I first try to work out what lies behind it. 5 4 3 2 1
44. I generally make good use of my time during the day. 5 4 3 2 1
45. I often have trouble in making sense of the things I have to remember. 5 4 3 2 1
46. I like to play around with ideas of my own even if they don't get me very far. 5 4 3 2 1
47. When I finish a piece of work, I check it through to see if it really meets the requirements. 5 4 3 2 1
48 Often I lie awake worrying about work I think I won't be able to do. 5 4 3 2 1
49 It's important for me to be able to follow the argument, or to see the reason behind things. 5 4 3 2 1
50. I don't find it at all difficult to motivate myself. 5 4 3 2 1
51. I like to be told precisely what to do in essays or other assignments. 5 4 3 2 1
52. I sometimes get 'hooked' on academic topics and feel I would like to keep on studying them. 5 4 3 2 1

C. Preferences for different types of course and teaching

***5** means definitely like (√) **4** = like to some extent (√**?**) **2** = dislike to some extent (**x?**) **1** = definitely dislike (**x**).*

*Try not to use **3** = unsure (**??**), unless you really have to, or if it cannot apply to you or your course.*

√ √? ?? x? x
a. lecturers who tell us exactly what to put down in our notes. 5 4 3 2 1
b. lecturers who encourage us to think for ourselves and show us how they themselves think 5 4 3 2 1
c. exams which allow me to show that I've thought about the course material for myself. 5 4 3 2 1
d. exams or tests which need only the material provided in our lecture notes. 5 4 3 2 1
e. courses in which it's made very clear just which books we have to read. 5 4 3 2 1
f. courses where we're encouraged to read around the subject a lot for ourselves. 5 4 3 2 1
g. books which challenge you and provide explanations which go beyond the lectures. 5 4 3 2 1
h. books which give you definite facts and information which can easily be learned. 5 4 3 2 1

Thank you very much for spending time completing this questionnaire: it is much appreciated.

Appendix F - Additional Questionnaires for Students

F.1. PBL Questionnaire for Students

Questionnaire for Students taking PBL based software development classes

The following questions relate to your opinions of Software Development (java programming) PBL classes. There are six groups of items, and those in each group pertain to the title at the top of that group. Please indicate your opinions using the scale provided:

PBL Group work

1. The tutorial group discussion is an important stimulus for my software development learning activities.

1	2	3	4	5
totally disagree	disagree	are neutral	agree	totally agree

2. The learning issues generated in the group tutorials are the most important starting point for my learning activities.

1	2	3	4	5
totally disagree	disagree	are neutral	agree	totally agree

3. I study to a large extent independently from the learning issues generated by my PBL group tutorials.

1	2	3	4	5
totally disagree	disagree	are neutral	agree	totally agree

4. The group climate facilitated the learning process.

1	2	3	4	5
totally disagree	disagree	are neutral	agree	totally agree

5. In the PBL tutorials I learned something that improved my software development skills.

1	2	3	4	5
totally disagree	disagree	are neutral	agree	totally agree

6. In the PBL group, I improved my communication skills.

1	2	3	4	5
totally disagree	disagree	are neutral	agree	totally agree

7. I would recommend PBL tutorials to other students.

1	2	3	4	5
totally disagree	disagree	are neutral	agree	totally agree

The PBL method

8. The PBL classes have motivated me to use additional learning resources.

1	2	3	4	5
totally disagree	disagree	are neutral	agree	totally agree

9. If you had had the possibility to choose before the course, would you have opted for the PBL-course or the lecture-based course? (Tick as appropriate)

1	2	3
yes	are neutral	no

10. After the experience of the course, would you now opt for the PBL-course or the lecture-based course if you had to choose again? (Tick as appropriate)

1	2	3
yes	are neutral	no

11. I felt well informed about the PBL method.

1	2	3	4	5
totally disagree	disagree	are neutral	agree	totally agree

12. I consider PBL to be an effective way of learning for myself.

1	2	3	4	5
totally disagree	disagree	are neutral	agree	totally agree

13. PBL was fun.

1	2	3	4	5
totally disagree	disagree	are neutral	agree	totally agree

14. Before the tutorials, I was open to the method.

1	2	3	4	5
totally disagree	disagree	are neutral	agree	totally agree

Student interest in Software Development

15. I am interested in the subject (Software Development) of the PBL tutorials.

1	2	3	4	5
totally disagree	disagree	are neutral	agree	totally agree

16. I consider the subject (Software Development) to be important within the frame of my studies.

1	2	3	4	5
totally disagree	disagree	are neutral	agree	totally agree

17. After class attendance, how much additional learning time did you invest each week in Software Development (time in hours).

Time (Hours)	

Course objectives and content

18. Topics covered during PBL classes stimulated my interest in Software Development.

1	2	3	4	5
totally disagree	disagree	are neutral	agree	totally agree

19. The content of the tutorials fitted the level of my knowledge.

1	2	3	4	5
totally disagree	disagree	are neutral	agree	totally agree

20. The problems used in the PBL classes illustrate Software Development concepts

1	2	3	4	5
totally disagree	disagree	are neutral	agree	totally agree

21. The questions included on past exams and continuous assessment for software development, to a large extent determine what I will study.

1	2	3	4	5
totally disagree	disagree	are neutral	agree	totally agree

22. The learning issues generated in the PBL classes are tuned to the subject matter to be tested.

1	2	3	4	5
totally disagree	disagree	are neutral	agree	totally agree

23. At the start of the Software Development course, I consulted the course objectives set out in the syllabus.

1	2	3	4	5
totally disagree	disagree	are neutral	agree	totally agree

24. At the end of the Software Development course, I consulted the course objectives to check whether I covered all the subject matter I was expected to cover.

1	2	3	4	5
totally disagree	disagree	are neutral	agree	totally agree

The PBL tutor

25. The PBL tutor has steered the group strongly.

1	2	3	4	5
totally disagree	disagree	are neutral	agree	totally agree

26. The PBL tutor’s interventions were adequate.

1	2	3	4	5
totally disagree	disagree	are neutral	agree	totally agree

27. The PBL tutor is enthusiastic about PBL.

1	2	3	4	5
totally disagree	disagree	are neutral	agree	totally agree

28. In general, the tutor stimulates students to make use of different sources of information.

1	2	3	4	5
totally disagree	disagree	are neutral	agree	totally agree

29. In general, the tutor stimulates my Software Development learning activities.

1	2	3	4	5
totally disagree	disagree	are neutral	agree	totally agree

Teaching resources

30. The class room, laboratories, and computer equipment were adequate.

1	2	3	4	5
totally disagree	disagree	are neutral	agree	totally agree

31. The Moodle e-learning environment supported my learning activities.

1	2	3	4	5
totally disagree	disagree	are neutral	agree	totally agree

32. I would like more timetabled PBL Software Development classes.

1	2	3	4	5
totally disagree	disagree	are neutral	agree	totally agree

33. In general do you have any other additional comments about PBL or Software Development? (write in the space provide below)

F.2. General Background Questionnaire for Students

The following questions relate to your background and your reasons for selecting a computing course. All answers are anonymous.

Please enter an X in the correct box:

Male		Female	

Nationality: _______________________

Home address: (please do not enter house numbers or name)

Street	Town	County	Post Code

Parents' occupation (optional):

Father ____________________ Mother ____________________

Please describe any previous non-programming computer experience you may have:

Please describe any previous programming experience you may have:

Where you encouraged by other people to pursue Computing as a career? YES/NO

If YES please give details:

Appendix G - Interview Questions for Staff

Q1: What for you are the essential characteristics of PBL?

Q2: For PBL then, what do you see at its main advantage…..main disadvantage?

Q3: Why did you “volunteer” to be a PBL tutor?

Q4: What makes a good PBL tutor? It might help to focus on your “main three elements”.

Q5: In your opinion, how have the students taken to the PBL environment?

Q6: Have you noticed any change in learner behaviour in the PBL lab classes as opposed to the traditional lab environment?

Q7: What do you understand by the term problem-solving skills?

Q8: How does “problem-solving” *per se* fit into your view of PBL?

Q9: In your opinion, have the students’ “problem-solving” skills improved more in the PBL lab classes as opposed to the traditional lab environment?

Q10: What is your opinion on the work load associated with PBL classes?

Q11: Have you anything further to say about PBL as a guiding philosophy for the curriculum?

Appendix H - Interview Questions for Students

Q1: Do you enjoy the PBL classes?

Q2: Did working together with other students in the PBL groups help you make friends?

Q3: Do you think the PBL group environment facilitates the learning process?

Q4: What was your opinion of the atmosphere of your PBL group meeting?

Q5: What was your opinion of the relationships between group members?

Q6: How did you find the distribution of the work between group members?

Q7: Did you feel isolated or uncomfortable in your PBL Group?

Q8. Did you fell well informed about the PBL method?

Q9: Do you think keeping a PBL journal is useful?

Q10: Do you consider Software Development to be an important subject?

Q11: How do you go about writing a Java solution to your programming problems?

Q12: What advice would you give to other students who are having problems with Java programming?

Q13: How much time do you spend studying Software Development outside of the class contact hours?

Q14: Is there anything you would like to add?

Appendix I - Interview and Observation Schedule

I.1. Interview Schedule for staff interviews:

All names used are pseudonyms:

Catherine: 10.30am, Monday, 12 May 2008

Stuart: 11.15am, Monday, 12 May 2008

Mary: 12.15am, Monday, 12 May 2008

David: 11.15am, Wednesday, 14 May 2008

Natasha: 12.15am, Wednesday, 14 May 2008

I.2. Interview Schedule for student interviews:

All names used are pseudonyms:

Sarah: 10.30am, Monday, 8 December 2008

Ahmed: 14.30am, Monday, 8 December 2008

Nichole: 10.30am, Wednesday, 10 December 2008

Darren: 14.30am, Wednesday 10 December 2008

Paul: 15.00am, Wednesday 10 December 2008

William: 15.30am, Wednesday 10 December 2008

I.3. Observation Schedule

Each week of the 15 week semester the researcher spent half an hour in PBL and half an hour in non-PBL Software Development laboratories observing participants' behaviour. This was done for each of four cohorts of learners over four academic years, resulting in a total of 80 hours of participant observation.

Appendix J - Interview Transcriptions

J.1. Staff Interview Transcriptions

Some staff members requested that small portions of their interviews that related to their views of college management and colleagues should not be transcribed. These portions have not been recorded in the transcriptions, but the researcher was cognisant of the issues. The location of the omitted portions is shown in the transcriptions.

J.1.1. David

Q1: What for you are the essential characteristics of PBL?

David: First of all the group work ... getting a group of students together to solve a common problem working together focusing on teamwork ...with an emphasis on problem-solving rather than low-level implementation so ... how are they going about solving the problem? What issues are they taking into consideration rather than building in our case in Software Development, a code solution to a real world problem. So it's just about taking a high-level view of a problem and working towards one or more solutions.

Q2: For PBL then, what do you see at its main advantage…..main disadvantage?

David: I suppose the main advantage would be the group work, building a common solution building on suggestions of other students and trying to see the positives and negatives for each suggestion and then building that into a solution without as I say getting bogged down in low level details of how it would be solved at a code level or whatever. So it's just focusing on the problem as a whole.

The main disadvantage that I would suggest is that sometimes they skip over low level details that actually require a bit more investigation. So sometimes students take a very simplistic view of some of the problems, which is fine in some cases because that's not the focus of the main activity. But sometimes you do want them to get down to a greater level of detail for some complex area. But it really depends on the scope of the activity.

Q3: Why did you volunteer to be a PBL tutor?

David: Eh… I was assigned the role.

Q4: What makes a good PBL tutor?

David: Essentially to get the best out of the students is to guide them towards a solution without giving it to them so to get them to come up with a solution but em to stand back a little bit and let them at it themselves. But if they're not getting anywhere to try and encourage them, to try and look at what they have and maybe suggest some alternatives without giving them the answer but really just trying to focus the direction of the group.

Q5: In your opinion, how have the students taken to the PBL environment?

David: I think they enjoyed it. Certainly in the latter stages they seemed to get a decent amount out of it especially the group work. Actually some of them might be a bit reluctant say in a lecture to speak up but in a small group setting may be more inclined to volunteer their opinion.

Q6: Have you noticed any change in learner behaviour in the PBL lab classes as opposed to the traditional lab environment?

David: Em, it's been a while since I would have done a direct comparison with the two but I guess some of the problems with the ... they're not really thinking about the larger problems in the low-level traditional labs. They're more stuck with syntax errors when they're starting programming. So they're bogged down in that rather than thinking about how they would solve the problem and worried about viewing bits of code that might do what they want rather than thinking about it in a logical step-by-step what do I need to do not what's this bit of code that I might need to use. So it kind of focuses their thinking a little bit more.

Q7: What do you understand by the term problem-solving skills?

David: Generally a logical approach to solving a problem. So step-by-step thinking about all the different issues, the assumptions, the prerequisites, what needs to happen in what sequence, what sequences are repeated, error conditions, what will happen in certain cases. Those kinds of core concepts that students need to learn for Software Development.

Q8: Do you think problem-solving *per se* fits into your view of PBL?

David: Yeah definitely. It's the very centre of it really. It's about solving the problem although you do have to give them some sort of solution near the end.

Q9: In your opinion, have the students' problem-solving skills improved more in the PBL lab classes as opposed to the traditional lab environment?

David: To be honest I wouldn't be able to really give a proper answer on that because eh I don't see the result of that em I'm just a facilitator I don't see the other side of it. I would hope so.

Q10: What is your opinion on the work load associated with PBL classes for you as a tutor?

David: Em there's very little to do. Really it depends on the amount of students you have. At the earlier stages there's a lot more work to do because you have to make sure that they're doing it right and one person isn't just shouting down the others, just to make sure that everyone's involved and not just sitting silently by, that they're all making a contribution. So from that point of view it depends on the number of students that you have to look after.

Interviewer: Did you ever notice many of them shouting at each other or is there tension in the groups?

David: Not normally, but sometimes you might get a loud mouth or someone who's a bit more vocal than the others or may have some prior knowledge or just natural ability and it's their way or no way and that can be a little bit unfortunate ... you need to get everybody contributing and not just agreeing with the leader. I haven't really seen too much of people being shouted down but there have been. It's more ... somebody might say something stupid and they get a little bit of slagging for it and they don't contribute anything more after that. So you have to try and kind of guide that in the right way.

Q11: Have you anything further to say about PBL as a guiding philosophy for the curriculum?

David: I think it definitely needs to be there. Certainly in the early stages. Not really sure about after say first year but it's no harm as an activity. Even at project stage it might be no harm in the initial stages for project planning, defining the goal of the project and how much you're going to do… where you get the group and maybe their project supervisor and maybe sit around a whiteboard and brainstorm.

J.1.2. Natasha

Q1: What for you are the essential characteristics of PBL?

Natasha: Well first of all it's that the students work in groups. Another thing is that the problems that are presented are ... they are different in that they are not like any of the problems that you encounter in the text books on Java. They are original in a way. And I think the third characteristic that comes out from the second one is that they come up with an original solution so there are never two solutions that are the same. Yeah.

Q2: For PBL then, what do you see at its main advantage…..main disadvantage?

Natasha: The advantage is that everyone is interacting all the time and that the tutor is near so that you don't have to wait for half an hour till the tutor comes to every student in the class as in the Java labs like, but in the PBL he's always there like and you can always get feedback from the tutor. And em the disadvantage is that the students who are weak, like, they don't really need to work in the PBL because the stronger students do everything for the group, well, if it comes out finally.

Q3: Did you volunteer to become a PBL tutor?

Natasha: Well, more or less. I like it.

Q4: What makes a good PBL tutor?

Natasha: Well first of all I think it must be the wish of the tutor to improve the knowledge and the skills. Like, the tutor must be motivated because if he doesn't care about what the students know or don't know then it won't work. Another thing is that it should be of course more social, communicative. And another thing, I think that the tutor should not be afraid to make a mistake because I think that many of the bad things come out of the fact that the tutor cannot admit something that he doesn't know because em the most important thing em is to be willing to help and not knowing everything by heart.

Q5: In your opinion, how have the students taken to the PBL environment?

Natasha: Do you mean they like it or not?

Interviewer: Yeah.

Natasha: Well, most people enjoy the environment but there are a few of them who feel it's even more complicated, I mean ones like who aren't really… who don't really like PBL at all.

Q6: Have you noticed any change in learner behaviour in the PBL lab classes as opposed to the traditional lab environment?

Natasha: Em yes, I think that in the PBL the students are more motivated to solve the problems by themselves while, when they write a program they more feel like asking like more questions and they are really sometimes trying to make the teacher make the solution instead of them, while in the PBL, well, it's more or less their work.

Q7: What do you understand by the term problem-solving skills?

Natasha: Well, I think first of all it is to break the task into a number of small steps, small and logical, and that's the most important. So sometimes you don't really have to know the language but you have to be able to make the task look logical.

Q8: How does problem-solving per *se* fit into your view of PBL? Do the students focus on problem-solving?

Natasha: Yes, more or less yes. But to my mind PBL might be a little bit more standardised or something.

Q9: In your opinion, have the students' problem-solving skills improved more in the PBL lab classes as opposed to the traditional lab environment?

Natasha: I would say rather yes.

Q10: What is your opinion on the work load associated with PBL classes?

Natasha: For me I think it's as much workload as in the classes. It's the same.

Interviewer: And for the students? Do you think they have more work or less work?

Natasha: I think they have more work. Just in Java labs they have more work without thinking. In the PBL there is more work going on outside like interacting, writing things and changing things, yeah.

Q11: Have you anything further to say about PBL as a guiding philosophy for the curriculum?

Natasha: No, I think no.

J.1.3. Mary

Q1: What for you are the essential characteristics of PBL?

Mary: Well really to get the students to step back from looking at problem-solving as a computing activity and to think about the problem itself as distinct from programming.

Q2: For PBL then, what do you see at its main advantage…..main disadvantage?

Mary: OK well really following on from the answer to the first question, it's getting the students to focus on the problem and to go through the whole brainstorming process; to go through all the specific cases without thinking about the programming and to get an understanding of the problem. I think there's a lot of advantages to it.

One of the main disadvantages that I've found in my work was to do with the way we structured the groups. I found that the groups were far too big, that a lot of people got lost within the group ... there's some very extrovert people and very quiet people and it didn't really suit some people. It suited some, and others it didn't, so you're losing some people in that setting.

Q3: Did you volunteer to be a PBL tutor?

Mary: Yeah I did actually. I think I put my name on a list. I was interested in problem-solving anyway, the different aspects of problem-solving for programming.

Q4: What makes a good PBL tutor? The characteristics?

Mary: First of all you need to have an understanding of the problem that you're presenting to the students so you want to be able to be a good problem-solver yourself. Also I find that not jumping in too quickly during the whole process ... to throw a few pointers in, to guide the students to solve the problem, and to go back to the students who are less likely to solve the problem, to bring them in to the whole process, to try to, you know, let them have a part.

Q5: In your opinion, how have the students taken to the PBL environment?

Mary: Mostly positive. I know that they say that it's a bit of a chore at times. Here we go again, we have to do all this writing and it's tedious, and there's that aspect to it definitely, and you know when after a few sessions with PBL they were able to go back to the computer, they were just absolutely dying to get back to programming. So you do have to force them to take the time to take that step back.

Q6: Have you noticed any change in learner behaviour in the PBL lab classes as opposed to the traditional lab environment?

Mary: Em, they're more likely to consider other aspects of the problem rather than how do I get to that solution so they look at extreme cases and unusual cases of the problem. And they might work through, I mean we coach them to work through specific problems with say numbers or something like that and they're more likely I think to do that.

Interviewer: "So they have a broader view?

Mary: I think so.

Q7: What do you understand by the term problem-solving skills?

Mary: Just reiterating what I said, to take a problem and to look at what you have and to look at all the different ways that you could possibly get to the solution that you want and then the different actions you could take and the means that you could take to get there.

Q8: How does problem-solving *per se* fit into your view of PBL? (question unintentionally omitted)

Q9: In your opinion, have the students' problem-solving skills improved more in the PBL lab classes as opposed to the traditional lab environment?

Mary: Definitely. They don't get lost in the coding. I think it's very very helpful.

Q10: What is your opinion on the work load associated with PBL classes?

Mary: Em, the workload outside the classes is not huge. You would need to go and get the problems and possibly work through it yourself but there's not a huge workload.

Interviewer: What do you think about the students' workload?

Mary: No, they come in and they're presented with a problem and all the work is done within the class. All the thinking work is done there so I don't think it's extra for them.

Q11: Have you anything further to say about PBL as a guiding philosophy for the curriculum?

Mary: It would probably be very useful in other areas. It's very useful in Computing. I'd probably see it as being useful in Engineering and Science, other disciplines.

J.1.4. Catherine

Q1: What for you are the essential characteristics of PBL?

Catherine: Group work. That's the main thing.

Q2: For PBL then, what do you see at its main advantage…..main disadvantage?

Catherine: Well...I think the good thing is seeing how involved with the problems the students get. And after they have tried their initial solutions, working with the students is enjoyable, much better than in the ordinary labs.

Interviewer: Why is that?

Catherine: Because they are actually trying to construct a solution that works rather than just stopping every time they have a problem and asking for help, so their labs are much better .

Interviewer: Any disadvantages?

Catherine: Yes, there is a lot of extra work preparing for a lab.

Interviewer: Why is that?

Catherine: Well in the ordinary labs you are never stuck for a solution as the questions are so simple. But in the PBL labs you come up against complex problems that you have to figure out on the spot. But look the worst thing is having to deal with X [other staff member]. [Section of transcription omitted – The omitted segment discussed work load between staff]

Interviewer: Any more disadvantages?

Catherine: There are many, only some students do the work, the rest are just too weak to work out the problems so they just tag along. Also most of the students don't know where to start, they have no idea.

Q3: Why did you "volunteer" to be a PBL tutor?

Catherine: I didn't. I just needed to make up hours. I would have tutored on any available course.

Q4: What makes a good PBL tutor?

Catherine: Being able to guide the students and help them understand the problem.

Q5: In your opinion, how have the students taken to the PBL environment? Do they like it?

Catherine: Yes most of them. Some don't like the group work as they can be shown to be weak in front of their friends.

Q6: Have you noticed any change in learner behaviour in the PBL lab classes as opposed to the traditional lab environment?

Catherine: In the PBL labs the students actually keep working on their problems and are not surfing the web and using Bebo. You don't have to constantly be asking them to stop messing.

Interviewer: Why do you think they spend so much time doing other things rather than working on their lab-sheets?

[Section of transcription omitted – The omitted segment discussed the lack of management support for staff disciplining students for not working solely on Software Development problems in lab time.]

Interviewer: Have you noticed any change in how students go about problem-solving?

Catherine: Yes well it is quite different. Em, firstly the PBL students are in a group and away from the machines. They are trying to get the steps in their algorithm right. Whereas the others are always stuck on syntax.

Q7: What do you understand by the term problem-solving skills?

Catherine: It means being able to break a problem down into its parts. Organising the facts you have and applying them to the problem to be solved.

Q8: How does problem-solving *per se* fit into your view of PBL?

Catherine: I think it is at the core of PBL.

Q9: In your opinion, have the students' problem-solving skills improved more in the PBL lab classes as opposed to the traditional lab environment?

Catherine: Yes I think they have. They have better critical reasoning skills.

Interviewer: Can you expand on that?

Catherine: I mean they can have a higher level focus on the problem and they work out their overall problem-solving approach before they start to code. They have a plan of what they want to do. Also they understand the underlying concepts better. They know what is meant by abstraction for example. They know what you mean when you talk about an object-oriented programme. They know what a class is.

Interviewer: Ok, any other comments on the students' problem-solving skills?

Catherine: One thing that is a problem is that…well they sometime try to Google the answers. And its is a problem because they don't understand what the code they find does. You can't just take it out of context and lots of it are full of errors, but they think it's perfect and they are just looking for a quick answer.

Interviewer: Is that just a problem for the PBL group?

Catherine: Er..no. Both groups do it, sorry.

Q10: What is your opinion on the work load associated with PBL classes?

Catherine: Yes, as I said earlier there is a lot of extra work preparing for a lab. Although haveing all the material online is a big help.

[Section of transcription omitted – The omitted segment discussed management support for PBL and recognition of the extra workload]

Q11: Have you anything further to say about PBL as a guiding philosophy for the curriculum?

Catherine: I think it has its place, especially for subjects that require the students to develop diagnostic skills.

Interviewer: Would you like to say anything else?

Catherine: Er..well. I think by doing the PBL module they get a greater interest in their course in general. I mean in all their other Computing subjects, and that seems to last even into second and third year.

J.1.5. Stuart

Q1: What for you are the essential characteristics of PBL?

Stuart: The team work. It gets the students to work as a team. That is the most important thing.

Q2: For PBL then, what do you see at its main advantage…..main disadvantage?

Stuart: Em, I think the team work is a big advantage, having the students working together is more like real life. And that's what we need to do prepare them for working in industry. Also it helps to create a love of Computing in the students. It really does and they don't lose that.

Interviewer: Any disadvantages?

Stuart: No not really.

Q3: Why did you volunteer to be a PBL tutor?

Stuart: I'm the PBL coordinator, not just a tutor.

Interviewer: Sorry, I mean why did you volunteer to be the PBL coordinator?

Stuart: I wanted us to implement PBL and I pushed for its introduction so it was natural that I'd lead on it.

Q4: What makes a good PBL tutor?

Stuart: Em, knowing how to lead the students to a solution, without giving them the answer. And having good problem-solving skills and having enthusiasm for PBL and the energy and drive to make it work.

Q5: In your opinion, how have the students taken to the PBL environment?

Stuart: Oh they love it.

Q6: Have you noticed any change in learner behaviour in the PBL lab classes as opposed to the traditional lab environment?

Stuart: Em, my labs are much more professional. The students love the problems and love working together. (Laughs) When I bring in the whiteboard they fight over the pens. They really enjoy the labs. The only thing is I'm so busy I never get a break, I have to get around to all the groups. Yesterday I missed lunch, but I don't care, it's fun.

Interviewer: Have you noticed any change in how students go about problem-solving?

Stuart: My students are much better team workers and they are better problem solvers because I make them work on their problems at a high level, working out the correct set of steps in pseudo code before they go near Java. Also they know the basic concepts of object-oriented programming, like classes, methods, abstraction etc.

Interviewer: All good then?

Stuart: Yes, well except some of them still try to get the answers on the web. That's no good. We tried to stop that by making them turn off the machines at the start of the lab, but then they couldn't get Moodle, and that was a problem, as they needed to access the course notes online. We couldn't do without Moodle so we had to allow them to turn on the machines again. Still it is better than the other labs, where they are just surfing the web all the time.

Interviewer: Why do you think they spend so much time surfing the web rather than working on their lab-sheets?

Stuart: I think they are bored. Those labs are not much fun for them, so they get bored and do other things. We have to keep them interested.

Q7: What do you understand by the term problem-solving skills?

Stuart: Well if you can't problem-solve you won't be able to program that's for sure.

Q8: How does problem-solving *per se* fit into your view of PBL?

Stuart: Well, that and working together as a team is what PBL is all about. Problem-solving as a group is the key activity.

Q9: In your opinion, have the students' problem-solving skills improved more in the PBL lab classes as opposed to the traditional lab environment?

Stuart: Absolutely. They are so much better than they were before. I think the PBL has really made a difference. The students design their solutions from the top down, they work out their plan first and start from there, rather than just hacking away on the computer. Also when they meet a problem they can look at it from a number of different angles and they find better solutions.

Q10: What is your opinion on the work load associated with PBL classes?

Stuart: There is an awful lot of work, but it's worth it, it really makes a difference.

Q11: Have you anything further to say about PBL as a guiding philosophy for the curriculum?

Stuart: Eh, no, I think I've said all the important stuff. I'm leading on it and I think it's crucial.

J.2. Student Interview Transcriptions

J.2.1. Paul

Q1: Did you enjoy the PBL classes?

Paul: I would say overall yes. At first more so than at the end.

Interviewer: Why was that?

Paul: At the end I think maybe I got bored with it. Like, it starts off at the start and you're interested. Half way through you think it's the best thing that ever happened and by the end you're just frustrated with it.

Interviewer: What's the cause of that frustration?

Paul: I don't know, maybe just doing the same thing again and again.

Q2: Did working together with other students in the PBL groups help you make friends?

Paul: Em, personally I don't think it helped that much for me.

Q3: Do you think the PBL group environment facilitates the learning process?

Paul: Absolutely, yes.

Interviewer: In what way?

Paul: In the teambuilding, well, the teamwork. And then, in the course of what we do, finding alternative ways for a solution. I think it's brilliant, yeah.

Q4: What was your opinion of the atmosphere of your PBL group meeting?

Paul: My personal group was an excellent group. Lovely mix of people, lovely mix of talents, so it was brilliant.

Q5: What was your opinion of the relationships between group members?

Paul: Again everybody had something different to bring to the table. It was all very positive. And those who were strong helped and waited for those who weren't as strong. So it was very nice.

Q6: How did you find the distribution of the work between group members?

Paul: The stronger ones took on more but everyone contributed in my personal group. I saw other groups where it was desperate, where two people did everything but in mine personally it was brilliant.

Q7: Did you feel isolated or uncomfortable in your PBL Group?

Paul: No, not at all.

Q8. Did you fell well informed about the PBL method?

Paul: Yeah, great information came through. There was great supervision of all groups, yeah.

Q9: Did you think keeping a PBL journal was useful?

Paul: Retrospectively no, but at the time possibly yes.

Interviewer: And useful in what way?

Paul: Well, at the time it was important because there were marks allocated to it. But em, maybe you could have traced back your previous problems to help.

Q10: Do you consider Software Development to be an important subject?

Paul: Yeah

Q11: How do you go about writing a Java solution to your programming problems?

Paul: I suppose you'd find the way. An element of PBL would be in the mind. First you'd figure out how you've got to overcome the problem and then put it down. You might draw pictures, you might map it out. And you pull out all the little complex bits and work them out. And then you pull it all together.

Interviewer: So you don't start on the computer?

Paul: No.

Q12: What advice would you give to other students who are having problems with Java programming?

Paul: (Laughs). Ask for help.

Q13: How much time do you spend studying Software Development outside of the class contact hours? (Question unintentionally omitted)

Q14: Is there anything you would like to add?

Paul: No, overall very good. It's a great tool.

J.2.2. William

Q1: Did you enjoy the PBL classes?

William: No. Not at all.

Interviewer: Why was that?

William: Cos like you're sitting there and if you didn't know anything and then the lecturer comes over and starts talking to you and you feel under pressure and it's like ooooh I don't know anything and everyone else is just standing around and they're doing everything and you just haven't got a clue what's going on so it's not really very nice just sitting there like that.

Q2: Did working together with other students in the PBL groups help you make friends?

William: Yeah, no, I did.

Q3: Do you think the PBL group environment facilitates the learning process?

William: No. I didn't learn much in it.

Interviewer: Why did you think that was? Was it as you were saying that you felt a bit lost?

William: Yeah, they knew everything and if you were lost and behind and all that and they were flying ahead of you so you don't know and they're moving on. You don't really know what they're on about like cos you're like you're falling behind. You're unsure.

Q4: What was your opinion of the atmosphere of your PBL group meeting? Were people helpful?

