Approaches to Support Student Learning in Introductory Programming Laboratory Classes

Ph.D. Thesis

September 2010

Adam Low

Technology Enhanced Learning Research Group
School of Engineering and Computing Sciences
Durham University

Abstract

Objectives: This thesis will explore some innovative solutions to communication difficulties that exist in higher education teaching of introductory programming. Communication between a teacher and student is important, as it is the main opportunity where a student can ask a teacher questions about a particular problem they have, and a teacher can give feedback to direct them towards a solution. It is expected that through utilising technology in laboratory practical classes, communication between teachers and student can be improved.

Methods: This thesis primarily explores the possibilities of using student compiler and method invocation data, collected during a practical class and sent directly to a teacher. This data maybe beneficial as a method of allowing teachers to see if a student requires help. This thesis utilises a variety of research methods including questionnaires, observations of classroom interactions and collection of data recorded from student and teachers interactions with the technology. The approaches are used during an investigation into the current approaches of laboratory practical teaching, before progressing onto investigations using the technology developed that accompanies this thesis.

Results: The results identified that the majority of the students and teachers who used the technology felt that it improved their ability to communicate within laboratory practical classes. The teachers felt that they could use the data collected by the technology to view activity from the students and see a student's progress. The teachers could interpret the data collected from the technology and students who needed help could be identified.

Conclusions: This thesis has demonstrated that technology has the potential to improve communication in laboratory classes, and enable teachers to support students more effectively. However, the technology developed in this thesis, does not eliminate the requirement for a teacher to interact with a student face-to-face, but rather its role is to act as an indicator of students who may need assistance.

Declaration of Authorship

I, Adam Low, declare that this thesis titled, 'Approaches to Support Student Learning in Introductory Programming Laboratory Classes' and the work presented in it are my own. I confirm that no part of the material provided has previously been submitted by the author for a higher degree in Durham University or any other University. All the work presented here is the sole work of the author.

This research has been documented or is related, in part, within the publications listed below:

- A. Low. 'TR-TEL-08-04: Technology to Support Interactivity in Introductory Programming Lectures' Technology Enhanced Learning Research Group Technical Report Series, Durham University, August 2008
- J. Lavery, & A. Low. **Concept Mapping in Lectures.** Proceedings of 9th Conference of the Subject Centre for Information and Computer Sciences, Liverpool, England. 2008.
- A. Low, L. Burd & A. Hatch. **Technologically Enhanced Demonstrator Support - Tools for Assisting Demonstrators in First Year Programming Laboratory Classes.** 10th Conference of the Subject Centre for Information and Computer Sciences, Kent, England. 2009.

Acknowledgements

I would like to acknowledge and thank my friends (especially Ryan Ford, Tam & Andy Burn, Christopher Foster and Stephen Cummins) who supported me and provided escapes from my work when it was needed.

Thank you to Professor Liz Burd And Doctor Andrew Hatch, for providing invaluable guidance and technical advice for this research.

I would like to thank Professor Keith Gallagher and Stephan Jamieson, for allowing me to test TEDS within their courses. Without their support this thesis would have been very short.

Special thanks to Marion Low, Elizabeth Low, Jane Low, Ramona Scheufele and Stephen Cummins for either proof reading my work or giving me advice on the statistical portion of this research. Their support and assistance was invaluable.

I would like to acknowledge the Centre for Excellence in Teaching and Learning: Active Learning in Computing for funding this research.

I would like to finally thank my family, who have always supported me when I have needed it, and especially my wife, Marion, who has put up with me and kept me going through all of the low points.

Contents

Abstract ii

Declaration of Authorship iii

Acknowledgements iv

Contents v

List of Figures xii

List of Tables xv

1. Introduction 1
   1.1. Research Phases 1
      1.1.1. Research Phase (i) – Analyse existing methods of teaching programming and identify issues with the current methods 2
      1.1.2. Research Phase (ii)– Develop technologies to overcome identified issues 2
      1.1.3. Research Phase (iii)– Carry out experiments to see the potential of the technologies to overcome any discovered issues 3
   1.2. Layout of the Thesis 3
   1.3. Aim of the thesis 4
   1.4. Demonstrators 4
   1.5. Scope 4
2. Literature Review 6
   2.1. Pedagogy 6
      2.1.1. Constructivism and Directed Guidance 6

2.1.2. Constructive Alignment 7

2.2. Modes of Teaching in Higher Education 10

2.2.1. Lectures 10

2.2.2. Practicals and problem based learning 19

2.3. Teaching Programming 22

2.3.1. The problems of teaching programming 22

2.3.2. Existing methods of teaching programming 25

2.3.3. Technology 25

2.4. Summary 29

3. Investigating Learning and Teaching In Current Practical Setting 30

3.1. Method 31

3.2. Results 32

3.2.1. Group A 32

3.2.2. Group B 35

3.3. Evaluation 37

3.3.1. Issue 1 – Students are afraid to openly initiate dialogue with teachers 37

3.3.2. Issue 2 – Communication Difficulties 41

3.3.3. Issue 3 – Students' Range of Programming Abilities 50

3.3.4. Issue 4– More Independent Learning 51

3.3.5. Issue 5– Visibility of Students' Progress 52

3.4. Summary 54

3.4.1. Research Question 1 – How highly do students value practicals? Do students and teachers find communication difficult in existing practicals? Do teachers find it difficult to track progress in the existing practicals setting? 54

3.4.2. Research Question 2 – How do students begin interactions with teachers in the existing practical setting? 55

3.4.3. Research Question 3 – To what extent can teachers perceive student status in the existing practical setting? 55

4. Technologically Enhanced Demonstrator Support 57

4.1. Design Principles 57

4.2. Design Investigations 58

4.2.1. Exploration of the capabilities of the BlueJ extension library 58

4.2.2. Focus groups and interviews 61

4.2.3. During Use 65

4.3. Features Identified 66

4.4. The Features of TEDS 67

4.4.1. Teacher Client 69

4.4.2. Feature A - Compiler Errors 72

4.4.3. Feature B - Method Invocation 78

4.4.4. Feature C - Code Snapshot 80

4.4.5. Feature D - Help Button 82

4.4.6. Feature E - Short Message Functionality 84

4.4.7. Feature F - Objective Setting 84

4.4.8. Feature G – Image Sending 86

4.5. Summary 87

5. Implementation 89

5.1. Design Issues 89

5.2. Server Architecture 91

5.2.1. Connection Manager 93

5.2.2. User Manager 93
5.2.3. Functions Manager 94
5.2.4. Database Manager 94
5.3. Clients 95
5.3.1. Teacher Client 95
5.3.2. Student Client 97
5.4. Summary 98
6. Case Studies Design 99
6.1. Methods 99
6.1.1. User Feedback 100
6.1.2. Observations 102
6.1.3. Automated Data Collection 104
6.2. Case Studies 105
6.3. Sample Selection 108
6.4. Limitations 108
6.5. Summary 112
7. Case Studies 113
7.1. Case Study One 113
7.1.1. Results 114
7.1.2. Evaluation 118
7.1.2.1. The benefits of smaller groups 118
7.1.2.2. Runtime Errors 119
7.1.2.3. Teachers Behaviours Using TEDS 120
7.2. Case Study Two 120

7.2.1. Results 121
7.2.1.1. Results from Group A Session (i) 121
7.2.1.2. Results from Group A Session (ii) 124
7.2.1.3. Results from Group B Session (i) 127
7.2.2. Evaluation 130
7.2.2.1. The majority of students' did compile/run code 130
7.2.2.2. Some students do not execute their code 131
7.2.2.3. Types of compiler errors are common across the cohort 132
7.2.2.4. Some students did not run or compile code 134
7.3. Case Study Three 135
7.3.1. Results 135
7.3.1.1. Results from Group A Session (iii) 136
7.3.1.2. Results from Group B Session (ii) 148
7.3.1.3. Aggregated Student Errors 152
7.3.1.4. Student Questionnaires 154
7.3.2. Evaluation 156
7.3.2.1. TEDS shows teachers students' status 157
7.3.2.2. TEDS reveals 'movers, 'stoppers and 'extreme movers' 166
7.3.2.3. TEDS enabled better teacher to student communication 172
7.3.2.4. The diagramming tool is beneficial for supporting the students 175
7.3.2.5. Teachers did not use all of the functions 176
7.3.2.6. Students' compiler errors were grouped together 177
7.3.2.7. Positive Responses in the questionnaires 178
7.4. Summary 179

7.4.1. Research Question 4 - In what areas can TEDS change the way that teachers track student status? 180

7.4.2. Research Question 5 - In what areas can TEDS change the way that teachers and students interact? 180

8. Conclusions 182

8.1. Research Phases 182

8.1.1. Research Question 1 – What are the students' and teachers' opinions on the pedagogic value of practicals? 182

8.1.2. Research Question 2 – How do students begin interactions with teachers in the existing practical setting? 183

8.1.3. Research Question 3 – To what extent can teachers perceive student status in the existing practical setting? 184

8.1.4. Research Question 4 – In what areas can TEDS change the way that teachers track student status? 184

8.1.5. Research Question 5 – In what areas can TEDS change the way that teachers and students interact? 185

8.2. Limitations 186

8.3. Further Work 186

8.4. Summary 187

Appendix 1 – Teacher Preliminary Case Study Questionnaire 189

Appendix 2 – Student Preliminary Case study Questionnaire 191

Appendix 3 – Teacher Post Main Case Study Questionnaire 193

Appendix 4 – Student Post Main Case Study Questionnaire 195

Appendix 5 – Observation Tally Chart Sheet 197

References.. 198

List of Figures

Figure 2.1: Student orientation, teaching method and level of engagement [Big02 pp 4] 9

Figure 2.2: Laurillard's Conversational Framework [Lau06 pp. 87] 12

Figure 2.3: Laurillard's Conversational Framework in regards to the traditional lecture format [Lau06] 14

Figure 2.4: The Problem Based Learning Cycle [Hme04 pp237] 20

Figure 3.1: Group A comparison between students' total time of interactions and the average time per interaction 33

Figure 3.2: Group A amount of student and teacher interactions and the breakdown of who initiated these interactions 34

Figure 3.3: Group B comparison between students' total time of interactions and the average time per interaction 35

Figure 3.4: Group B amount of student and teacher interactions and the breakdown of how these interactions began 36

Figure 3.5: Example Classroom 42

Figure 3.6: Thematic map of Group A's student interaction by total amount 43

Figure 3.7: Thematic map of the frequency of interactions initiated by teachers in Group A 44

Figure 3.8: Thematic map of the frequency of interactions initiated by students in Group A 45

Figure 3.9: Thematic map of Group B's student interaction by amount 46

Figure 3.10: Cutts version of Laurillard Conversational Framework [Cut05] 48

Figure 4.1: Teacher Console Version 1 70

Figure 4.2: Teacher Console Version 2 70

Figure 4.3: Teacher client 72

Figure 4.4: View of compiler status 75

Figure 4.5: Students' computer ID's 76

Figure 4.6: Graph to show a student's percentage of successful compiles 77

Figure 4.7: List of groups of compiler errors 77

Figure 4.8: Pie chart of combined groups compiler errors 78

Figure 4.9: View of method invocation status 80

Figure 4.10: Example of student code view 82

Figure 4.11: View of the student help box 83

Figure 4.12: View of the teachers view rows associated with the 'help button" feature 83

Figure 4.13: Teacher objective setting window 86

Figure 4.14: Student view of the set objective 86

Figure 4.15: Image creating tool 87

Figure 5.1: Server Managers 92

Figure 7.1: Comparison between students' total time of interactions with teachers and average time per interaction in a practical 136

Figure 7.2: The amount of interactions by students and who initiated the interactions 137

Figure 7.3: Showing the total groups breakdown of how interactions began 138

Figure 7.4: Comparison between student's total time of interactions and average time per interaction in a practical. (Students not using TEDS) 139

Figure 7.5: The amount of interactions by students and how these interactions began (Students not using TEDS) 140

Figure 7.6: Showing the breakdown of how interactions began (Students not using TEDS) 141

Figure 7.7: Comparison between students' total time of interactions and average time per interaction in a practical. (Students using TEDS) 142

Figure 7.8: The amount of interactions by students and how these interactions began (Students using TEDS) 143

Figure 7.9: Showing the breakdown of how interactions began (Students using TEDS 144

Figure 7.10: Student A's compile success rate 158

Figure 7.11: Student A's compile success rate and method invocations 159

Figure 7.12: Student B's compile success rate 162

Figure 7.13: Example of a stopper 169

Figure 7.14: Example of "Extreme Mover" 171

Figure 7.15: Screenshot of Image for student 176

List of Tables

Table 3.1: Table presenting comparison of observations taken from Group A and Group B 39

Table 4.1: Presenting link between features and investigations 67

Table 4.2: Map of features to issues 69

Table 5.1: Advantages and Disadvantages of different portable computer options 96

Table 6.1: Table of different case studies by groups 106

Table 7.1: Case Study One session data 115

Table 7.2: Case Study One, grouped compiler errors 117

Table 7.3: Group A Session (i), student data 122

Table 7.4: Group A Session (i), grouped compiler errors 123

Table 7.5: Group A Session (ii), student data 124

Table 7.6: Group A Session (ii), grouped compiler errors 126

Table 7.7: Group B Session (i), student data 127

Table 7.8: Group B Session (i), groups combined compiler errors 129

Table 7.9: Group A Session (iii), student data 145

Table 7.10: Group A Session (iii), collated groups compiler errors 147

Table 7.11: Group B Session (ii), student data 149

Table 7.12: Group B Session (ii), collated compiler errors 151

Table 7.13: Whole set of compiler errors collected by TEDS over the three case studies 153

Table 7.14: T-Tests on data collected by TEDS during Case studies two and three 160

Table 7.15: T-Tests on observation data taken during Case Study Three 164

1. Introduction

Communication is a vital part of a student's learning [Lau06] at any level, either by the teacher conveying knowledge to a student or a student asking questions of a teacher. Advancements in ubiquitous computing over recent times have enabled developers to create systems that assist communication in the classroom. One instance of an application of ubiquitous computing to improve communication in the classroom is the use of Personal Response Systems (PRS) [Cut01].

Along with communication, teaching introductory programming within higher education is also viewed as difficult, and has been the focus of recent research. For example Scheele [Sch05] and Cutts [Cut01] explore developing interactive lectures to increase their efficacy using tools such as PRS, while researchers like Jadud [Jad05] and English [Eng09] investigate using technology to increase the efficacy of *practicals*. *Practicals* in the context of this work are computer laboratory classes, where students are required to complete work based on what they have been taught in lectures. *Practicals* are differentiated from the term practical by the use of italics.

This thesis bridges these areas and focuses on developing new technologies to improve student support within *practicals* on teaching programming.