William: Yeah, they were. They were really friendly. Like they do try to help you. Sometimes if you're stuck and you didn't understand the other person ... I think you need a lecturer *beside you. Like if there was a lecturer like there'd be two between four or five groups and that's not enough. I think you need more lecturers. Like I think you need to look at that and there should be less people within the class rather than having so many groups and the lecturer not being able to help you that much like.*

Interviewer: Ok, so you'd like to see the lecturers help a bit more.

William: Yeah, and less students in the PBL groups. We had eight in our group sometimes and that's too much. You're not going to learn much.

Q5: What was your opinion of the relationships between group members? Were people happy with the people in their group?

William: Yeah, they were. It was OK.

Q6: How did you find the distribution of the work between group members?

William: Yeah, it was ok but there was always certain people. Like, half of them would know what they were doing and the other half was just sitting there not knowing what was going on so the people that knew what they were doing would communicate between themselves and we kind of just sat back and you may say the odd thing whether it was right or wrong but people who knew what they were doing they just moved ahead really. We were falling behind still.

Q7: Did you feel isolated or uncomfortable in your PBL Group?

William: Yeah, I did, because if you don't know what you're doing you just feel, you know, you're just sitting there.

Q8. Did you fell well informed about the PBL method?

William: Yeah, I understand like what it was all about and all but like sometimes I didn't get the concepts, like the steps… I didn't know … there were certain steps and I was like wooh I didn't really get it to be honest. I knew all the bits of the general thing but I didn't know how to do it like I didn't have a good idea of how to do each step like.

Q9: Do you think keeping a PBL journal is useful?

William: Em, kind of. It was all right. Like you could look back at things but I suppose if you don't understand it, em…

Q10: Do you consider Software Development to be an important subject?

William: It is if you want to go down that road of doing games and all that and if you don't it's not of use at all but it wouldn't be of use to me now because I don't want anything to do with it. I hate it.

Q11: How do you go about writing a java solution to your programming problems?

William: Em… how did I go about it? I dunno, I'd ask questions, of the people sitting beside me or whatever.

Interviewer: So you'd ask other people in the group.

William: Yeah.

Q12: What advice would you give to other students who are having problems with java programming?

William: Listen from the very beginning and take out loads of books, and study. And take out books and all. And ask loads of questions as well.

Q13: How much time do you spend studying Software Development outside of the class contact hours?

William: Very little, I've got a job and that takes up most of my time.

Q14: Is there anything you would like to add, other than what you've already said?

William: Classes should be smaller.

J.2.3. Darren

Q1: Do you enjoy the PBL classes?

Darren: Well, they get you involved and are a good way of learning, but I can think of better things to be doing.

Q2: Did working together with other students in the PBL groups help you make friends?

Darren: Oh yes, you get to meet all the other people in the class and as you're working together you get to really know them. You wouldn't be friends with them all but you get to know people and you'll know who to ask if you get a problem .

Q3: Do you think the PBL group environment facilitates the learning process?

Darren: Yeah.

Q4: What was your opinion of the atmosphere of your PBL group meeting?

Darren: Great, yeah.

Q5: What was your opinion of the relationships between group members?

Darren: It was great, yeah, great.

Q6: How did you find the distribution of the work between group members?

Darren: Ok, it was ok, yeah, no problem.

Q7: Did you feel isolated or uncomfortable in your PBL Group?

Darren: Me? No.

Q8. Did you fell well informed about the PBL method?

Darren: Yes, yeah.

Q9: Do you think keeping a PBL journal is useful?

Darren: Yeah.

Q10: Do you consider software development to be an important subject?

Darren: Yes, so many people fail it, it's scary but you just have to get through it. You'll never get a job if you can't program.

Q11: How do you go about writing a Java solution to your programming problems? (Question unintentionally omitted)

Q12: What advice would you give to other students who are having problems with Java programming?

Darren: I'd tell them that they need to work on their own.

Q13: How much time do you spend studying Software Development outside of the class contact hours?

Darren: Er..em, I don't spend that much, maybe an hour or two a week. I have go to work in the evening and that takes a lot of time.

Q14 Is there anything you would like to add?

Darren: No.

J.2.4. Nichole

Q1: Do you enjoy the PBL classes?

Nichole: No not really, they are ok, but they're a bit boring. You can't do anything but talk about the problems. That's not fun, that's work. I'd rather be doing something else.

Q2: Did working together with other students in the PBL groups help you make friends?

Nichole: Yes it did.

Q3: Do you think the PBL group environment facilitates the learning process?

Nichole: Yes in a way. It's good to be in the group because you get help solving the problems, but the guys are always trying to show off who knows the most, and impress the tutors. I couldn't be bothered with all that stuff. The problems are so hard, it's too much effort, I mean, you can't take a break or anything, and we have to solve the problems on our own. The tutors don't give us much help. I think they are a bit lazy or maybe they don't know how to solve some of the problems. It's not right that we have to do it all on our own, the tutors should give us more help.

Interviewer: When you ask the tutors for help what do they do?

Nichole: They never give you the right answer, they just say try this or that, I think they don't know the right answer.

Q4: What was your opinion of the atmosphere of your PBL group meeting?

Nichole: It was ok.

Q5: What was your opinion of the relationships between group members?

Nichole: Er..we all got on like.

Q6: How did you find the distribution of the work between group members?

Nichole: You can get away with doing nothing. Sometimes I would do nothing.

Interviewer: Why didn't you take an active part in the sessions?

Nichole: I couldn't be bothered. I'm a bit lazy in the mornings. I'll ask someone to go over it with me later on, and then I'll know the solutions for the exams. I don't like talking in the big group; if you make a mistake, they will always say you're wrong. Not everyone like, but some of them will.

Q7: Did you feel isolated or uncomfortable in your PBL Group?

Nichole: No, not at all, working in a group was good, but the problems were hard but I made friends and stuff.

Q8. Did you fell well informed about the PBL method?

Nichole: Yeah, I know what it is.

Q9: Do you think keeping a PBL journal is useful?

Nichole: Yeah, I could use it to look over the solutions.

Q10: Do you consider Software Development to be an important subject?

Nichole: Yeah, it is a very difficult subject, but if you can get through it, you have made it, because it is the subject people fail.

Q11: How do you go about writing a Java solution to your programming problems?

Nichole: Just start working out what has to be done…I just get someone to help me when I get stuck.

Q12: What advice would you give to other students who are having problems with Java programming?

Nichole: Get help with the problems from someone who knows the answers.

Q13: How much time do you spend studying Software Development outside of the class contact hours?

Nichole: I've no time for that, what with work and everything.

Q14: Is there anything you would like to add?

Nichole: No.

J.2.5. Sarah

Q1: Do you enjoy the PBL classes?

Sarah: Yes, they are enjoyable. Some of the problems are very difficult and you feel great when you get them sorted out.

Q2: Did working together with other students in the PBL groups help you make friends?

Sarah: Yes.

Q3: Do you think the PBL group environment facilitates the learning process?

Sarah: Yeah, but not everyone does the work. Some of the guys just sit back and let others do all the work. They don't do any preparation or nothing, but they still get credit when we get the right answer. It's not fair, the tutors don't do anything about it.

Interviewer: Did you mention it to the tutors?

Sarah: No, I'm not going to get into a fight over it. Like, it doesn't bother me that much. And some of the guys are me mates anyway, so I'm not going to cause trouble. You know what I mean? Anyway the tutors are paid to do the job, aren't they? They should notice and do something about it.

Q4: What was your opinion of the atmosphere of your PBL group meeting?

Sarah: Mainly good, sometimes we would have an argument about what to do next but overall it was ok.

Q5: What was your opinion of the relationships between group members?

Sarah: Like I say, it was ok.

Q6: How did you find the distribution of the work between group members? Was it fairly distributed?

Sarah: As I said before, no. Some people just sat back and let others do all the work.

Q7: Did you feel isolated or uncomfortable in your PBL Group?

Sarah: No that's not what I'm saying. I wasn't isolated or uncomfortable, I just didn't like it that I had to do more work than some of the others.

Q8. Did you fell well informed about the PBL method?

Sarah: Yes, but I thought we would be able to pick some of the problem areas ourselves, but the tutors always set the problems. We had no say in choosing them.

Q9: Do you think keeping a PBL journal is useful?

Sarah: Yeah, you can use it to study the answers and it shows you how much you have learnt. Yeah, the journal is a good idea.

Q10: Do you consider Software Development to be an important subject?

Sarah: Yes of course. If you can't program you can't do anything. You need to know how to program to be able to handle other subjects. […] Once you know how to program then you feel like you have made it.

Q11: How do you go about writing a Java solution to your programming problems?

Sarah: I'd get all the notes and read them and then I'd start working out the steps on paper. Once I had all that done, I'd start writing the code.

Q12: What advice would you give to other students who are having problems with Java programming?

Sarah: They need to sit down and think about what they're doing, try the problems themselves.

Q13: How much time do you spend studying Software Development outside of the class contact hours?

Sarah: I try to do at least an hour a day, and go over what we did in class. I'd like to do more but I have to work. I work four evenings a week and Saturday mornings so it's difficult. I need the money. I try to do some revision when I get home from work but I'm usually too tired.

Q14: Is there anything you would like to add?

Sarah: No.

J.2.6. Ahmed

Q1: Do you enjoy the PBL classes?

Ahmed: I know a lot about Java and I enjoy showing the others how to solve the problems.

Interviewer: What about the PBL rather than just Java programming?

Ahmed: Yes I get to help all the others in the group. I like that.

Q2: Did working together with other students in the PBL groups help you make friends?

Ahmed: No, not really.

Q3: Do you think the PBL group environment facilitates the learning process?

Ahmed: Yes.

Q4: What was your opinion of the atmosphere of your PBL group meeting?

Ahmed: It was good. Yes.

Q5: What was your opinion of the relationships between group members?

Ahmed: A good relationship, we all had a good working relationship.

Q6: How did you find the distribution of the work between group members?

Ahmed: Well some people can't solve the problems and can't do the work, the group would be better without them.

Q7: Did you feel isolated or uncomfortable in your PBL Group?

Ahmed: No.

Q8. Did you fell well informed about the PBL method?

Ahmed: Yes I did, but I think the tutors didn't. You see I did Java before coming here and I wanted to pick problems on arrays and methods but they didn't allow that. I had to do the same problems as everyone else.

Q9: Do you think keeping a PBL journal is useful?

Ahmed: It was good. Yes. I was proud to record my solutions.

Q10: Do you consider Software Development to be an important subject?

Ahmed: Yes, the most important subject. If you can't program you'll never make much money. Some of the others don't realize that, that's why they fail. It is easy once you work at it. You have to do all the lab problems and study the notes. You can't learn it from a book, you just have to do the labs.

Q11: How do you go about writing a Java solution to your programming problems?

Ahmed: I just start writing the code… I work out the solution in code and just program it

Q12: What advice would you give to other students who are having problems with Java programming?

Ahmed: Sit down and work it out on paper first and so you really understand what is happening.

Q13: How much time do you spend studying Software Development outside of the class contact hours?

Ahmed: About two hours a week.

Q14: Is there anything you would like to add?

Ahmed: No.

Appendix K - Analysis of Learner Attainment Scores

Statistical tests were carried out using Minitab 15 and MS Excel. Descriptive statistics, t-tests and F-tests were carried out using MS Excel, while the residual gain analysis and the general linear model were carried out using Minitab 15. The results are presented below.

K.1. Tests Carried Out on the Overall Attainment Scores of the Four Cohorts 2005/2009

K.1.1. Overall Exam Score: Descriptive Statistics, F-tests and t-tests

K.1.1.1. Descriptive statistics

Group A Exam	
Mean	50.91803279
Standard Error	1.690589772
Median	50
Mode	57
Standard Deviation	18.67317436
Sample Variance	348.6874407
Kurtosis	-1.01979063
Skewness	0.284887117
Range	80
Minimum	16
Maximum	96
Sum	6212
Count	122
Confidence Level(95.0%)	3.346968326

Group B Exam	
Mean	52.34710744
Standard Error	1.834949603
Median	50
Mode	34
Standard Deviation	20.18444564
Sample Variance	407.4118457
Kurtosis	-0.75824461
Skewness	0.077572055
Range	80
Minimum	12
Maximum	92
Sum	6334
Count	121
Confidence Level(95.0%)	3.633072468

K.1.1.2. F-tests

F-Test Two-Sample for Variances

	Exam A	*Exam B*
Mean	50.91803	52.34711
Variance	348.6874	407.4118
Observations	122	121
df	121	120
F	0.85586	
P(F<=f) one-tail	0.196987	
F Critical one-tail	0.740253	

K.1.1.3. t-tests

t-Test: Two-Sample Assuming Equal Variances

	Exam A	*Exam B*
Mean	50.91803279	52.34710744
Variance	348.6874407	407.4118457
Observations	122	121
Pooled Variance	377.9278084	
Hypothesized Mean Difference	0	

df	241
t Stat	-0.572954263
P(T<=t) one-tail	0.283604921
t Critical one-tail	1.651200843
P(T<=t) two-tail	0.567209842
t Critical two-tail	1.969856158

K.1.2. Overall CA Score: Descriptive Statistics, F-tests and t-tests

K.1.2.1. Descriptive Statistics

Group A CA		*Group B CA*	
Mean	64.27868852	Mean	56.37066116
Standard Error	1.765605792	Standard Error	1.781945406
Median	63.5	Median	58
Mode	46	Mode	72
Standard Deviation	19.50175338	Standard Deviation	19.60139947
Sample Variance	380.318385	Sample Variance	384.2148612
Kurtosis	-0.81430402	Kurtosis	-0.92061104
Skewness	-0.14102747	Skewness	-0.30300265
Range	77	Range	76
Minimum	21	Minimum	16
Maximum	98	Maximum	92
Sum	7842	Sum	6820.85
Count	122	Count	121
Confidence Level(95.0%)	3.495482322	Confidence Level(95.0%)	3.528127848

K.1.2.2. F-tests

F-Test Two-Sample for Variances

	CA A	*CA B*
Mean	64.27869	56.37066
Variance	380.3184	384.2149
Observations	122	121
df	121	120
F	0.989859	
P(F<=f) one-tail	0.477638	
F Critical one-tail	0.740253	

K.1.2.3. t-tests

t-Test: Two-Sample Assuming Equal Variances

	CA A	*CA B*
Mean	64.27868852	56.37066116
Variance	380.318385	384.2148612
Observations	122	121
Pooled Variance	382.2585392	
Hypothesized Mean Difference	0	

df	241
t Stat	3.152528558
P(T<=t) one-tail	0.000911932
t Critical one-tail	1.651200843
P(T<=t) two-tail	0.001823863
t Critical two-tail	1.969856158

K.1.3. Leaving Certificate Score: Descriptive Statistics, and t-tests

K.1.3.1. Descriptive Statistics

Group A LC Points	
Mean	64.27868852
Standard Error	1.765605792
Median	63.5
Mode	46
Standard Deviation	19.50175338
Sample Variance	380.318385
Kurtosis	-0.81430402
Skewness	-0.14102747
Range	77
Minimum	21
Maximum	98
Sum	7842
Count	122
Confidence Level(95.0%)	3.495482322

Group B LC Points	
Mean	56.37066116
Standard Error	1.781945406
Median	58
Mode	72
Standard Deviation	19.60139947
Sample Variance	384.2148612
Kurtosis	-0.92061104
Skewness	-0.30300265
Range	76
Minimum	16
Maximum	92
Sum	6820.85
Count	121
Confidence Level(95.0%)	3.528127848

K.1.3.2. t-test tests

t-Test: Two-Sample Assuming Equal Variances

	Group A LC Points	*Group B LC Points*
Mean	257.4180328	254.9586777
Variance	4550.096532	4336.039945
Observations	122	121
Pooled Variance	4443.512339	
Hypothesized Mean Difference	0	
df	241	
t Stat	0.28755936	
P(T<=t) one-tail	0.386965586	
t Critical one-tail	1.651200843	
P(T<=t) two-tail	0.773931172	
t Critical two-tail	1.969856158	

LC points Correlation = 0.602177

K.2. General Linear Model

Output from the General Linear Model (CA, Exam versus Group, Year) is given below:

Factor	**Type**	**Levels**	**Values**
Group	fixed	2	A, B
Year	fixed	4	05/06, 06/07, 07/08, 08/09

K.2.1. Analysis of Variance for CA, using Adjusted SS for Tests

Source	DF	Seq SS	Adj SS	Adj MS	F	P
Points	1	23660.9	23706.0	23706.0	82.59	0.000
Group	1	3457.2	3477.3	3477.3	12.11	0.001
Year	3	777.0	777.0	259.0	0.90	0.441
Error	237	68028.3	68028.3	287.0		
Total	242	95923.4				

S = 16.9422 R-Sq = 29.08% R-Sq(adj) = 27.58%

Term	**Coef**	**SE Coef**	**T**	**P**
Constant	22.106	4.346	5.09	0.000
Points	0.14935	0.01643	9.09	0.000

Unusual Observations for CA

Obs	**CA**	**Fit**	**SE Fit**	**Residual**	**St Resid**
17	87.0000	47.9106	3.1689	39.0894	2.35 R
23	27.0000	65.8320	2.6744	-38.8320	-2.32 R
72	39.0000	75.5431	2.5925	-36.5431	-2.18 R
101	87.0000	47.5375	2.8266	39.4625	2.36 R
105	53.0000	86.3673	3.4247	-33.3673	-2.01 R
113	85.0000	49.0310	2.7310	35.9690	2.15 R
123	20.0000	70.9583	3.1378	-50.9583	-3.06 R
134	76.0000	42.5827	3.0076	33.4173	2.00 R
146	25.0000	62.7443	2.7584	-37.7443	-2.26 R
161	23.0000	57.1707	2.4771	-34.1707	-2.04 R
190	23.0000	62.0012	2.4153	-39.0012	-2.33 R
207	89.0000	51.5471	2.4934	37.4529	2.23 R

R denotes an observation with a large standardized residual.

K.2.2. Analysis of Variance for Exam, using Adjusted SS for Tests

```
Source DF    Seq SS     Adj SS     Adj MS     F      P
Points  1    22785.0    22996.5    22996.5    80.50  0.000
Group   1    194.2      184.8      184.8      0.65   0.422
Year    3    524.3      524.3      174.8      0.61   0.608
Error   237  67701.1    67701.1    285.7
Total   242  91204.7

S = 16.9014   R-Sq = 25.77%   R-Sq(adj) = 24.20%

Term         Coef       SE Coef    T      P
Constant     14.152     4.336      3.26   0.001
Points       0.14709    0.01639    8.97   0.000

Unusual Observations for CA

Obs    Exam       Fit        SE Fit     Residual    St Resid
12     96.0000    54.9433    2.6615      41.0567    2.46 R
17     85.0000    38.0276    3.1613      46.9724    2.83 R
44     85.0000    44.9843    2.5470      40.0157    2.39 R
73     83.0000    44.7996    2.4531      38.2004    2.28 R
97     26.0000    59.8993    2.5044     -33.8993    -2.03 R
123    20.0000    69.9266    3.1302     -49.9266    -3.01 R
136    34.0000    69.9266    3.1302     -35.9266    -2.16 R
145    81.0000    47.1271    2.7732      33.8729    2.03 R
188    90.0000    48.0154    2.4102      41.9846    2.51 R

R denotes an observation with a large standardized residual.
```

K.2.3. Means for Covariates

```
Covariate    Mean   StDev
Points       256.2  66.53
```

K.2.4. Least Squares Means

```
       ------CA------      -----Exam-----
Group  Mean   SE Mean      Mean   SE Mean
A      64.15  1.543        50.96  1.540
B      56.58  1.546        52.71  1.543
```

K.3. Additional Tests Carried out on the Overall Attainment Scores of the First Cohort 2005/2006

- Group A's attainment results (for Semester 1) from both final exam and continuous assessment were compared against historical (Semester 1) attainment data for the Software Development module.

Firstly Group A was compared against the Software Development class of 04/05 overall results for Semester 1. An F-test to test the two samples for variances was carried out. As the P value (0.508) is not less than .05 we can accept that the variances are equal. Next a t-test to test two samples assuming equal variances was carried out. As the P value (0.109) on the one tail test is not less than .05 we can accept the null hypothesis that there is no significant difference between the means.

Secondly Group A was compared against the Software Development class of 03/04 overall results for Semester 1. An F-test to test the two samples for variances was carried out. As the P value (0.396) is not less than .05 we can accept that the variances are equal. A t-test to test two-samples assuming equal variances was carried out. As the P value (0.178) on the one tail test is again not less than .05 we can accept the null hypothesis that the means are the same and accept that there is no significant difference between the means.

- First year engineering students were used as a control group. They also take the Software Development module but do not use PBL. The Engineering students have very similar course entry points to the Computing students (achieved in the Irish Leaving Certificate or equivalent). Group A's attainment results (for Semester 1) from both final exam and continuous assessments were compared against the Engineering Group's results.

Again an F-test to test the two samples for variances was carried out. As the P value (0.184) is not less than .05 we can accept that the variances are equal. However, to be conservative, a t-test to test two samples assuming unequal variances was carried out. As the P value (0.045) on the one tail test is less than .05 we can reject the null hypothesis that the means are the same and accept that there is a significant difference between the means.

Note: In the above t-tests, a one tail test was used in each case, since the null hypothesis (indirectly) predicts the direction of the difference (i.e. that the PBL group will have a higher mean).

K.4. Effect Size Calculations on the Attainment Scores (Overall, Exam and CA) of the Four Cohorts 2005/2009

DATA ENTRY							RAW DIFFERENCE						STANDARDISED EFFECT SIZE					
Outcome measure	Treatment group			Control group			pooled standard deviation	p-value for difference in SDs	Mean Difference	p-value for mean diff (2-tailed T-test)	Confidence Interval for Difference		Effect Size	Bias corrected (Hedges)	Standard Error of E.S. estimate	Confidence Interval for Effect Size		Effect Size based on control gp SD
	mean	n	SD	mean	n	SD					lower	upper				lower	upper	
exam 0607	50.72	29	17.22	51.79	29	23.08	20.36	0.06	-1.07	####	####	9.64	- 0.05	**- 0.05**	0.26	-0.57	0.46	- 0.05
ca 0607	63.14	29	20.74	57.79	29	20.63	20.69	0.49	5.34	0.33	-5.54	16.23	0.26	**0.25**	0.26	-0.26	0.77	0.26
exam 0506	57.13	24	23.29	51.24	25	17.25	20.43	0.08	5.89	0.32	-5.86	17.63	0.29	**0.28**	0.29	-0.28	0.85	0.34
ca 0506	66.96	24	21.89	52.31	25	18.38	20.17	0.20	14.64	0.01	3.05	26.24	0.73	**0.71**	0.29	0.14	1.29	0.80
exam 0708	49.69	32	17.58	52.34	32	22.69	20.30	0.08	-2.66	####	####	7.49	- 0.13	**- 0.13**	0.25	-0.62	0.36	- 0.12
ca 0708	63.72	32	19.3	60.78	32	20.79	20.06	0.34	2.94	0.56	-7.09	12.96	0.15	**0.14**	0.25	-0.35	0.64	0.14
exam 0809	48.11	37	17.13	53.6	35	17.8	17.46	0.41	-5.49	####	####	2.72	- 0.31	**- 0.31**	0.24	-0.78	0.15	- 0.31
ca 0809	63.91	37	17.61	54.06	35	18.31	17.96	0.41	9.85	0.02	1.41	18.29	0.55	**0.54**	0.24	0.07	1.01	0.54
overall exam	50.91	122	18.67	52.35	121	20.18	19.44	0.20	-1.44	####	-6.35	3.47	- 0.07	**- 0.07**	0.13	-0.33	0.18	- 0.07
overall ca	64.28	122	19.5	56.37	121	19.6	19.55	0.48	7.91	0.00	2.97	12.85	0.40	**0.40**	0.13	0.15	0.66	0.40

Appendix L - Analysis of Learning Self-Regulation, Programming Self-Efficacy, Approaches to Learning, and Preferences for Types of Teaching Scores

Statistical tests were carried out using Minitab 15 and MS Excel. Descriptive statistics, t-tests and F-tests were carried out using MS Excel, while the residual gain analysis and the general linear model were carried out using Minitab 15. The results are presented below.

L.1. Tests Carried Out on the Overall and Yearly Self-Efficacy Scores of the Two Cohorts 2007/2009, Including Pre- and Post-Teaching Results

L.1.1. Overall Self-Efficacy Scores: Descriptive Statistics and t-tests

L.1.1.1. Descriptive Statistics of Pre-teaching Scores

Group A Overall Self Efficacy Pre	
Mean	123.4203
Standard Error	3.076283
Median	121
Mode	119
Standard Deviation	25.55352
Sample Variance	652.9825
Kurtosis	-0.50964
Skewness	-0.04018
Range	118
Minimum	67
Maximum	185
Sum	8516
Count	69

Group B Overall Self Efficacy Pre	
Mean	125.4478
Standard Error	2.21639
Median	126
Mode	147
Standard Deviation	18.14193
Sample Variance	329.1298
Kurtosis	-1.18428
Skewness	-0.30852
Range	57
Minimum	95
Maximum	152
Sum	8405
Count	67

L.1.1.2. Descriptive Statistics of Post-teaching Scores

Group A Overall Self Efficacy Post	
Mean	180.0435
Standard Error	2.653589
Median	183
Mode	179
Standard Deviation	22.04236
Sample Variance	485.8657
Kurtosis	1.014831
Skewness	-0.80992
Range	116
Minimum	103
Maximum	219
Sum	12423
Count	69

Group B Overall Self Efficacy Post	
Mean	145.4328
Standard Error	2.229609
Median	147
Mode	159
Standard Deviation	18.25013
Sample Variance	333.0674
Kurtosis	-0.68099
Skewness	-0.51162
Range	69
Minimum	102
Maximum	171
Sum	9744
Count	67

L.1.1.3. t-test on Pre-teaching Scores

t-Test: Two-Sample Assuming Equal Variances

	Group A Overall Self Efficacy Pre	*Group B Overall Self Efficacy Pre*
Mean	123.4202899	125.4477612
Variance	652.9825234	329.1298055
Observations	69	67
Pooled Variance	493.4729758	
Hypothesized Mean Difference	0	
df	134	
t Stat	-0.532127286	
P(T<=t) one-tail	0.29775951	
t Critical one-tail	1.656304542	
P(T<=t) two-tail	0.595519019	
t Critical two-tail	1.97782573	

L.1.1.4. t-test on Post-Teaching Scores

t-Test: Two-Sample Assuming Equal Variances

	Group A Overall Self Efficacy Post	*Group B Overall Self Efficacy Post*
Mean	180.0434783	145.4328358
Variance	485.8657289	333.0673903
Observations	69	67
Pooled Variance	410.6068457	
Hypothesized Mean Difference	0	
df	134	
t Stat	9.958388391	
P(T<=t) one-tail	3.98102E-18	
t Critical one-tail	1.656304542	
P(T<=t) two-tail	7.96205E-18	
t Critical two-tail	1.97782573	

L.1.1.5. t-tests on Group Scores

t-Test: Two-Sample Assuming Equal Variances

	Group A Overall Self Efficacy Pre	*Group A Overall Self Efficacy Post*
Mean	123.4202899	180.0434783
Variance	729.9889173	485.8657289
Observations	69	69
Pooled Variance	607.9273231	
Hypothesized Mean Difference	0	
df	136	
t Stat	-14.10349296	
P(T<=t) one-tail	1.07156E-28	
t Critical one-tail	1.656134989	

P(T<=t) two-tail	2.14313E-28
t Critical two-tail	1.977560747

t-Test: Two-Sample Assuming Equal Variances

	Group B Overall Self Efficacy Pre	*Group B Overall Self Efficacy Post*
Mean	125.4477612	145.4328358
Variance	329.1298055	333.0673903
Observations	67	67
Pooled Variance	331.0985979	
Hypothesized Mean Difference	0	
df	132	
t Stat	-6.356960414	
P(T<=t) one-tail	1.54674E-09	
t Critical one-tail	1.65647927	
P(T<=t) two-tail	3.09348E-09	
t Critical two-tail	1.978098814	

L.1.2. 07/08 Self-Efficacy Scores: Descriptive Statistics and t-tests

L.1.2.1. Descriptive Statistics of Pre-teaching Scores

Group A Self Efficacy Pre 07/08	
Mean	132.3125
Standard Error	4.913862
Median	135.5
Mode	149
Standard Deviation	27.797
Sample Variance	772.6734
Kurtosis	-0.22878
Skewness	-0.49918
Range	118
Minimum	67
Maximum	185
Sum	4234
Count	32

Group B Self Efficacy Pre 07/08	
Mean	132.4375
Standard Error	3.023076
Median	140
Mode	148
Standard Deviation	17.1011
Sample Variance	292.4476
Kurtosis	-0.23015
Skewness	-0.90583
Range	57
Minimum	95
Maximum	152
Sum	4238
Count	32

L.1.2.2. t-test on Pre-teaching Scores

t-Test: Two-Sample Assuming Equal Variances

	Group A Self Efficacy Pre 07/08	*Group B Self Efficacy Pre 07/08*
Mean	132.3125	132.4375
Variance	772.6733871	292.4475806
Observations	32	32
Pooled Variance	532.5604839	
Hypothesized Mean Difference	0	
df	62	
t Stat	-0.021666339	

P(T<=t) one-tail	0.491391847
t Critical one-tail	1.669804163
P(T<=t) two-tail	0.982783695
t Critical two-tail	1.998971498

L.1.2.3 Descriptive Statistics of Post-teaching Scores

Group A Self Efficacy Post 07/08	
Mean	181.0625
Standard Error	3.615585
Median	182
Mode	179
Standard Deviation	20.45284
Sample Variance	418.3185
Kurtosis	-0.53529
Skewness	-0.40022
Range	78
Minimum	141
Maximum	219
Sum	5794
Count	32

Group B Self Efficacy Post 07/08	
Mean	153.5313
Standard Error	2.567235
Median	156.5
Mode	147
Standard Deviation	14.52247
Sample Variance	210.9022
Kurtosis	1.558101
Skewness	-1.1061
Range	63
Minimum	108
Maximum	171
Sum	4913
Count	32

L.1.2.4. t-test on Post-teaching Scores

t-Test: Two-Sample Assuming Equal Variances

	Group A Self Efficacy Post 07/08	*Group B Self Efficacy Post 07/08*
Mean	181.0625	153.53125
Variance	418.3185484	210.9022177
Observations	32	32
Pooled Variance	314.6103831	
Hypothesized Mean Difference	0	
df	62	
t Stat	6.208681703	
P(T<=t) one-tail	2.46282E-08	
t Critical one-tail	1.669804163	
P(T<=t) two-tail	4.92564E-08	
t Critical two-tail	1.998971498	

L.1.3. 08/09 Self-Efficacy Scores, Descriptive Statistics and t-tests

L.1.3.1. Descriptive Statistics of Pre-teaching Scores

Group A Self Efficacy Pre 08/09	
Mean	115.7297
Standard Error	3.436609
Median	114
Mode	107
Standard Deviation	20.90408
Sample Variance	436.9805
Kurtosis	-0.50533
Skewness	-0.04623
Range	86
Minimum	69
Maximum	155
Sum	4282
Count	37

Group B Self Efficacy Pre 08/09	
Mean	119.0571
Standard Error	2.849721
Median	121
Mode	95
Standard Deviation	16.85918
Sample Variance	284.2319
Kurtosis	-1.22418
Skewness	0.046098
Range	52
Minimum	95
Maximum	147
Sum	4167
Count	35