1.1. Research Phases

Three research phases are completed in this thesis. Supporting these research phases, five research questions further clarify what the research aims to discover during the completion of the research phases. The five research questions are addressed during the completion of each research phase. The following three points summarise the research phases:

Research Phase i. Analyse existing methods of teaching programming and identify issues with the current methods in *practicals*.

Research Phase ii. Develop technologies to overcome these issues in *practicals*.

Research Phase iii. Carry out experiments to see the potential of the technologies to overcome any discovered issues in *practicals*.

Each of these research phases are explored in more detail below by explaining the motivation for each theme and presenting the research questions that are asked on the topic of each phase.

1.1.1. Research Phase (i) – Analyse existing methods of teaching programming and identify issues with the current methods

The motivation behind the Research Phase (i) is to explore the advantages and disadvantages of the current methods of teaching introductory programming. The method of finding these advantages and disadvantages are analysed through the literature within the research area and by investigating a live introductory programming module.

In regards to Research Phase (i) the following questions and areas are considered:

Research Question 1. What are the students' and teachers' opinions on the pedagogic value of *practicals*?

Research Question 2. How do students begin interactions with teachers in the existing practical setting?

Research Question 3. To what extent can teachers perceive student status in the existing practical setting?

These questions assist in providing answers that dictate the technology created for Research Phase (ii). In the context of this work the term *status* is used, status relates to how a student is doing at a certain point in *practicals*, for instance, if their last compile or method invocation was successful or unsuccessful.

1.1.2. Research Phase (ii)– Develop technologies to overcome identified issues

Research Phase (ii) explores whether technology can be developed to overcome the issues that currently exist in *practicals*. None of the research questions are directly

associated to this phase, yet the developed tool is used in relation to Research Phase (iii).

1.1.3. Research Phase (iii)– Carry out experiments to see the potential of the technologies to overcome any discovered issues

The motivation behind the Research Phase (iii) is to investigate the potential that any developed technologies have in improving teachers' support of students in *practicals*. Research Questions 4 and 5 are used in relation to Research Phase (iii) to explore the opinions that both the students and the teachers have of the developed technology (TEDS), especially in regards to its potential as a teaching tool.

In response to the Research Phase (iii) the following areas were considered:

Research Question 4. In what areas can TEDS change the way that teachers track student status?

Research Question 5. In what areas can TEDS change the way that teachers and students interact?

1.2. Layout of the Thesis

The thesis has chapters that can be mapped to each of the research phases.

Chapters 2 and 3, which are the literature chapters and the existing *practicals* setting, look into the existing ways of teaching programming within higher education. As a result, the first two chapters focus on Research Phase (i).

Chapters 4 and 5, are the design and implementation chapters, and look at the development of a tool called 'Technologically Enhanced Demonstrator Support' (TEDS), which is designed to help teachers in *practicals* setting. For the duration of the research the system is referred to as TEDS and is the tool that is created for examining the data collected in Research Phase (iii).

Chapters 6 and 7 are concerned with the investigations that test the efficacy of TEDS within a practical setting, to investigate Research Phase (iii).

1.3. Aim of the thesis

In summary the main aim of this work is to investigate if technology can improve teachers' support of students in programming *practicals*. In response to this aim the thesis uses the three research phases and their related research questions.

During Research Phase (i) any issues in the literature and during investigations into introductory programming *practicals* are identified. These findings lead to the creation of technologies (TEDS) and then investigations into their capability to improve student support within programming *practicals*.

1.4. Demonstrators

In Durham University the teaching staff that assist students in *practicals* are known as demonstrators. Demonstrators are usually postgraduate students from within the department and in some cases have previously studied the modules that they now assist.

The role the demonstrator plays in *practicals* is that of a teacher helping the students with problems and leading them to solutions, therefore teacher is a suitable and more widely understood title for the demonstrators at Durham University. In this thesis, since 'demonstrator' is a term that is not common to all institutions, the term *teacher* will be used.

1.5. Scope

The environment where the research takes place dictates the scope of this thesis. In the case of this thesis the environment is the introductory programming module at Durham University. Caution has to be taken when producing any generalisations from the findings of this work, as the research needs to take into account the environment in which the research is carried out. For instance, students studying introductory programming at Durham University are taught the Java programming language, whereas other institutions may use different languages.

Additional factors that could influence the scope of this work involve applying the research to different students (i.e. with stronger or weaker students), different teaching staff or different teaching methods (i.e. more or fewer lectures/*practicals*).

Despite these limitations to the scope of this work, the environment does still have some benefits. Durham University uses a typical teaching method of lectures supported by *practicals*, which is common in most universities. Also Java is a widely taught programming language in introductory programming courses.

2. Literature Review

This chapter presents relevant teaching techniques used in higher education. Responses in literature are provided as to why University taught courses are designed with an emphasis on lectures. Some examples of how courses are taught at University are also presented.

This work is cross disciplinary with a focus on Computer Science, but has elements of Education and Psychology as well, which is reflected in this chapter.

The chapter is split into three areas:

1. Pedagogy – Considering different teaching theories related to teaching students in higher education.
2. Methods of teaching in higher education – Exploring different methods used to teach in higher education institutions.
3. Teaching programming – Focusing on how introductory programming is taught in higher education institutions.

Exploring each of these areas gives an overview of the context in which this thesis takes place.

2.1. Pedagogy

In present day higher education, two theories have shaped the currently accepted best practice. Here the two theories are presented as well as their culmination in constructive alignment (section 2.1.2).

2.1.1. Constructivism and Directed Guidance

Constructivism [Ben01] and Directed Guidance [Kir06] differ in the explanation of how knowledge acquisition occurs in learners. The way the two pedagogies differ is in how students acquire and assimilate knowledge.

Advocates of Constructivism believe that the student can actively construct knowledge, rather than the students' passively absorbing knowledge from textbooks and lectures. The construction builds recursively on knowledge that the student already has, which means that each student will construct his or her own personalised version of the knowledge [Ben01 pp45]. In essence, Constructivism relies on the students being allowed to construct their own versions of the knowledge and teachers acting as facilitators for the students in developing their versions.

Directed Guidance is much more prescriptive. Concepts and procedures of the topic are explained fully through lectures and textbooks. In Directed Guidance, rather than the knowledge being constructed, the act of learning is defined as "a change in long-term memory" [Kir06 pp75]. Through Directed Guidance the students are taught the procedures and the concepts that are required for the material on the course.

In higher education a range of teaching methods are used. These include both methods that relate to Directed Guidance (lectures) and Constructivism (problem based learning) as they operate together in Constructive Alignment. Constructive Alignment is the framework widely used in higher education in the UK.

2.1.2. Constructive Alignment

Constructive Alignment is the approach Biggs [Big02] outlines as the most appropriate for teaching students within higher education. Biggs defines Constructive Alignment as a system where teaching methods and assessment are aligned to the learning objectives of the course [Big02, page 11].

Wide acceptance of Constructive Alignment can clearly be seen by the stance adopted by the QAA (Quality Assurance Agency for Higher Education) [Jac02]. Constructive Alignment consists of two dimensions *"What the teacher does to learn and promote students' learning"* and *"what the students do to learn and promote their own learning"* [Jac02]. The term Constructive Alignment can be analysed in terms of how it relates to the students and the teacher. The *constructive* element refers to what the learner does to construct their own meanings through the teaching and learning activities in which they participate. The *alignment* element refers to the

teacher and his or her goal to create a learning environment that will allow the students to achieve the desired learning outcomes. Jackson [Jac02] admits that both of the terms could apply equally to student and lecturer, where lecturers can construct meaning through their teaching and students can align their learning to the teachers' learning outcomes.

The main aim of Constructive Alignment and its associated teaching techniques is to encourage deeper approaches to learning and discourage surface learning [Big02]. Biggs argues that Constructive Alignment would allow for the change from the current higher educational teaching framework of supporting a surface style to deep learning. Surface learning is an approach to learning where facts are learnt so that they can then be recited or memorised and regurgitated during assessment, for instance, within examinations. Through the implementation of Constructive Alignment, an environment would be created where the students would be actively encouraged to explore deeper approaches to learning. Deep approaches to learning include theorizing, reflecting, generating and applying. In this 'Deep' learning, students are encouraged to use more thought rather than just memory. Deeper approaches are more widely seen during seminar discussions and *practicals* where the application of knowledge supersedes the learning of the knowledge [Big02].

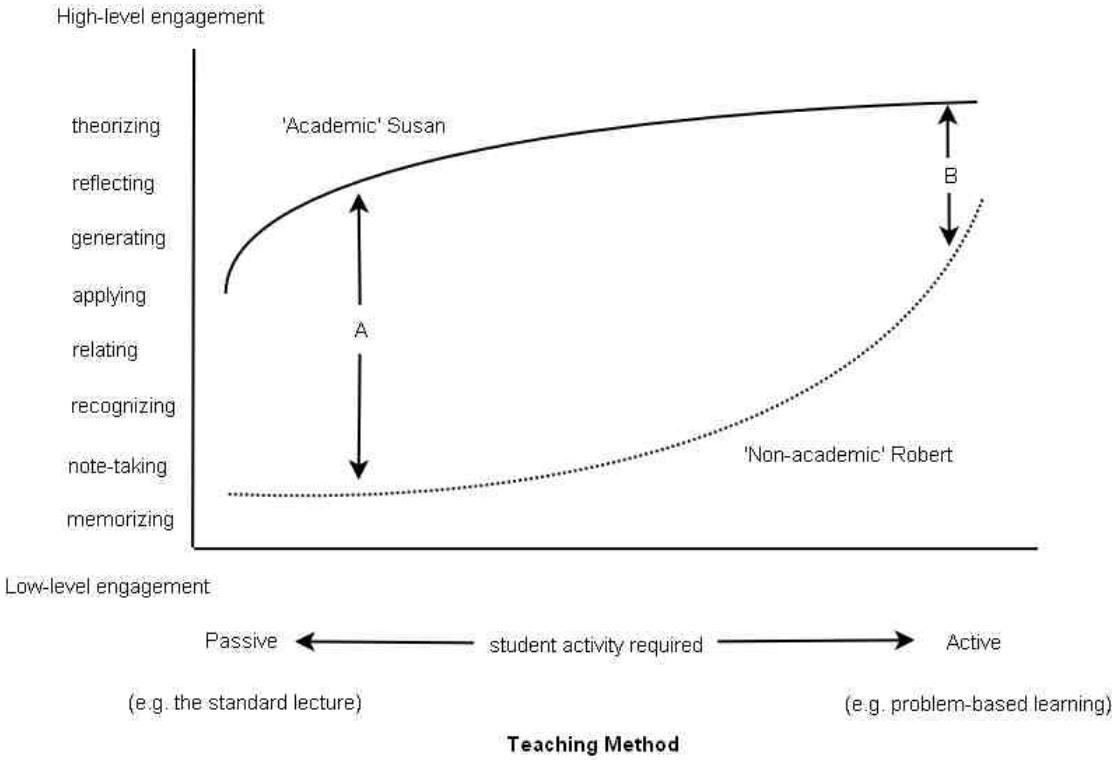


Figure 2.1: Student orientation, teaching method and level of engagement [Big02 pp 4]

Figure 2.1 shows what Biggs views as the two spectrums of student that study within higher education. At the bottom is the 'non-academic', Robert, who seeks to do the bare minimum to pass the course to go on to get a good job and is more inclined to adopt surface learning techniques. At the top is the 'academic', Susan, who has more intrinsic motivation to be successful at an academic level and has a higher inclination to adopt deep learning techniques. Figure 2.1's presents how these two different types of learners react within different teaching methods. Arrow A highlights that the largest gap between the two learners is during the passive standard lecture and the smallest gap is indicated by arrow B that is during active learning. An example of active learning is students actively applying knowledge the students have been told or read. Biggs seeks to show within Figure 2.1 that active learning can motivate students to use deeper methods of learning which passive learning, for example the expository lecture, does not [Big02]. Yet this is not an indication that lectures are all bad; there are ways of making lectures more active and thus stimulating deeper

learning within them. It is important to note that although some students may in some modules use deeper learning methods this does not necessarily mean that they will use deep learning methods for all modules.

2.2. Modes of Teaching in Higher Education

Universities in the UK vary in the way that they teach courses. Some put increased focus on seminars and support *practicals* to assist students whereas others choose lectures as the main way of delivering content. It not only varies from institution to institution but also from department to department, and module to module. At the School of Engineering and Computing Sciences in Durham [Dur10], a typical programme begins with more support *practicals* at level 1 than lectures. At level 2 students get fewer support *practicals* and at level 3 the students have almost no support *practicals* on the modules, just lectures. Student support classes are gradually phased out as students make progress. The principal aim for this practice is to increase student independence and make them more prepared for work after University.

Lectures have existed as a method of teaching for centuries and are *"an established and popular way of knowledge transfer"* [Sch05 pp6]. Yet they have been a widely criticised teaching method, due to their passive nature, where students are asked to just listen to the lecturer [Big02, Bli98].

Lectures and their support classes are explored below.

2.2.1. Lectures

Modern day higher education courses are dominated by lectures, where a lecturer explains a concept to a large group of students who passively listen and these are supported by *practicals*, where a student actively applies the knowledge that they have been taught within the lecture. According to the module descriptions at Durham University, the majority of modules in level 1 computer science are split almost equally between lectures and 'hands-on' support *practicals*. For example, in the single module first year *Introduction to programming* course, there is a 50-50 split

between lectures and *practicals*. In the double module of *Introductory Physics*, also at Durham University, there are 116 hours of lectures, 20 hours of tutorials and 10 hours of workshops [Dur10]. Durham University's modules are indicative of current practice, where the majority of teaching consists of lectures and *practicals*.

Lectures are often seen as a method for inspiring surface learning. The expository style of the lecturer, 'exposing' facts to their students, allows them to either memorise or take notes for reference, in preparation for exams [Gib92, Bli98]. Biggs agrees that lectures do appeal more to surface learning yet believes that with a combination of a good lecturer and students more open to deeper learning techniques, lectures can still be successful. The conveyance and description of facts and methods during lectures also aligns them to the Directed Guidance pedagogy (section 2.1.1).

Lectures are seen as one of the best methods for a teacher to transmit his or her knowledge to a large group of students in the most efficient way possible in regards to both lecture hall space (student numbers do not matter) and teacher work hours [Big02]. The passivity of the lecture is often criticised, as active learning is what most students require to learn effectively. The passivity of the lecture leads to the students being asked to listen to the teacher for the duration of the lecture, which many studies show, is not an effective method for acquiring knowledge [Llo68].