L.1.3.2. t-test on Pre-teaching scores

t-Test: Two-Sample Assuming Equal Variances

	Group A Self Efficacy Pre 08/09	*Group B Self Efficacy Pre 08/09*
Mean	115.7297297	119.0571429
Variance	436.9804805	284.2319328
Observations	37	35
Pooled Variance	362.7883287	
Hypothesized Mean Difference	0	
df	70	
t Stat	-0.740881428	
P(T<=t) one-tail	0.230621735	
t Critical one-tail	1.66691448	
P(T<=t) two-tail	0.46124347	
t Critical two-tail	1.994437086	

L.1.3.3. Descriptive Statistics of Post-teaching Scores

Group A Self Efficacy Post 08/09	
Mean	179.1622
Standard Error	3.875796
Median	183
Mode	206
Standard Deviation	23.57554
Sample Variance	555.8063
Kurtosis	1.748393
Skewness	-1.03079
Range	114
Minimum	103
Maximum	217
Sum	6629
Count	37

Group B Self Efficacy Post 08/09	
Mean	138.0286
Standard Error	3.09879
Median	138
Mode	126
Standard Deviation	18.33269
Sample Variance	336.0874
Kurtosis	-1.02299
Skewness	-0.04289
Range	67
Minimum	102
Maximum	169
Sum	4831
Count	35

L.1.3.4. t-test on Post-teaching Scores

t-Test: Two-Sample Assuming Equal Variances

	Group A Self Efficacy Post 08/09	*Group B Self Efficacy Post 08/09*
Mean	179.1621622	138.0285714
Variance	555.8063063	336.087395
Observations	37	35
Pooled Variance	449.0856922	
Hypothesized Mean Difference	0	
df	70	
t Stat	8.231910676	
P(T<=t) one-tail	3.41579E-12	
t Critical one-tail	1.66691448	
P(T<=t) two-tail	6.83158E-12	
t Critical two-tail	1.994437086	

L.2. Tests Carried out on the Overall Relative Autonomy Scores of the two Cohorts 2007/2009, Including Pre- and Post-Teaching Results

L.2.1. Descriptive Statistics of Pre-teaching and Post-teaching scores

Group A Overall Relative Autonomy Index Pre	
Mean	11.08695652
Standard Error	1.021193315
Median	12
Mode	12
Standard Deviation	8.482668756
Sample Variance	71.95566922
Kurtosis	-0.641952234
Skewness	-0.131114926
Range	34
Minimum	-5
Maximum	29
Sum	758
Count	69

Group B Overall Relative Autonomy Index Pre	
Mean	12.3880597
Standard Error	1.097457121
Median	13
Mode	23
Standard Deviation	8.983073686
Sample Variance	80.69561284
Kurtosis	-1.084322285
Skewness	-0.3076217
Range	32
Minimum	-5
Maximum	27
Sum	830
Count	67

Group A Overall Relative Autonomy Index Post	
Mean	13.985507
Standard Error	1.1588447
Median	13
Mode	11
Standard Deviation	9.626087
Sample Variance	92.661552
Kurtosis	-0.156612
Skewness	0.1394151
Range	43
Minimum	-6
Maximum	37

Group B Overall Relative Autonomy Index Post	
Mean	11.73134
Standard Error	1.19201
Median	12
Mode	6
Standard Deviation	9.757021
Sample Variance	95.19946
Kurtosis	-0.74435
Skewness	-0.00865
Range	42
Minimum	-9
Maximum	33

Sum	965	Sum	786
Count	69	Count	67

L.2.2. t-tests on Pre-teaching and Post-teaching Scores

t-Test: Two-Sample Assuming Equal Variances

	Group A Overall Relative Autonomy Index Pre	*Group B Overall Relative Autonomy Index Pre*
Mean	11.08695652	12.3880597
Variance	77.10997442	80.69561284
Observations	69	67
Pooled Variance	78.87603514	
Hypothesized Mean Difference	0	
df	134	
t Stat	-0.854145181	
P(T<=t) one-tail	0.197274477	
t Critical one-tail	1.656304542	
P(T<=t) two-tail	0.394548955	
t Critical two-tail	1.97782573	

t-Test: Two-Sample Assuming Equal Variances

	Group A Overall Relative Autonomy Index Post	*Group B Overall Relative Autonomy Index Post*
Mean	13.98550725	11.73134328
Variance	92.66155158	95.19945726
Observations	69	67
Pooled Variance	93.91156482	
Hypothesized Mean Difference	0	
df	134	
t Stat	1.356183501	
P(T<=t) one-tail	0.088660801	
t Critical one-tail	1.656304542	
P(T<=t) two-tail	0.177321602	
t Critical two-tail	1.97782573	

t-Test: Two-Sample Assuming Equal Variances

	Group A Overall Relative Autonomy Index Pre	*Group A Overall Relative Autonomy Index Post*
Mean	11.08695652	13.98550725
Variance	71.95566922	92.66155158
Observations	69	69
Pooled Variance	82.3086104	
Hypothesized Mean Difference	0	
df	136	
t Stat	-1.942263584	
P(T<=t) one-tail	0.027085978	
t Critical one-tail	1.656134989	
P(T<=t) two-tail	0.054171957	
t Critical two-tail	1.977560747	

t-Test: Two-Sample Assuming Equal Variances

	Group B Overall Relative Autonomy Index Pre	*Group B Overall Relative Autonomy Index Post*
Mean	12.3880597	11.73134328
Variance	80.69561284	95.19945726
Observations	67	67
Pooled Variance	87.94753505	
Hypothesized Mean Difference	0	
df	132	
t Stat	0.405311048	
P(T<=t) one-tail	0.342952519	
t Critical one-tail	1.65647927	
P(T<=t) two-tail	0.685905039	
t Critical two-tail	1.978098814	

L.3. Tests Carried Out on the Overall Approaches to Learning Scores of the 2008/2009 Cohort, including pre and post-teaching results

L.3.1. Tests Carried Out on the Overall Deep Approach to Learning Scores of the 2008/2009 Cohort, Including Pre- and Post-Teaching Results

L.3.1.1. Descriptive Statistics of Deep Approach Pre-teaching and Post-teaching Scores

Overall Group A Deep Approach Pre	
Mean	57.13043478
Standard Error	1.057170749
Median	58
Mode	64
Standard Deviation	8.781519772
Sample Variance	77.11508951
Kurtosis	0.172967773
Skewness	-0.571836026
Range	42
Minimum	34
Maximum	76
Sum	2114
Count	37

Overall Group B Deep Approach Pre	
Mean	57.20895522
Standard Error	0.929816008
Median	56
Mode	59
Standard Deviation	7.610872035
Sample Variance	57.92537313
Kurtosis	1.356553438
Skewness	-0.015770148
Range	40
Minimum	35
Maximum	75
Sum	2002
Count	35

L.3.1.2. t-tests on Deep Approach Pre-teaching and Post-teaching Scores

t-Test: Two-Sample Assuming Equal Variances

	Overall Group A Deep Approach Pre	*Overall Group B Deep Approach Pre*
Mean	57.13043478	57.20895522
Variance	77.11508951	57.92537313
Observations	37	35
Pooled Variance	67.66343816	

Hypothesized Mean Difference	0	
df	70	
t Stat	-0.055654232	
P(T<=t) one-tail	0.477850078	
t Critical one-tail	1.656304542	
P(T<=t) two-tail	0.955700157	
t Critical two-tail	1.97782573	

t-Test: Two-Sample Assuming Equal Variances

	Overall Group A Deep Approach Post	*Overall Group B Deep Approach Post*
Mean	58.89855072	54.59701493
Variance	167.3572038	128.3048394
Observations	37	35
Pooled Variance	148.1224571	
Hypothesized Mean Difference	0	
df	70	
t Stat	2.060656391	
P(T<=t) one-tail	0.020635116	
t Critical one-tail	1.656304542	
P(T<=t) two-tail	0.041270232	
t Critical two-tail	1.97782573	

t-Test: Two-Sample Assuming Equal Variances

	Overall Group A Deep Approach Pre	*Overall Group A Deep Approach Post*
Mean	57.13043478	58.89855072
Variance	85.00809889	167.3572038
Observations	37	35
Pooled Variance	126.1826513	
Hypothesized Mean Difference	0	
df	70	
t Stat	-1.091247156	
P(T<=t) one-tail	0.138546827	
t Critical one-tail	1.656134989	
P(T<=t) two-tail	0.277093654	
t Critical two-tail	1.977560747	

t-Test: Two-Sample Assuming Equal Variances

	Overall Group B Deep Approach Pre	*Overall Group B Deep Approach Post*
Mean	57.20895522	54.59701493
Variance	57.92537313	128.3048394
Observations	67	67
Pooled Variance	93.11510629	
Hypothesized Mean Difference	0	
df	132	
t Stat	1.566662968	

P(T<=t) one-tail	0.059793842
t Critical one-tail	1.65647927
P(T<=t) two-tail	0.119587683
t Critical two-tail	1.978098814

L.3.2. Tests Carried Out on the Overall Strategic Approach to Learning Scores of the 2008/2009 Cohort, Including Pre- and Post-teaching Results

L.3.2.1. Descriptive Statistics of Strategic Approach Pre-teaching and Post-teaching Scores

Overall Group A Strategic Approach Pre	
Mean	71
Standard Error	1.156544814
Median	71
Mode	68
Standard Deviation	9.606982755
Sample Variance	92.29411765
Kurtosis	0.10306458
Skewness	0.259796656
Range	47
Minimum	49
Maximum	96
Sum	2627
Count	37

Overall Group B Strategic Approach Pre	
Mean	69.86567164
Standard Error	0.739740784
Median	70
Mode	62
Standard Deviation	6.05503928
Sample Variance	36.66350068
Kurtosis	-1.037097221
Skewness	0.043919905
Range	21
Minimum	60
Maximum	81
Sum	2445
Count	35

Overall Group A Strategic Approach Post	
Mean	66.05797101
Standard Error	1.610372927
Median	65
Mode	79
Standard Deviation	13.37676218
Sample Variance	178.9377664
Kurtosis	0.101111177
Skewness	0.476286249
Range	65
Minimum	39
Maximum	104
Sum	2444
Count	37

Overall Group B Strategic Approach Post	
Mean	71.01492537
Standard Error	1.299983046
Median	73
Mode	73
Standard Deviation	10.64081983
Sample Variance	113.2270466
Kurtosis	-0.562200636
Skewness	-0.163531171
Range	49
Minimum	48
Maximum	97
Sum	2486
Count	35

L.3.2.2. t-tests on Strategic Approach Pre-teaching and Post-teaching scores

t-Test: Two-Sample Assuming Equal Variances

	Overall Group A Strategic Approach Pre	*Overall Group B Strategic Approach Pre*
Mean	71	69.86567164
Variance	92.29411765	36.66350068
Observations	37	35
Pooled Variance	64.89396302	
Hypothesized Mean Difference	0	

df	70
t Stat	0.82097349
P(T<=t) one-tail	0.206559762
t Critical one-tail	1.656304542
P(T<=t) two-tail	0.413119524
t Critical two-tail	1.97782573

t-Test: Two-Sample Assuming Equal Variances

	Overall Group A Strategic Approach Post	*Overall Group B Strategic Approach Post*
Mean	66.05797101	71.01492537
Variance	178.9377664	113.2270466
Observations	37	35
Pooled Variance	146.572785	
Hypothesized Mean Difference	0	
df	70	
t Stat	-2.38715568	
P(T<=t) one-tail	0.00918791	
t Critical one-tail	1.656304542	
P(T<=t) two-tail	0.01837582	
t Critical two-tail	1.97782573	

t-Test: Two-Sample Assuming Equal Variances

	Overall Group A Strategic Approach Pre	*Overall Group A Strategic Approach Post*
Mean	71	66.05797101
Variance	117.5029838	178.9377664
Observations	69	69
Pooled Variance	148.2203751	
Hypothesized Mean Difference	0	
df	70	
t Stat	2.041687897	
P(T<=t) one-tail	0.021557939	
t Critical one-tail	1.656134989	
P(T<=t) two-tail	0.043115879	
t Critical two-tail	1.977560747	

t-Test: Two-Sample Assuming Equal Variances

	Overall Group B Strategic Approach Pre	*Overall Group B Strategic Approach Post*
Mean	69.86567164	71.01492537
Variance	36.66350068	113.2270466
Observations	67	67
Pooled Variance	74.94527363	
Hypothesized Mean Difference	0	
df	70	
t Stat	-0.768362571	
P(T<=t) one-tail	0.221822137	
t Critical one-tail	1.65647927	

P(T<=t) two-tail	0.443644275
t Critical two-tail	1.978098814

L.3.3. Tests Carried Out on the Overall Surface Apathetic Approach to Learning Scores of the 2008/2009 Cohort, Including Pre- and Post-teaching results

L.3.3.1. Descriptive Statistics of Surface Apathetic Approach Pre-teaching and Post-teaching Scores

Overall Group A Surface Apathetic Approach Pre		*Overall Group B Surface Apathetic Approach Pre*	
Mean	46.20289855	Mean	46.79104478
Standard Error	1.224323623	Standard Error	0.795321381
Median	47	Median	47
Mode	56	Mode	53
Standard Deviation	10.16999582	Standard Deviation	6.509986067
Sample Variance	103.428815	Sample Variance	42.37991859
Kurtosis	-0.791828787	Kurtosis	-1.087905414
Skewness	-0.210033122	Skewness	-0.246923749
Range	42	Range	22
Minimum	26	Minimum	35
Maximum	68	Maximum	57
Sum	1710	Sum	1638
Count	37	Count	35

Overall Group A Surface Apathetic Approach Post		*Overall Group B Surface Apathetic Approach Post*	
Mean	40.10144928	Mean	47.62686567
Standard Error	1.406073922	Standard Error	0.970351021
Median	42	Median	46
Mode	44	Mode	46
Standard Deviation	11.67972719	Standard Deviation	7.94266542
Sample Variance	136.4160273	Sample Variance	63.08593397
Kurtosis	-0.73501194	Kurtosis	-0.163949852
Skewness	-0.301549907	Skewness	-0.043346088
Range	45	Range	33
Minimum	14	Minimum	32
Maximum	59	Maximum	65
Sum	1484	Sum	1667
Count	37	Count	35

L.3.3.2. t-tests on Surface Apathetic Approach Pre-teaching and Post-teaching Scores

t-Test: Two-Sample Assuming Equal Variances

	Overall Group A Surface Apathetic Approach Pre	*Overall Group B Surface Apathetic Approach Pre*
Mean	46.20289855	46.79104478

Variance	103.428815	42.37991859
Observations	37	35
Pooled Variance	73.35995558	
Hypothesized Mean Difference	0	
df	70	
t Stat	-0.400357784	
P(T<=t) one-tail	0.344765284	
t Critical one-tail	1.656304542	
P(T<=t) two-tail	0.689530568	
t Critical two-tail	1.97782573	

t-Test: Two-Sample Assuming Equal Variances

	Overall Group A Surface Apathetic Approach Post	*Overall Group B Surface Apathetic Approach Post*
Mean	40.10144928	47.62686567
Variance	136.4160273	63.08593397
Observations	37	35
Pooled Variance	100.2982201	
Hypothesized Mean Difference	0	
df	70	
t Stat	-4.381031893	
P(T<=t) one-tail	1.18173E-05	
t Critical one-tail	1.656304542	
P(T<=t) two-tail	2.36346E-05	
t Critical two-tail	1.97782573	

t-Test: Two-Sample Assuming Equal Variances

	Overall Group A Surface Apathetic Approach Pre	*Overall Group A Surface Apathetic Approach Post*
Mean	46.20289855	40.10144928
Variance	113.5029838	136.4160273
Observations	69	69
Pooled Variance	124.9595055	
Hypothesized Mean Difference	0	
df	70	
t Stat	3.426796554	
P(T<=t) one-tail	0.000404095	
t Critical one-tail	1.656134989	
P(T<=t) two-tail	0.00080819	
t Critical two-tail	1.977560747	

t-Test: Two-Sample Assuming Equal Variances

	Overall Group B Surface Apathetic Approach Pre	*Overall Group B Surface Apathetic Approach Post*
Mean	46.79104478	47.62686567
Variance	42.37991859	63.08593397

Observations	67	67
Pooled Variance	52.73292628	
Hypothesized Mean Difference	0	
df	70	
t Stat	-0.666184758	
P(T<=t) one-tail	0.253227811	
t Critical one-tail	1.65647927	
P(T<=t) two-tail	0.506455623	
t Critical two-tail	1.978098814	

L.4. Tests Carried Out on the Overall Preferences for Types of Teaching Scores of the 2008/2009 Cohort, Including Pre- and Post-teaching Results

L.4.1. Tests Carried Out on the Overall Supporting Understanding Teaching Scores of the 2008/2009 Cohort, Including Pre- and Post Teaching Results

L.4.1.1. Descriptive Statistics of Supporting Understanding Pre-teaching and Post-Teaching Scores

Group A Supporting Understanding Pre	
Mean	13.97101449
Standard Error	0.330961949
Median	14
Mode	16
Standard Deviation	2.749176425
Sample Variance	7.557971014
Kurtosis	-0.502991024
Skewness	-0.226906751
Range	12
Minimum	8
Maximum	20
Sum	517
Count	37

Group B Supporting Understanding Pre	
Mean	14.537313
Standard Error	0.2263992
Median	15
Mode	15
Standard Deviation	1.8531575
Sample Variance	3.4341927
Kurtosis	-1.0489293
Skewness	-0.3920331
Range	6
Minimum	11
Maximum	17
Sum	509
Count	35

Group A Supporting Understanding Post	
Mean	15.42028986
Standard Error	0.454871237
Median	16
Mode	12
Standard Deviation	3.778444269
Sample Variance	14.27664109
Kurtosis	-0.507909959
Skewness	-0.06238063
Range	18
Minimum	6
Maximum	24
Sum	571
Count	37

Group B Supporting Understanding Post	
Mean	14.223881
Standard Error	0.346737
Median	15
Mode	15
Standard Deviation	2.8381647
Sample Variance	8.0551787
Kurtosis	-1.0358511
Skewness	-0.1358235
Range	10
Minimum	9
Maximum	19
Sum	498
Count	35

L.4.1.2. t-tests on Supporting Understanding Pre-teaching and Post-teaching Scores

t-Test: Two-Sample Assuming Equal Variances

	Group A Supporting Understanding Pre	*Group B Supporting Understanding Pre*
Mean	13.97101449	14.53731343
Variance	7.557971014	3.434192673
Observations	37	35
Pooled Variance	5.526856309	
Hypothesized Mean Difference	0	
df	70	
t Stat	-1.404426949	
P(T<=t) one-tail	0.081252708	
t Critical one-tail	1.656304542	
P(T<=t) two-tail	0.162505417	
t Critical two-tail	1.97782573	

t-Test: Two-Sample Assuming Equal Variances

	Group A Supporting Understanding Post	*Group B Supporting Understanding Post*
Mean	15.42028986	14.2238806
Variance	14.27664109	8.055178652
Observations	37	35
Pooled Variance	11.2123387	
Hypothesized Mean Difference	0	
df	70	
t Stat	2.083167318	
P(T<=t) one-tail	0.019568222	
t Critical one-tail	1.656304542	
P(T<=t) two-tail	0.039136444	
t Critical two-tail	1.97782573	

t-Test: Two-Sample Assuming Equal Variances

	Group A Supporting Understanding Pre	*Group A Supporting Understanding Post*
Mean	13.97101449	15.42028986
Variance	7.557971014	14.27664109
Observations	37	35
Pooled Variance	10.91730605	
Hypothesized Mean Difference	0	
df	70	
t Stat	-2.576337258	
P(T<=t) one-tail	0.005525996	
t Critical one-tail	1.656134989	
P(T<=t) two-tail	0.011051992	
t Critical two-tail	1.977560747	

t-Test: Two-Sample Assuming Equal Variances

	Group B Supporting Understanding Pre	*Group B Supporting Understanding Post*
Mean	14.53731343	14.2238806
Variance	3.434192673	8.055178652
Observations	37	35
Pooled Variance	5.744685663	
Hypothesized Mean Difference	0	
df	70	
t Stat	0.756891731	
P(T<=t) one-tail	0.225231802	
t Critical one-tail	1.65647927	
P(T<=t) two-tail	0.450463605	
t Critical two-tail	1.978098814	

L.4.2. Tests Carried Out on the Overall Transforming Information Teaching Scores of the 2008/2009 Cohort, Including Pre- and Post-teaching Results

L.4.2.1. Descriptive Statistics of Transforming Information Pre-teaching and Post-Teaching Scores

Group A Transforming Information Pre	
Mean	16.92753623
Standard Error	0.338190638
Median	17
Mode	17
Standard Deviation	2.80922242
Sample Variance	7.891730605
Kurtosis	1.992382167
Skewness	-1.300904502
Range	13
Minimum	8
Maximum	21
Sum	625
Count	37

Group B Transforming Information Pre	
Mean	17.626866
Standard Error	0.5125454
Median	17
Mode	20
Standard Deviation	4.1953648
Sample Variance	17.601085
Kurtosis	36.884943
Skewness	5.2474721
Range	34
Minimum	13
Maximum	47
Sum	617
Count	35

Group A Transforming Information Post	
Mean	15.79710145
Standard Error	0.479138413
Median	16
Mode	16
Standard Deviation	3.980022577
Sample Variance	15.84057971
Kurtosis	-0.324334342
Skewness	-0.317032795
Range	17
Minimum	6
Maximum	23

Group B Transforming Information Post	
Mean	17.820896
Standard Error	0.2629937
Median	18
Mode	17
Standard Deviation	2.1526965
Sample Variance	4.6341022
Kurtosis	-0.6078111
Skewness	0.2484895
Range	8
Minimum	14
Maximum	22

Sum	585	Sum	624
Count	37	Count	35

L.4.2.2. t-tests on Transforming Information Pre-teaching and Post-teaching Scores

t-Test: Two-Sample Assuming Equal Variances

	Group A Transforming Information Pre	*Group B Transforming Information Pre*
Mean	16.92753623	17.62686567
Variance	7.891730605	17.60108548
Observations	37	35
Pooled Variance	12.67395017	
Hypothesized Mean Difference	0	
df	70	
t Stat	-1.145297482	
P(T<=t) one-tail	0.127063667	
t Critical one-tail	1.656304542	
P(T<=t) two-tail	0.254127334	
t Critical two-tail	1.97782573	

t-Test: Two-Sample Assuming Equal Variances

	Group A Transforming Information Post	*Group B Transforming Information Post*
Mean	15.79710145	17.82089552
Variance	15.84057971	4.634102216
Observations	37	35
Pooled Variance	10.32097139	
Hypothesized Mean Difference	0	
df	70	
t Stat	-3.672810532	
P(T<=t) one-tail	0.000172805	
t Critical one-tail	1.656304542	
P(T<=t) two-tail	0.000345611	
t Critical two-tail	1.97782573	

t-Test: Two-Sample Assuming Equal Variances

	Group A Transforming Information Pre	*Group A Transforming Information Post*
Mean	16.92753623	15.79710145
Variance	7.891730605	15.84057971
Observations	37	35
Pooled Variance	11.86615516	
Hypothesized Mean Difference	0	
df	70	
t Stat	1.927525127	
P(T<=t) one-tail	0.027999473	
t Critical one-tail	1.656134989	
P(T<=t) two-tail	0.055998947	
t Critical two-tail	1.977560747	

t-Test: Two-Sample Assuming Equal Variances

	Group B Transforming Information Pre	*Group B Transforming Information Post*
Mean	17.62686567	17.82089552
Variance	17.60108548	4.634102216
Observations	37	35
Pooled Variance	11.11759385	
Hypothesized Mean Difference	0	
df	70	
t Stat	-0.336810445	
P(T<=t) one-tail	0.368397349	
t Critical one-tail	1.65647927	
P(T<=t) two-tail	0.736794698	
t Critical two-tail	1.978098814	

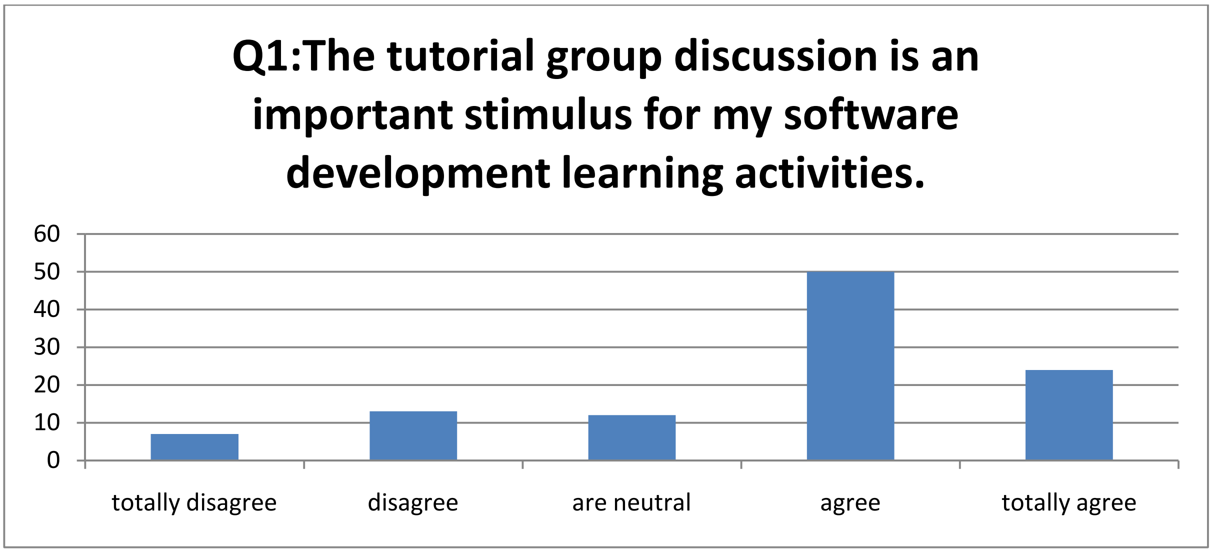
L.5. Effect Size Calculations of the Learning Self-Regulation, Programming Self-Efficacy, Approaches to Learning, and Preferences for Types of Teaching Scores

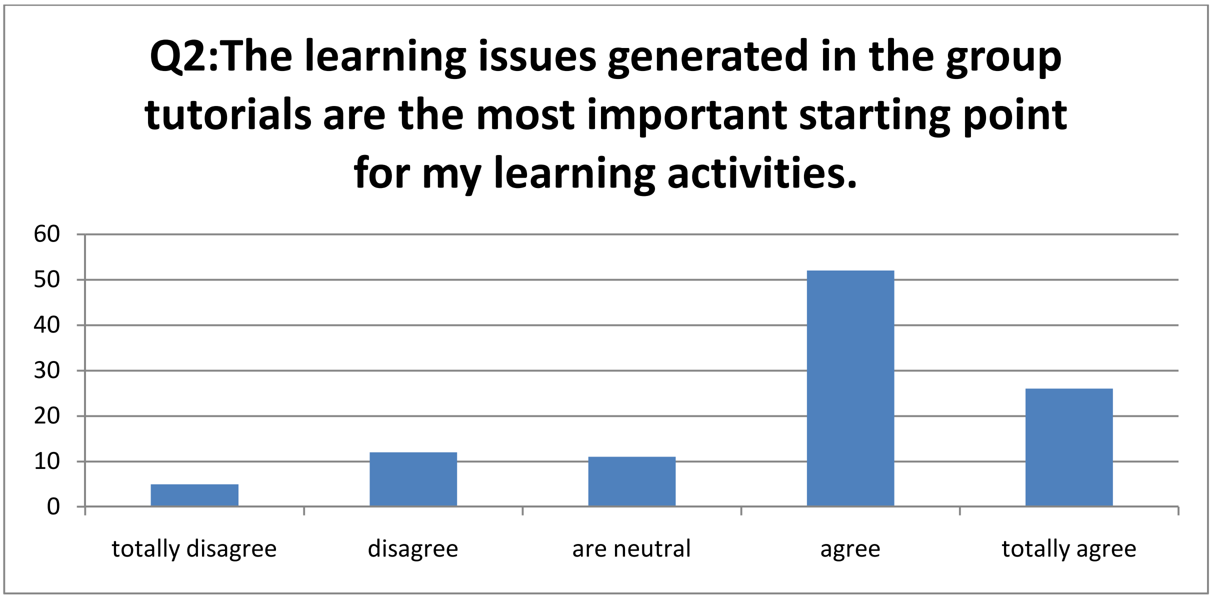
DATA ENTRY							RAW DIFFERENCE						STANDARDISED EFFECT SIZE					
Outcome measure	Treatment group			Control group			pooled standard deviation	p-value for difference in SDs	Mean Difference	p-value for mean diff (2-tailed T-test)	Confidence Interval for Difference		Effect Size	Bias corrected (Hedges)	Standard Error of E.S. estimate	Confidence Interval for Effect Size		Effect Size based on control gp SD
	mean	n	SD	mean	n	SD					lower	upper				lower	upper	
Self Efficacy post overall	180	69	22	145	67	18	20.26	0.06	34.61	0.00	27.74	41.48	1.71	**1.70**	0.20	1.31	2.09	1.90
Self Efficacy 0708	181	32	20	154	32	15	17.74	0.03	27.53	0.00	18.67	36.40	1.55	**1.53**	0.28	0.98	2.09	1.90
Self Efficacy 0809	179	37	24	138	35	18	21.19	0.07	41.13	0.00	31.17	51.10	1.94	**1.92**	0.28	1.36	2.48	2.24
Relative Autonomy Index	14	69	10	12	67	10	9.69	0.46	2.25	0.18	-1.03	5.54	0.23	**0.23**	0.17	-0.11	0.57	0.23
deep	59	37	13	55	35	11	12.17	0.14	4.30	0.04	0.17	8.43	0.35	**0.35**	0.17	0.01	0.69	0.38
strategic	66	37	13	71	35	11	12.11	0.03	-4.96	####	-9.06	-0.85	-0.41	**-0.41**	0.17	-0.75	-0.07	-0.47
surface apathetic	40	37	12	48	35	8	10.01	0.00	-7.53	####	####	-4.13	-0.75	**-0.75**	0.18	-1.09	-0.40	-0.95
Supporting Understanding	15	37	4	14	35	3	3.35	0.01	1.20	0.04	0.06	2.33	0.36	**0.36**	0.17	0.02	0.69	0.42
Transforming Information	16	37	4	18	35	2	3.21	0.00	-2.02	####	-3.11	-0.93	-0.63	**-0.63**	0.18	-0.97	-0.28	-0.94

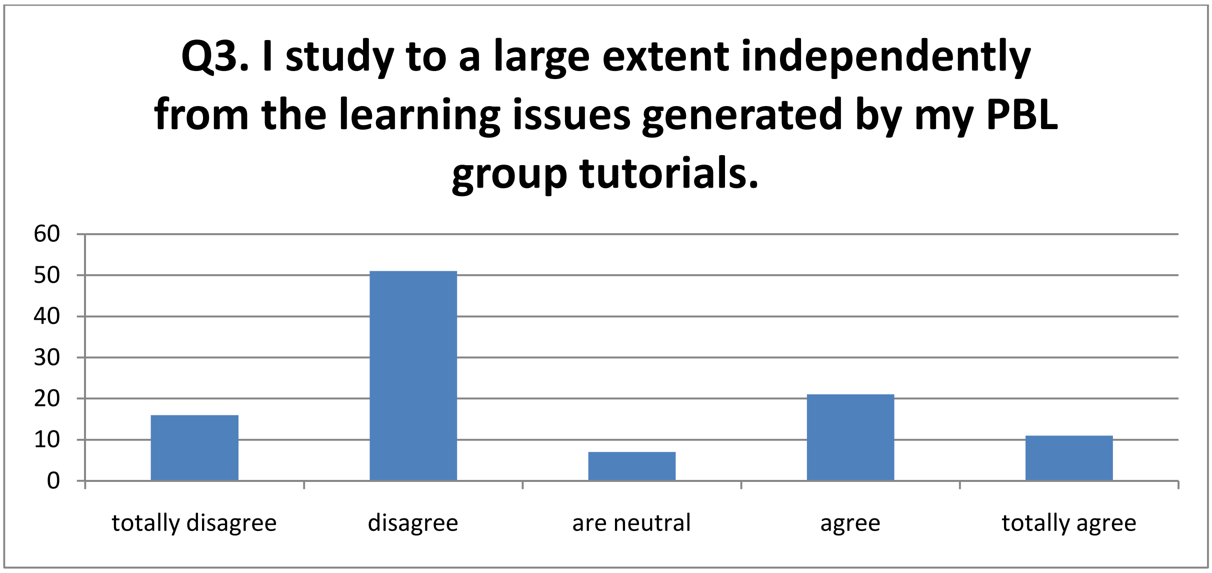
Appendix M : Results of the PBL Questionnaire for Students

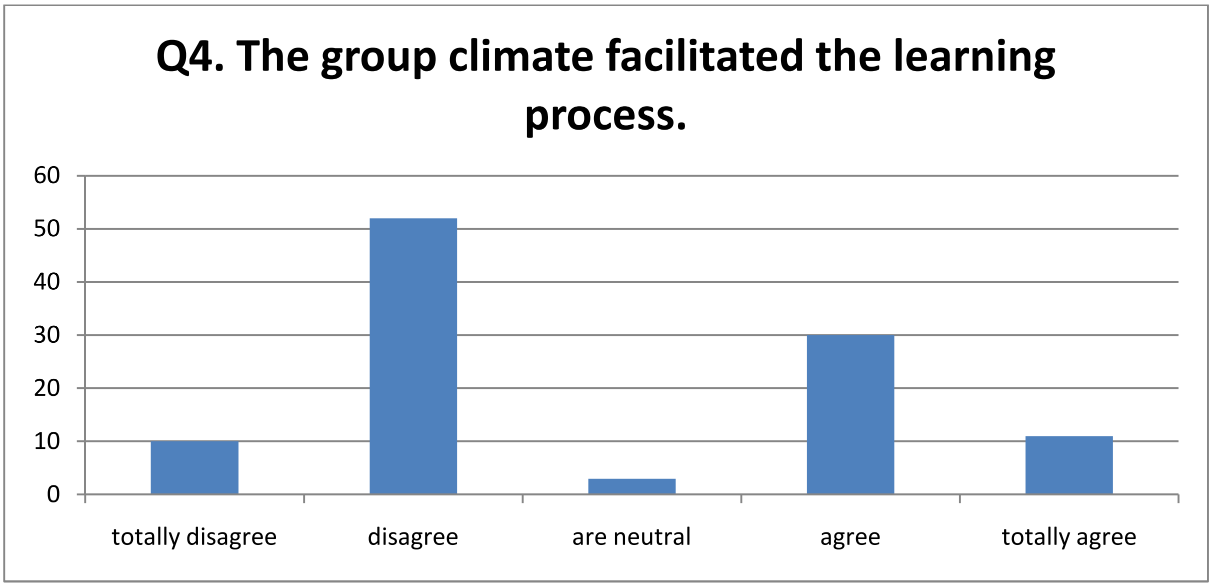
All graphs show 106 respondents.