Laurillard [Lau06] and, to a certain extent, Biggs [Big02] support a view that if we were to consider higher education teaching again with a clean slate, the lecture would not be the method we would eventually agree on to be the dominant method of teaching. They both agree that students require more interaction to achieve the deeper forms of learning, that Biggs would like students to achieve whilst studying in a constructively aligned learning environment.

The limitations of the lecture can be lessened by the adaptation of more interactive forms of lecturing [Lau06, Big02, Bli98]. These interactive elements have the tri-purpose of maintaining student attention, instigating students into exploring deeper learning styles and maintaining a link between the student and the lecturer where any

cognitive conflicts can be identified and resolved [Bli98, Sch05]. This final point of interactive elements will allow a form of conversation to develop between the lecturer and the student. This method is one that has been identified by Laurillard and formulated into a theory called the Conversational Framework [Lau06]. A diagram of the ideal implementation of a Conversational Framework can be seen in Figure 2.2.

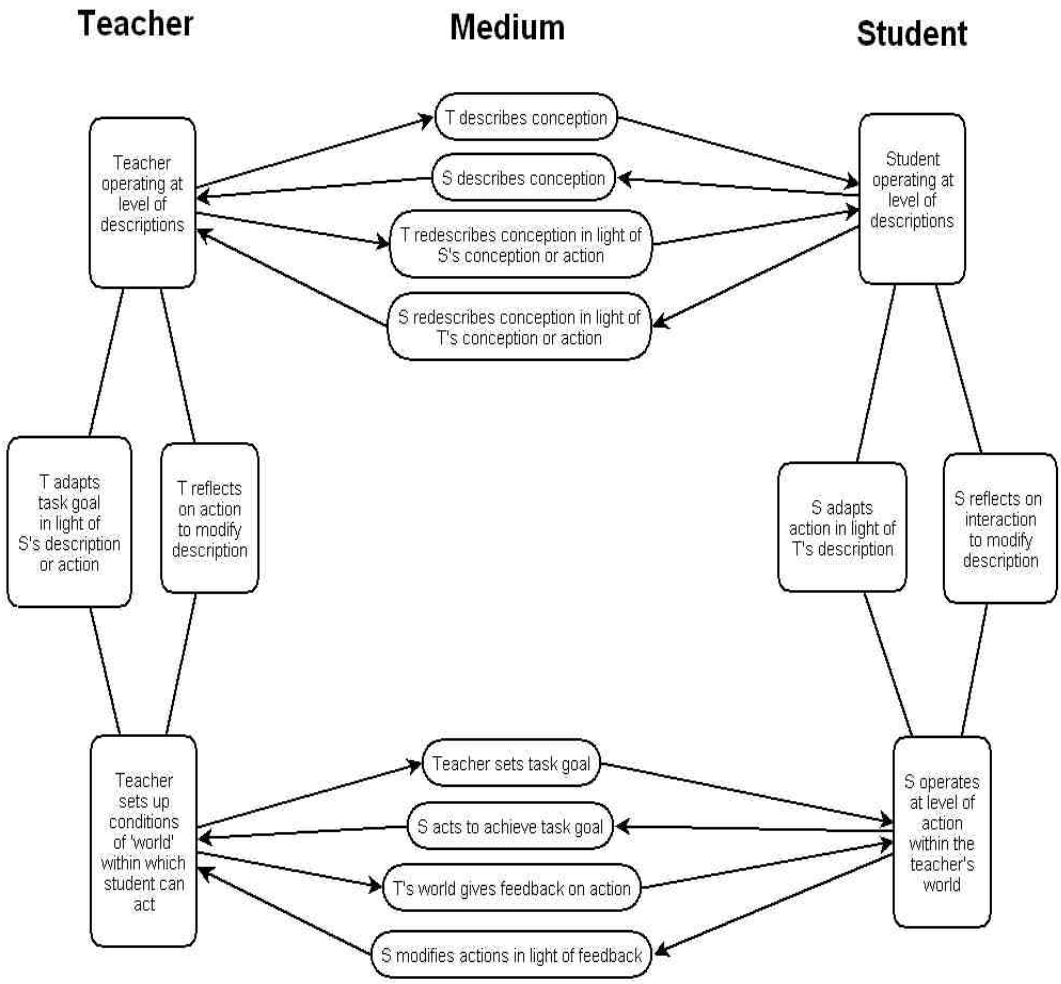


Figure 2.2: Laurillard's Conversational Framework [Lau06 pp. 87]

Laurillard's model can be seen as a form of the Socratic dialogue model, in the way that the second part of the conversation is concerned with the teacher setting a problem for the student to attempt to complete. The Socratic dialogue model is described as being *"teaching by questioning"* by the Joint Information Systems

Committee (JISC) and developed at Harvard University to attempt to re-engage students [JISC06], more detail in regards to the Socratic dialogue method are given by Ravenscroft et al [Rav00]. Laurillard does not demonstrate links to the Socratic dialogue method within her Conversational Framework, as, despite her method also being a dialogue-based method of teaching, she questions whether the Socratic dialogue method has value as a teaching method.

Laurillard's [Lau06 pp75] Conversational Framework is constructivist as the teacher uses existing 'things' in the students' current mental world, to allow them to construct meanings for the new 'things' they are being taught. Laurillard argues that Socratic dialogue struggles to be successful as it is still merely the lecturer lecturing to the students. She accepts that it may be in a discursive way, yet still the lecturer leads the students to the answer rather than them constructing the answer themselves, which is what Laurillard seeks to do with her Conversational Framework.

As Biggs and Laurillard show, lectures are not altogether deficient as a teaching method because the main problems can be overcome within the lecture format. Though both prefer more active learning methods of teaching examples, Figure 2.1 and Figure 2.2 referenced from both Biggs and Laurillard support this statement. Figure 2.1 shows that in Biggs' opinion deeper learning is achieved through active learning methods. Figure 2.2 shows that Laurillard's Conversational Framework is easier to implement in a setting where a communication link between a student and teacher is maintained, for example in support *practicals* and interactive lectures.

In summary, lectures play a large part in the current constructively aligned higher education teaching system, yet the lecture is flawed in a number of ways. The next subsection explores how active learning in lectures can be achieved.

Active Learning within Lectures and the Lecture Hall

Lectures are often unidirectional with the lecturer talking to the passive student, with theory being taught to students with the aid of handouts or other audio-visual devices [Lau06]. Figure 2.3 shows how Laurillard views the current format of lectures in relation to her Conversational Framework.

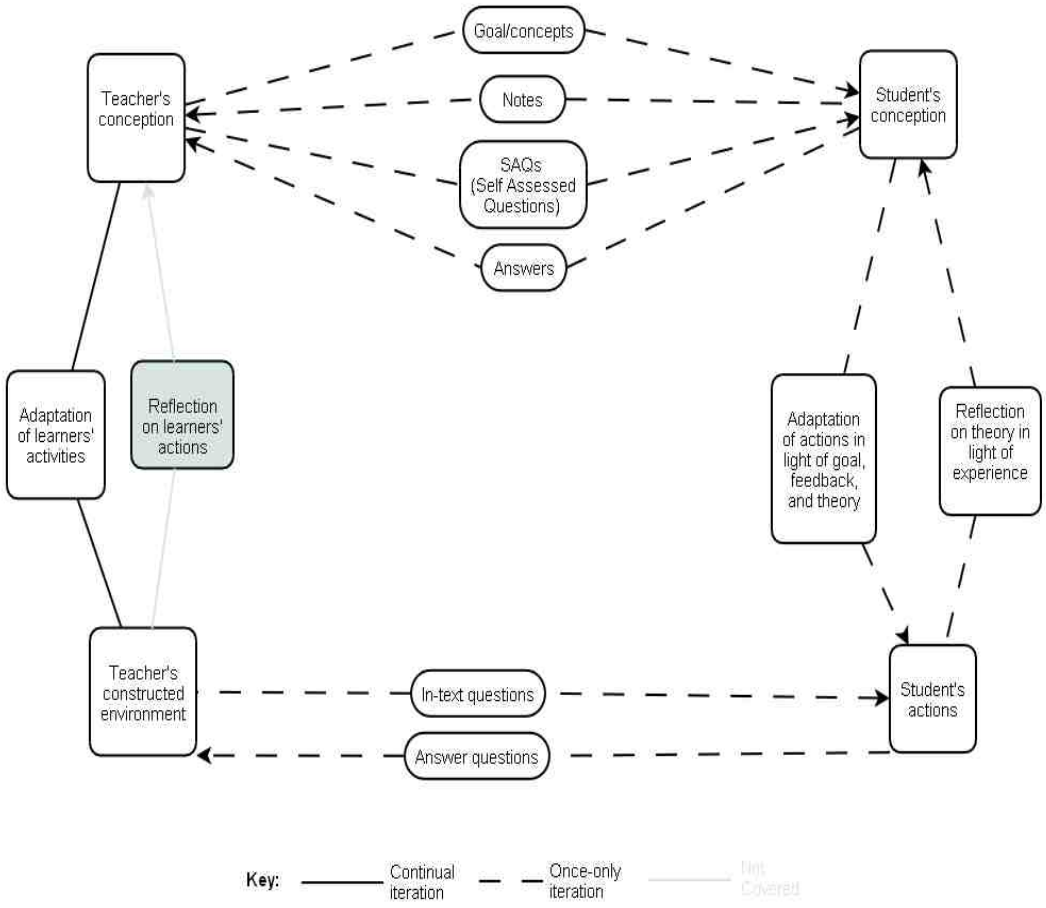


Figure 2.3: Laurillard's Conversational Framework in regards to the traditional lecture format [Lau06]

Laurillard believes that the lecturer and the student need to perform a conversation, so that the lecturer can judge whether the student understands the point that the lecturer is intending to teach. She believes the best way to do this is through questions from the lecturer to the students, which carry on until the student's conception matches the lecturer's conception. Figure 2.3 shows the traditional lecture in relation to the Conversational Framework. The lecturer's questions may be set in the form of, 'Goal/concepts', 'self assessed questions' and 'in-text questions'. The problem with these questions is that they are often left unanswered by students. Any questions that are answered have no capability to be given to the lecturer within the lecture due to constraints with: technology, student timidity and time [Gib81, Sch05],

which prevent the Conversational Framework from being implemented within the traditional lecture.

Introducing interactive elements to lectures can help to maintain student attention [Smi01], as these breaks in the lecture to answer questions allow students to recover more quickly from what Lloyd refers to as the 'middle sag' where student attention drops. Bligh shows a break away from the lecture can bring this attention up again. Smith supports the view that the break could be an active learning component in a lecture [Smi01].

Bonwell identifies seven major characteristics that show why active learning is more beneficial than passive learning. They are:

1. Students are involved in the class more than passive listening
2. Students are engaged in activities (e.g., reading, discussing, writing)
3. There is less emphasis placed on information transmission and greater emphasis placed on developing student skills
4. There is greater emphasis placed on the exploration of attitudes and values
5. Student motivation is increased (especially for adult learners)
6. Students can receive immediate feedback from their instructor
7. Students are involved in higher order thinking (analysis, synthesis, evaluation)

   [Bon03, pp. 2]

By applying to different types of learners, the seven characteristics above show that active learning addresses a range of students with differing learning styles.

The problem with the traditional lecture format is that it does not allow the student to apply the received information during the lecture [Gag65]. This problem is far less prevalent within an interactive lecture since all seven of Bonwell's characteristics

show, the main goal of an interactive lecture is to allow a student to immediately apply the information that they receive during the lecture.

Interaction can help resolve the majority of the issues with the lecture, yet the layout of the lecture hall could make the implementation of some forms of interaction difficult [Cut01]. In reference to this issue, Cutts [Cut01] observes the problems associated with interacting in large lecture theatre typically with around 300 students. He explains how these lecture theatres and their formation have been used in the same way for centuries and are not really designed for interaction, but rather for the lecturer conveying knowledge to the students, vocally and visually.

Cutts [Cut01] introduced interactivity to his computer science lectures by using a paper method. This method consists of the lecturer outlining the process of creating a piece of source code and then allowing the students the opportunity to write down on paper their own example of the code, in relation to a set problem [Cut01]. Cutts concludes that this attempt at interaction was not entirely successful as the majority of students did not participate and "would rather wait for the answer". Further he concluded that the room layout was not conducive for interaction on a one to one basis.

The problem of the room not being conducive for interaction is an issue that is addressed by interactive lecture technologies, including Personal Response Systems (PRS) and others. In more recent papers Cutts [Cut05] discusses how he used a PRS in his module to improve interaction between the lecturer and the students, and to help in creating a simplified form of the Laurillardian Conversational Framework. Cutts uses a PRS to enable the lecturer to apply the Conversational Framework through the use of Multiple Choice Questions (MCQs). This allows the lecturer to see how much the student understands the subject and also allows the student to actively participate in the lecture [Cut05]. PRS allows interaction within lectures, although there is some doubt that this interaction inspires deeper learning, which better supports student learning [Big02]. Biggs argues that MCQs can lead students towards a surface approach to learning potentially causing students to just revise potential questions and memorising the answers. He argues that this does not prompt

them to think more deeply by asking questions that promote the students to use deeper learning methods, such as reflection [Big02].

Some teachers argue that well phrased MCQs can be used to elicit deeper thinking [Wit03]. Wit gives examples of ways that questions can be asked to engage students in deep learning and claims that MCQs can address all levels of Bloom's taxonomy of educational objectives [Wit03].

Duncan [Dun06] refers to an example of active learning with interactive lecture technologies that has been shown to increase attention within *practicals*. During the study 100 students were given a lecture in the traditional style, by the lecturer reciting material to passive students (referred to in the paper as highly motivated managerial trainees). As in previous studies by Lloyd [Llo68], the students in Duncan's experiment also showed a slide of attention after 20 minutes and throughout the lecture the number of students who were paying attention averaged 47 out of 100. The lecture was repeated with some non-technological interactive elements added and the attention level of the class rose to 68 of 100. Yet during the lecture few students actually contributed to the interactive elements on a consistent basis, with 10 – 20% of the students dominating the lecture and the other 80 – 90% only occasionally contributing, the 'silent student' can account for this. A silent student is defined by Wit as being afraid of contributing publicly in front of a large group of their peers, for fear of getting answers wrong and facing ridicule [Wit03]. To further improve the lecture and to try and increase the amount of student contribution to it, in the third iteration of the lecture an interactive classroom technology was used on the students. This improved attention in the lecture to 83 out of 100 students. One final observation was that students in the interactive lecture scored more highly than those in the traditional lecture. This would seem to highlight that interactive lectures are more successful than the expository lecture method [Dun04].

As identified previously a problem for active learning within lectures is the learning space of the lecture theatre [Cut01, Bli98]. Despite the lecture theatre being a problem for active learning, the ever-improving technology available or present

within lecture theatres has begun to alleviate this problem [Mil07]. Milne comments that higher education has reached the "fourth wave" of computing system evolution which he describes as *"many devices, many users"* [Mil07 pp 16]. For instance, wireless technology allows many users to connect to the Internet or a Local Area Network (LAN) if they are within the signal range of a wireless router. Wireless technology allows students within a lecture theatre to connect to the Internet and participate during an interactive lecture using a range of devices, such as PDAs and laptops. This ubiquitous computing is used by Scheele [Sch05] highlights that, due to the improving technology, the lecture theatre as learning space can be used for active learning within an interactive lecture. This is despite the fact that the lecture hall is not designed or built for it.

Active learning is important within lectures as:

1. It allows students to use deeper learning techniques through the application of knowledge, rather than just passively listening to knowledge being spoken to them [Big02].
2. It allows attention to be maintained by breaking up the traditional expository style of lectures [Llo68, Bli98].
3. It maintains motivation by bringing in interesting and challenging elements to lectures through quizzes [Cut05, Sch05].
4. It allows closer working relationships in lectures between the lecturer and the students, promoting feedback to both especially on how the lecture is proceeding, allowing adaptivity within the lecture i.e. for a teacher to put more focus on particular topic which the students are having difficulties understanding [Lau06, Sch05].
5. Interactive lectures appeal to the different learning styles thereby they satisfy the majority of students, whereas the traditional expository lecture style may exclude many different types of learners. This problem is addressed with interactive lectures [Bon03].

With these five points considered it is clearly beneficial to introduce active elements to lectures and increase the efficacy of them.

The positives of active learning shown in this subsection can be replicated in lecture support *practicals* as well. This is addressed in the next section, which focuses on *practicals*.

2.2.2. Practicals and problem based learning

Numerous different types of teaching formats often support lectures, the main three are seminars, tutorials and *practicals*. The fact that lectures require support lessons at all has led some to argue that if lectures are that great a method of teaching, why do they need a support class at all [Gib81]. Gibbs [Gib81] argues that eventually it is the support *practicals* themselves, where students are required to actively participate that are more important than the lectures to the overall learning process.

In Biggs' [Big02] Constructive Alignment system he highlights that the support *practicals* allow the students to actively apply the knowledge related to them in lectures. Biggs further asserts that the support *practicals* allow the instructors the opportunity to inspire the less academic students into using deeper methods of learning. Biggs supports lectures, as doing one important task of passing knowledge to the students, yet he supports Gibbs' view that eventually it is the support *practicals* that are more influential elements of a student's learning in Constructive Alignment.

Laboratory *practicals* usually take the form of giving students a set problem, then students spend the allotted time trying to solve. This is active learning, as the students are actively trying to overcome problems. This is also a form of problem based learning or PBL [Ben01].

PBL is a member of the constructivist family of teaching methods, which is also the basis of Constructive Alignment. As discussed in section 2.1.

Figure 2.4 shows the constructivist-based problem based learning cycle, showing the different steps which students should go through whilst in a PBL class.

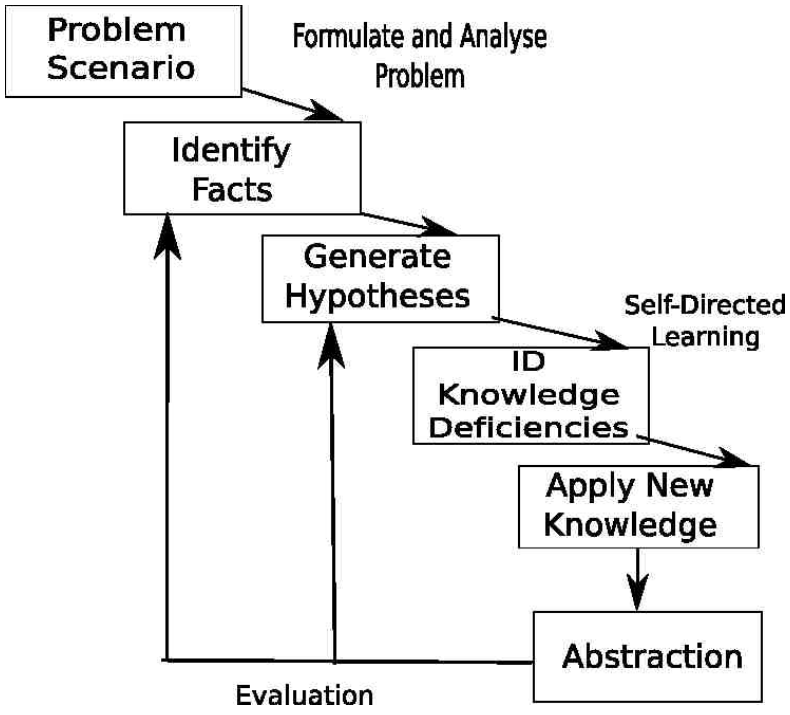


Figure 2.4: The Problem Based Learning Cycle [Hme04 pp237]

Hmelo-Silver [Hme04] is an advocate of this system as it allows students to achieve the higher levels of learning championed by Biggs [Big02], where the more active the learning method is, the more likely it is to stimulate higher levels of learning.

The communication between the student and the teacher is important in PBL although Figure 2.4 does not implicitly identify where communication between the teachers and the students would happen in the cycle. Hmelo-Silver comments that a teacher during PBL has two main roles: the first role of the facilitator is to guide the students to using higher order thinking skills, they do this by getting the students to justify their theories; the second role is to encourage self reflection by asking appropriate questions to the individual students [Hme04 pp245]. These roles mean that during PBL the teachers are only there to help and to provide guidance to the students. The teacher also ensures that they mentally understand the solution to the problem that they have been set. This description is similar to the role that the teacher has in Laurillard's Conversational Framework [Lau06]. In the Conversational Framework teachers ask questions probing the students' knowledge and then, if they

give incorrect responses, the teacher rephrases the answer to them and questions them again, until the student gives a correct answer.

PBL has faced criticism especially from advocates of Directed Learning [Kir06]. Directed Learning is where the main parts of the learning come through the teacher telling the students directly about concepts in a passive way. The advocates of this Directed Learning argue that due to minimal guidance methods, students have problems in acquiring the skills they require for the course. Kirschner et al argue that there is not much research that supports that PBL is actually capable of doing what it claims to do, this claim is that through setting a problem students learn skills through the solving of the problem [Kir06].

PBL and Anchored Instruction

There are many different instantiations of PBL with differences in the ways that the teachers/facilitators of the course interact with the students and also with the type of problems that students are asked to solve. One example is Anchored Instruction [Hme04, You92]. This differs in a number of ways to traditional PBL, but retains similarities in regards to the learning cycle shown in Figure 2.5. Hmelo-Silver defines the similarities between the two methods are that they both use a common problem and the teacher is key arbitrator of the learning process. PBL and anchored learning differ in the: "*terms of the type and role of the problem, the problem-solving process, and the specific tools that are employed*" [Hme04 pp237]. The key difference between traditional PBL is that the problems that students have to solve in Anchored Instruction are a lot more structured than in PBL. The structure provides the stages that a student must complete to resolve the problem.

The teachers' role also changes as well under Anchored Instruction. In Anchored Instruction they act from a position of having the knowledge from having experience of solving the problem in a real situation. The teachers then use this knowledge to lead the students to solutions that they may not consider, due to them previously not experiencing these kinds of problems [You92 pp7]. Anchored Learning is more

aligned to Laurillard's Conversational Dialogue than basic problem based learning in the way that the teacher provides a structure and guidance to the students.

2.3. Teaching Programming

There are two areas that this work is concerned with in relation to teaching programming:

- The discussion in literature of teaching novice programmers
- Current course leaders' strategies to teach novice programmers

Teaching programming is 'notoriously difficult' [Rob03] and many reasons have been discussed why this is the case. Section 2.3.1 looks at the different problems involved in teaching novice programmers and section 2.3.2 focuses on some techniques course leaders have implemented in their courses to alleviate these problems.

2.3.1. The problems of teaching programming

There has been much discussion on why it is difficult to teach novice programmers how to program, especially on higher education introductory programming courses [Rob03]. Robins et al in their review of the literature on the topic, look at a range of the different issues that course developers face in teaching introductory programming. They focus on two issues in particular:

- The difficulties that novice programmers have learning how to program
- Difficulties in teaching novice programmers

Robins et al report a number of issues they have found from their own experiences teaching novice programmers and also through the literature. The problems include:

- Which languages, if any, are taught on the course
- Unrealistic targets set for the courses [Rob03 pp157]
- The way that students cope with problems

The programming language used to teach students does not necessarily alter the modes of teaching. Languages vary from institute to institute with some choosing not to use languages at all in their introductory course. Some examples include:

- Object orientated programming [Kol03]
- Procedural languages [Rob03 pp145]
- Not using programming languages [Dji89]

Despite this range, the method of delivery is entirely independent. Institutions would still adopt lectures, *practicals*, or both.

One of the questions this work is concerned with is making *practicals* more effective. To this end the latter two issues, discussed by Robins et al [Rob03], that is, the way students cope with problems and the unrealistic targets set for courses, are more important for this work.

Winslow [Win96] in his investigation into programming pedagogy makes the observation that we cannot expect a student to become an 'expert' programmer over the course of a 4-year undergraduate programme. Winslow states that it could take a student 10 years to become an 'expert' programmer [Win96 pp 18]. Winslow adds that during an undergraduate programme the best a student can be expected to become is competent or proficient on the 5-stage programmer development scale proposed by Dreyfus and Dreyfus [Dre86]. The Dreyfus scale is:

- Novice – Who complete systems based on the set objective facts and features
- Advanced Beginner – Start to develop their own strategies but not quite comprehending what they are doing
- Competence – Are capable of considering the whole problem and choosing a suitable plan for achieving it
- Proficiency – No longer have to consciously think through a system step by step before they begin
- Expert – Knows what to do based on experience

Winslow sees university novice programming courses as the beginning of a student's learning development, which they build on when they move into industry.

Perkins et al [Per85] looked at how students cope when faced with problems when producing software source code. During this study Perkins et al identified three types of students who each apply different strategies when they are faced with a problem. These three types of students are:

- Stopper – This student faces a problem and stops with not being able to see the solution or attempt to find it.
- Mover – This student faces a problem and then solves it by trying something that they know already, or by working out a solution.
- Extreme Mover – This student is a subset of movers but rather than successfully solving problems, they try to find solutions without considering the logic of the change and could end up creating more problems [Per85 pp11]

During Perkins et al's study, Students exhibited these types of behaviours during an observed programming practical. Perkins et al [Per85] note that students who, for instance, exhibit "stopper" behaviour will always exhibit this behaviour, but that in some cases they just need assistance from teaching staff. In an example, Perkins et al records how a particular student when faced with a problem skipped past it without attempting to overcome it. A researcher then asked the student why they did this and the student replied that they could not do it, but when pressed by the researcher they managed to solve it [Per85 pp24]. This highlights that with assistance and prompting, students can be helped.

This subsection provides an overview of some of the problems facing course developers in creating novice-programming courses, what they teach and coping with student behaviours.

2.3.2. Existing methods of teaching programming

Novice programming modules are usually taught through lectures supported by *practicals*. *Practicals* are used to reinforce the theory taught in lectures and also to allow students in a supported way to develop working software. Typically in higher education teachers, who are postgraduate students and knowledgeable in the subject, usually provide support in *practicals*. In some cases the teachers have taken the course themselves, and can call on this experience to assist the students.

The course developer in Durham describes the course in the disciplinary commons portfolio on teaching programming, which was chaired by Fincher during 2005 / 2006 [Fin06]. The aim of the disciplinary commons was for teachers and lecturers of programming courses throughout the UK and the USA to share their approaches and experiences of teaching programming to undergraduate students. With this shared resource the course leaders could consider if they could improve their courses by integrating some ideas from their peers.

Fincher [Fin06] identifies some of the unique elements of each course. An example of this is from the University of Abertay and its creation of a hybrid lecture/practical setting. A further example is from the University of Glasgow where technology is used within lectures to make them more interactive and more effective, which was referenced in section 2.2.1.

The element of interest at Durham University is the use of the 'Personal Project', which allows the students to direct and develop their own projects for assessment. The goal of the 'Personal Project' is to enable a student to gain ownership of their work and therefore motivate them to learn more to improve their programming skills [Fin06 pp157].

2.3.3. Technology

Technology is used when teaching programming in a number of different ways. One example is personal response systems (PRS) [Cut05] to enable interactive lectures. A number of other tools have also been used to assist teaching and learning of

programming within higher education specifically for within *practicals*. These three in particular were chosen as they are; currently used, have been the topic of recent research and are in the area of research that this work is in i.e. improving *practicals* with the use of technology. The tools that are discussed are the following:

- Checkpoint [Eng06]
- SNOOPIE [Fin06]
- BlueJ extended version [Jad05]

These tools are summarised below.

Checkpoint

Checkpoint is a system implemented by English [Eng06] at the University of Brighton. It is used as a tool to automatically assess students' work both in *practicals* and for homework based assignments.

Checkpoint allows for two different types of questions:

- Fixed response questions – Where the tool can be used to ask multiple-choice questions or 'fill in the blanks'.
- Free text questions – Where the students have to complete questions, which could have a number of correct answers. An example could be to write some code. This code can go through the process of being automatically:
    - Compiled and run
    - Checked for functional correctness
    - Assessed for stylistic aspects
    - Timed to assess the efficiency of the code [Eng06]

Checkpoint can either automatically mark any answers submitted to it or allow teachers to manually check the answers. After the work has been assessed, Checkpoint provides the results to the teaching staff for example if the student completed the work successfully.

The University of Brighton uses a 'little and often' assessment model where students are given short assignments to complete on a fortnightly basis [Eng09], Checkpoint is designed to support this. Checkpoint is used as an assessment checker with the capability of making assessment easier for the teaching staff, as well as quickly creating automated feedback for the students.

SNOOPIE

SNOOPIE is designed with the aim of overcoming two issues students face learning how to program: "first formulating a (working) program at all and second formulating the right program to address the problem" [Fin06 pp154]. SNOOPIE does this by checking students' submitted code and checking the coding style. SNOOPIE can also further explain compiler errors to students, which in some cases are quite obscure and not very easy for novice programmers to interpret and fix.

SNOOPIE has been used in *practicals* in Abertay University [Bow06]. In Bown's contribution to the disciplinary commons portfolio, notes how the system is available for students to assist with their individual learning. He reports that the students accept the tool and the support it provides and that they preferred using SNOOPIE rather than, as he puts it, to 'pester' the lecturer for assistance.

In comparison to Checkpoint, SNOOPIE is similar in the way it provides support for the teachers through automatically checking students' submissions. However, the tools differ in the type of work submitted to them, with SNOOPIE being able to assist with more open submissions whereas Checkpoint is more focused on assignments with set variables.