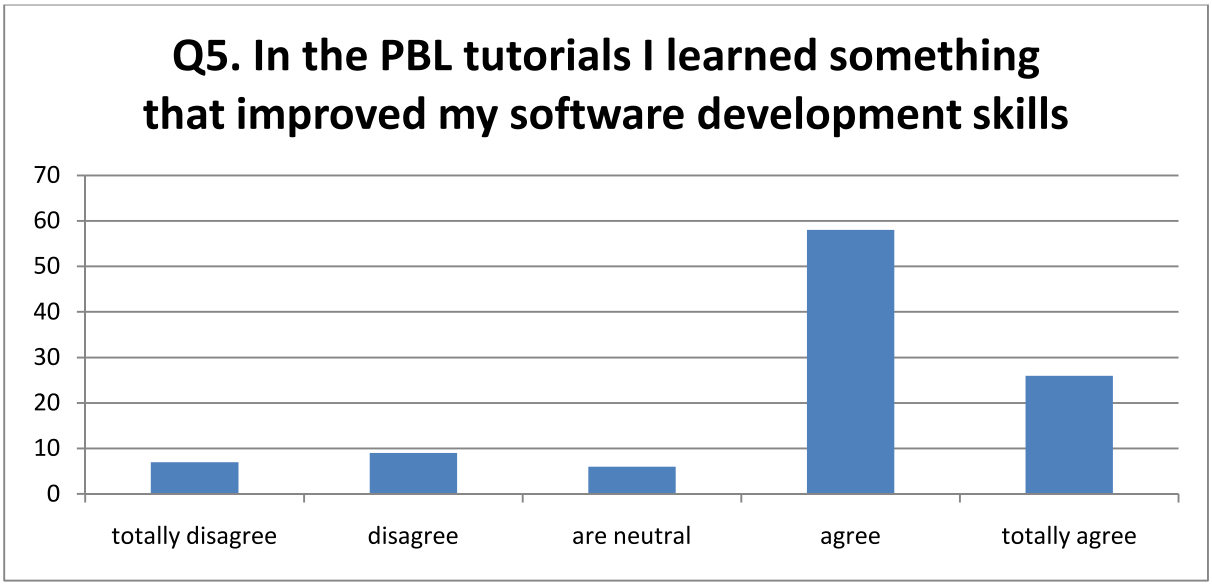
M.1. The PBL Group

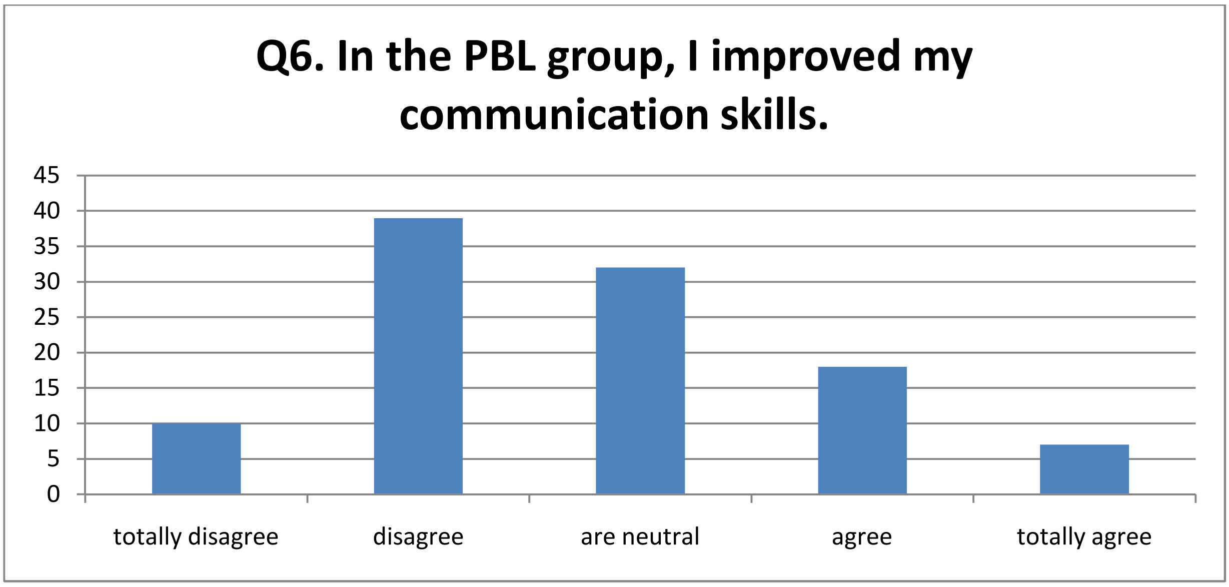


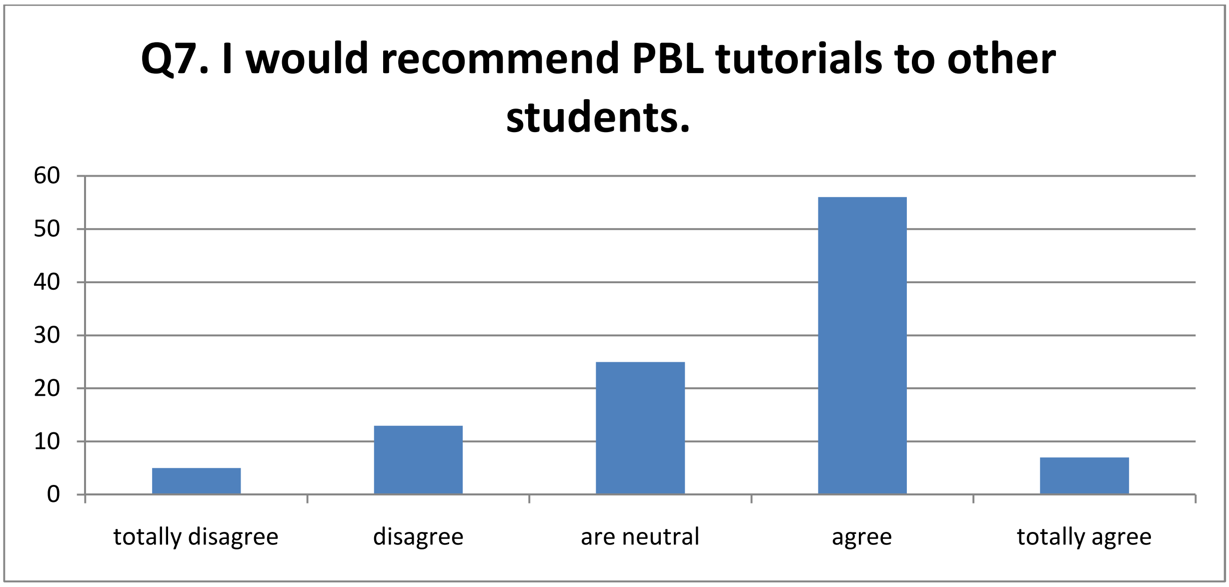



Q3. I study to a large extent independently from the learning issues generated by my PBL group tutorials.
60
50
40
30
20
10
0
totally disagree
disagree
are neutral
agree
totally agree

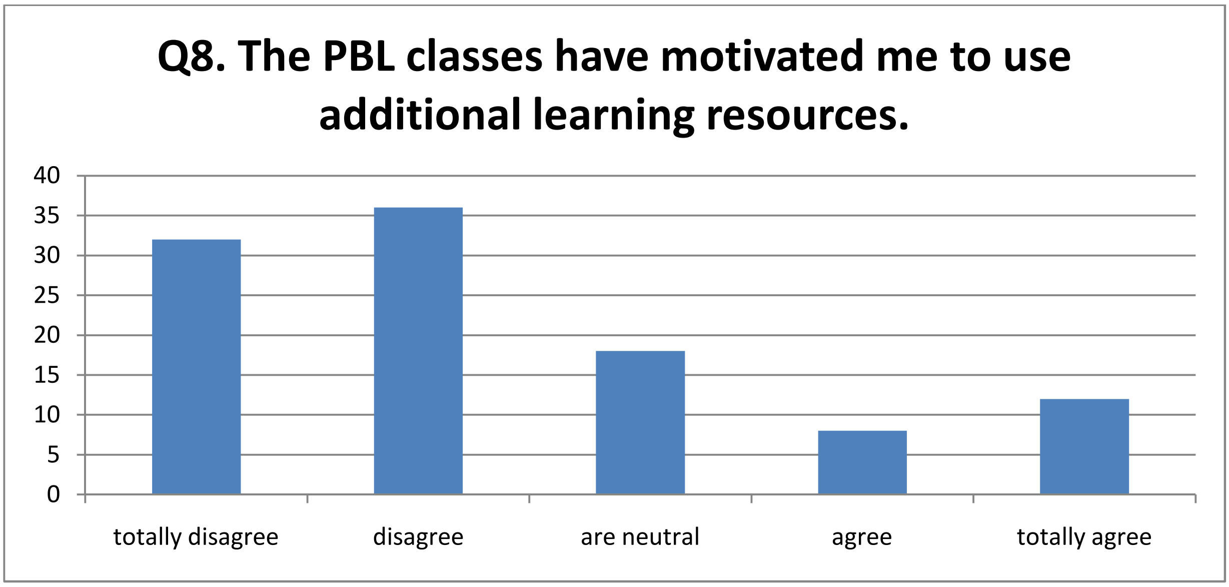

Q4. The group climate facilitated the learning process.
60
50
40
30
20
10
0
totally disagree
disagree
are neutral
agree
totally agree

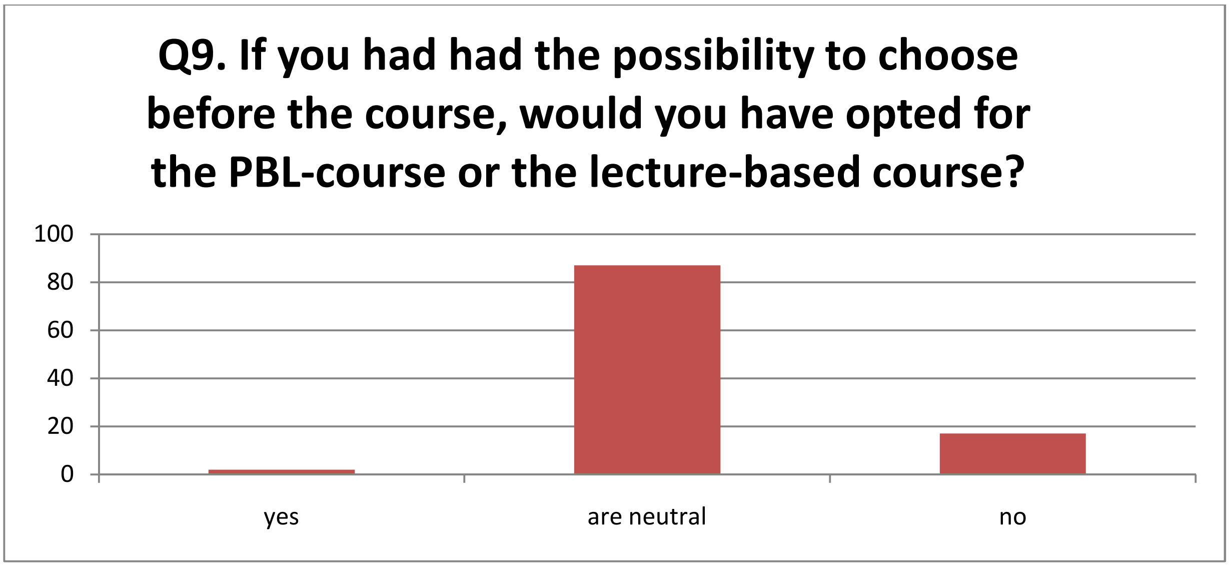

Q5. In the PBL tutorials I learned something that improved my software development skills
70
60
50
40
30
20
10
0
totally disagree
disagree
are neutral
agree
totally agree

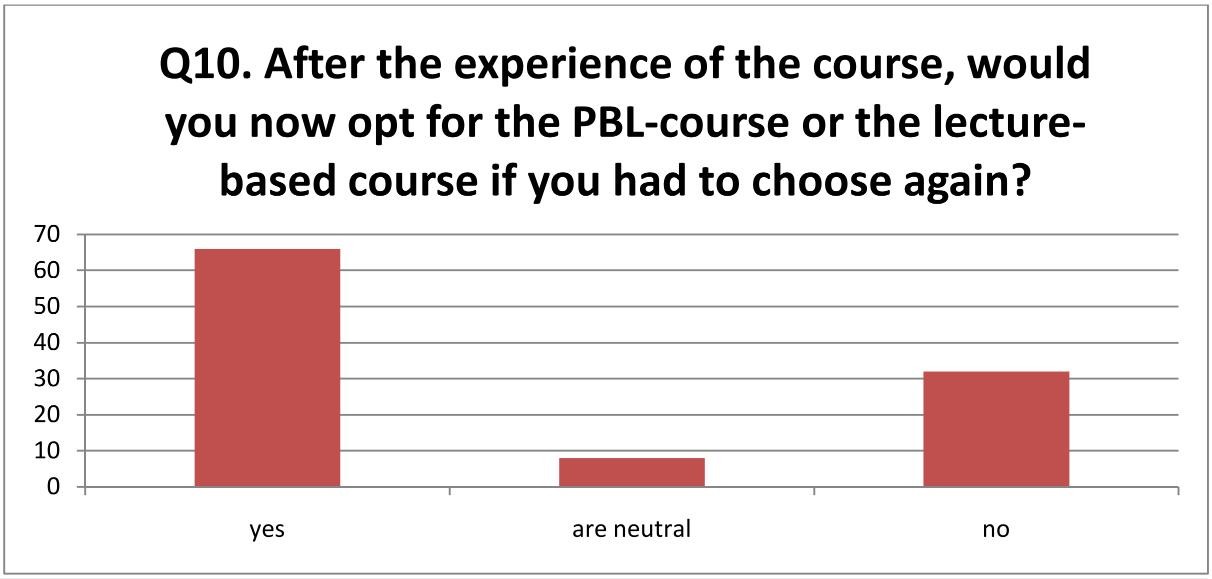

Q6. In the PBL group, I improved my communication skills.
45
40
35
30
25
20
15
10
5
0
totally disagree
disagree
are neutral
agree
totally agree

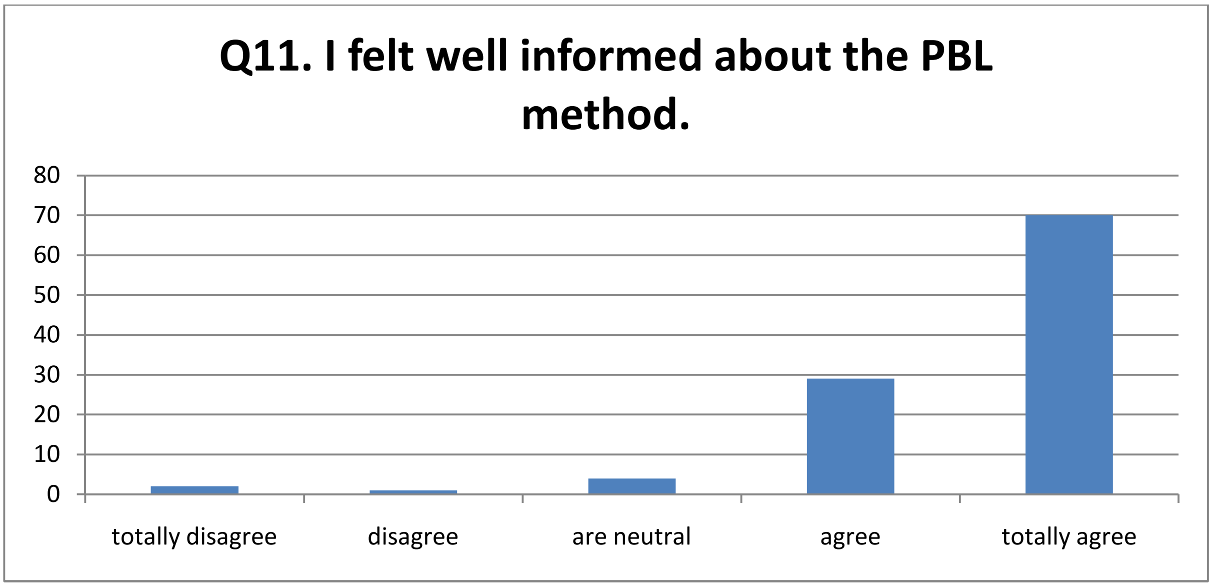

Q7. I would recommend PBL tutorials to other students.
60
50
40
30
20
10
0
totally disagree
disagree
are neutral
agree
totally agree

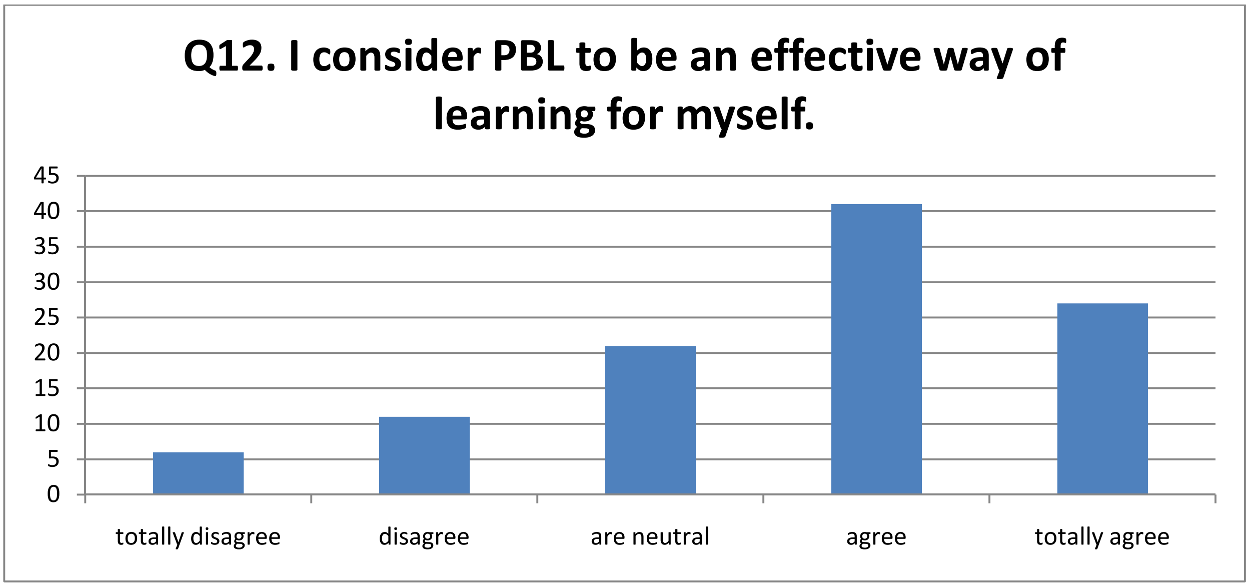
M.2. The PBL Method

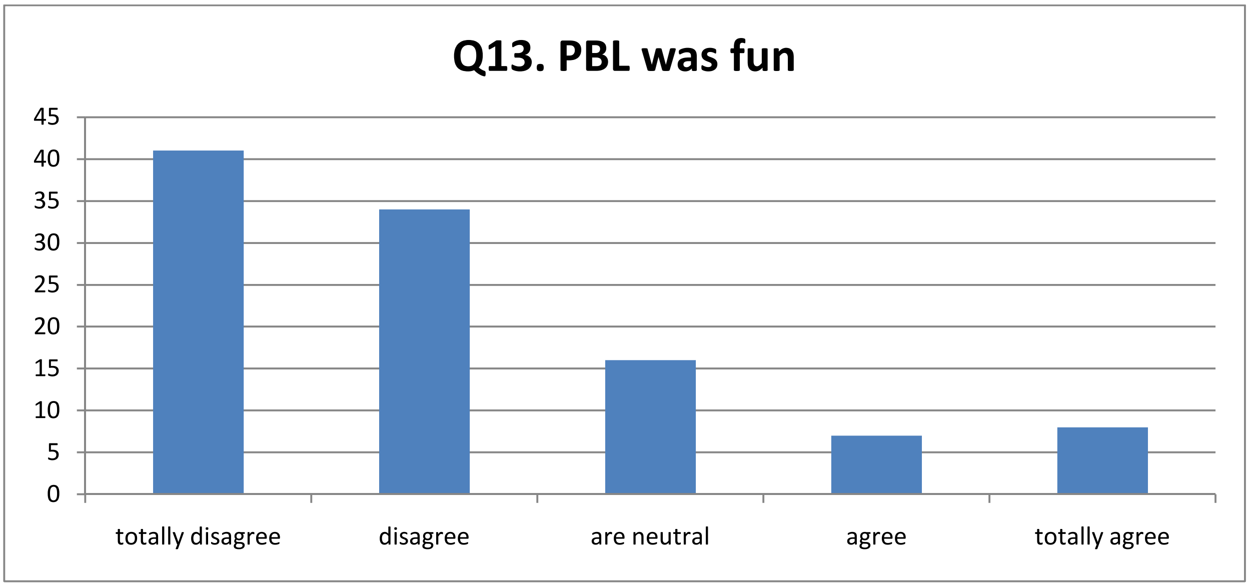


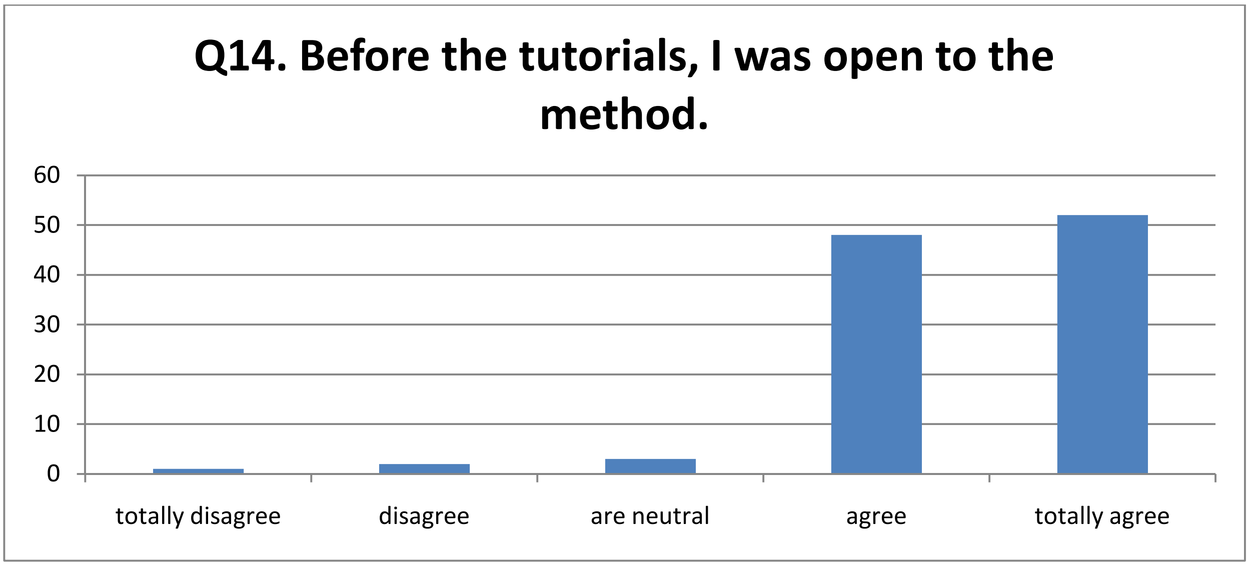




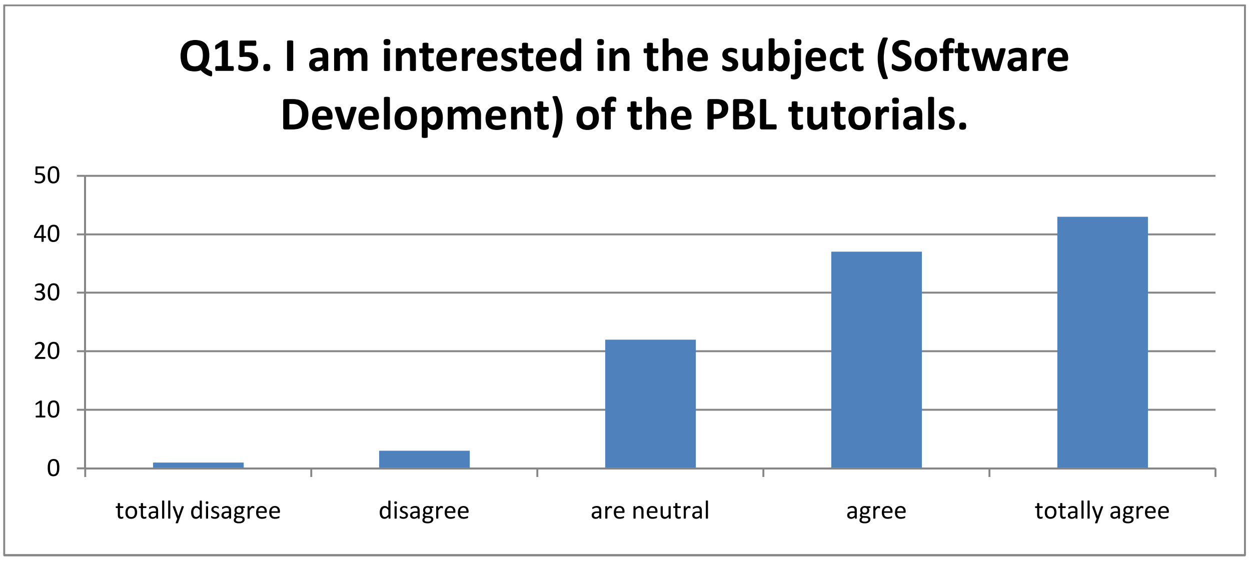

Q11. I felt well informed about the PBL method.
80
70
60
50
40
30
20
10
0
totally disagree
disagree
are neutral
agree
totally agree

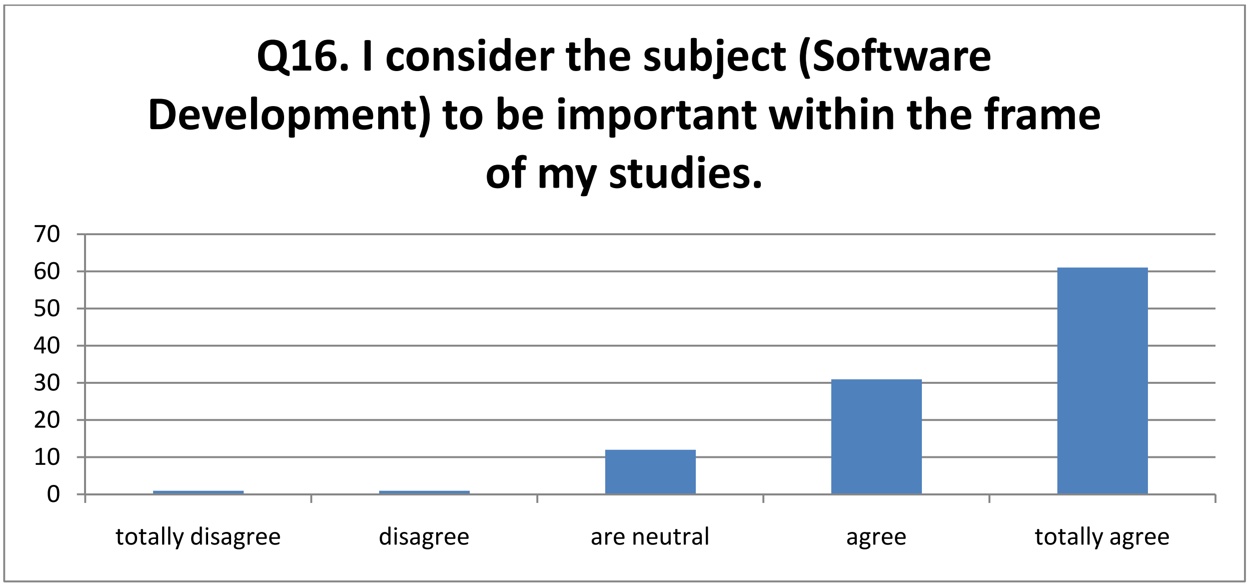

Q12. I consider PBL to be an effective way of learning for myself.
45
40
35
30
25
20
15
10
5
0
totally disagree
disagree
are neutral
agree
totally agree