BlueJ

BlueJ is a tool used to help novice programmers to learn how to program [Kol05]. BlueJ has also been used within *practicals* to collect data on student behaviour [Jad05].

The investigations carried out by Jadud [Jad05] used the BlueJ extension library [Blu10] to create an extension for BlueJ, which collected data on a student's

compilation reports. The data was stored and analysed after *practicals*. The data was used to view similarities and trends in novice compiler behaviour.

Jadud's investigations found that some compiler errors, in particular, dominated the list of collected errors. The investigation's top 5 compiler errors were:

- Missing semicolons (18%),
- Unknown symbol: variable (12%),
- Bracket expected (12%),
- Illegal start of expression (9%),
- Unknown symbol: class (7%) [Jad05 pp30-31]

Jadud carried out further analysis into the repetition of compiler errors and also the time between compiler errors.

The extension to BlueJ, unlike the previous tools, is not focussed as a teaching or student support tool. Despite this the data it collects can be used to support teachers and students. The analysis of the data can identify 'bad' novice programmer behaviours and the teachers can use this to help the students, although the researcher's focus is to identifying, rather than correcting the behaviours.

Technology Summary

The three tools explored during section 3.1.3 have identified both the need for, and integration of, technology into higher education programming courses.

Both Checkpoint and SNOOPIE assist the teachers by automatically assessing work and providing automated feedback, which reduces the workload of teachers.

Checkpoint and SNOOPIE support students by providing automated feedback. SNOOPIE provides feedback by further explaining compiler errors to the students and Checkpoint provides assistance on source code style. The automation of feedback assists students in two ways:

- The feedback is quicker, as students do not have to wait for teachers to mark work

- The feedback is tailored to the students' work and the mistakes that the tools finds

Therefore, both Checkpoint and SNOOPIE deal with the issues of supporting students whilst lessening the teaching staff's workload.

2.4. Summary

Chapter 2 has presented an overview of how some researchers believe courses should be implemented within higher education. In particular, it has addressed lectures and how these are integral at University level to distribute course content to a high number of students at one time and one place. The chapter has also outlined some of the problems with the lecture. Specifically, it is often carried out in a unidirectional way, where a student can lose focus and interest.

Despite the problems with the lecture format, strategies have been attempted to improve the efficacy of lectures, namely with the introduction of active elements. As section 2.2.2 has presented, this has been done with the use of systems like PRS, where students can be questioned in a way that allows lecturers to see how effective their description of concepts has been.

The introduction of active elements to lectures is one method that has been successful to an extent in improving teaching, yet lecture support *practicals* are still important as a method of enabling students to use techniques in a supported environment.

The last section of this chapter has focused on introductory programming courses and how such courses have been implemented. It has also described three examples of technologies that have been implemented to assist teaching programming.

The following chapter, Chapter 3, examines one example of an introductory programming module and how *practicals* are managed on it.

3. Investigating Learning and Teaching In Current Practical Setting

Practicals and other types of support lessons have the function of reinforcing the material that students have been taught in lectures. On programming courses they are the primary place where the students are supported when they are programming themselves [Jam06]. This makes *practicals* very important and the course leader in Durham and course leaders at other establishments place great emphasis on them.

At Durham University students are supported by teachers known in Durham as demonstrators (see section 1.4) and by lecturers. Typically during a practical there are at least ten students to one teacher. A practical in Durham University is 120 minutes in length and there are normally three teachers in attendance. This gives 360 minutes of combined teacher time, available for student interaction.

This chapter presents an investigation of how *practicals* in Durham University's introductory programming module operate. This investigation addresses Research Phase (i), which comprises of the following three research questions:

Research Question 1. What are the students' and teachers' opinions on the pedagogic value of *practicals*?

Research Question 2. How do students begin interactions with teachers in the existing practical setting?

Research Question 3. To what extent can teachers perceive student status in the existing practical setting?

To explore these, data was obtained during an investigation using the first year programming course at Durham University. The chapter outlines the method used to investigate *practicals*, and then display and evaluate the results collected during the investigations.

3.1. Method

To explore the research questions outlined above both participant observations and questionnaires were used. Observations were used during *practicals* at Durham University in the start of the 2008/2009 academic year. These observations were used to investigate the format of the current practical setting and to provide evidence of how teachers and students view *practicals*. The observations were also used to see how teachers and students operated within *practicals*. The approach used in the observations is described in more detail in the next section.

The data collected from the observations, presented in this chapter, was collected from two practical groups. The observations consisted of records of the frequency that interactions took place within *practicals*. These observations in particular recorded:

- Who initiated the interactions (student or teacher)
- The duration of each interaction

These observations identified some problems that exist within the current practical setting.

There are two differences between the groups:

- Group B is smaller (14 students in comparison to 25)
- Both groups have three teachers, but in Group B two of the teachers appeared to be more reactive (reacting to students asking for help rather than searching for students who may need help)

The two aims of collecting the data were to, firstly observe how open students were in asking teachers for help and secondly, to view if any students dominated the teachers' time during a practical. The observations are also important to answer research questions 1, 2 and 3. The observations do this by providing further quantitative and qualitative data. The sheet used to record these observations can be found in Appendix 5.

Questionnaires were given to both the teachers and the students during the investigations into current *practicals*. These were used to obtain both quantitative and qualitative responses to assist in answering research questions 1,2 and 3. The research questions rely on finding the opinions of the teachers and students, on three areas of *practicals*: how highly they value *practicals*, do they find it difficult to communicate in *practicals* and how easy or difficult do teachers find tracking status in *practicals*. The teacher and student questionnaires from the investigations into current *practicals* are recorded in Appendix 1 and Appendix 2. They use the same method as the questionnaire used in the main case studies of this work (described in detail in section 6.1.1) where both open and closed questions are asked to get the opinions of the respondents.

3.2. Results

The following sections present the data that was successfully obtained from conducting the investigation outlined above. Results are presented by group, rather than for each category of data. This approach better illustrates the differences that group size and teacher style had on variables measured during the observation.

3.2.1. Group A

Total Time of Interaction by Student

Figure 3.1 presents each student's individual total interaction time with the teachers in *practicals* and also the average time per interaction. In the chart the values on the X-axis relate to the student and the assigned computer number at which they were sat. For instance, in Group A there was no student sat at computer 6, so this number is not included on the figure. In Group A there was a student at computer 9 but they did not interact with a teacher at any point in *practicals*, so they are included on the Figure with no interactions.

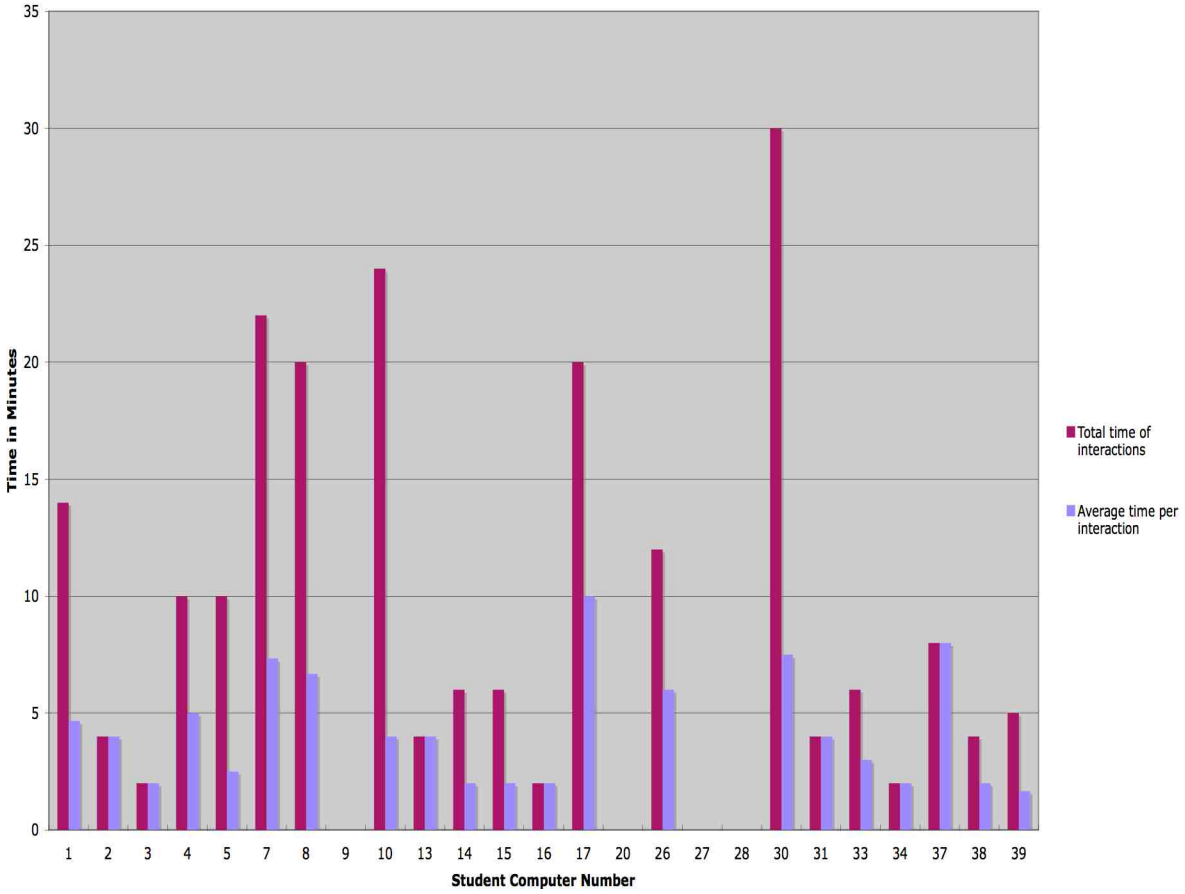


Figure 3.1: Group A comparison between students' total time of interactions and the average time per interaction

In Group A (with 25 students) there were 215 minutes of interactions recorded by the researcher. What additionally needs to be taken into consideration for the timings is that during the first ten minutes of *practicals* students were allowed to prepare or arrive for the lesson and the last ten minutes of *practicals* the students were allowed to complete to go to their next lesson. This means that there are 60 minutes of interaction time that can be discounted, as the three teachers in effect individually lose 10 minutes at the beginning and the end of *practicals*. After discounting these 60 minutes, 85 minutes of the teachers' interaction time were not used for interaction with the students. The majority of the 85 minutes unused interaction time was from just one of the three teachers who tended to be more reactive, waiting for requests for help rather than for searching for students who may require help.

Figure 3.1 highlights one of the expected outputs that some students did dominate the teachers' time. As was noted while there is only a limited amount of time available with the teachers and in Group A, five students accessed 20 or more minutes of interaction time with a teacher. In Group A one student (Student 30) alone had 30 minutes of the total interaction time. The majority of the students had ten or fewer minutes of interaction (18 students) and four students had no interactions within *practicals*. These results, especially the students with longer teacher interaction times, are further discussed in section 3.3, as their behaviour is notable.

Interaction Frequency

Figure 3.2 is concerned with how the interactions were initiated. Teachers' perceptions are that the students are often not willing to ask for help and that this would lead to the teachers being required to initiate interactions. However, the observations revealed that opposite was the case. In Figure 3.2 the student numbers relate to the computer that they were sat at and omitted numbers are computers that no student was sat at.

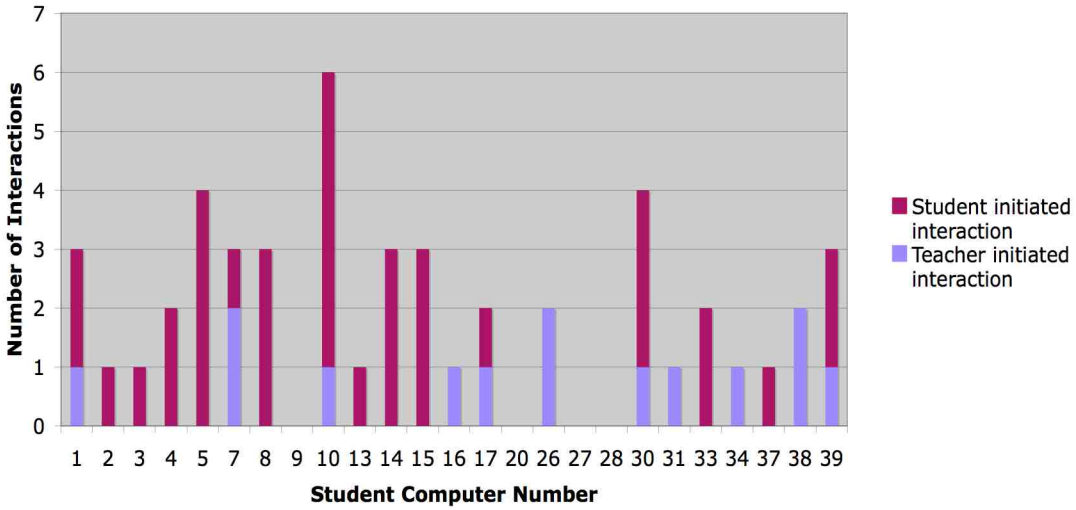


Figure 3.2: Group A amount of student and teacher interactions and the breakdown of who initiated these interactions

During the observations of Group A, there were 49 interactions. These 49 interactions consisted of 35 initiated by the students and 14 initiated by the teachers.

As Figure 3.2 highlights there were nine of the 24 students who did not ask for help during *practicals* including the four students who had no interactions at all. Over 50 % of the students did ask for help and this was unexpected, but the nature of how the students asked for help may explain this result. Many of the interactions were initiated by the student 'catching the eye' of the teacher as they walked around the laboratory. By doing this, the students began interactions in a less obvious way rather than openly asking for help, such as by putting their hand up.

3.2.2. Group B

Total Time of Interaction by Student

Figure 3.3 presents the total and average time of the interactions between the students and the teachers.

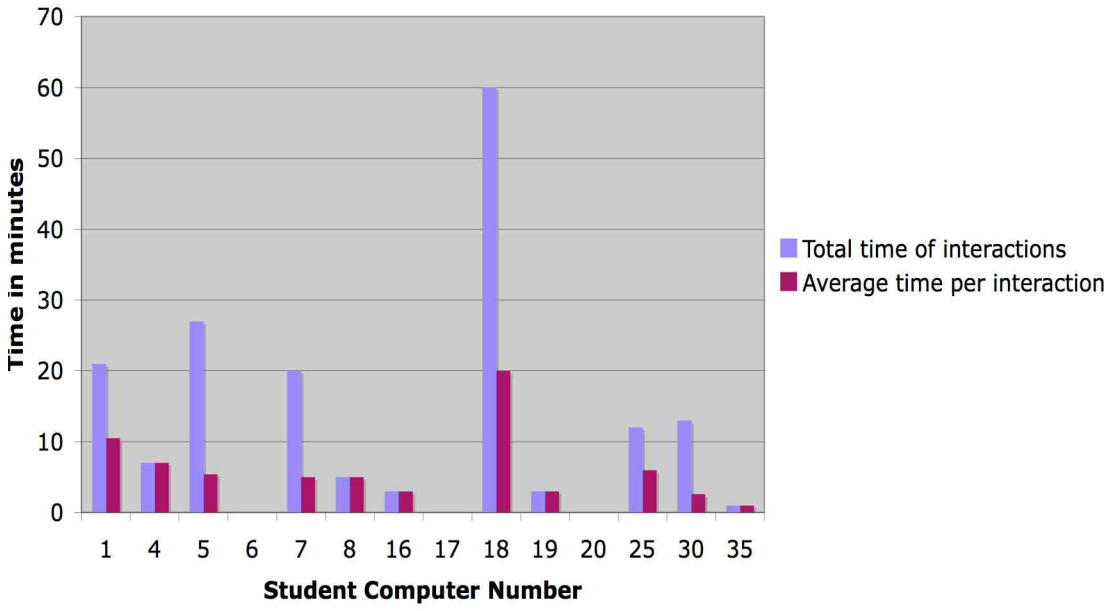


Figure 3.3: Group B comparison between students' total time of interactions and the average time per interaction

The total time of interactions in Group B was 172 minutes and as Figure 3.3 highlights one student (18) alone accounted for 60 minutes of these interactions. The interactions took the form of the teacher sitting behind, rather than next to, the student and assisting them with their work. This is discussed in further detail in

section 3.3, as the teachers' adoption of one to one assistance was possible in Group B, due to it being a smaller group.

As with Group A, there was a mixture of students with low and high amounts of interactions. Group B had four students with 20 or more minutes of interactions and eight students with fewer than ten minutes of interactions including three students with no interactions at all. The high proportion of on average low interaction times may be because the majority of teachers in this practical adopted a reactive approach and waited for requests from the students.

Interaction Frequency

Figure 3.4 further supports the fact that the students had to ask the teachers more for help in Group B to get assistance from the teachers.

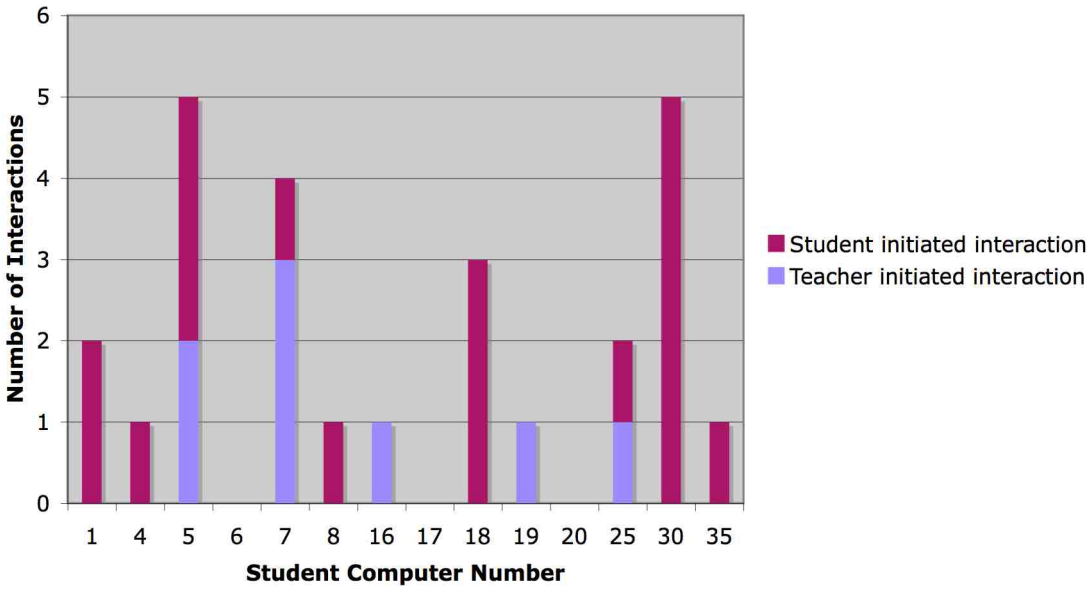


Figure 3.4: Group B amount of student and teacher interactions and the breakdown of how these interactions began

During the observations of Group B there were 26 interactions. These interactions consisted of 18 initiated by the students and only eight initiated by the teachers. As in Group A these numbers further support the observation that students initiated the majority of interactions.

As Figure 3.4 shows five of the students did not ask for help during the class and three of these students did not have any interactions with the teachers.

3.3. Evaluation

Five issues were identified during the observations taken from Group A and Group B. These issues are described in this section with the evidence, which supports them. These are:

Issue 1. Students are afraid to openly initiate dialogue with teachers

Issue 2. Communication difficulties between teachers and students

Issue 3. Teaching made difficult by the range of different skill levels of students

Issue 4. The independence of learning in *practicals* leading to students not knowing what to do

Issue 5. Visibility of students' progress

These issues are explained more fully below.

3.3.1. Issue 1 – Students are afraid to openly initiate dialogue with teachers

During the qualitative observations it was discovered that some students were apprehensive about openly initiating dialogue with the teachers, but the recorded observations show that they still did initiate dialogue. The quantitative observations showed that the students initiated the majority of interactions in both Group A and Group B.

The qualitative observations of the two groups revealed that there are two behaviours exhibited by the teachers, whilst they were in *practicals*:

- Standing in a specific area of the classroom for the majority of the class time waiting for the students to ask for help or put their hands up.
- Actively 'patrolling' the classroom and enquiring on the status of individual students.

The teachers tended to have an inclination towards one of the two behaviours. In recognition of this, a classification was created to group teachers. Teachers who were more inclined to wait for the students to ask for help were classified as *reactive*, and teachers who were more inclined to inquire on the status of the students were classified as *proactive*.

The qualitative observations taken of the two groups found that there was a mixture of proactive and reactive teachers with a 2:1 ratio in Group A and 1:2 in Group B. This ratio was formed through observing how the teachers acted within the learning environment.

T-tests were carried out between Group A and Group B to compare if there was any significant difference in:

- The total time of interactions per student
- Total frequency of interactions per student
- Frequency of teacher initiated interactions per student
- Frequency of student initiated interactions per student

	N	Mean	Sd	T	P - Value
Total time interactions per student Group A	25	8.60	8.48	-0.79	Not Significant
Total time interactions per student Group B	14	12.29	16.32		
Total frequency of interactions per student (Group A)	25	1.96	1.49	0.19	Not Significant
Total frequency of interactions per student (Group B)	14	1.86	1.75		
Frequency of teacher initiated interactions (Group A)	25	0.56	0.71	-0.04	Not Significant
Frequency of teacher initiated interactions (Group B)	14	0.57	0.94		
Frequency of student initiated interactions (Group A)	25	1.40	1.44	0.23	Not Significant
Frequency of student initiated interactions (Group B)	14	1.29	1.49		

Table 3.1: Table presenting comparison of observations taken from Group A and Group B

As Table 3.1 presents, despite the qualitative observations revealing the proactive and reactive teachers, none of the T-tests revealed any significant difference between groups A or B. These results could be due to the size of the groups being relatively small.

In Group A, 49 interactions took place between the students and the teachers. A paired T- Test was conducted comparing who initiated interactions in Group A. There was a significant difference in the scores for student initiated ($M = 1.40$, $SD = 1.44$) and teacher initiated interactions ($M = 0.56$, $SD = 0.71$) conditions; $t(24) = 2.44$, $p = 0.021$. The p value reveals that there is significantly more student who

initiated interactions than those initiated by teachers. This suggests that the students were not afraid to request help from the teachers, although typically their requests for help involved approaches other than the student putting their hand up to attract the attention of the teacher. More commonly students' requested help by asking a teacher when teachers passed close to them as the teachers monitored classroom activity. The importance of this observation is that the proactive teachers were more likely to be asked for help. This is due to the fact that the more reactive teachers, who wait and look for hands to go up, are not so accessible to the students who seem to prefer a more 'covert' method of attracting attention.

Unstructured interviews with the teachers carried out after this observation supported the view that if the teachers did not make themselves approachable to the students then they were rarely approached. The teachers in the interviews also speculated that the students sit together in friendship groups and that they are afraid of putting their hands up and revealing that they are having problems in front of their friends. A student and their peers may well perceive a casual request to a passing teacher as being less associated with personal failure and more related to interest in the topic.

Questionnaires given to students within the observed *practicals* revealed that only 9% of the students admitted to finding it difficult to ask the teachers for assistance. These results seem contrary to the observations that were noted. However, the way the students asked for help from the teachers highlights that they might be just apprehensive to openly ask for help, rather than not being prepared to ask for help at all.

In summary, the students did initiate interactions with the teachers but it was the mode of asking for assistance that was notable. The questionnaire responses further highlighted that students do not see a problem asking for help. Yet the observations of *practicals* groups revealed that some of the students preferred to not openly ask the teachers for help, they would rather 'catch the eye' of a teacher as they patrol the classroom. This method works when the teachers in the classroom are proactive and walk around the classroom, however, when they adopted a reactive approach it was observed that they were less likely to have student initiated interactions.

3.3.2. Issue 2 – Communication Difficulties

The observations highlighted the following three problems, which caused communication difficulties:

1. The room layout is not conducive to the type of assistance that can be provided by the teachers
2. Students have difficulties describing their problems
3. Teachers have problems describing solutions to students

Although *practicals* are more suited for interaction than lecture theatres, the teachers noted a number of problems with the learning space. For instance, one teacher noted that: “Currently it is awkward for teachers to talk to a student as the rooms are too cramped and the teacher will have to talk on their knees or over the student’s shoulder.” This was noted during the observations as well, where certain computers were difficult for the teachers to reach to help students. Such as classroom design makes it difficult for the teachers to provide feedback and also to allow them to gain an appreciation that students have understood their instruction. Facial feedback can be useful for teachers to judge whether students have understood something. It is not easy to get this feedback in *practicals* as the students sit facing their computer screen.

Figure 3.5 presents the layout of Durham University’s CG65 classroom that was used as the experiment laboratory and highlights some of the difficulties that teachers can have interacting with the students.

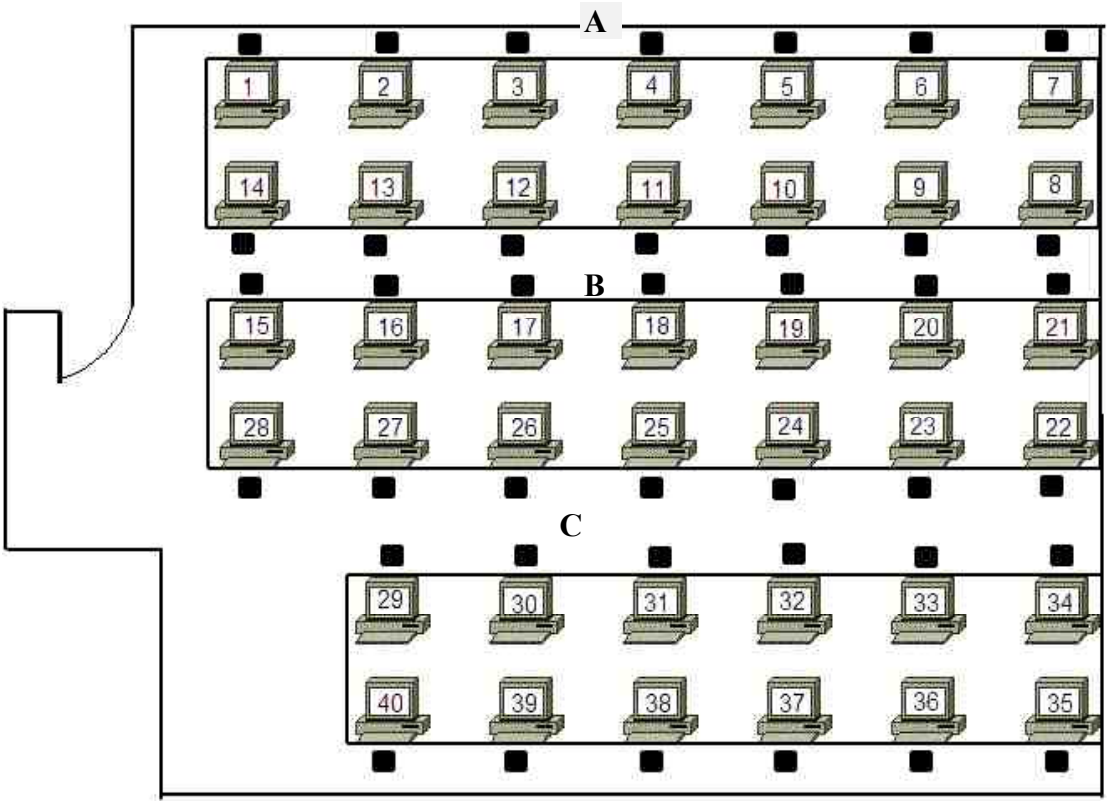


Figure 3.5: Example Classroom

There are two rows of computers that present particular problems, marked as Column A and Column B in Figure 3.5. Figure 3.5 depicts these problems in the classroom where the teachers have to disturb students in Column A to be able to help students at computers 2 to 7. This is due to the narrow space behind the computers, which makes it difficult for the teachers to get to the students to help them. This is also the case with column B, where to assist students seated at computers 13 to 8 and 16 to 21 teachers would have to disturb students seated at the top of column B e.g. computers 14 and 15. The column that is easy for the teachers to interact with the students is column C. In the case of column C there is a larger gap between the two rows of students, which enables the teachers to monitor the students without disturbing the other students seated on the column.

The problems in the classroom layout are reflected by observations that the students seated at some computers seemed less likely to have interaction with the teachers if they did not request help by putting up their hands.