Q13. PBL was fun
45
40
35
30
25
20
15
10
5
0
totally disagree
disagree
are neutral
agree
totally agree



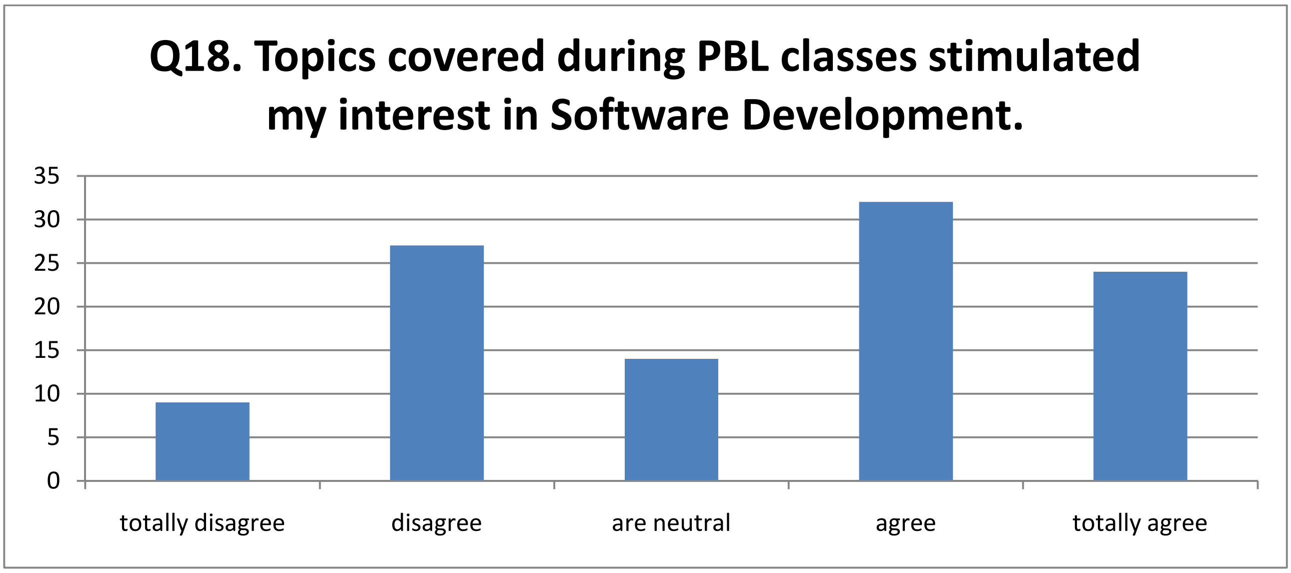
M.3. Student Interest in Software Development

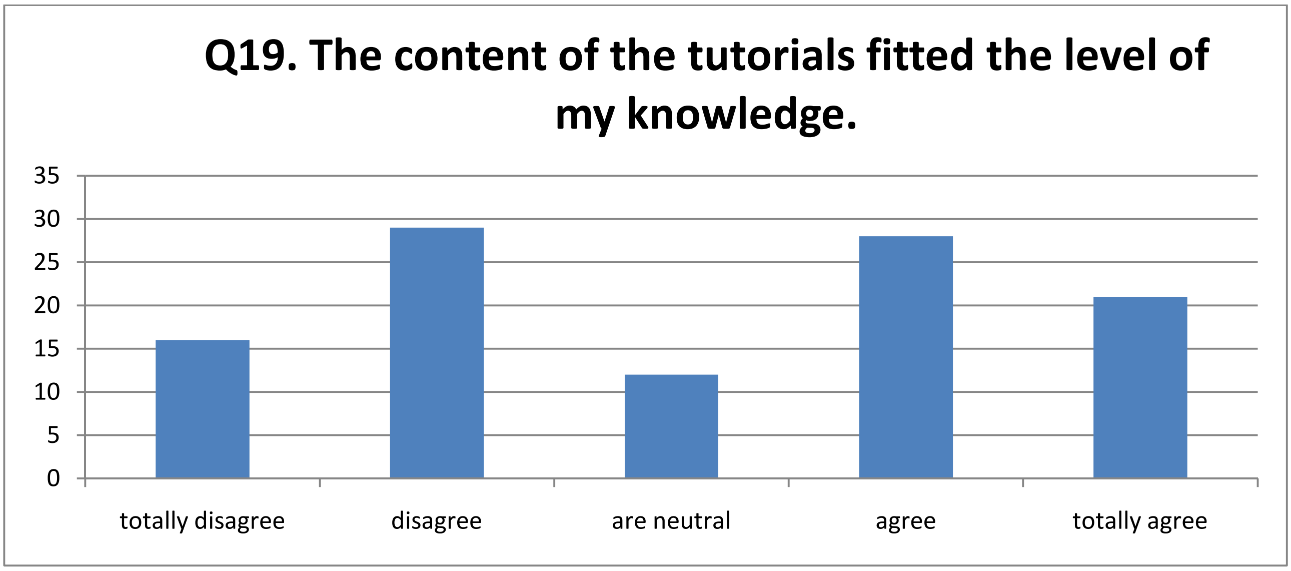


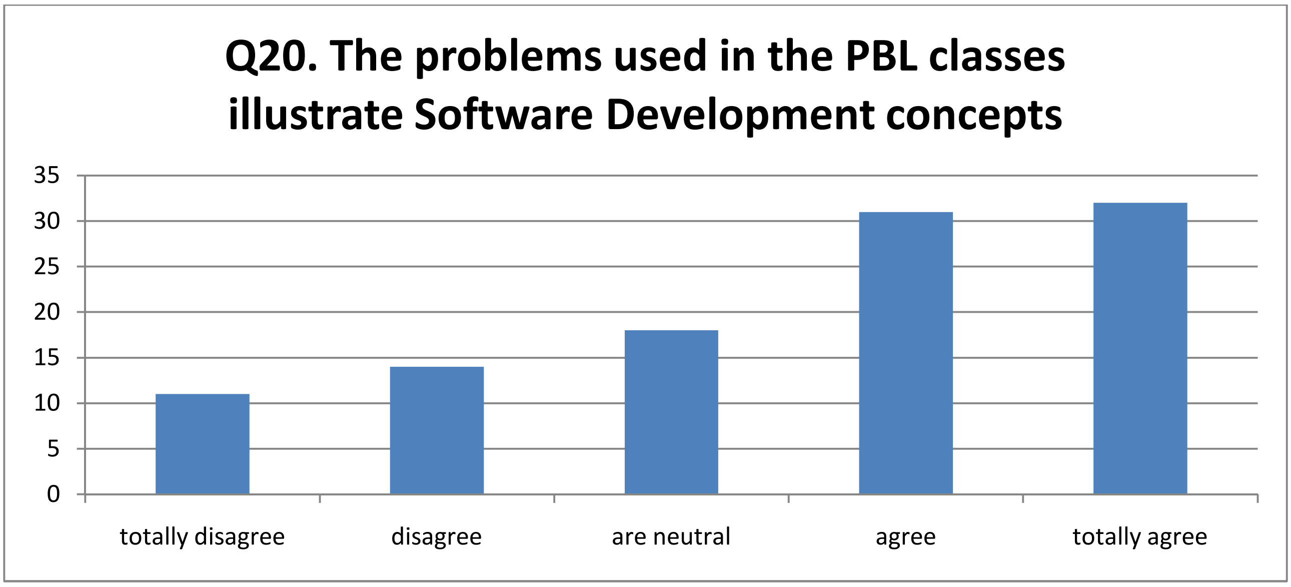


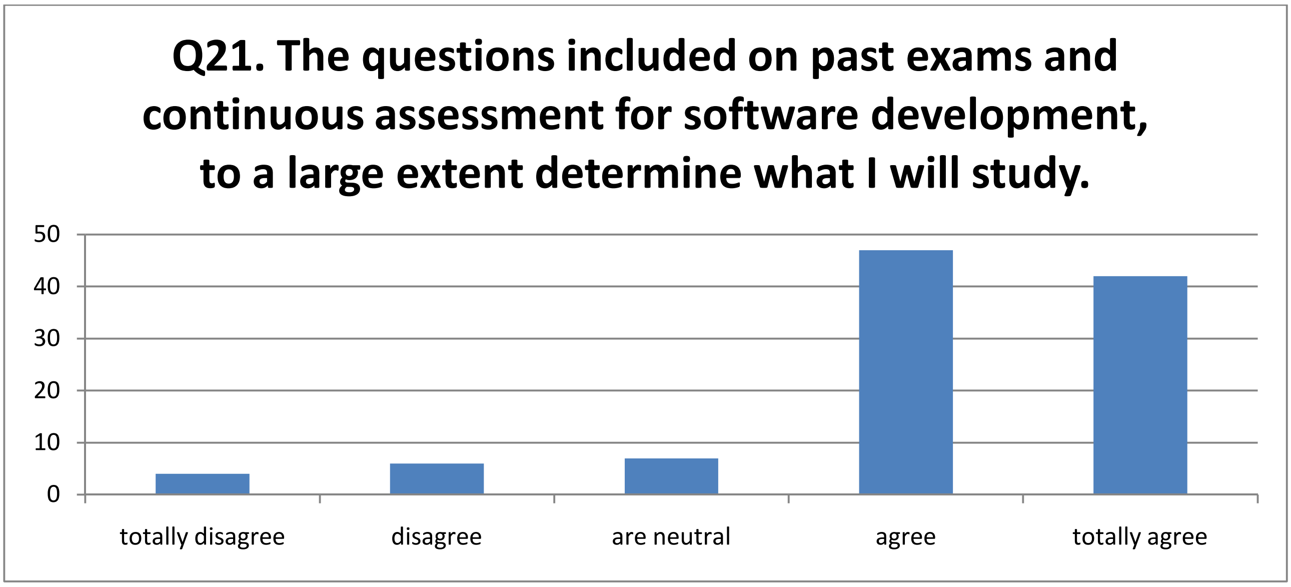
Q17. After class attendance, how much additional learning time did you invest each week in Software Development (time in hours). The mean answer was 2 hours.

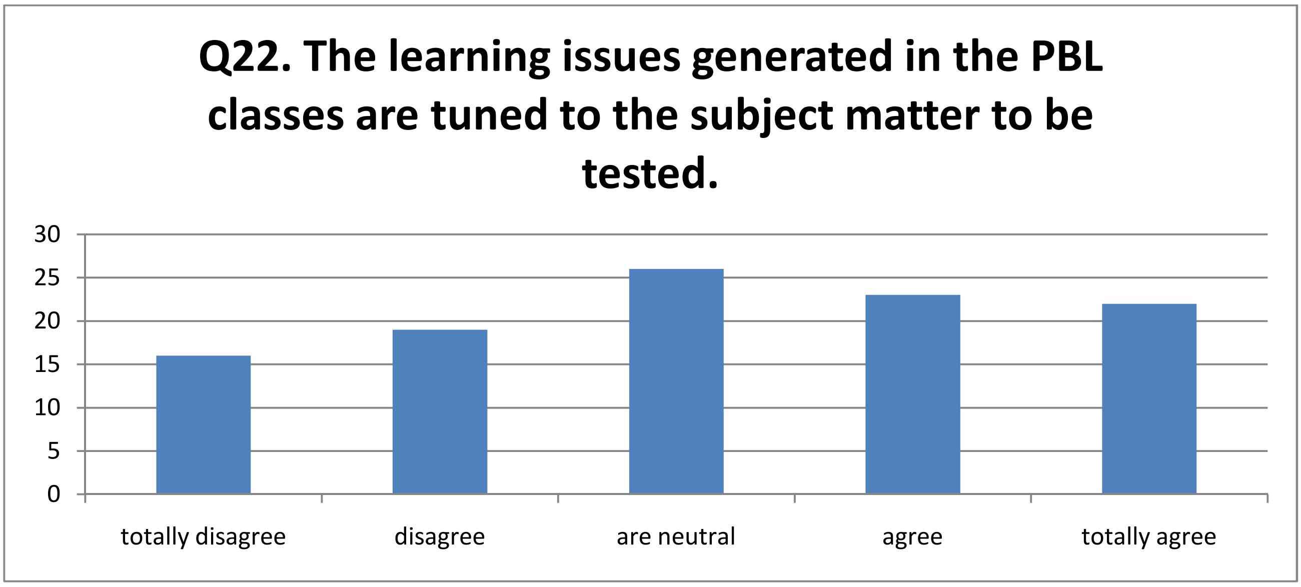
M.4. Course Objectives and Content



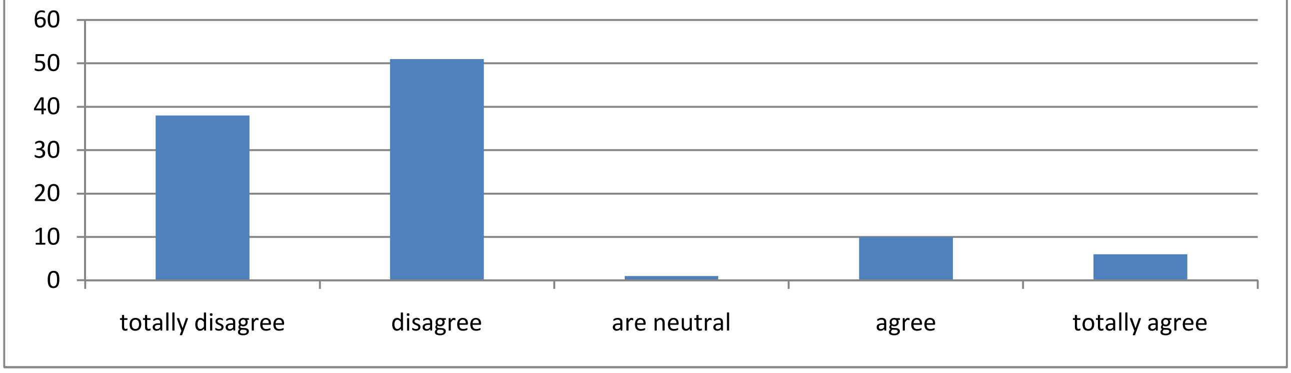



Q20. The problems used in the PBL classes illustrate Software Development concepts

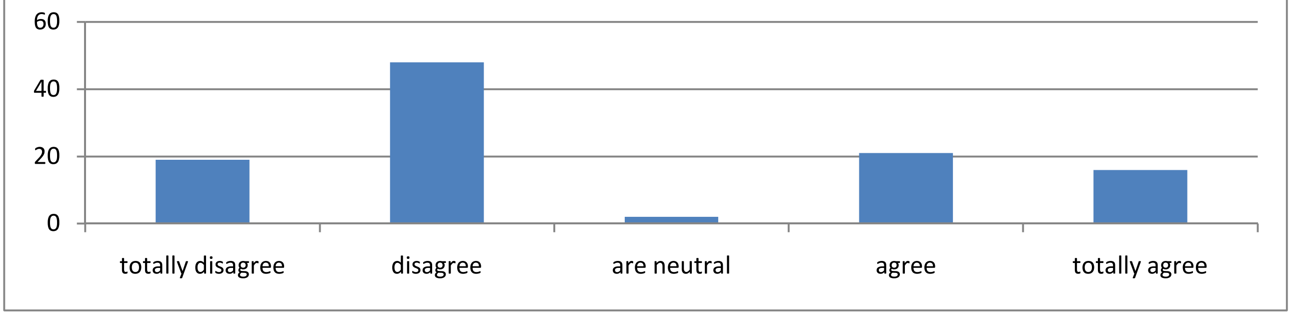

Q21. The questions included on past exams and continuous assessment for software development, to a large extent determine what I will study.


Q22. The learning issues generated in the PBL classes are tuned to the subject matter to be tested.

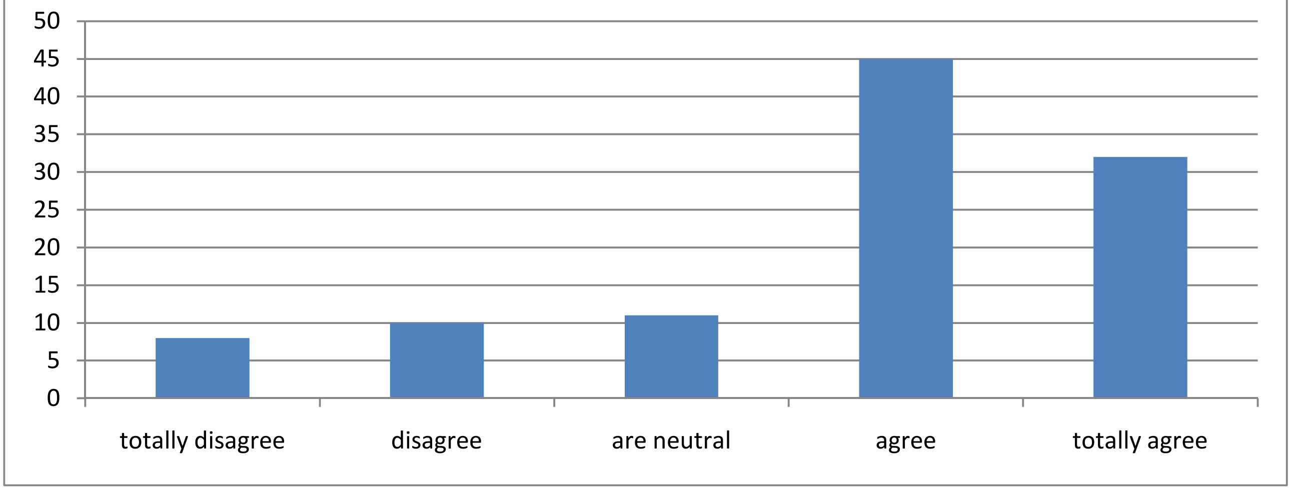
Q23. At the start of the Software Development course, I consulted the course objectives set out in the syllabus.

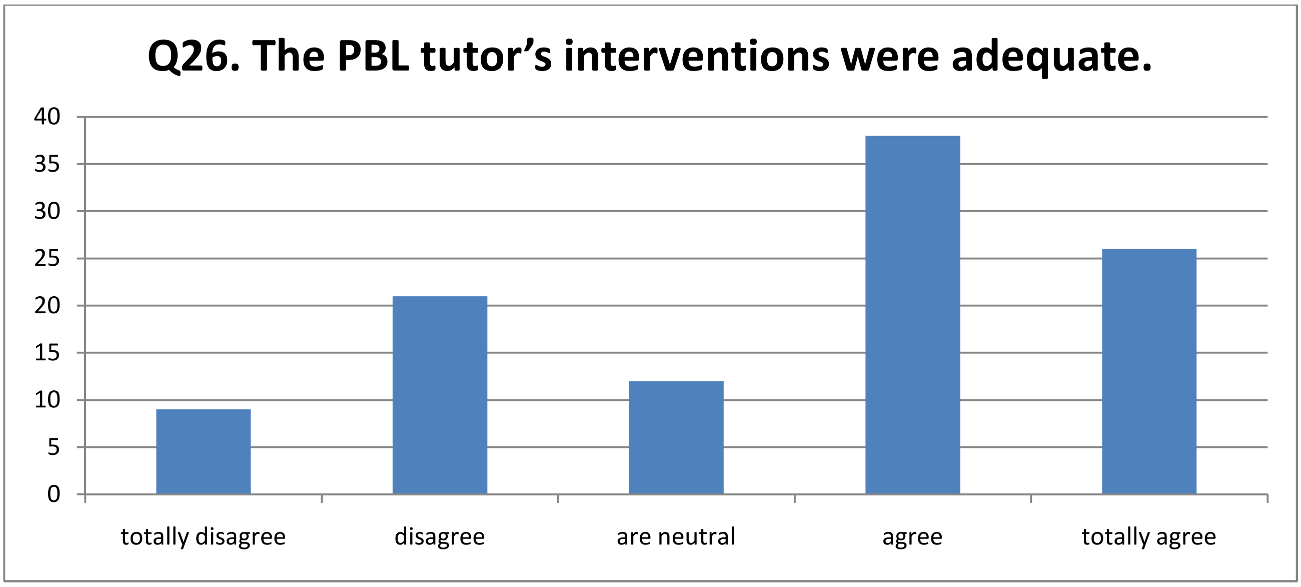


Q24. At the end of the Software Development course, I consulted the course objectives to check whether I covered all the subject matter I was expected to cover.

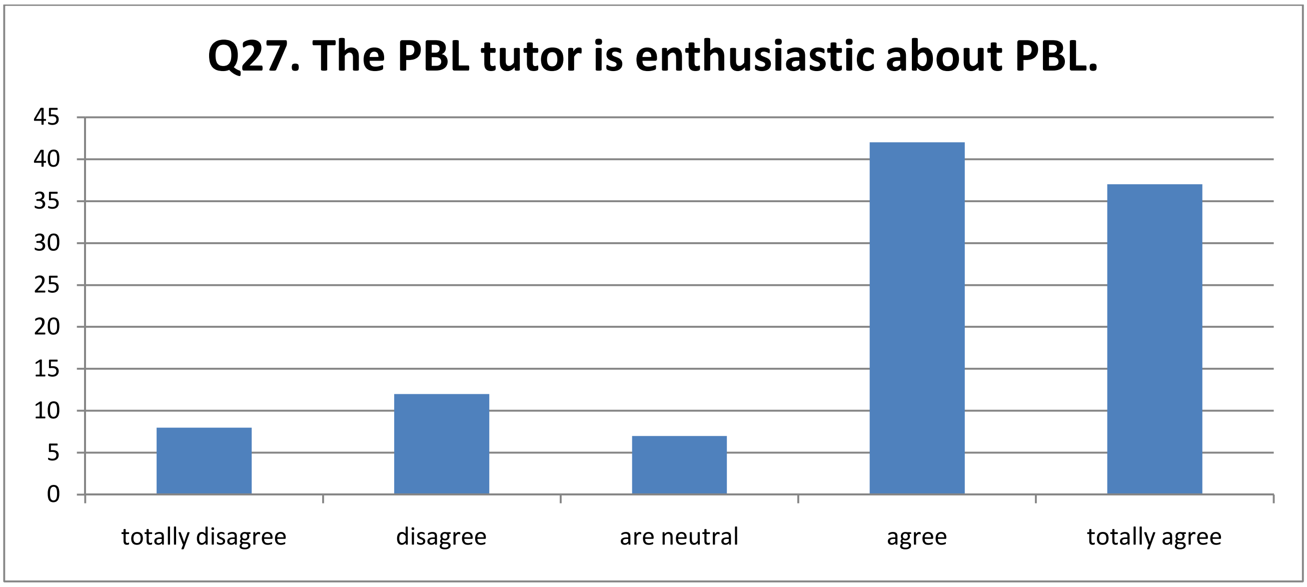


Q25. The PBL tutor has steered the group strongly

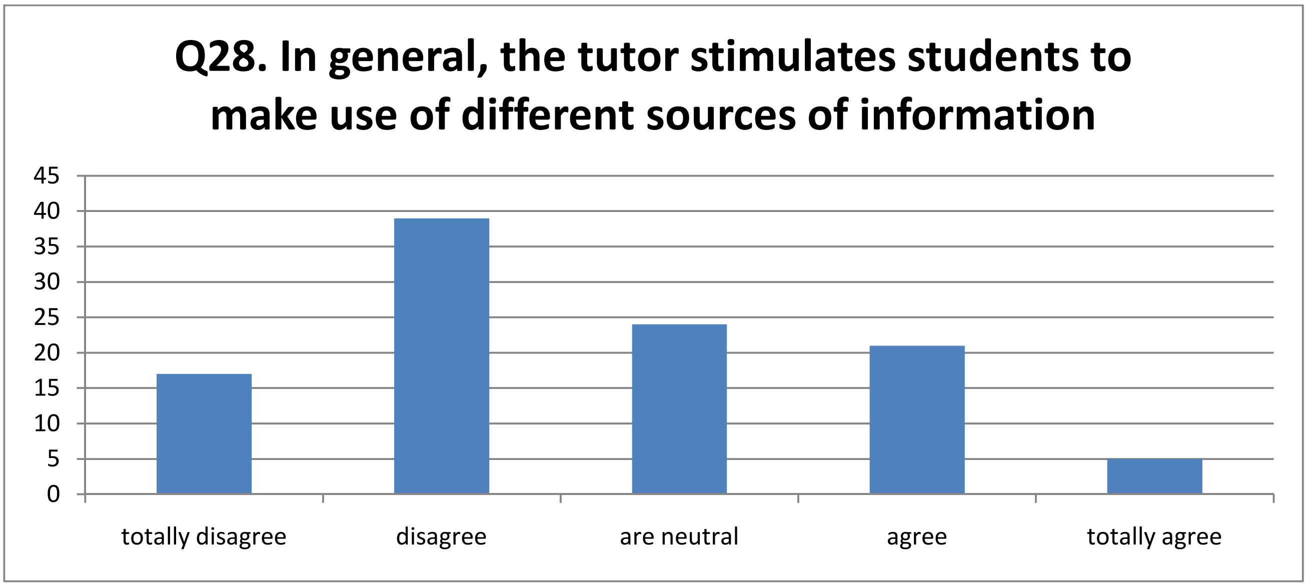




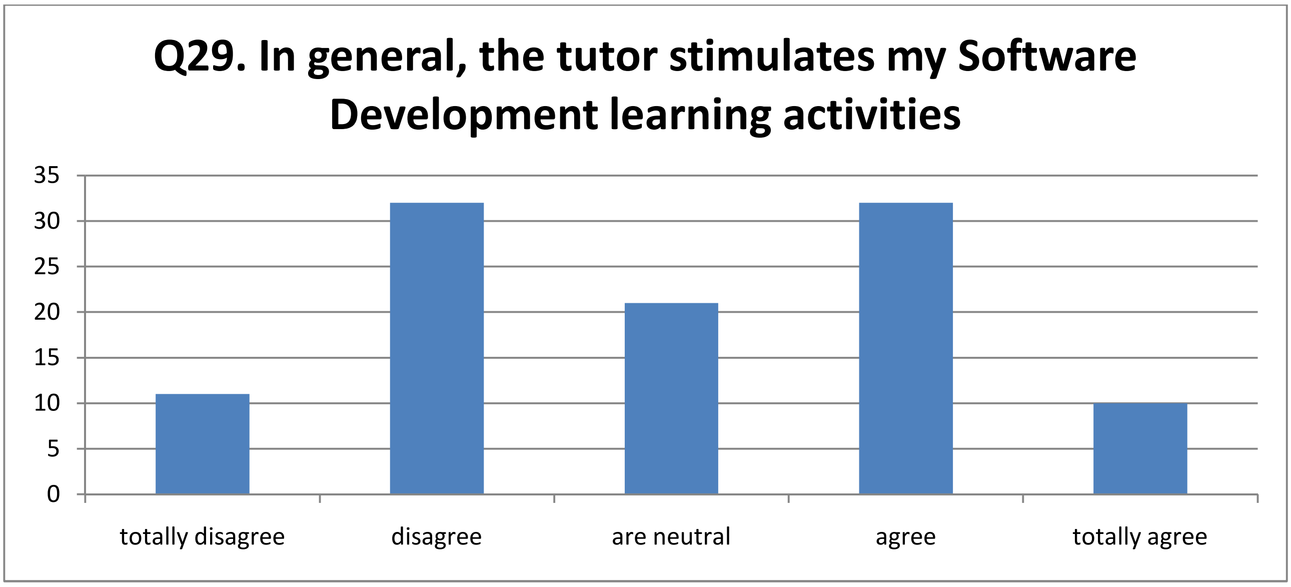
Q26. The PBL tutor's interventions were adequate.



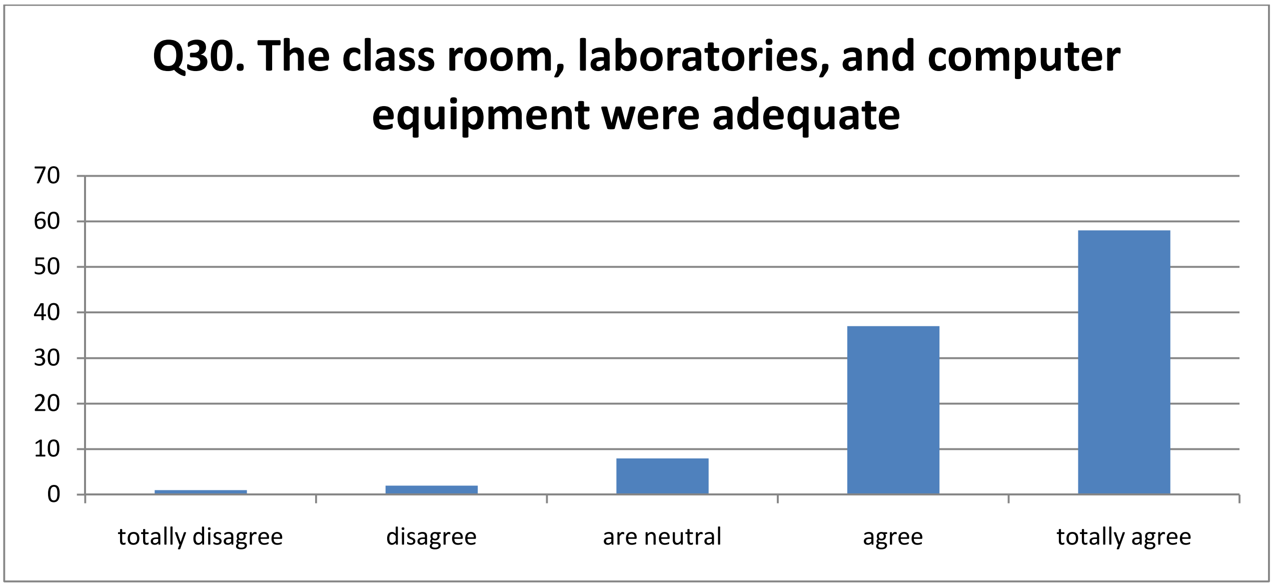
Q27. The PBL tutor is enthusiastic about PBL.