To highlight the frequencies of interactions based on where the students were located in the classroom thematic maps from both groups have been generated using the observation data,. Figure 3.6 shows the thematic map of Group A's interactions, and presents the total frequency of each student's interactions during *practicals*:

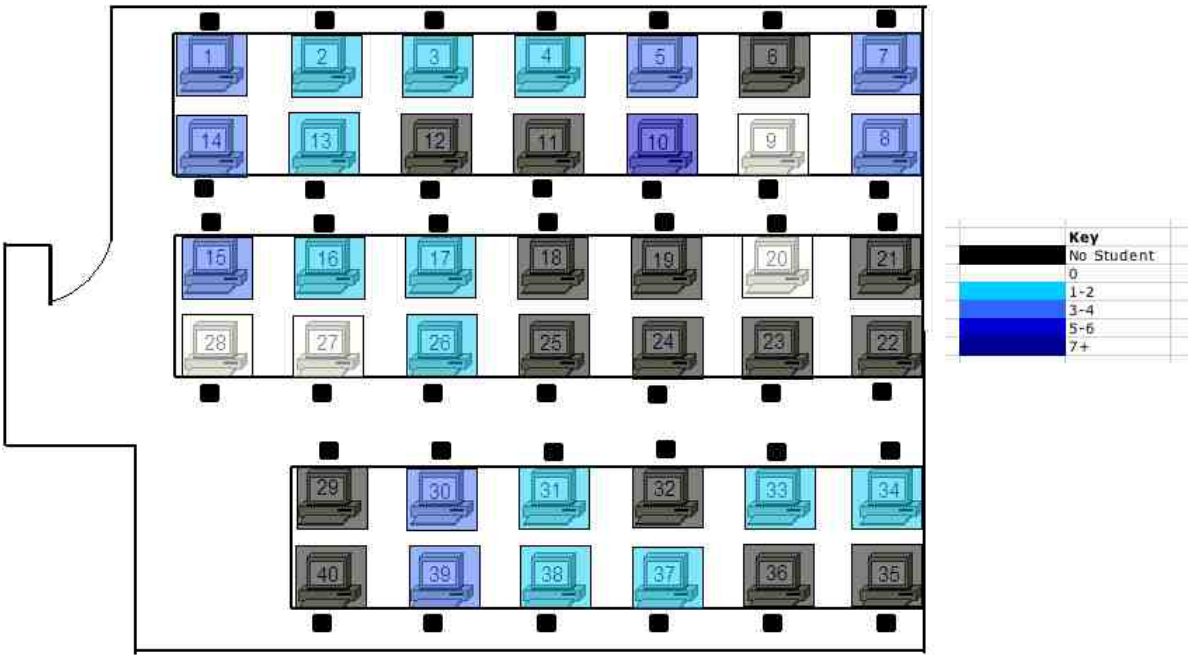


Figure 3.6: Thematic map of Group A's student interaction by total amount

As Figure 3.6 highlights there are no clusters of students with a high volume of interactions in Group A, even in the computer locations where it was expected that the teachers could have difficulties to reach as the darker blue colours, which represent higher frequencies of interactions, are distributed throughout the classroom. However, the thematic maps that display who initiated the interactions (Figures 3.7 and 3.8) reveal that certain locations required students to initiate interactions more frequently. Figure 3.7 presents the thematic map of the frequency of interactions initiated by the teachers:

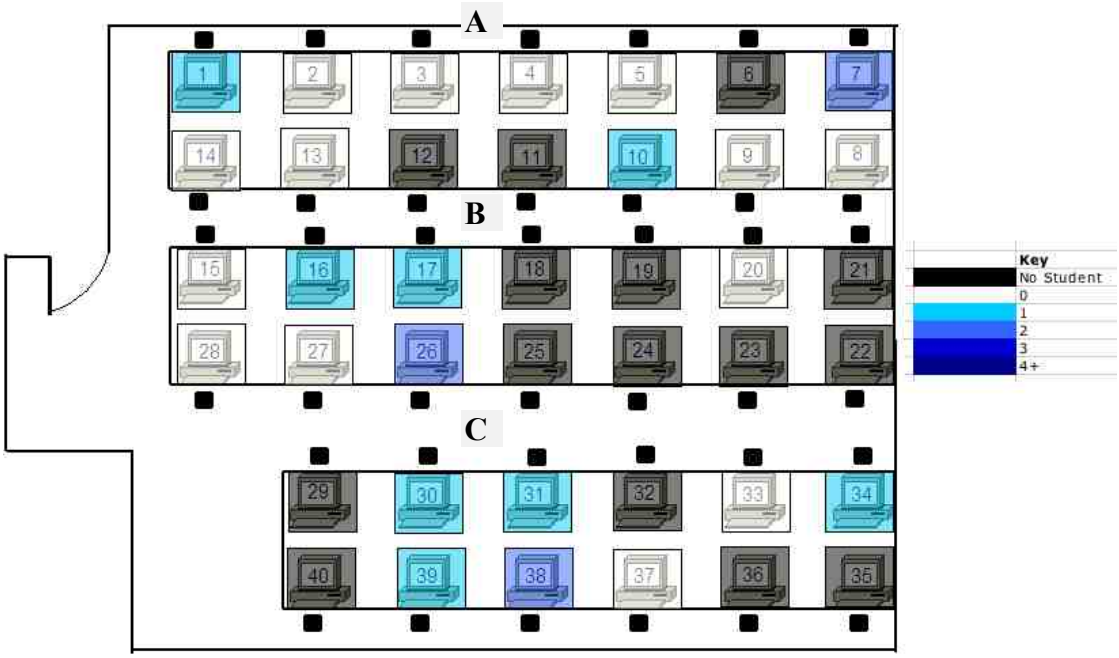


Figure 3.7: Thematic map of the frequency of interactions initiated by teachers in Group A

Figure 3.7 shows that the students who were located on column A were less likely to have interactions initiated by teachers. This compares to locations in column C, which are easier for the teachers to reach. In column C it was recorded that the students had more interactions initiated by the teachers in comparison to Columns A and B. These observations correspond to the problem rows identified in Figure 3.5.

Figure 3.8 presents the thematic map of Group A's student initiated interactions:

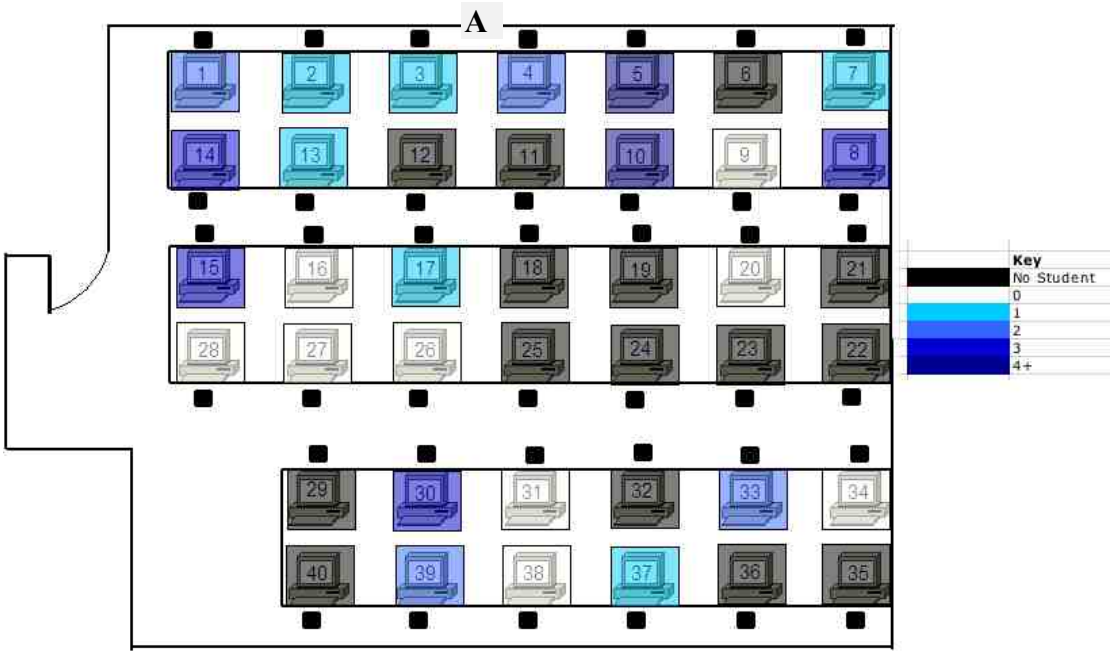


Figure 3.8: Thematic map of the frequency of interactions initiated by students in Group A

Figure 3.8 confirms that with the students having fewer interactions initiated by teachers, they had to formally request help, which is especially notable in column A. The teachers attempted to explain this observation, during semi-structured interviews. They reported that students in inconvenient locations are less likely to be interacted with unless there are obvious request help.

The smaller size of Group B makes it difficult to illicit viable results from the interactions as it is sparsely populated, but it did provide notable qualitative observations into how interactions in the class took place in a small class context. The thematic map of the interactions recorded during the Group B practical did highlight how students seem to prefer to sit in column A. The thematic map also reveals that with fewer students some of the issues with the learning space are reduced.

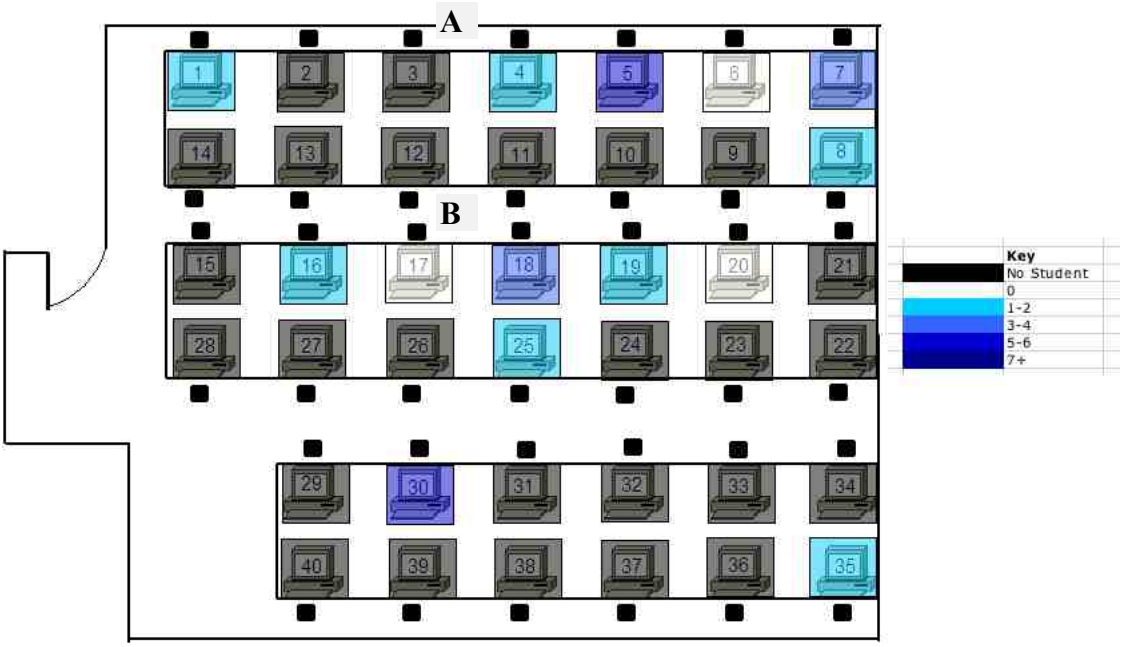


Figure 3.9: Thematic map of Group B's student interaction by amount

Figure 3.9 highlights that the students sat on only one side of column B, the qualitative observations taken by the researcher found that it was easier for the teachers to interact with the students. The thematic map shows that some students did group, for example on columns A and B, but others were more spread out. For instance, those students sat at computers 25, 30 and 35.

The observations show that the student sat at computer 35 in Group B had only a single interaction lasting for only one minute with a teacher. It can only be speculated as to why that student sat in that location but one possible reason is that they might wish to avoid teacher interactions unless they themselves chose to request it. Alternatively the student sat at computer 30 had a high frequency of interactions. In this case as may they positioned themselves close to the teachers so they could easily access help when needed.

Another notable example was in Group B for a student sat at computer 7. In this case, the first interaction was instigated by the student putting their hand up and getting the teacher to come to them. Thereby the student initiating the interaction made the teachers aware of the difficulties they were in and so henceforth the next three were inquiries initiated by the teacher. The important point about this example

is that only when the student actually asked for help did the teacher became aware of the problems they faced. So, when the teachers identified that the student required extra help, they paid particular attention to them.

The observations of *practicals* showed that the majority of the interactions were short, averaging between two to three minutes per interaction, yet there were a few students who seemed to require longer interactions. The interactions instigated by the teachers were in the majority of instances shorter. In these cases typically a student would either report that they had completed the work successfully or otherwise presented issues that were solvable in a short time. The problem with short interactions is that it could be the teacher just provides the student the answer, which although it allows the student to get over the issue quickly, it does not help the student's learning. Providing the students with an answer may deter them from trying to solve the problems themselves in the future.

In both Group A and Group B some of the students, on average, took more of their teacher's time. In Group A one student had an average of ten minutes of interactions, divided into two interactions and another student had four interactions with an average of seven and a half minutes each. These examples could be a result of the student not understanding the solution, so as a consequence the teacher has to try various ways of leading the students to the answer. There are a number of methods for the teachers to help the student, for example:

- Showing them the Application Programming Interface (API)
- Drawing students a picture to illustrate the solution
- Standing over a student's shoulder offering them advice when it is required.

These are all valid and accepted ways for teachers to assist the students, but there has to be a balance so that the students are not too reliant on such help. One of the main differences between higher and secondary school education is that the students are expected to be work more independently at university.

In the groups that were observed two of the students receive one-to-one tuition. One student from Group A had 30 minutes of interaction (Student A) and in another example one student from Group B had 60 minutes of interaction (Student B) with one teacher. The researcher asked the teachers involved in both cases to provide an outline of the reasons for the large amount of interaction time with these students. Both teachers reported that the students appeared to be having serious problems with their work and teachers felt that they had been unsuccessful in explaining the solution to the student. They noted this was especially the case for student A. Student A had already had one of the other teachers try and explain a solution to them before the longer later interaction. Student B was a different situation, in this case the teacher made use of smaller group teaching involving two other students who sat close to student B, and all of whom were having similar issues. With student B the teacher was also able to stay looking over the shoulder of the student whilst they coded and then so to offer advice when it was needed.

As well as the learning space issue, an additional issue in relation to communication difficulties is the issue that many students have difficulty describing their problems to the teachers.