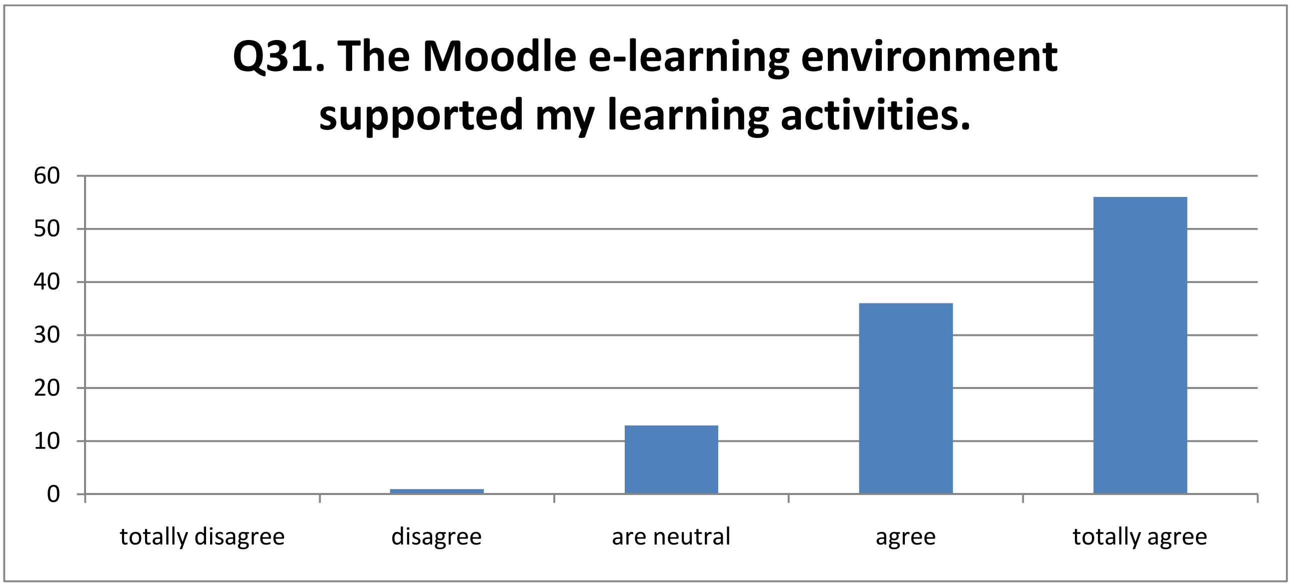
Q28. In general, the tutor stimulates students to make use of different sources of information



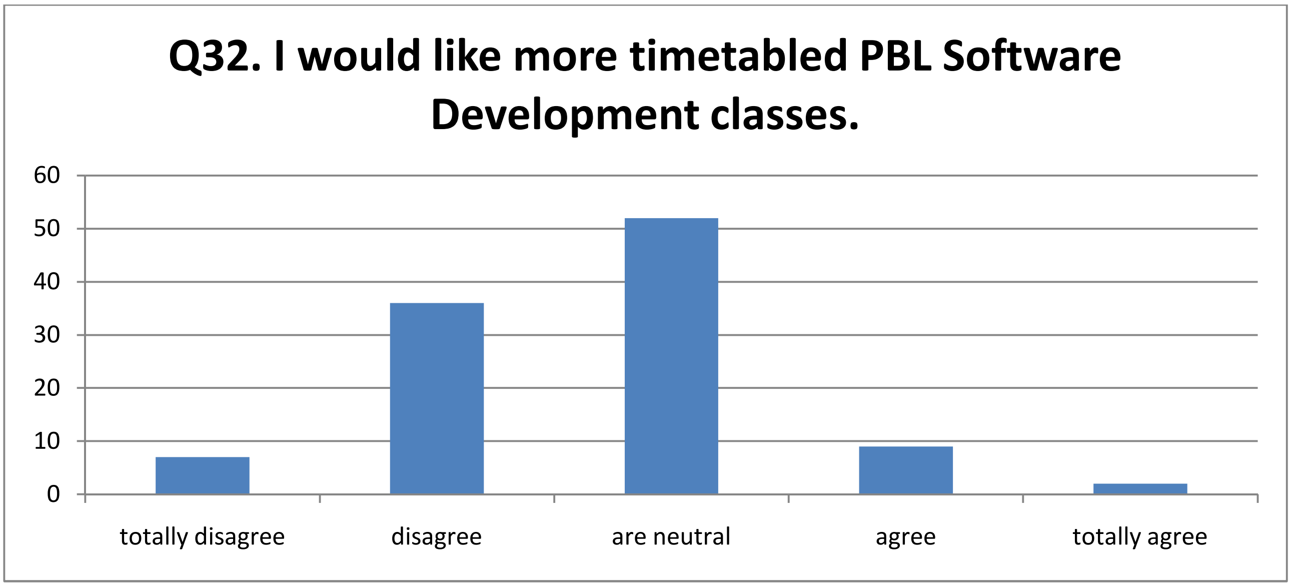
Q29. In general, the tutor stimulates my Software Development learning activities



Q30. The class room, laboratories, and computer equipment were adequate



Q31. The Moodle e-learning environment supported my learning activities.



Q33 asked students if in general they had any other additional comments about PBL or Software Development.

M.5. Actual Responses

Q	yes	are neutral	no	Total
Q9. If you had had the possibility to choose before the course, would you have opted for the PBL-course or the lecture-based course?	2	87	17	106
Q10. After the experience of the course, would you now opt for the PBL-course or the lecture-based course if you had to choose again?	66	8	32	106

Q	totally disagree	disagree	are neutral	agree	totally agree	Total
Q1. The tutorial group discussion is an important stimulus for my software development learning activities	7	13	12	50	24	106
Q2. The learning issues generated in the group tutorials are the most important starting point for my learning activities	5	12	11	52	26	106
Q3. I study to a large extent independently from the learning issues generated by my PBL group tutorials	16	51	7	21	11	106
Q4. The group climate facilitated the learning process	10	52	3	30	11	106
Q5. In the PBL tutorials I learned something that improved my software development skills	7	9	6	58	26	106
Q6. In the PBL group, I improved my communication skills.	10	39	32	18	7	106
Q7. I would recommend PBL tutorials to other students.	5	13	25	56	7	106
Q8. The PBL classes have motivated me to use additional learning resources.	32	36	18	8	12	106
Q11. I felt well informed about the PBL method.	2	1	4	29	70	106
Q12. I consider PBL to be an effective way of learning for myself.	6	11	21	41	27	106

Q13. PBL was fun	41	34	16	7	8	106
Q14. Before the tutorials, I was open to the method.	1	2	3	48	52	106
Q15. I am interested in the subject (Software Development) of the PBL tutorials.	1	3	22	37	43	106
Q16. I consider the subject (Software Development) to be important within the frame of my studies.	1	1	12	31	61	106
Q18. Topics covered during PBL classes stimulated my interest in Software Development.	9	27	14	32	24	106
Q19. The content of the tutorials fitted the level of my knowledge.	16	29	12	28	21	106
Q20. The problems used in the PBL classes illustrate Software Development concepts	11	14	18	31	32	106
Q21. The questions included on past exams and continuous assessment for software development, to a large extent determine what I will study.	4	6	7	47	42	106
Q22. The learning issues generated in the PBL classes are tuned to the subject matter to be tested.	16	19	26	23	22	106
Q23. At the start of the Software Development course, I consulted the course objectives set out in the syllabus.	38	51	1	10	6	106
Q24. At the end of the Software Development course, I consulted the course objectives to check whether I covered all the subject matter I was expected to cover.	19	48	2	21	16	106
Q25. The PBL tutor has steered the group strongly	8	10	11	45	32	106
Q26. The PBL tutor’s interventions were adequate.	9	21	12	38	26	106
Q27. The PBL tutor is enthusiastic about PBL.	8	12	7	42	37	106
Q28. In general, the tutor stimulates students to make use of different sources of information	17	39	24	21	5	106
Q29. In general, the tutor stimulates my Software Development learning activities	11	32	21	32	10	106
Q30. The class room, laboratories, and computer equipment were adequate	1	2	8	37	58	106
Q31. The Moodle e-learning environment supported my learning activities.	0	1	13	36	56	106
Q32. I would like more timetabled PBL Software Development classes.	7	36	52	9	2	106

Bibliography

Abelson, H., & Sussman, G. J. (1996). *Structure and Interpretation of Computer Programs.* Cambridge, Massachusetts, USA: MIT Press.

Alasuutari, P. (1995). *Researching Culture: Qualitative Method and Cultural Studies* (1st ed.). London: Sage.

Albanese, M. A. (2000). Problem-based learning: why curricula are likely to show little effect on knowledge and clinical skills. *Medical Education, 34*, 729-738.

Albanese, M. A., & Mitchell, S. (1993). Problem-based learning: a review of literature on its outcomes and implementation issues. *Academic Medicine, 68*(8), 52-81.

Alexander, P. A., & Murphy, P. K. (2000). A motivated exploration of motivation terminology. *Contemporary Educational Psychology, 25*, 3–53.

Alexander, S., Clark, M., Loose, K., Amillo, J., Daniels, M., Boyle, R., *et al.* (2003). Case studies in admissions to and early performance in computer science degrees. *ACM SIGCSE Bulletin, 35*(4), 137-147.

Ames, C. A. (1984). Competitive, cooperative, and individualistic goal structures: A cognitive-motivational analysis. In C. A. Ames & R. Ames (Eds.), *Research on motivation in education: Student Motivation* (Vol. 1, pp. 177-208). Greenwich: Academic Press.

Ames, C. A. (1990). Motivation: What Teachers Need to Know. *Teacher's College Record, 91*(3).

Ames, C. A. (1992). Achievement goals, motivational climate and motivational processes. In G. C. Roberts (Ed.), *Motivation in Sport and Exercise* (pp. 161–176). Champaign, IL: Human Kinetics.

Ames, C. A., & Archer, J. (1987). Mother's beliefs about the role of ability and effort in school learning. *Journal of Educational Psychology, 18*, 409-414.

Ames, C. A., & Archer, J. (1988). Achievement goals in the classroom: Students' learning strategies and motivation process. *Journal of Educational Psychology, 80*(3), 260 - 267.

Anazi, Y., & Uesato, Y. (1982). Is Recursive Computation difficult to Learn. *CIP paper No. 439, Dept. of Psychology, Carnegie-Mellon University, Pittsburg.*

Anderson, J. R. (1976). *Language, Memory, and Thought*. Hillsdale, NJ: Lawrence Erlbaum Associates.

Anderson, J. R. (1993). *Rules of the Mind*. Hillsdale, NJ: Lawrence Erlbaum Associates.

Anderson, J. R. (1996). *The Architecture of Cognition*. Hillsdale, NJ: Lawrence Erlbaum Associates.

Antepohl, W., & Herzig, S. (1999). Problem-based learning versus lecture-based learning in a course of basic pharmacology: a controlled, randomized study. *Medical Education, 33*(2), 106-113.

Askar, P., & Davenport, D. (2009). An Investigation of Factors Related To Self-Efficacy For Java Programming Among Engineering Students. *The Turkish Online Journal of Educational Technology, 8*(1).

Association for Computing Machinery (2008). Curricula Recommendations Retrieved 16 November, 2008, from http://www.acm.org/education/curricula-recommendations

Ayres, F. (2002). Problem-based learning: the benefits to students and organisations. *Training Journal*, 20-22.

Baca, E., Mennin, S. P., Kaufman, A., & Moore-West, M. (1990). Comparison between a problem-based, community-oriented track and a traditional track within one medical school. In Z. H. Nooman, H. G. Schmidt & E. S. Ezzat (Eds.), *Innovation in medical education: An evaluation of its present status* (pp. 9–26). New York: Springer.

Baecker, R. (1998). Sorting Out Sorting: A Case Study of Software Visualization for Teaching Computer Science. In J. T. Stasko, J. B. Domingue, M. H. Brown & B. A. Price (Eds.), *Software Visualization: Programming as a Multimedia Experience* (pp. 369-381). Cambridge, Massachusetts, USA: MIT Press.

Bandura, A. (1994). Self-efficacy. In V. Ramachaudran (Ed.), *Encyclopaedia of Human Behaviour* (Vol. 4, pp. 71-81). New York: Academic Press.

Barr, R. B., & Tagg, J. (1995). From teaching to learning - a new paradigm for undergraduate education. *Change, 27*(6), 12-25.

Barrett, T., Mac Labhrainn, I., & Fallon, H. (Eds.). (2005). *Handbook of Enquiry and Problem-based Learning: Irish Case studies and International Perspectives* (1st ed.). Galway: CELT NUI Galway.

Barron, K. E., & Harackiewicz, J. M. (2001). Achievement goals and optimal motivation: Testing multiple goal models. *Journal of Personality and Social Psychology, 80*, 706–722.

Barros, J. P., Estevens, L., Dias, R., Pais, R., & Soeiro, E. (2003). *Using lab exams to ensure programming practice in an introductory programming course.* Paper presented at the Annual Joint Conference Integrating Technology into Computer Science Education, Thessaloniki, Greece.

Barrows, H., & Tamblyn, R. (1980). *Problem-based learning: an approach to medical education.* New York: Springer Publishing Company.

Barrows, H. S. (1986). A taxonomy of problem-based learning methods. *Medical Education, 20*(6), 481-486.

Beckers, J. J., & Schmidt, H. G. (2001). The structure of computer anxiety: a six-factor model. *Computers in Human Behavior, 17*(.), 35-49.

Bell, J. (1993). *Doing Your Research Project: A Guide for First-Time Researched in Education and Social Science* (2 ed.). Milton Keynes, England: Open University Press.

Ben-Ari, M. (2001). Constructivism in computer science education. *Journal of Computers in Mathematics and Science Teaching, 20*(1), 45-73.

Ben-Ari, M., & Reich, N. (1997). Recursion: From Drama to Program. *Journal of Computer Science Education, 11*(3), 9–12.

Benbow, E. W., & McMahon, R. F. T. (2001). Mature students? In P. Schwartz, S. Mennin & G. Webb (Eds.), *Problem-based Learning. Case Studies, Experience and Practice* (pp. 119-125). London: Kogan Page.

Bennedsen, J., & Caspersen, M. E. (2007). Failure rates in introductory programming. *ACM SIGCSE Bulletin, 39*(2), 32-36.

Bentley, J. F., Lowry, G. R., & Sandy, G. A. (1999). *Towards the compleat information systems graduate: a problem based learning approach.* Paper presented at the 10th Australasian Conference on Information System.

BERA (2004). *British Educational Research Association (BERA) ethical guidelines.*

Bereiter, C., & Scardamalia, M. (1985). Cognitive coping strategies and the problem of inert knowledge. In J. W. Segal, S. F. Chipman & R. Glaser (Eds.), *Thinking and learning skills: Current Research and Open Questions* (Vol. 2, pp. 65-80). Hillsdale, NJ: Lawrence Erlbaum Associates.

Bergin, J., Brodie, K., Patiño-Martínez, M., McNally, M., Naps, T., Rodger, S.*, et al.* (1996). *An overview of visualization: its use and design: report of the working group in visualization.* Paper presented at the 1st conference on integrating technology into computer science education, Barcelona, Spain.

Bergin, S., & Reilly, R. (2005). *The influence of motivation and comfort-level on learning to program.* Paper presented at the 17th Workshop of the Psychology of Programming Interest Group, Sussex University.

Bergin, S., & Reilly, R. (2006). Predicting introductory programming performance: A multi-institutional multivariate study. *Computer Science Education, 16*(4), 303-323.

Berkson, L. (1993). Problem-based learning: have the expectations been met? *Academic Medicine, 68*(10 Supplement), S79-88.

Bernstein, P., Tipping, J., Bercovitz, K., & Skinner, H. A. (1995). Shifting students and faculty to a PBL curriculum: Attitudes changed and lessons learned. *Academic Medicine, 70*(3), 245-247.

Bhuiyan, S., Greer, J. E., & McCalla, G. I. (1994). Supporting the Learning of Recursive Problem Solving. *Interactive Learning Environments, 4*(2), 115-139.

Biermann, A. W. (1997). *Great Ideas in Computer Science: A Gentle Introduction*. Cambridge, Massachusetts, USA: MIT Press.

Biggs, J. B. (1979). Individual differences in study processes and the quality of learning outcomes. *Higher Education, 8*(4), 381-394.

Biggs, J. B. (1987). *Learning Process Questionnaire Manual. Student Approaches to Learning and Studying*. Hawthorn, Australia: Australian Council for Educational Research Ltd.

Biggs, J. B. (1988). Assessing student approaches to learning. *Australian Psychologist, 23*(2), 197-206.

Biggs, J. B. (1993). What do inventories of students' learning processes really measure? A theoretical review and clarification. *British Journal of Educational Psychology, 63*(1), 3-19.

Bishop-Clark, C. (1995). Cognitive style, personality, and computer programming. *Computers in Human Behavior, 11*(2), 241-260.

Black, A. E., & Deci, E. L. (2000). The effects of instructors' autonomy support and students' autonomous motivation on learning organic chemistry: A self-determination theory perspective. *Science Education, 84*(6), 740-756.

Black, S. R. (2003). *Predictors of first year computing science student failure*. Glasgow: University of Glasgow

Blight, J. (1995). Problem based, small group learning: an idea whose time has come. *British Medical Journal, 311*(7001), 342-343.

Blumberg, P., & Michael, J. (1992). Development of self-directed learning behaviours in a partially teacher-directed problem-based learning curriculum. *Teaching and Learning in Medicine*(4), 3-8.

Boehm, B. W. (1981). *Software engineering economics*. Englewood Cliffs, NJ: Prentice-Hall

Bonar, J., & Soloway, E. (1985). Preprogramming Knowledge: A Major Source of Misconceptions in Novice Programmers. *Human-Computer Interaction, 1*(2), 133-161.

Boud, D. (1985). *Problem-based Learning in Education for the Professions*: HERDSA.

Boud, D., & Feletti, G. (Eds.). (1998). *The Challenge of Problem-Based Learning* (2nd ed.). London: Routledge.

Bouffard, T., Boisvert, J., Vezeau, C., & Larouche, C. (1995). The impact of goal orientation on self-regulation and performance among college students. *British Journal of Educational Psychology, 65*, 317-329.

Bouffard, T., Vezeau, C., & Bordeleau, L. (1998). A developmental study of the relation between combined learning and performance goals and students' self-regulated learning. *British Journal of Educational Psychology, 68*, 309-319.

Boustedt, J., Eckerdal, A., McCartney, R., Moström, J. E., Ratcliffe, M., Sanders, K., *et al.* (2007). Threshold concepts in computer science: do they exist and are they useful? *ACM SIGCSE Bulletin, 39*(1), 504-508.

Box, R., & Whitelaw, M. (2000). *Experiences when migrating from structured analysis to object-oriented modelling.* Paper presented at the Australasian conference on Computing education, Melbourne, Australia.

Bransford, J. D., & Schwartz, D. L. (1999). Rethinking transfer: A simple proposal with interesting implications. In A. Iran-Nejad & P. D. Pearson (Eds.), *Review of Research in Education* (Vol. 24, pp. 61–100). Washington, DC: American Educational Research Association.

Bråten, I., Samuelstuen, M. S., & Strømsø, H. I. (2004). Do Students' Self-Efficacy Beliefs Moderate the Effects of Performance Goals on Self-Regulatory Strategy Use? *Educational Psychology, 24*(2), 231-247.

Bridges, E. M., & Hallinger, P. (1991, September). *Problem-based learning in medical and managerial education.* Paper presented at the Cognition and School Leadership Conference of the National Center for Educational Leadership and the Ontario Institute for Studies in Education, Nashville, TN.

Brooks, F. P. (1995). The Mythical Man-Month, Essays on Software Engineering, 20th Anniversary Edition (pp. 336). Reading, Massachusetts: Addison-Wesley.

Brooks, R. E. (1983). Towards a theory of the comprehension of computer programs. *International Journal of Man-Machine Studies, 18*(6), 543-554.

Brooks, R. E. (1990). Categories of programming knowledge and their application. *International Journal of Man-Machine Studies, 33*(3), 241-246.

Brooks, R. E. (1999). Towards a theory of the cognitive processes in computer programming. *International Journal of Human-Computer Studies, 51*(2), 197-211.

Brookshear, J. G. (2007). *Computer science: an overview* (10th ed.). Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc.

Brophy, J. (2004). *Motivating Students to Learn* (2nd ed.). Mahwah, NJ: Lawrence Erlbaum.

Brophy, J. (2005). Goal theorists should move on from performance goals. *Educational Psychologist, 40*(3), 167-176.

Brunel, P. C. (1999). Relationship between achievement goal orientations and perceived motivational climate on intrinsic motivation. *Scandinavian Journal of Medicine & Science in Sports, 9*(6), 365-374.

Bruner, J. S. (1960). *The Process of Education*: Harvard University Press.

Burns, A. (1999). *Collaborative action research for English language teachers*: Cambridge University Press.

Butler, R. (1992). What young people want to know when: Effects of mastery and ability goals on interest in different kinds of social comparisons. *Journal of Personality & Social Psychology, 62*, 934-943.

Butler, R., Inman, D., & Lobb, D. (2005). Problem-based learning and the medical school: another case of the emperor's new clothes? *Advances in Physiology Education*(29), 194-196.

Butler, S. (1999). Catalysing student autonomy through action research in a problem centred learning environment. *Research in Science Education, 29*(1).

Cañas, J. J., Bajo, M. T., & Gonzalvo, P. (1994). Mental models and computer programming. *International Journal of Human-Computer Studies*(40), 795-811.

Cantwell-Wilson, B., & Shrock, S. (2001). *Contributing to success in an introductory computer science course: a study of twelve factors.* Paper presented at the 32nd SIGCSE technical symposium on Computer Science Education Charlotte, North Carolina, United States.

Carmines, E. G., & Zeller, R. A. (1979). *Reliability and validity assessment.* Thousand Oaks, CA: Sage Publications.

Carter, J., & Jenkins, T. (2002). Gender differences in programming? *ACM SIGCSE Bulletin, 34*(3), 188-192.

Chou, H. W. (2001). Effects of Training Method and Computer Anxiety on Learning Performance and Self-Efficacy. *Computers in Human Behavior, 17*, 51-69.

Church, M. A., Elliot, A. J., & Gable, S. L. (2001). Perceptions of classroom environment, achievement goals and achievement outcomes. *Journal of Educational Psychology, 93*(1), 43-54.

Clancy, M. (2004). Misconceptions and Attitudes that Interfere with Learning to Program. In S. Fincher & M. Petre (Eds.), *Computer Science Education Research* (pp. 239). London: Routledge.

Clark, R. (1996). Data Protection in Ireland. *The Journal of Information, Law and Technology, 11*(2), 203 - 220.

Cohen, J. (1988). *Statistical Power Analysis for the Behavioral Sciences* (2nd ed.). New York: Lawrence Erlbaum.

Cohen, L., Manion, L., & Morrison, K. R. B. (2000). *Research Methods in Education* (5 ed.). London: Routledge.

Colby, A., Ehrlich, T., Beaumont, E., & Stephens, J. (2003). *Educating Citizens: Preparing America's Undergraduates for Lives of Moral and Civic Responsibility*. San Francisco, CA: Jossey-Bass.

Colliver, J. A. (2000). Effectiveness of Problem-based Learning Curricula. *Academic Medicine*(75), 259-266.

Colorado State University (2009). Writing Guides Reliability & Validity Retrieved 14 March, 2009, from http://writing.colostate.edu/guides/research/relval/index.cfm

Compeau, D. R., & Higgins, C. A. (1995). Computer Self-Efficacy: Development of a Measure and Initial Test. *MIS Quarterly, 19*(2), 189-211.

Connelly, F. M., & Clandinin, D. J. (1990). Stories of Experience and Narrative Inquiry. *Educational Researcher, 19*(5), 2-14.

Connolly, C., Murphy, E., & Moore, S. (2008). Programming Anxiety Amongst Computing Students - A Key in the Retention Debate? *IEEE Transactions on Education*(99), 1-4.

Connolly, T. M., & Begg, C. E. (2006). A constructivist-based approach to teaching database analysis and design. *Journal of Information Systems Education, 17*(1), 43.

Cooper, S., Dann, W., & Pausch, R. (2003). *Teaching objects-first in introductory computer science.* Paper presented at the 34th SIGCSE technical symposium on Computer science education, Reno, Navada, USA.

Covington, M. V. (1984). Strategic thinking and the fear of failure. In J. W. Segal, S. F. Chipman & R. Glaser (Eds.), *Thinking and Learning Skills: Relating instruction to research* (Vol. 1, pp. 389-416). Hillsdale, NJ: Lawrence Erlbaum Associates.

Creswell, J. W. (1998). *Qualitative inquiry and research design: Choosing among five traditions*. Thousand Oaks: Sage Publications.

Creswell, J. W. (2003). *Research Design: Qualitative, Quantitative, and Mixed Method Approaches* (2nd ed.). London: Sage Publications Inc.

Curzon, P., & Rix, J. (1998). Why do students take programming modules? *ACM SIGCSE Bulletin, 30*(3), 59-63.

Data Protection (Amendment) Act (2003). *Oireachtas Éireann*.

Davies, S. P. (1993). Models and theories of programming strategy. *International Journal of Man-Machine Studies, 39*(2), 237-267.

De Grave, W. S., Dolmans, D. H. J. M., & van der Vleuten, C. P. M. (2001). Student perceptions about the occurrence of critical incidents in tutorial groups. *Medical Teacher, 23*(1), 49-54.

De Volder, M. L., Schmidt, H. G., Moust, J. H. C., & De Grave, W. S. (1986). Problem-based-learning and intrinsic motivation. In J. H. C. van der Berchen, T. C. M. Bergen & E. E. I. de Bruyn (Eds.), *Achievement and task motivation* (pp. 25-32). The Netherlands: Swets.

Deci, E. L. (1971). Effects of externally mediated rewards on intrinsic motivation. *Journal of Personality and Social Psychology, 18*(1), 105-115.

Deci, E. L., & Ryan, R. M. (1985). *Intrinsic Motivation and Self-Determination in Human Behavior*. New York: Plenum Press.

Deci, E. L., & Ryan, R. M. (2000). The "what" and "why" of goal pursuits: Human needs and the self-determination of behaviour. . *Psychological Enquiry, 11*(4), 227-268.

Deek, F. P., & Kimmel, H. (1993). *Changing the Students' Role: From Passive Listeners to Active Participants.* Paper presented at the 23rd Frontiers in Education Conference, Washington, D.C., USA.

Deek, F. P., Kimmel, H., & McHugh, J. A. (1998). Pedagogical Changes in the Delivery of the First-Course in Computer Science: Problem Solving, Then Programming. *Journal of Engineering Education, 87*(3), 313-320.

Denning, P. J. (2004). Great Principles in Computing Curricula. *ACM SIGCSE Bulletin, 36*(1), 336-341.

Denzin, N. K. (2006). *Sociological Methods: A Sourcebook*: Aldine Transaction.

Des Marchais, J. (1993). A student-centred, problem-based curriculum: 5 years' experience. *Canadian Medical Association Journal* (148), 1567-1572.

Détienne, F. (1990). Expert Programming Knowledge: a Schema-Based Approach. In J. Hoc, T. R. G. Green, R. Samurcay & D. J. Gilmore (Eds.), *Psychology of programming* (pp. 205-222): Academic Press.

Détienne, F. (1997). Assessing the cognitive consequences of the object-oriented approach: a survey of empirical research on object-oriented design by individuals and teams. *Interacting with Computers, 9*(1), 47-72.

Détienne, F., & Soloway, E. (1990). An Empirically-Derived Control Structure for the Process of Program Understanding. *International Journal of Man-Machine Studies, 33*(3), 323-342.

Dewey, J. (1998). *Experience and education: The 60th Anniversary Edition*: Kappa Delta Pi.

Dicheva, D., & Close, J. (1996). Mental Models of Recursion. *Journal of Educational Computing Research, 14*(1), 1-24.

Diener, C. I., & Dweck, C. S. (1978). An analysis of learned helplessness: Continuous changes in performance, strategy, and achievement cognitions following failure. *Journal of Personality & Social Psychology, 36*(5), 451-462.

Diener, C. I., & Dweck, C. S. (1980). An analysis of learned helplessness II. The processing of success. *Journal of Personality and Social Psychology, 39*(5), 940-952.

Dijkstra, E. W. (1989). On the Cruelty of Really Teaching Computing Science. *Communications of the ACM, 32*(12), 1398-1404.

Dochy, F., Segers, M., Van den Bossche, P., & Gijbels, D. (2003). Effects of problem-based learning: a meta-analysis. *Learning and Instruction, 13*(5), 533-568.

Dolmans, D. H. J. M., De Grave, W. S., Wolfhagen, I. H. A. P., & van der Vleuten, C. P. M. (2005). Problem-based learning: future challenges for educational practice and research. *Medical Education, 39*(7), 732-741.

Dolmans, D. H. J. M., & Schmidt, H. G. (2000). What directs self-directed learning in a problem-based curriculum? In D. Evensen & C. E. Hmelo-Silver (Eds.), *Problem-Based Learning. A Research Perspective on Learning Interactions* (pp. 251-262). Mahwah New Jersey: Lawrence Erlbaum.

Dolmans, D. H. J. M., & Schmidt, H. G. (2006). What Do We Know About Cognitive and Motivational Effects of Small Group Tutorials in Problem-Based Learning? *Advances in Health Sciences Education, 11*(4), 321-336.

Donnelly, K. (2008, Thursday August 21). Record number of college courses unfilled. *Irish Independent*. Retrieved 6 October 2008, from http://www.independent.ie/national-news/record-number-of-college-courses-unfilled-1460197.html

Donnelly, K., & Walshe, J. (2008, Monday August 18). Thousands of jobs 'lost' as courses snubbed. *Irish Independent*. Retrieved 6 October 2008, from http://www.independent.ie/national-news/thousands-of-jobs-lost-as-courses-snubbed-1457757.html

Donner, R. S., & Bickley, H. (1993). Problem-based learning in American medical education: an overview. *Bulletin of the Medical Library Association, 81*(3), 294-298.

Doody, J. R., O'Reilly, D., Cardiff, J., & Magee, P. (2006, September). *Reflections On Teaching And Learning In A Virtual Learning Environment Using Learning Objects In Face-To-Face And Distance Learning Programmes.* Paper presented at the 22nd ICDE World Conference on Open Learning and Distance Education, Rio de Janeiro, Brazil.

Dreyfus, H. L., Dreyfus, S. E., & Anthanasiou, T. (2000). *Mind over Machine: The Power of Human Intuition and Expertise in the Era of the Computer*: Simon & Schuster.

du Boulay, B. (1989). Some difficulties of learning to program. In E. Soloway & J. C. Spohrer (Eds.), *Studying the novice programmer* (pp. 283-299). Hillsdale, NJ: Lawrence Erlbaum.

du Boulay, B., O'Shea, T., & Monk, J. (1989). The black box inside the glass box: presenting computing concepts to novices. In E. Soloway & J. C. Spohrer (Eds.), *Studying the novice programmer* (pp. 431-446). Hillsdale, NJ: Lawrence Erlbaum.

Duch, B. J., Groh, S. E., & Allen, D. E. (2001). Why problem-based learning? A case study of institutional change in undergraduate education. In B. J. Duch, S. E. Groh & D. E. Allen (Eds.), *The Power of Problem-Based Learning: A Practical" How To" for Teaching Undergraduate Courses in Any Discipline* (pp. 3-11). Sterling, VA: Stylus Publishing, LLC.

Duff, A. (1997). A note on the reliability and validity of a 30-item version of Entwistle & Tait's Revised Approaches to Studying Inventory. *British Journal of Educational Psychology, 67*(4), 529-539.

Dunlap, J. C. (2005). Problem-based learning and self-efficacy: How a capstone course prepares students for a profession. *Educational Technology Research and Development, 53*(1), 65-83.

Dweck, C. S. (1975). The role of expectations and attributions in the alleviation of learned helplessness. *Journal of Personality and Social Psychology, 31*(4), 674-685.

Dweck, C. S. (1986). Motivational processes affecting learning. *American Psychologist, 41*, 1040-1048.

Dweck, C. S., & Elliott, E. S. (1983). Achievement motivation. In P. Mussen & E. M. Hetherington (Eds.), *Handbook of child psychology: Socialization, Personality, and Social Development* (Vol. IV, pp. 643–691). New York: Wiley.

Dweck, C. S., & Leggett, E. L. (1988). A Social–Cognitive Approach to Motivation and Personality. *Psychological Review, 95*(2), 256–273.

Dweck, C. S., & Reppucci, N. D. (1973). Learned helplessness and reinforcement responsibility in children. *Journal of Personality and Social Psychology, 25*, 109-116.

Eckerdal, A. (2004). *On the understanding of Object and Class. Technical Report 2004-058.*: Dept. of Information Technology, Uppsala University, Sweden.

Eckerdal, A., McCartney, R., Moström, J. E., Ratcliffe, M., Sanders, K., & Zander, C. (2006). Putting threshold concepts into context in computer science education. *ACM SIGCSE Bulletin, 38*(3), 103-107.

Eckerdal, A., & Thuné, M. (2005). *Novice Java programmers' conceptions of 'object' and 'class', and variation theory.* Paper presented at the 10th annual SIGCSE conference on Innovation and technology in computer science education, Caparica, Portugal.

Eclipse Foundation (2004). Eclipse Retrieved 11 Janurary 2009, from http://www.eclipse.org/

Edward, N. S. (2001). Evaluation of a constructivist approach to student induction in relation to students' learning styles. *European Journal of Engineering Education, 26*(4), 429 - 440.

Efklides, A. (2003, August). *Metacognition and affect: What can metacognitive experiences tell us about the learning process?* Paper presented at the 10th EARLI Conference, University of Padova, Italy.

Efklides, A. (2006). Metacognition, affect, and conceptual difficulty. In J. H. F. Meyer & R. Land (Eds.), *Overcoming Barriers to Student Understanding: Threshold Concepts and Troublesome Knowledge* (pp. 48-69). London: Routledge.

Eisenstaedt, R. S., Barry, W. E., & Glanz, K. (1990). Problem-based learning: Cognitive retention and cohort traits of randomly selected participants and decliners. *Academic Medicine, 65*(9), 11–12.

Eisner, E. (1993). Objectivity in educational research. In M. Hammersley (Ed.), *Educational research: Current issues* (Vol. 1, pp. 49-56): Sage.

Elliot, A. J. (1999). Approach and Avoidance Motivation and Achievement Goals. *Educational Psychologist, 34*(3), 169-189.

Elliot, A. J. (2005). A conceptual history of the achievement goal construct. In A. J. Elliot & C. S. Dweck (Eds.), *Handbook of competence and motivation* (pp. 52-72). New York: Guilford Press.

Elliot, A. J., & Church, M. A. (1997). A hierarchical model of approach and avoidance achievement motivation. *Journal of Personality & Social Psychology, 72*(1), 218-232.

Elliot, A. J., & Covington, M. V. (2001). Approach and Avoidance Motivation. *Educational Psychology Review, 13*(2), 73-92.

Elliot, A. J., & Dweck, C. S. (2005). Competence and motivation: Competence as the core of achievement motivation. In A. J. Elliot & C. S. Dweck (Eds.), *Handbook of competence and motivation* (pp. 3–14). New York: Guilford Press.

Elliot, A. J., & Harackiewicz, J. M. (1996). Approach and Avoidance Achievement Goals and Intrinsic Motivation: A Mediational Analysis. *Journal of Personality and Social Psychology, 70*(3), 461-475.

Elliot, A. J., & McGregor, H. A. (1999). Test anxiety and the hierarchical model of approach and avoidance achievement motivation. *Journal of Personality and Social Psychology, 76*(3), 628-644. .

Elliot, A. J., & McGregor, H. A. (2001). A 2 x 2 Achievement Goal Framework. *Journal of Personality and Social Psychology, 80*(3), 501–519.

Elliot, A. J., & McGregor, H. A. (2002). Achievement goals as predictors of achievement-related processes prior to task engagement. *Journal of Educational Psychology, 94*, 381-395.

Elliot, A. J., McGregor, H. A., & Gable, S. L. (1999). Achievement goals, study strategies, and exam performance: A mediational analysis. *Journal of Educational Psychology, 91*(3), 549-563.

Elliot, A. J., & Reis, H. T. (2003). Attachment and exploration in adulthood. *Journal of Personality and Social Psychology, 85*, 317-331.

Elliot, A. J., & Thrash, T. M. (2001). Achievement goals and the Hierarchical Model of achievement motivation. *Educational Psychology Review, 13*(2), 139-156.

Ellis, A., Carswell, L., Bernat, A., Deveaux, D., Frison, P., Meisalo, V., *et al.* (1998). Resources, tools, and techniques for problem based learning in computing. [Special issue on the working group reports of the 3rd annual SIGCSE/SIGCUE ITiCSE conference]. *ACM SIGCUE Outlook, 26*(4), 41-56.

Ellis, S., & Dick, P. (2000). *Introduction to Organisational Behaviour* (3rd ed.). London: McGraw-Hill.

Ellsworth, E. (1997). Teaching Positions: Difference, Pedagogy, and the Power of Address. New York: Teachers College Press.

Enterprise Ireland (2004, December). Software: Why Ireland? Retrieved 19 October, 2008, from http://www.enterprise-ireland.com/SourceIreland/Ireland/Software.htm

Entwistle, N. J. (1997). The Approaches and Study Skills Inventory for Students (ASSIST), *Centre for Research on Learning and Instruction, University of Edinburgh, Edinburgh*.

Entwistle, N. J., & McCune, V. (2004). The Conceptual Bases of Study Strategy Inventories. *Educational Psychology Review, 16*(4), 325-345.

Entwistle, N. J., McCune, V., & Tait, H. (2006). *Approaches and Study Skills Inventory for Students (ASSIST) Report of the development and use of the inventory*

Entwistle, N. J., & Ramsden, P. (1983). *Understanding student learning*. London,: Croom Helm.

Entwistle, N. J., & Tait, H. (1994). The Revised Approaches to Studying Inventory. *Centre for Research into Learning and Instruction, University of Edinburgh, Edinburgh*.

Entwistle, N. J., Tait, H., & McCune, V. (2000). Patterns of Response to an Approaches to Studying Inventory across Contrasting Groups and Contexts. *European Journal of Psychology of Education, 15*(1), 33-48.

Entwistle, N. J., & Waterston, S. (1988). Approaches to Studying and Levels of Processing in University Students. *British Journal of Educational Psychology, 58*(pt3), 258-265.

Evans, G. E., & Simkin, M. G. (1989). What best predicts computer proficiency? *Communications of the ACM, 32*(11), 1322-1327.

Expert Group on Future Skills Needs (2008). *Strong Future and Opportunities for ICT Sector in Ireland*. Retrieved 19 October 2008. from http://www.skillsireland.ie/press/releases/2008-06-23-future_ict_skills.html.

Eylon, B.-S., & Linn, M. C. (1988). Learning and Instruction: An Examination of Four Research Perspectives in Science Education. *Review of Educational Research, 58*(3), 251.

Feldgen, M., & Clua, O. (2003). *New motivations are required for freshman introductory programming.* Paper presented at the 33rd Annual Frontiers in Education Conference, , Boulder, CO, USA.

Fincher, S. (1999a). Analysis of design: an exploration of patterns and pattern languages for pedagogy. *Journal of Computers in Mathematics and Science Teaching, 18*(3), 331-348.

Fincher, S. (1999b). *What are we doing when we teach programming?* Paper presented at the 29th Annual Frontiers in Education Conference, 1999. FIE '99., San Juan, Puerto Rico.

Fincher, S., Baker, B., Box, I., Cutts, Q., de Raadt, M., Haden, P., *et al.* (2005). *Programmed to succeed?: a multi-national, multi-institutional study of introductory programming courses*.

Fincher, S., & Utting, I. (2002). Pedagogical Patterns: Their Place in the Genre. *ACM SIGCSE Bulletin, 34*(3), 199-202.

Finney, S. J., Pieper, S. L., & Barron, K. E. (2004). Examining the Psychometric Properties of the Achievement Goal Questionnaire in a General Academic Context. *Educational and Psychological Measurement, 64*, 365-382.

Finucane, P. M., Johnson, S. M., & Prideaux, D. J. (1998). Problem-based learning: its rationale and efficacy. *Medical Journal of Australia*(168), 445-448.

Fisher, R. C. (1994). The Potential for Problem-Based Learning in Pharmacy Education: A Clinical Therapeutics Course in Diabetes. *American Journal of Pharmaceutical Education, 58*, 183-183.

Fleury, A. E. (2000). *Programming in Java: student-constructed rules.* Paper presented at the thirty-first SIGCSE technical symposium on Computer science education, Austin, Texas, United States.

Fleury, A. E. (2001). Encapsulation and Reuse as Viewed by Java Students. *ACM SIGCSE Bulletin, 33*(1), 189-193.

Ford, M. (1992). *Motivating humans: Goals, emotions, and personal agency beliefs*. Newbury Park, CA: Sage.

Fraenkel, J. R., & Wallen, N. E. (2005). *How to Design and Evaluate Research in Education* (6th ed.). Boston: McGraw-Hill.

Gagné, E. D. (1978). Long-term retention of information following learning from prose. *Review of Educational Research, 48*(4), 629–665.

Gamma, E., Helm, R., Johnson, R., & Vlissides, J. M. (1995). *Design patterns: elements of reusable object-oriented software*. Reading, MA: Addison-Wesley.

Gardner, H. E. (1993). *The Unschooled Mind: How Children Think and How Schools Should Teach*. New York: Basic Books.

Geertz, C. (1977). *The Interpretation of Culture*. New York: Basic Books.

Gentner, D., & Stevens, A. L. (Eds.). (1983). *Mental Models* (1st ed.). Hillsdale, NJ: Lawrence Erlbaum Associates.

Gibbs, G. (1992). *Improving the quality of student learning*. Bristol, UK.: Technical and Educational Services.

Gilmore, D. J. (1990). Expert programming knowledge: a strategic approach. In J.-M. Hoc, T. R. G. Green, R. Samurcay & D. J. Gilmore (Eds.), *Psychology of programming* (pp. 223–234). London: Academic Press.

Gilmore, D. J., & Green, T. R. G. (1984). Comprehension and recall of miniature programs. *International Journal of Man-Machine Studies, 21*(1), 31-48.

Gomez-Albarran, M. (2005). The teaching and learning of programming: a survey of supporting software tools. *The Computer Journal, 48*(2), 130-144.

Gotschi, T., Sanders, I., & Galpin, V. (2003). Mental Models of Recursion. *ACM SIGCSE Bulletin, 35*(1), 346-350.

Green, T. R. G. (1990). Programming languages as information structures. In J.-M. Hoc, T. R. G. Green, R. Samurcay & D. J. Gilmore (Eds.), *Psychology of programming* (pp. 117-137). London: Academic Press.

Grolnick, W. S., & Ryan, R. M. (1987). Autonomy in Children's learning: An experimental and individual difference investigation. *Journal of Personality and Social Psychology, 52*(5), 890-898.

Groves, M. (2005). Problem-Based Learning and Learning Approach: Is There a Relationship? *Advances in Health Sciences Education, 10*(4), 315-326.

Guba, E. G., & Lincoln, Y. S. (1989). *Fourth Generation Evaluation*. Newbury Park: Sage Publications.

Guindon, R. (1990). Knowledge exploited by experts during software system design. *International Journal of Man-Machine Studies, 33*(3), 279-304.

Haberman, B., & Averbuch, H. (2002). The case of base cases: why are they so difficult to recognize? student difficulties with recursion. *ACM SIGCSE Bulletin, 34*(3), 84-88.

Hadjerrouit, S. (1998a). A Constructivist Framework for Integrating the Java Paradigm into the Undergraduate Curriculum. *ACM SIGCSE Bulletin, 30*(3), 105-107.

Hadjerrouit, S. (1998b). Java as First Programming Language: A Critical Evaluation. *ACM SIGCSE Bulletin, 30*(2), 43-47.

Hagan, D., & Markham, S. (2000). *Does it help to have some programming experience before beginning a computing degree program?* Paper presented at the 5th annual SIGCSE/SIGCUE ITiCSE conference on Innovation and technology in computer science education Helsinki, Finland.

Hammersley, M. (1993). *Educational Research: Current Issues* (Vol. 1): Sage.

Hammersley, M. (1998). *Reading Ethnographic Research: A Critical Guide* (2nd ed.). London: Longman.

Harackiewicz, J. M., Barron, K. E., Carter, S. M., Lehto, A. T., & Elliot, A. J. (1997). Predictors and consequences of achievement goals in the college classroom: Maintaining interest and making the grade. *Journal of Personality and Social Psychology, 73*, 1284–1295.

Harackiewicz, J. M., Barron, K. E., & Elliot, A. J. (1998). Rethinking achievement goals: When are they adaptive for college students and why? *Educational Psychologist, 33*, 1-21.

Harackiewicz, J. M., Barron, K. E., Pintrich, P. R., Elliot, A. J., & Thrash, T. M. (2002a). Revision of Achievement Goal Theory: Necessary and Illuminating. *Journal of Educational Psychology, 94*(3), 638–645.

Harackiewicz, J. M., Barron, K. E., Tauer, J. M., Carter, S. M., & Elliot, A. J. (2000). Short-Term and Long-Term Consequences of Achievement Goals: Predicting Interest and Performance Over Time. *Journal of Educational Psychology, 92*(2), 316–330.

Harackiewicz, J. M., Barron, K. E., Tauer, J. M., & Elliot, A. J. (2002b). Predicting Success in College: A Longitudinal Study of Achievement Goals and Ability Measures as Predictors of Interest and Performance From Freshman Year Through Graduation. *Journal of Educational Psychology, 94*(3), 562–575.

Harackiewicz, J. M., & Elliot, A. J. (1993). Achievement goals and intrinsic motivation. *Journal of Personality and Social Psychology, 65*, 904-915.

Harackiewicz, J. M., & Sansone, C. (1991). Goals and intrinsic motivation: You can get there from here. In M. L. Maeher & P. R. Pintrich (Eds.), *Advances in motivation and achievement: Goals and self-regulatory processes* (Vol. 7, pp. 21-50). Greenwich, CT: JAI Press.

Haskell, R. E. (2001). *Transfer of Learning: Cognition, Instruction, and Reasoning*. London: Academic Press.

Headrick, L., Kaufman, A., Stillman, P., Wilkerson, L., & Wigton, R. (1994). Teaching and learning methods for new generalist physicians. *Journal of General Internal Medicine* (9), S42-S49.

Hendry, G. D., Frommer, M., & Walker, R. A. (1999). Constructivism and problem-based learning. *Journal of further and higher education, 23*(3), 369-371.

Hendry, G. D., Ryan, G., & Harris, J. (2003). Group problems in problem-based learning. *Medical Teacher, 25*(6), 609-616.

Hidi, S., & Harackiewicz, J. M. (2000). Motivating the academically unmotivated: A critical issue for the 21st century. *Review of Educational Research, 70*(2), 151-179.

Hitchcock, M. A., & Anderson, A. S. (1997). Dealing with dysfunctional tutorial groups. *Teaching and Learning in Medicine, 9*(1), 19-24.

Hmelo-Silver, C. E. (2004). Problem-Based Learning: What and How Do Students Learn? *Educational Psychology Review, 16*(3), 235-266.

Hoc, J.-M., Green, T. R. G., Samurcay, R., & Gilmore, D. J. (Eds.). (1990). *Psychology of programming*. London: Academic Press.

Hoc, J.-M., & Nguyen-Xuan, A. (1990). Language semantics, mental models and analogy. In J.-M. Hoc, T. R. G. Green, R. Samurcay & D. J. Gilmore (Eds.), *Psychology of programming* (pp. 139–156). London: Academic Press.

Holland, S., Griffiths, R., & Woodman, M. (1997). Avoiding Object Misconceptions. *ACM SIGCSE Bulletin, 29*(1), 131-134.

Holloway, I. (1997). *Basic Concepts for Qualitative Research*: Blackwell Publishing.

Humphrey, W. S. (1999). *Introduction to the Team Software Process* (1st ed.). Reading, Massachusetts: Addison-Wesley Professional.

Irani, L. (2004). Understanding gender and confidence in CS course culture. *ACM SIGCSE Bulletin, 36*(1), 195-199.

Irons, A., & Alexander, S. (Eds.). (2004). *Effective learning and teaching in computing*. London, UK: RoutledgeFalmer.

Jackson, M. (2003). Why software writing is difficult and will remain so. *Information Processing Letters, 88*(1-2), 13 - 25.

JavaPLT Group (2008). About DrJava Retrieved 11 January 2009, from http://www.cs.rice.edu/~javaplt/drjava/

Jenkins, T. (2001). *The motivation of students of programming.* Paper presented at the 6th annual conference on Innovation and technology in computer science education, Canterbury, UK.

Jenkins, T. (2002, August 27- 29). *On the difficulty of learning to program.* Paper presented at the 3rd Annual conference of the LTSN Centre for Information and Computer Sciences, Loughborough, UK.

Johnson-Laird, P. N. (1983). *Mental models: towards a cognitive science of language, inference, and consciousness* (Vol. 6). Cambridge: Harvard University Press.

Kahney, H. (1983). *What do novice programmers know about recursion.* Paper presented at the the SIGCHI conference on Human Factors in Computing Systems, Boston, Massachusetts, United States.

Kaplan, A., & Middleton, M. J. (2002). Should Childhood Be a Journey or a Race? Response to Harackiewicz et al. (2002). *Journal of Educational Psychology, 94*(3), 646-648.

Karabenick, S. A. (2003). Seeking help in large college classes: A person centered approach. *Contemporary Educational Psychology, 28*, 37-58.

Karabenick, S. A. (2004). Perceived achievement goal structure and college student help seeking. *Journal of Educational Psychology, 96*, 569-581.

Kaufman, A., Mennin, S., Waterman, R., Duban, S., Hansbarger, C., Silverblatt, H., *et al.* (1989). The New Mexico experiment: Educational innovation and institutional change. *Academic Medicine, 64*(6), 285–294.

Kaufman, D. M. (1995). Preparing faculty as tutors in problem-based learning. *Teaching Improvement Practices—Successful Strategies for Higher Education (eds. W. A. Wright & Associates)*, 101-125.

Kawasaki, G., & Williams, A. (2008). *Software Driving Global Business Opportunities*. Paper presented at the ISA Annual Conference.

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

Khalife, J. T. (2006, September 2006). *Threshold for the introduction of programming: Providing learners with a simple computer model.* Paper presented at the 18th Workshop of the Psychology of Programming Interest Group, University of Sussex.

Kieras, D. E., & Bovair, S. (1984). The role of a mental model in learning to operate a device. *Cognitive Science, 8*(3), 255-273.

Kinnunen, P., & Malmi, L. (2005). Problems in Problem-Based Learning - Experiences, Analysis and Lessons Learned on an Introductory Programming Course. *Informatics in Education, 4*(2), 193-214.

Klem, A. M., & Connell, J. P. (2004). Relationships Matter: Linking Teacher Support to Student Engagement and Achievement. *Journal of School Health, 74*(7), 262-273.

Koestner, R., Zuckerman, M., & Koestner, J. (1987). Praise, involvement and intrinsic motivation *Journal of Personality and Social Psychology, 53*, 383-390.

Kolb, D. A. (1984). *Experiential learning: Experience as the source of learning and development*. Englewood Cliffs, NJ: Prentice-Hall.

Kölling, M. (1999). The problem of teaching object-oriented programming. *Journal of Object Oriented Programming, 11*(8), 8-15.

Konvalina, J., Wileman, S. A., & Stephens, L. J. (1983). Math proficiency: a key to success for computer science students. *Communications of the ACM, 26*(5), 377-382.

Kurland, D. M., Pea, R. D., Clement, C., & Mawby, R. (1989). A study of the development of programming ability and thinking skills in high school students. In E. Soloway & J. C. Spohrer (Eds.), *Studying the novice programmer* (pp. 283-299). Hillsdale, NJ: Lawrence Erlbaum.

Laurillard, D. (1979). The processes of student learning. *Higher Education, 8*(4), 395-409.

Lawrence, R. (2004). Teaching data structures using competitive games. *IEEE Transactions on Education, 47*(4), 459-466.

Letovsky, S. (1986). Cognitive Processes in Program Comprehension. In E. Soloway & S. Iyengar (Eds.), *Empirical studies of programmers, First Workshop* (pp. 58-79). Norwood, NJ: Intellect Books.

Levy, D., & Lapidot, T. (2000). Recursively speaking: analyzing students' discourse of recursive phenomena. *ACM SIGCSE Bulletin, 32*(1), 315-319.

Lewis, K. E., & Tamblyn, R. M. (1987). The problem-based learning approach in Baccalaureate nursing education: How effective is it? *Nursing Papers, 19*(2), 17–26.

Lieberman, D., & Remedios, R. (2007). Do undergraduates' motives for studying change as they progress through their degrees? *British Journal of Educational Psychology, 77*(2), 379-395.

Lincoln, Y. S., & Guba, E. G. (1985). *Naturalistic Inquiry* (1st ed.): Sage Publications Inc.

Linge, N., & Parsons, D. (2006). Problem-based learning as an effective tool for teaching computer network design. *IEEE Transactions on Education, 49*(1), 5-10.

Linn, M. C., & Dalbey, J. (1989). Cognitive consequences of programming instruction. In E. Soloway & J. C. Spohrer (Eds.), *Studying the novice programmer* (pp. 57-81). Hillsdale, NJ: Lawrence Erlbaum.

Lipsey, M. W., & Wilson, D. B. (1993). The efficacy of psychological, educational, and behavioral treatment. Confirmation from meta-analysis. *American Psychologist, 48*(12), 1181-1209.

Lohr, S. (2001). *Go To: The story of the math majors, bridge players, engineers, chess wizards, maverick scientists and iconoclasts - the programmers who created the software revolution.* New York. USA.: Basic Books.

Long, W. F. (2003). Dissonance Detected by Cluster Analysis of Responses to the Approaches and Study Skills Inventory for Students. *Studies in Higher Education, 28*(1), 21-35.

Lopez, D. F. (1999). Social cognitive influences on self-regulated learning: The impact of action-control beliefs and academic goals on achievement-related outcomes. *Learning and Individual Differences, 11*(3), 301-319.

Luker, P. A. (1994). *There's more to OOP than syntax!* Paper presented at the twenty-fifth SIGCSE symposium on Computer science education, Phoenix, Arizona, United States.

Maeher, M. L. (1983). On doing well in science: Why Johnny no longer excels, why Sarah never did. In S. G. Paris, G. M. Olsen & H. W. Stevenson (Eds.), *Learning and Motivation in the Classroom* (pp. 179-210). Hillsdale, NJ: Erlbaum.

Maeher, M. L., & Midgley, C. (1991). Enhancing student motivation: A schoolwide approach. *Educational Psychologist, 26*(3/4), 399-427.

Mahmoud, Q. H., Dobosiewicz, W., & Swayne, D. (2004). *Redesigning introductory computer programming with HTML, JavaScript, and Java.* Paper presented at the 35th SIGCSE technical symposium on Computer science education, Norfolk, Virginia, USA.

Malterud, K. (2001). Qualitative research: standards, challenges, and guidelines. *The Lancet, 358*(9280), 483-488.

Mamone, S. (1992). Empirical study of motivation in a entry level programming course. *ACM SIGPLAN Notices, 27*(3), 54-60.

Maricopa Community Colleges Center for Learning and Instruction (2001). PBL Definition Retrieved 30 Sept 2008, from http://www.mcli.dist.maricopa.edu/pbl/info.html

Marsh, H. W. (1987). Student's evaluations of university teaching: research findings, methodological issues, and directions for future research. *International Journal of Educational Research, 11*(3), 253-388.

Marton, F. (1976). What does it take to learn? Some implications of an alternative view of learning. In N. J. Entwistle (Ed.), *Strategies for Research and Development in Higher Education* (pp. 32-43). Amsterdam: Swets and Zeitlinger.

Marton, F., Dall'Alba, G., & Beaty, E. (1993). Conceptions of Learning. *International Journal of Educational Research, 19*, 277-300.

Marton, F., & Säljö, R. (1976). On Qualitative Differences in Learning: 1 - Outcome and Process. *British Journal of Educational Psychology, 46*, 4-11.

Marton, F., & Säljö, R. (1997). Approaches to learning. In F. Marton, D. J. Hounsell & N. J. Entwistle (Eds.), *The experience of learning* (2nd ed.). Edinburgh: Scottish Academic Press.

Maudsley, G. (1999). Do we all mean the same thing by "problem-based learning"? A review of the concepts and a formulation of the ground rules. *Academic Medicine, 74*(2), 178.

Maudsley, G. (2002). Making sense of trying not to teach: an interview study of tutors' ideas of problem-based learning. *Academic Medicine, 77*(2), 162-172.

Mauffette, Y., Kandlbinder, P., & Soucisse, A. (2004). The problem in problem-based learning is the problems: But do they motivate students? In M. Savin-Baden & K. Wilkie (Eds.), *Challenging Research into Problem-based learning* (pp. 11-25): Buckingham: SRHE and Open University Press.

Mayer, R. E. (1989). The psychology of how novices learn computer programming. In E. Soloway & J. C. Spohrer (Eds.), *Studying the novice programmer* (pp. 129–159). Hillsdale, NJ: Lawrence Erlbaum.

Mayer, R. E. (2004). Should there be a three-strikes rule against pure discovery learning. *American Psychologist, 59*(1), 14-19.

Mayer, R. E., Dyck, J. L., & Vilberg, W. (1989). Learning to program and learning to think: what’s the connection? In E. Soloway & J. C. Spohrer (Eds.), *Studying the novice programmer* (pp. 113-124). Hillsdale, NJ: Lawrence Erlbaum.

Mays, N., & Pope, C. (1995). Qualitative Research: Rigour and qualitative research. *British Medical Journal, 311*(6997), 109-112.

Mazlack, L. J. (1980). Identifying potential to acquire programming skill. *Communications of the ACM, 23*(1), 14-17.

McAllister, G., & Alexander, S. (2003). Key aspects of teaching and learning in information and computer sciences. In H. Fry, S. Ketteridge & S. Marshall (Eds.), *A handbook for teaching and learning in higher education: enhancing academic practice* (2nd ed.). London, UK: Kogan Page.

McCracken, M., Almstrum, V., Diaz, D., Guzdial, M., Hagan, D., Kolikant, Y. B. D*., et al.* (2001). A multi-national, multi-institutional study of assessment of programming skills of first-year CS students. [Report by the ITiCSE 2001 Working Group on Assessment of Programming Skills of first-year CS Students.]. *ACM SIGCSE Bulletin, 33*(4), 125-180.

McCracken, M., Newstetter, W., & Chastine, J. (1999). Misconceptions of designing: a descriptive study. *ACM SIGCSE Bulletin, 31*(3).

McKeough, A., Lupart, J. L., & Marini, A. (Eds.). (1995). *Teaching for Transfer: Fostering Generalization in Learning*: Lawrence Erlbaum Associates.

McLean, L., Myers, M., Smillie, C., & Vaillancourt, D. (1997). Qualitative Research Methods: An essay review. *Education Policy Analysis Archives, 5*(13).

Meece, J. L., & Holt, K. (1993). A pattern analysis of students' achievement goals. *Journal of Educational Psychology, 85*, 582-590.

Mendelsohn, P., Green, T. R. G., & Brna, P. (1990). Programming languages in education: The search for an easy start. In J.-M. Hoc, T. R. G. Green, R. Samurcay & D. J. Gilmore (Eds.), *Psychology of programming* (pp. 175-199). London: Academic Press.

Mertens, D. M. (1998). *Research Methods in Education and Psychology: Integrating Diversity with Quantitative & Qualitative Approaches*: Sage Publications.

Meyer, B. (1997). *Object-oriented software construction* (2 ed.). Upper Saddle River, NJ: Prentice Hall.

Meyer, J. H. F., & Land, R. (2005). Threshold concepts and troublesome knowledge (2): Epistemological considerations and a conceptual framework for teaching and learning. *Higher Education, 49*(3), 373-388.

Meyer, J. H. F., & Land, R. (2006). Threshold concepts and troublesome knowledge: An introduction. In J. H. F. Meyer & R. Land (Eds.), *Overcoming Barriers to Student Understanding: Threshold Concepts and Troublesome Knowledge* (pp. 3-18): Routledge.

Middleton, M. J., & Midgley, C. (1997). *Avoiding the Demonstration of Lack of Ability: An Under-Explored Aspect of Goal Theory*. Paper presented at the Annual meeting of the American Education Research Association.

Midgley, C., Kaplan, A., & Middleton, M. J. (2001). Performance-approach goals: Good for what, for whom, under what circumstances, and at what cost? *Journal of Educational Psychology, 93*, 77–86.

Midgley, C., Maeher, M. L., Hruda, L. Z., Anderman, E., Anderman, L., Freeman, K. E*., et al.* (2000). *Manual for the Patterns of Adaptive Learning Scales (PALS)*. Ann Arbor, MI:: University of Michigan.

Miles, M. B., & Huberman, M. A. (1984). *Qualitative data analysis: A sourcebook of new methods.* Beverly Hills: Sage Publications.

Miserandino, M. (1996). Children who do well in school: Individual differences in perceived competence and autonomy in above-average children. *Journal of Educational Psychology, 88*(2), 203-214.

Mitchell, J. E., Smith, J., & Kenyon, A. J. (2005). 'It's not for lazy students like me.'. *International Journal of Electrical Engineering Education, 42*(1), 41-51.

Mitchell, M., Sheard, J., & Markham, S. (2000). *Student motivation and positive impressions of computing subjects.* Paper presented at the Australasian conference on Computing education, Melbourne, Australia.

Moran, M. A., & Crowley, M. J. (1979). The Leaving Certificate and First Year University Performance. *Journal of the Statistical and Social Inquiry Society of Ireland, 24*, 231-266.

Morgan, A., Gibbs, G., & Taylor, E. (1981). *What do Open University students initially understand about learning* (No. 8). Walton Hall, Milton Keynes England: The Open University.

Newble, D. I., & Clarke, R. M. (1986). The approaches to learning of students in a traditional and in an innovative problem-based medical school. *Medical Education, 20*(4), 267-273.

Newman, M. (2002). Software Errors Cost U.S. Economy $59.5 Billion Annually Retrieved 27 August, 2009, from http://www.nist.gov/public_affairs/releases/n02-10.htm

Newman, M. (2004a). *A pilot systematic review and meta-analysis on the effectiveness of Problem Based Learning*: The Learning and Teaching Support Network subject centre for Medicine, Dentistry and Veterinary Medicine.

Newman, M. (2004b). *Problem-based learning: An exploration of the method and evaluation of its effectiveness in a continuing nursing education programme*: Middlesex University.

Nicholls, J. G. (1976). Effort Is Virtuous, But It's Better to Have Ability: Evaluative Responses to Perceptions of Effort and Ability. *Journal of Research in Personality, 10*(3), 306-315.

Nicholls, J. G. (1978). The Development of the Concepts of Effort and Ability, Perception of Academic Attainment, and the Understanding That Difficult Tasks Require More Ability. *Child Development, 49*(3), 800-814.

Nicholls, J. G. (1979). Quality and equality in intellectual development: The role of motivation in education. *American Psychologist, 34*, 1071-1084.

Nicholls, J. G. (1980). The Development of the Concept of Difficulty. *Merrill-Palmer Quarterly, 26*(3), 271-281.

Nicholls, J. G. (1989). *The competitive ethos and democratic education*. Cambridge, MA: Harvard University Press.

Nickerson, R. S. (1982). Computer programming as a vehicle for teaching thinking skills. *Thinking: The Journal of Philosophy for Children, 4*(3), 42-48.

Nolen, S. B. (1988). Reasons for studying: Motivational orientations and study strategies. *Cognition and Instruction, 5*(4), 269-287.

Norman, G. R., & Schmidt, H. G. (1992). The psychological basis of problem-based learning: a review of the evidence. *Academic Medicine, 67*(9), 557-565.

Ntoumanis, N. (2001). Empirical links between achievement goal theory and self-determination theory in sport. *Journal of Sports Sciences, 19*, 397-409.

O'Kelly, J. (2005). Designing a hybrid problem-based learning (PBL) course: A case study of first year computer science in NUI Maynooth. In T. Barrett, I. Mac Labhrainn & H. Fallon (Eds.), *Handbook of Enquiry and Problem-based Learning: Irish Case studies and International Perspectives*.

O'Kelly, J., Mooney, A., Ghent, J., Gaughran, P., Dunne, S., & Bergin, S. (2004). *An Overview of the Integration of Problem Based Learning into an existing Computer Science Programming Module.* Paper presented at the Problem-Based Learning International. Conference 2004: Pleasure by Learning, 2004, Cancun, Mexico.

O'Neill, K., Singh, G., & O'Donoghue, J. (2004). Implementing eLearning Programmes for Higher Education: A Review of the Literature. *Journal of Information Technology Education, 3*.

Or-Bach, R., & Lavy, I. (2004). Cognitive Activities of Abstraction in Object Orientation: An Empirical Study. *ACM SIGCSE Bulletin, 36*(2), 82-86.

Ormerod, T. C. (1990). Human cognition and programming. In J.-M. Hoc, T. R. G. Green, R. Samurcay & D. J. Gilmore (Eds.), *Psychology of programming* (pp. 63-92). London: Academic Press.

Palmer, R. E. (2001, 29 May). The Liminality of Hermes and the Meaning of Hermeneutics Retrieved 7 November, 2008, from http://www.mac.edu/faculty/richardpalmer/liminality.html

Patton, M. Q. (1990). *Qualitative evaluation and research methods* (2nd ed.). London: Sage.

Patton, M. Q. (1999). Enhancing the quality and credibility of qualitative analysis. *Health Services Research, 34*(5 Pt 2), 1189-1208.

Pennington, N. (1987a). Comprehension Strategies in Programming. In G. M. Olsen, S. Sheppard & E. Soloway (Eds.), *Empirical Studies of Programmers: 2nd Workshop* (pp. 100-112). Norwood, NJ: Ablex Publishing Corp.

Pennington, N. (1987b). Stimulus structures and mental representations in expert comprehension of computer programs. *Cognitive Psychology, 19*(3), 295-341.

Perkins, D. N. (1992). *Smart Schools: From Training Memories to Educating Minds*. New York: Free Press.

Perkins, D. N. (1999). The Many Faces of Constructivism. *Educational Leadership, 57*(3), 6-11.

Perkins, D. N. (2006). Constructivism and troublesome knowledge. In J. H. F. Meyer & R. Land (Eds.), *Overcoming Barriers to Student Understanding:*

Threshold Concepts and Troublesome Knowledge (pp. 33-47). London: Routledge.

Perkins, D. N., Hancock, C., Hobbs, R., Martin, F., & Simmons, R. (1989). Conditions of learning in novice programmers. In E. Soloway & J. C. Spohrer (Eds.), *Studying the novice programmer* (pp. 261-279). Hillsdale, NJ: Lawrence Erlbaum.

Perkins, D. N., & Martin, F. (1986). Fragile knowledge and neglected strategies in novice programmers. In E. Soloway & S. Iyengar (Eds.), *Empirical studies of programmers, First Workshop* (pp. 213-229). Norwood, NJ: Intellect Books.

Perkins, D. N., Schwartz, S., & Simmons, R. (1988). Instructional Strategies for the Problems of Novice Programmers. In R. E. Mayer (Ed.), *Teaching and Learning Computer Programming: Multiple Research Perspectives* (pp. 153-178): Lawrence Erlbaum Associates.

Perkins, D. N., & Simmons, R. (1988). Patterns of Misunderstanding: An Integrative Model for Science, Math, and Programming. *Review of educational research, 58*(3), 303.

Phillips, D. C. (Ed.). (2000). *Constructivism in Education: Opinions and Second Opinions on Controversial Issues. Ninety-Ninth Yearbook of the National Society for the Study of Education*. Chicago: University of Chicago Press.

Pincus, K. V. (1995). Introductory Accounting: Changing the First Course. *New Directions for Teaching and Learning*(61), 89-98.

Pintrich, P. R. (2000a). An Achievement Goal Theory Perspective on Issues in Motivation Terminology, Theory, and Research. *Contemporary Educational Psychology, 25*, 92-104.

Pintrich, P. R. (2000b). Multiple goals, multiple pathways: The role of goal orientation in learning and achievement. *Journal of Educational Psychology, 92*, 544–555.

Pintrich, P. R. (2000c). The role of goal orientation in self-regulated learning. In B. M., P. R. Pintrich & Z. M. (Eds.), *Handbook of self-regulation* (pp. 451-502). San Diego: Academic Press.

Pintrich, P. R., & Garcia, T. (1991). Student goal orientation and self-regulation in the college classroom. In M. L. Maeher & P. R. Pintrich (Eds.), *Advances in motivation and achievement: Goals and self-regulatory processes* (Vol. 7, pp. 371-402). Greenwich, CT: JAI Press.

Pintrich, P. R., Smith, D., Garcia, T., & McKeachie, W. J. (1991). *A manual for the use of the Motivated Strategies for Learning Questionnaire (MSLQ)*. Ann Arbor: University of Michigan, School of Education.

Pole, C., & Morrison, M. (2003). *Ethnography for Education*: Open University Press.

Postmes, T., Tanis, M., & de Wit, B. (2001). Communication and commitment in organizations: A social identity approach. *Group Processes & Intergroup Relations, 4*(3), 227-246.

Proulx, V. K. (2000). *Programming patterns and design patterns in the introductory computer science course.* Paper presented at the thirty-first SIGCSE technical symposium on Computer science education, Austin, Texas, United States

Radio Telefís Éireann (2005, Thursday, 26 May 2005). Not enough graduates for IT jobs Retrieved 19 October, 2008, from http://www.rte.ie/business/2005/0526/technology.html

Ragonis, N., & Ben-Ari, M. (2002). *Teaching constructors: A difficult multiple choice.* Paper presented at the 16th European Conference on Object-Oriented Programming, Universidad de Málaga, Málaga, Spain.

Ramalingam, V., & Wiedenbeck, S. (1998). Development and Validation of Scores on a Computer Programming Self-Efficacy Scale and Group Analyses of Novice Programmer Self-Efficacy. *Journal of Educational Computing Research, 19*(4), 367-381.

Ramsden, P. (1979). Student learning and perceptions of the academic environment. *Higher Education, 8*(4), 411-427.

Ramsden, P. (1991). A performance indicator of teaching quality in higher education: The Course Experience Questionnaire. *Studies in Higher Education, 16*(2), 129-150.

Rasch, R. H., & Tosi, H. L. (1992). Factors Affecting Software Developers' Performance: An Integrated Approach. *MIS Quarterly, 16*(3), 395-413.

Rawsthorne, L. J., & Elliot, A. J. (1999). Achievement goals and intrinsic motivation: a meta-analytic review. *Personality and Social Psychology Review, 3*, 326-344.

Razmov, V., & Anderson, R. (2006). *Pedagogical techniques supported by the use of student devices in teaching software engineering.* Paper presented at the 37th SIGCSE technical symposium on Computer science education, Houston, Texas, USA

Reed, D. (1998). *Incorporating problem-solving patterns in CS1.* Paper presented at the twenty-ninth SIGCSE technical symposium on Computer science education, Atlanta, Georgia, United States.

Reeve, J., Bolt, E., & Cai, Y. (1999). Autonomy-supportive teachers: How they teach and motivate students. *Journal of Educational Psychology, 91*, 537-548.

Rehder, B., Pennington, N., & Lee, A. Y. (1995). Cognitive Activities and Levels of Abstraction in Procedural and Object-Oriented Design. *Human-Computer Interaction, 10*(2), 171-226.

Reimann, N., & Jackson, I. (2003). *Threshold concepts in economics–a case study*. Paper presented at the 10th Conference of the European Association for Research on Learning and Instruction (EARLI), .

Report of The Joint Task Force on Computing Curricula (2001). *Computing Curricula 2001: Computer Science*: IEEE Computer Society and The Association for Computing Machinery.

Richardson, J. T. E. (1990). Reliability and replicability of the Approaches to Studying Questionnaire. *Studies in Higher Education, 15*(2), 155-168.

Richardson, J. T. E. (1996). Measures of effect size. *Behavior Research Methods, Instruments, & Computers, 28*(1), 12-22.

Richardson, J. T. E. (2000). *Researching Student Learning: Approaches to Studying in Campus-based and Distance Education*: Society for Research into Higher Education & Open University Press.

Richardson, J. T. E. (2005). The future of research in problem-based learning. In H. Crabtree, A. Darvill, K. Holland, S. MacKay, M. McLoughlin, D. Oakley & J. Supyk (Eds.), *Problem-based Learning 2004 A Quality Experience?* (pp. 41-59): University of Salford.

Rist, R. S. (1990). Variability in Program Design: The Interaction of Process with Knowledge. *International Journal of Man-Machine Studies, 33*(3), 305-322.

Rist, R. S. (1995). Program structure and design. *Cognitive Science, 19*(4), 507-562.

Rist, R. S. (1996). Teaching Eiffel as a first language. *Journal of Object-Oriented Programming, 9*(3), 30-41.

Robertson, S. P., & Yu, C.-C. (1990). Common Cognitive Representations of Program Code Across Tasks and Languages. *International Journal of Man-Machine Studies, 33*(3), 343-360.

Robins, A. (1996). Transfer in cognition. *Connection Science, 8*(2), 185-203.

Robins, A., Rountree, J., & Rountree, N. (2003). Learning and Teaching Programming: A Review and Discussion. *Computer Science Education, 13*(2), 137-172.

Roddan, M. (2002). *The Determinants of Student Failure and Attrition in First Year Computing Science.* Unpublished Final-year undergraduate project., University of Glasgow, Glasgow.

Roethlisberger, F. J., & Dickson, W. J. (1939). *Management and the Worker*. Cambridge, Mass: Harvard University Press.

Rogalski, J., & Samurçay, R. (1990). Acquisition of programming knowledge and skills. In J.-M. Hoc, T. R. G. Green, R. Samurcay & D. J. Gilmore (Eds.), *Psychology of programming* (pp. 157-174). London: Academic Press.

Rogers, C. R. (1969). *Freedom to learn: A view of what education might become*. Columbus, Ohio: Merrill.

Rosenthal, R. (1994). Parametric measures of effect size. In H. Cooper & L. V. Hedges (Eds.), *The handbook of research synthesis* (pp. 231-244). New York: Russell Sage Foundation.

Rosenthal, R., & Jacobson, L. (1992). *Pygmalion in the Classroom: Teacher Expectation and Pupils' Intellectual Development.* New York: Irvington Publishers.

Rößling, G., & Naps, T. L. (2002). *A testbed for pedagogical requirements in algorithm visualizations.* Paper presented at the 7th annual conference on Innovation and technology in computer science education (ITiCSE), Aarhus, Denmark.

Rosson, M. B., & Alpert, S. R. (1990). The Cognitive Consequences of Object-Oriented Design. *Human-Computer Interaction, 5*(4), 345-379.

Rountree, J., & Rountree, N. (2009). *Issues Regarding Threshold Concepts in Computer Science.* Paper presented at the The Eleventh Australasian Computing Education Conference (ACE2009), Wellington, New Zealand.

Rountree, J., Rountree, N., & Robins, A. (2002). Predictors of Success and Failure in a CS1 Course. *ACM SIGCSE Bulletin, 34*(4), 121-124.

Rowbottom, D. P. (2007). Demystifying threshold concepts. *Journal of Philosophy of Education, 41*(2), 263-270.

Ryan, R. M., & Connell, J. P. (1989). Perceived locus of causality and internalization: Examining reasons for acting in two domains. *Journal of Personality and Social Psychology, 57*(5), 749-761.

Ryan, R. M., & Grolnick, W. S. (1986). Origins and pawns in the classroom: Self-report and projective assessments of children's perceptions. *Journal of Personality and Social Psychology, 50*(3), 550-558. .

Sackman, H. (1970). *Man-computer problem solving*. Princeton, NJ: Auerbach.

Sackrowitz, M. G., & Parelius, A. P. (1996). An unlevel playing field: women in the introductory computer science courses. *ACM SIGCSE Bulletin, 28*(1), 37-41.

Sadlo, G. (1997). Problem-based learning enhances the educational experiences of occupational therapy students. *Education for Health, 10*(1), 101–114.

Sadlo, G., & Richardson, J. T. E. (2003). Approaches to Studying and Perceptions of the Academic Environment in Students Following Problem-Based and

Subject-Based Curricula. *Higher Education Research & Development, 22*(3), 253-274.

SAI (2008). *The Sociological Association of Ireland (SAI) ethical guidelines.*

Säljö, R. (1979). *Learning in the Learner's Perspective: Some Common-sense Conceptions* (No. 76). Göteborg: Institute of Education, University of Göteborg.

Salomon, G., & Perkins, D. N. (1989). Rocky Roads to Transfer: Rethinking Mechanism of a Neglected Phenomenon. *Educational Psychologist, 24*(2), 113-142.

Samurcay, R. (1989). The concept of variable in programming: Its meaning and use in problem solving by novice programmers. In E. Soloway & J. C. Spohrer (Eds.), *Studying the novice programmer* (pp. 161–178). Hillsdale, NJ: Lawrence Erlbaum.

Savery, J. R., & Duffy, T. M. (1995). Problem Based Learning: An Instructional Model and its Constructivist Framework. *Educational Technology, 35*(5), 31-38.

Savin-Baden, M. (2000). *Problem-based Learning in Higher Education: Untold Stories*. Buckingham: Society for Research into Higher Education & Open University Press.

Savin-Baden, M. (2006). Troublesome knowledge in problem-based learning. In J. H. F. Meyer & R. Land (Eds.), *Overcoming Barriers to Student Understanding: Threshold Concepts and Troublesome Knowledge* (pp. 160-172): Routledge.

Schmidt, H. G. (1990). Innovative and conventional curricula compared: What can be said about their effects? In Z. H. Nooman, H. G. Schmidt & E. S. Ezzat (Eds.), *Innovation in medical education: An evaluation of its present status* (pp. 1-7). New York: Springer.

Schmidt, H. G. (1994). Resolving inconsistencies in tutor expertise research: does lack of structure cause students to seek tutor guidance? *Academic Medicine, 69*(8), 656-662.

Schmidt, H. G., Henny, P. A., & de Vries, M. W. (1992). Comparing problem-based with conventional education: A review of the University of Limburg medical school experiment. *Annals of Community-Oriented Education, 5*, 193-198.

Schmidt, H. G., Loyens, S. M. M., van Gog, T., & Paas, F. (2007). Problem-Based Learning is Compatible with Human Cognitive Architecture: Commentary on Kirschner, Sweller, and Clark (2006). *Educational Psychologist, 42*(2), 91-97.

Schneider, M. G., & Gersting, J. L. (2006). *Invitation to Computer Science: Java Version* (3rd ed.). Florence, KY, USA: Course Technology, Cengage Learning.

Schoenfeld, A. H. (1979). Explicit Heuristic Training as a Variable in Problem-Solving Performance. *Journal for Research in Mathematics Education, 10*(3), 173-187.

Schoenfeld, A. H. (1980). Teaching Problem-Solving Skills. *The American Mathematical Monthly, 87*(10), 794-805.

Schoenfeld, A. H., & Herrmann, D. J. (1982). Problem perception and knowledge structure in expert and novice mathematical problem solvers. *Journal of Experimental Psychology: Learning, Memory and Cognition, 8*(5), 484-494.

Schunk, D. H. (1991). *Learning Theories: An Educational Perspective*. New York: McMillan Publishing Company.

Schwill, A. (1994). Fundamental Ideas of Computer Science. *Bulletin European Association for Theoretical Computer Science, 53*, 274-295.

Schwill, A. (1997). Fundamental Ideas: Rethinking Computer Science Education. *Learning and Leading with Technology, 25*(1), 28-31.

Seale, C. (1999). *The Quality of Qualitative Research: Introducing Qualitative Methods*. London: Sage.

Shaft, T. M., & Vessey, I. (1998). The relevance of application domain knowledge: characterizing the computer program comprehension process. *Journal of Management Information Systems, 15*(1), 51-78.

Sharp, H., Manns, M. L., & Eckstein, J. (2003). Evolving Pedagogical Patterns: The Work of the Pedagogical Patterns Project. *Computer Science Education, 13*(4), 315-330.

Sheil, B. A. (1981). The Psychological Study of Programming. *ACM Computing Surveys (CSUR), 13*(1), 101-120.

Shin, J. H., Haynes, R. B., & Johnson, M. E. (1993). The effect of problem-based, self-directed undergraduate education on lifelong learning. *Canadian Medical Association Journal.*(148), 969-976.

Shneiderman, B., & Mayer, R. E. (1979). Syntactic/semantic interactions in programmer behavior: A model and experimental results. *International Journal of Parallel Programming, 8*(3), 219-238.

Simon, S., Fincher, S., Robins, A., Baker, B., Box, I., Cutts, Q.*, et al.* (2006). *Predictors of success in a first programming course.* Paper presented at the 8th Australian conference on Computing Education Hobart, Australia.

Skaalvik, E. M. (1997). Self-Enhancing and Self-Defeating Ego Orientation: Relations With Task and Avoidance Orientation, Achievement, Self-Perceptions, and Anxiety. *Journal of Educational Psychology, 89*(1), 71-81.

Skelly, B. (2006, 27 July 2006). Weird science as students ignore buoyant tech sector Retrieved 19 October, 2008, from http://www.siliconrepublic.com/news/news.nv?storyid=single6811

Smith, G., & Escott, E. (2006). Using Animations to Support Teaching in IT. In C. Bruce, G. M. Mohay, G. Smith, I. Stoodley & R. Tweedale (Eds.), *Transforming IT Education: Promoting a Culture of Excellence* (pp. 243-255): Informing Science.

Smith, J. P., diSessa, A. A., & Roschelle, J. (1994). Misconceptions Reconceived: A Constructivist Analysis of Knowledge in Transition. *The Journal of the Learning Sciences, 3*(2), 115-163.

Solomon, Y. (1998). Teaching Mathematics: Ritual, Principle and Practice. *Journal of Philosophy of Education, 32*(3), 377-390.

Soloway, E., Adelson, B., & Ehrlich, K. (1988). Knowledge and processes in the comprehension of computer programs. In M. Chi, R. Glaser & M. Farr (Eds.), *The Nature of Expertise* (pp. 129-152). Hillsdale, NJ: Lawrence Erlbaum.

Soloway, E., Bonar, J., & Ehrlich, K. (1989). Cognitive strategies and looping constructs. In E. Soloway & J. C. Spohrer (Eds.), *Studying the novice programmer* (pp. 191-207). Hillsdale, NJ: Lawrence Erlbaum.

Soloway, E., Ehrlich, K., Bonar, J., & Greenspan, J. (1982). What do novices know about programming. In B. Shneiderman & A. Badre (Eds.), *Directions in Human-Computer Interactions* (pp. 27-54). Norwood, NJ: Ablex.

Soloway, E., & Spohrer, J. C. (1989). Novice mistakes: are the folk wisdoms correct? In E. Soloway & J. C. Spohrer (Eds.), *Studying the novice programmer* (pp. 401-416). Hillsdale, NJ: Lawrence Erlbaum.

Sooriamurthi, R. (2001). Problems in comprehending recursion and suggested solutions. *ACM SIGCSE Bulletin, 33*(3), 25-28.

Spohrer, J. C., Soloway, E., & Pope, E. (1989). A goal/plan analysis of buggy Pascal programs. *Human-Computer Interaction., 1*(2), 163 - 207.

Sproull, L., Kiesler, S., & Zubrow, D. (1984). *Encountering an Alien Culture*: Carnegie-Mellon University, Committee on Social Science Research in Computing.

Strauss, A., & Corbin, J. (1990). *Basics of Qualitative Research: Grounded Theory Procedures and Techniques* (2nd ed.). Thousand Oaks: Sage Publications.

Sun Microsystems (2008, October 2008). Java Everywhere Retrieved 19 October 2008, from http://www.sun.com/java/everywhere/

Sweller, J., Kirschner, P. A., & Clark, R. E. (2006). Why Minimal Guidance During Instruction Does Not Work: An Analysis of the Failure of Constructivist, Discovery, Problem-Based, Experiential, and Inquiry-Based Teaching. *Educational Psychologist, 41*(2), 75-86.

Sweller, J., Kirschner, P. A., & Clark, R. E. (2007). Why Minimally Guided Teaching Techniques Do Not Work: A Reply to Commentaries. *Educational Psychologist, 42*(2), 115–121.

Sweller, J., & Sweller, S. (2006). Natural information processing systems. *Evolutionary Psychology, 4*, 434–458.

Tait, H., & Entwistle, N. J. (1996). Identifying students at risk through ineffective study strategies. *Higher Education, 31*(1), 97-116.

Talbot, L. A. (1995). *Principles and practice of nursing research*: C. V. Mosby.

Tharp, A. L. (1981). *Getting more oomph from programming exercises.* Paper presented at the Twelfth SIGCSE technical symposium on Computer science education, St. Louis, Missouri, USA.

The Pedagogical Patterns Project (2001). The Pedagogical Patterns Project Retrieved 11 December, 2008, from http://www.pedagogicalpatterns.org/

Thomas, J. W. (2000). *A review of research on project-based learning*. San Rafael, CA: Autodesk Foundation.

Thomas, L., Ratcliffe, M., & Robertson, A. (2003). *Code warriors and code-a-phobes: a study in attitude and pair programming.* Paper presented at the 34th SIGCSE technical symposium on Computer science education, Reno, Navada, USA.

Tinto, V. (1975). Dropout from Higher Education: A Theoretical Synthesis of Recent Research. *Review of educational research, 45*(1), 89.

Torgerson, C. J., & Torgerson, D. J. (2001). The Need for Randomised Controlled Trials in Educational Research. *British Journal of Educational Studies, 49*(3), 316-328.

Torp, L., & Sage, S. (2002). *Problems As Possibilities: Problem-Based Learning for K-16 Education* (2nd ed.): Association for Supervision & Curriculum Development.

Trigwell, K., & Prosser, M. (1996). Changing approaches to teaching: A relational perspective. *Studies in Higher Education, 21*(3), 275-284.

Turner, V. W. (1995). *The Ritual Process: Structure and Anti-Structure*: Aldine Transaction.

UK National Audit Office (2007). *Value for Money Report: Executive Summary, Staying the course: The retention of students in higher education*. from http://www.nao.org.uk/publications/nao_reports/06-07/0607616es.htm.

Urdan, T., & Turner, J. C. (2005). Competence motivation in the classroom. . In A. J. Elliot & C. S. Dweck (Eds.), *Handbook of competence and motivation* (pp. 297-317). New York: Guilford Press.

Urdan, T. C. (1997). Achievement goal theory: Past results, future directions. In M. L. Maeher & P. R. Pintrich (Eds.), *Advances in motivation and achievement* (Vol. 10, pp. 99-142). Greenwich, CT: JAI Press.

Urdan, T. C., & Maeher, M. L. (1995). Beyond a Two-Goal Theory of Motivation and Achievement: A Case for Social Goals. *Review of Educational Research, 65*(33), 213-243.

Urdan, T. C., & Mestas, M. (2006). The Goals Behind Performance Goals. *Journal of Educational Psychology, 98*(2), 354-365.

Utley, A. (2004, 28 May 2004). Problem method has high dropout*.* *The Times,* p. 13, from http://www.timeshighereducation.co.uk/story.asp?storyCode=188968&sectioncode=26

Vallerand, R. J., & Bissonnette, R. (1992). Intrinsic, extrinsic, and amotivational styles as predictors of behaviour: A prospective study. *Journal of Personality, 60*(33), 599-620.

Vallerand, R. J., Pelletier, L. G., Blais, M. R., Briere, N. M., Senecal, C., & Vallieres, E. F. (1992). The Academic Motivation Scale: A measure of intrinsic, extrinsic, and amotivation in education. *Educational and Psychological Measurement, 52*, 1003-1017.

van Dijk, T. A., & Kintsch, W. (1983). *Strategies of discourse comprehension*. New York: Academic Press.

van Gennep, A. (2004). *The Rites Of Passage* (M. Vizedom & G. L. Caffee, Trans. 2nd ed.). London: Routledge.

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

van Grinsven, L., & Tillema, H. (2006). Learning opportunities to support student self-regulation: comparing different instructional formats. *Educational Research, 48*(1), 77 - 91.

van Rossum, E. J., & Taylor, I. P. (1987). *The relationship between conceptions of learning and good teaching: A scheme of cognitive development.* Paper presented at the Annual meeting of the American Educational Research Association, Washington DC, USA.

van Roy, P., & Haridi, S. (2004). *Concepts, Techniques, and Models of Computer Programming*. Cambridge, Massachusetts, USA: MIT Press.

Velázquez-Iturbide, J. Á. (2000). *Recursion in gradual steps (is recursion really that difficult?).* Paper presented at the thirty-first SIGCSE technical symposium on Computer science education, Austin, Texas, United States.

Venkatesh, V., & Davis, F. D. (1996). A Model of the Antecedents of Perceived Ease of Use: Development and Test. *Decision Sciences, 27*(3), 451-481.

Vernon, D. T., & Blake, R. L. (1993). Does problem-based learning work? A meta-analysis of evaluative research. *Academic Medicine, 68*, 550-563.

Visser, W. (1990). More or Less Following a Plan During Design: Opportunistic Deviations in Specification. *International Journal of Man-Machine Studies, 33*(3), 247-278.

Visser, W., & Hoc, J.-M. (1990). Expert software design strategies. In J.-M. Hoc, T. R. G. Green, R. Samurcay & D. J. Gilmore (Eds.), *Psychology of programming* (pp. 235–250). London: Academic Press.

von Mayrhauser, A., & Vans, A. M. (1994). *Program Understanding-A Survey* (No. CS-94-120): Colorado State University Computer Science Technical Report CS94-120.

von Mayrhauser, A., & Vans, A. M. (1995a). Program comprehension during software maintenance and evolution. *Computer, 28*(8), 44-55.

von Mayrhauser, A., & Vans, A. M. (1995b). Program Understanding: Models and Experiments. *Advances in Computers, 40*(4), 25-46.

von Mayrhauser, A., Vans, A. M., & Howe, A. E. (1997). Program Understanding Behaviour during Enhancement of Large-scale Software. *Journal of Software Maintenance Research and Practice, 9*(5), 299-327.

Vygotsky, L. S. (1978). *Mind in Society: Development of Higher Psychological Processes* (14th ed.): Harvard University Press.

Waite, W. M., Jackson, M. H., & Diwan, A. (2003). *The conversational classroom.* Paper presented at the The 34th SIGCSE technical symposium on Computer science education, Reno, Navada, USA.

Wallace, M. J. (1998). *Action research for language teachers*: Cambridge University Press.

Weinberg, G. M. (1971). *The psychology of computer programming*: Van Nostrand Reinhold.

Weiner, B. (1983). Some Methodological Pitfalls in Attributional Research. *Journal of Educational Psychology, 75*(4), 530-543.

Wentzel, K. R. (1989). Adolescent classroom goals, standards for performance, and academic achievement: An interactionist perspective. *Journal of Educational Psychology, 81*, 131-142.

Wentzel, K. R. (1993). Social and academic goals at school: Motivation and achievement in early adolescence. *Journal of Early Adolescence, 13*(1), 4-20.

Wheelan, S. A., & Williams, T. (2003). Mapping dynamic interaction patterns in work groups. *Small Group Research, 34*(4), 443-467.

Widowski, D., & Eyferth, K. (1986). Comprehending and recalling computer programs of different structural and semantic complexity by experts and novices. In H. P. Willumeit (Ed.), *Human decision making and manual control* (pp. 267-275). Amsterdam: North-Holland Elsevier.

Wiedenbeck, S. (1988). Learning Recursion As a Concept and As a Programming Technique. *ACM SIGCSE Bulletin, 20*(1), 275-278.

Wiedenbeck, S., LaBelle, D., & Kain, V. N. R. (2004, April). *Factors affecting course outcomes in introductory programming.* Paper presented at the 16th Annual Workshop of the Psychology of Programming Interest Group, Carlow, Ireland.

Wiedenbeck, S., & Ramalingam, V. (1999). Novice comprehension of small programs written in the procedural and object-oriented styles. *International Journal of Human-Computer Studies, 51*(1), 71-87.

Wiedenbeck, S., Ramalingam, V., Sarasamma, S., & Corritore, C. L. (1999). A comparison of the comprehension of object-oriented and procedural programs by novice programmers. *Interacting with Computers, 11*(3), 255-282.

Wilkerson, L., & Feletti, G. (1989). Problem-based learning: One approach to increasing student participation. The Department Chairperson's Role in Enhancing College Teaching. *New Directions for Teaching and Learning* (37), 51-60.

Williams, G. C., & Deci, E. L. (1996). Internalization of biopsychosocial values by medical students: A test of self-determination theory. *Journal of Personality and Social Psychology, 70*, 767-779.

Williams, G. C., & Deci, E. L. (2007a, 27-Nov-2006). Learning Self-Regulation Questionnaire (SRQ-L) Retrieved 29 January, 2009, from http://www.psych.rochester.edu/SDT/measures/selfreg_lrn.html

Williams, G. C., & Deci, E. L. (2007b). Self-determination theory Retrieved 2 August 2007, from http://www.psych.rochester.edu/SDT/theory.html

Williams, L. (2007). IT undergraduates have UK's highest dropout rate Retrieved 19 October 2008, from http://www.computing.co.uk/computing/news/2195437/undergraduates-uk-highest

Williams, L., Wiebe, E., Yang, K., Ferzli, M., & Miller, C. (2002). In Support of Pair Programming in the Introductory Computer Science Course. *Computer Science Education, 12*(3), 197-212.

Wills, C. E., Deremer, D., McCauley, R. A., & Null, L. (1999). Studying the Use of Peer Learning in the Introductory Computer Science Curriculum. *Computer Science Education, 9*(2), 71-88.

Winslow, L. E. (1996). Programming pedagogy—a psychological overview. *ACM SIGCSE Bulletin, 28*(3), 17-22.

Wittrock, M. C. (1989). Generative processes of comprehension. *Educational Psychologist, 24*(4), 345–376.

Wolters, C. A., Yu, S. L., & Pintrich, P. R. (1996). The relation between goal orientation and students' motivational beliefs and self-regulated learning. *Learning and Individual Differences, 8*, 211-238.

Woods, D. R. (1996). *Problem-based Learning: How to gain the most from PBL.* Ontario, Canada: Waterdown.

Woods, D. R., Hall, F. L., Eyles, C. H., Hrymak, A. N., & Duncan-Hewitt, W. C. (1996). Tutored versus tutorless groups in problem-based learning. *American Journal of Pharmaceutical Education, 60*, 231-238.

Yehezkel, C., Ben-Ari, M., & Dreyfus, T. (2005). *Computer architecture and mental models*. Paper presented at the 36th SIGCSE technical symposium on Computer science education.

Zendler, A., & Spannagel, C. (2008). Empirical Foundation of Central Concepts for Computer Science Education. *Journal on Educational Resources in Computing, 8*(2), 1-15.