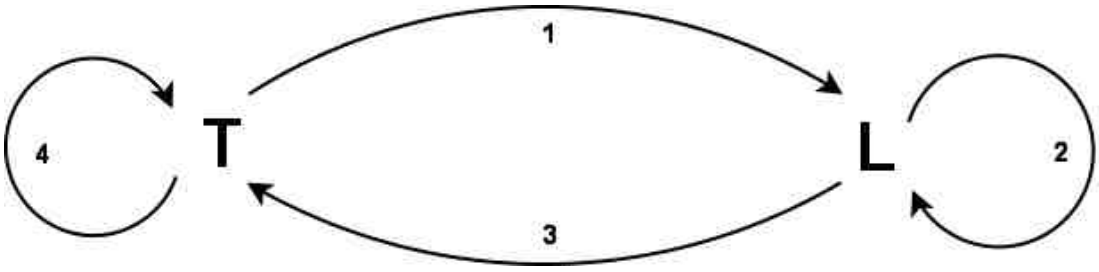


Figure 3.10: Cutts version of Laurillard Conversational Framework [Cut05]

Figure 3.10 presents a simplified version of the Laurillardian dialogic framework [Cut05], which demonstrates how PRS can facilitate the framework in lectures. The T refers to the teacher and L the learner and the numbers refer to the steps in the framework:

1. Where the teacher explains a concept to a student
2. Where the student attempts to comprehend the concept

3. The student explains their understanding of the concept back to the teacher
4. The teacher judges whether the student has understood the concept and if not they go back to stage 1 and explain the concept again maybe in a new way.

The Laurillardian dialogic framework can be applied more easily in *practicals* than in lectures, as teachers interact on a one-to-one level with students. For instance, the teachers interacting directly to a student and they interact directly back to the teacher, these interactions facilitate steps 1 and 3 in the Laurillardian Conversational Framework, yet as this section has revealed these steps are still difficult in *practicals*.

In the questionnaires given to the students, three answered that there are communication difficulties in step 1 in particular of the Laurillard Conversational Framework, one student said: "Questions are sometimes not directly answered and can leave you more confused" this reveals that students are sometimes unhappy with the assistance that they get from some of the teachers. This introduces an interesting problem that is explored in more detail in Issue 4, as one of the roles of the teacher is to lead the students to the answer rather than just give them the answer. Thus, the students still need to be able to solve the *practical* problems themselves. The student response highlighted above could reveal that the students feel that the methods that the teachers use to lead them towards solutions are actually more confusing than helpful.

The questionnaire responses from the teachers also highlighted some difficulties with communication at Laurillard's point 3 (Student to teacher interaction) where 50% of the respondents disagreed with the statement: "Students are good at describing the problems that they have with their code." This response highlights that the teachers do have concerns regarding at least some student's ability to describe their problems.

In summary the classroom design can cause communication difficulties between teachers and students. Although despite the room, a form of the Laurillardian Conversational Framework does exist where teachers ask the student questions and offer advice if it is needed. This work has also found that students and teachers sometimes find it difficult to describing their points to each other.

3.3.3. Issue 3 – Students' Range of Programming Abilities

An issue, which exists in introductory courses at any level and on any subject is the prerequisite knowledge that the students have before the course [Rob03]. The questionnaire given to the students explored what level of programming experience students had prior to coming to university. The results showed that the range of experiences were noticeable, from students who had a lot of prior experience including those whom had worked in industry to those students whom had no prior experience. These responses add another element of challenge for the teacher, as students with experience may be able to program but not explain why they need to they need to do it in certain way which is knowledge they require to pass the course, while students with no experience would need assistance with even the simple tasks.

Ten students, of the 35 students surveyed, admitted to having very little or no experience of programming prior to University, and a positive correlation can be seen between those students and those that found *practicals* and the course more difficult. In total six students responded that they found *practicals* difficult all of whom has no prior programming experience. In comparison none of the students with prior experience of programming admitted to *practicals* being difficult and two even reported that the course was too easy.

With a combination of the room set up and the apprehension of the students to openly ask for help, teachers have problems identifying which students require help. An example was seen in Group A where early in the course (within two months) some of the students were having problems with coding such as writing calling methods and adding fields and variables. To further illustrate the range of abilities in the cohort, one of the students spent a long portion of a practical trying to set up an advanced IDE on their computer. This resulted in the student acquiring, during *practicals*, 24 minutes of teacher time. This case raises the issue of what priorities the teachers should have in deciding how to apportion their time. Should teachers prioritise those beginners to ensure all students understand the basics, or do they also seek to help the advanced students to ensure they are able to push themselves, and

not get frustrated. While this in an interesting topic, it is outside the scope of this thesis.

In the questionnaire given to the teachers the question: “Do you think in the current practical setting it is difficult to judge how well one student is doing?” the teachers responded unanimously that they could judge the stronger students and through the fortnightly assessment lessons they could identify most of the weaker students. A number of the teachers highlighted that the students ‘in the middle’ were the hardest to identify and that it is difficult for them to see what level they are working at. The teachers reported that this middle group could either do well or they could fail. The following statement from one teacher summarises this point: “Over the last 3 years I’ve not had any difficulty in picking out the better and the worse programmers in *practicals*. However the ‘in-betweeners’ are harder to work out, and indeed some of them have gone on to fail or score badly in exams unexpectedly, and some have scored higher than expected.” So this group of individuals clearly need to be closely monitored.

3.3.4. Issue 4– More Independent Learning

The fourth issue found in *practicals* is with the element of independent learning. Independent learning makes it difficult for the teacher as it means that they are faced with a number of students who are all working on different projects and at different levels. The independence in *practicals* is beneficial because it allows the students to work at their own pace [Cut07]. It is difficult for the teachers to judge how a student is doing in some contexts. For instance, there could be situations where the students are reading the content of the course book but are not able to translate this into a capability for them to create their own programmes, whereas others may adopt this skill readily.

The students, in the questionnaire responses, seem to enjoy the independence of *practicals* setting. In the questionnaires that they completed one student noted: “More independent -> less pressured -> can learn in own time”. This is the sentiment presented by many students when they were asked to compare their secondary school

practicals to University *practicals*. Though the majority of the students like the independence, a few noted that there were negative impacts. One of these is that they were not sure what they were required to do, as these students' note; "PDS2 (introductory programming module) *practicals* are generally too vague", "Sometimes [it is] difficult to know what work is". A further student presented a reflective insight into the issues that face the students in the transition between school and higher education: "I find it [*practicals*] more difficult than a school lesson, purely due to the pace that is expected". These highlight that some students are used to being told exactly what to do in their lessons, as in secondary school, and have problems when they are set more open targets to complete at their own pace and over a longer period time.

Practicals as a method of independent learning is important but presents challenges to appropriately support students' learning. The student need to be supported only to an extent to allow them to be independent, but whilst also enabling them to be able to speedily progress with their learning.

Such issues are noted by Biggs [Big02] who writes that university is a place where students have the opportunity to get away from the lower order learning, which is sometimes apparent within secondary school and college. In secondary school emphasis is focused on getting the students prepared for exams. In higher education more emphasis is on enabling the students to learn independently and explore more, higher order, learning strategies, for example generating their own strategies for dealing with problems rather than having the answers given to them.

3.3.5. Issue 5– Visibility of Students' Progress

The final issue is a combination and a result of the previous four issues and is that teachers have difficulty in judging and viewing progress made by the students.

In Durham *practicals* operate by having a marking session every second practical, to allow the teachers to view which students are falling behind and which are working to or above the required level. Of equal importance is to give the students formative and summative feedback. One of the reasons that the course leaders had

implemented this marking session was due to the room issues addressed in issue 2, which makes it difficult for teachers to judge how well a student is doing in *practicals*. In the questionnaires completed by the teachers a couple of the responses related to the ease of judging a student's progress in a practical. One teacher noted: "Teachers cannot always see what the students are doing", which leads to the need for the marking sessions to check on how a student is progressing.

Teacher's opinion and also the observations taken by the researcher were mixed in regards to the success of marking sessions. The teachers unanimously thought that the marking sessions were a good idea as it allowed them to see how well a student was progressing. However, some teachers and the researcher argued that *practicals* was the wrong place to do it. The researcher observed one marking session and found that each student took an average of ten minutes. This time was dependent on the level of the student, longer for a weaker student and not as long for a stronger student. Group A had three teachers and a class of 25 students, so out of the 360 teacher minutes, 300 minutes could be taken up by marking work. The researcher noted that during this time some students wasted time while waiting for the teacher to mark their work and instead spent *practicals* time surfing the Internet. The teacher questionnaires seemed to support this, one in particular noting a desire to: "Remove marking from the actual practical settings" and later explaining their desire to, "Have marking sessions outside of *practicals* with *practicals* being mainly problem sessions and feedback sessions". However, if marking sessions were removed from *practicals*, an alternative way for the teachers to see levels of progress may be needed.

The teacher was further interviewed and noted that the marking session is good for seeing how a student is progressing, but that takes up so much time that support cannot be offered to the other students whose work is not being marked in the session. So, for instance, a student could be marked in the first ten minutes of *practicals* and then get stuck with their work but due to the marking requirement placed on teachers there is no one free to assist them. This may mean that they do not

receive the help from the experienced programmer, which is one of the main benefits of *practicals*.

The current practical setting presents the challenge that for teachers to understand how a student is progressing requires that the teachers regularly assess them, but this causes staff-time constraint difficulties. Technology could be used to assist the teachers to judge a student's status in *practicals* and is explored during the remainder of this thesis.

3.4. Summary

This chapter investigated a practical within a typical introductory programming course at Durham University. The investigation identified five issues with the way that *practicals* operate. The last issue of these five issues, Issue 5 – Visibility of students' progress, was seen as both a result and cause of a number of the other issues. This issue should form a primary focus of the development of the technology to support *practicals*.

The remainder of this summary discusses the three research questions that were asked in Chapter 1 that were addressed in Chapter 3.

3.4.1. Research Question 1 – What are the students' and teachers' opinions on the pedagogic value of practicals?

The questionnaires responses from the students and the teachers revealed that they do value *practicals* highly. The students and the teachers view the importance that they can apply knowledge they have read and been taught in lectures, whilst being supported by teaching staff. The students also value the way *practicals* allow them to work at their own pace and on particular parts of the topics that they feel they have difficulties with. This was discussed in more detail in section 3.3.4.

The University also accepts the importance of *practicals*, by making *practicals* compulsory for students when lectures are not.

3.4.2. Research Question 2 – How do students begin interactions with teachers in the existing practical setting?

The results taken during Research Phase (i) found that the students began interactions in a number of different ways with the teachers. The two main methods were through:

1. The students putting their hands up and requesting help,
2. The students intercepting the teachers as they patrolled the classroom.

The first method was an expected observation when the students have a problem that they raise their hand to get the attention of the teacher, but the second method and the way that some students asked for help was more notable.

The observations of this second method found that some students preferred to not openly ask for help, but instead preferred to ask 'covertly'. This is sometimes made difficult by another observation that some teachers are reactive rather than proactive in the classroom. Reactive teachers were seen to wait in one spot of the classroom waiting for hands to go up, whereas proactive demonstrators would patrol the classroom. The proactive teachers were much more likely to be asked for assistance by the students who preferred to ask for help 'covertly'.

Another factor that could impact on a student's capability to use this second method could be the room layout. This is especially as there is not much space behind the students' seats, which makes it difficult to interact. In some cases it is also required to disturb a number of students to assist an individual.

3.4.3. Research Question 3 – To what extent can teachers perceive student status in the existing practical setting?

Two main factors were found during Research Phase (i) that both hinder the teachers' ability to track the students' statuses in the *practical*:

- Communication difficulties
- Room layout

As a result of communication difficulties in the existing *practical* setting, the teachers admitted to having some difficulties in tracking how some students were progressing. The teachers in the questionnaires answered that it was usually easy to pick the students who were strong and the ones who were weak but there was a middle group majority, which could either do very well or very poorly. The teachers noted that it was difficult to judge how well the students were doing. This was discussed in more detail in section 3.3.2.

The room layout causes problems as it is difficult to see a student's work on the screen and quickly see if it is good or not, which is compounded by the students, who have problems describing their work. The ability to describe their code is one of the skills they are developing during the introductory programming course, which should improve over the course.

The existing setting attempts to overcome the difficulties in tracking progress by having marking sessions in *practicals*. Although these marking sessions are flawed as they occupy teachers for the duration of a lesson with the consequence that they cannot provide the support to students who are not being marked. The teachers' support to the students is the main benefit of *practicals*, so limiting this is not beneficial.

In Chapter 4, components of Technologically Enhanced Demonstrator Support (TEDS) are described. TEDS is used to help teachers support students in *practicals*. It explores ways of lessening the impact of the issues identified in this chapter.

4. Technologically Enhanced Demonstrator Support

Technologically Enhanced Demonstrator Support (TEDS) is an approach, realised by a software system that is designed to address the five issues for existing practice, as identified in Chapter 3. In trying to overcome the issues outlined in Chapter 3, TEDS is used to address research questions 4 and 5.

The focus of this chapter is on the features of the TEDS software that are designed to provide support for those teaching programming. A *feature*, in the context of this thesis, is a component of TEDS that is specifically designed and implemented to fulfil the identified needs of the teachers. The process of implementing the features is later explained in Chapter 5.

4.1. Design Principles

During design, a set of principles were developed and used to guide the inclusion or exclusion of potential features, and set the approach by which the TEDS software would be constructed. These principles are:

Design Principle 1 – Minimal change to students' method of working. It is imperative that the software does not change the way in which students normally work, and has no impact on the instructional design of the course.

Design Principle 2 – Provide a maximum amount of data to the teacher. Different teachers may find different data useful in evaluating the learning of students. By providing as much data as possible, teachers have access to a number of metrics.

Design Principle 3 – Ensure data is accessible. Given principle (2), it is important that the volume of data is accessible and does not present a further cognitive overhead during teaching. For instance the data must be presented in a way that the teachers can effectively use it during the *practical*.

Design Principle 4 – Facilitate improved communication. It is necessary for the software to improve communication. In this case improving communication could ease some of the issues identified in Chapter 3.

The goal of this design is to adhere as closely as possible to the four presented design principles, which seek to lessen the impact of the issues identified in Chapter 3.

4.2. Design Investigations

Features included in TEDS were derived from an explicit design process. The design process consisted of three investigations:

1. *Exploration of the capabilities of the BlueJ extension library*. This investigation aimed to reveal what data would be available to the TEDS system via the BlueJ extension library and for determining what data could be gathered from the students' activities.
2. *Focus groups and interviews*. This investigation concerned direct consultation with users. It gathered data from focus groups and interviews with teaching staff from the department as a means to determining which features to include in the design of the TEDS system.
3. *User evaluations*. Early versions of the software were, in a rapid-prototyping fashion, evaluated by users. This process served as both a validation exercise for the implemented features, and a way of capturing further feature needs.

Each of the three investigations identified above are explained in more detail below, showing where each feature in TEDS was identified as a requirement.

4.2.1. Exploration of the capabilities of the BlueJ extension library

Students in the introductory programming module use BlueJ as their IDE. This prompted research into the possibility of using BlueJ as the foundation of TEDS, adhering to design principle 1. It would not be appropriate to select any other IDE as

this would have a significant effect on the course design and the learning activities that the students engage with. This investigation, therefore, is to fully understand the potential of the BlueJ extension library system in providing the necessary data for TEDS.

BlueJ has a built in extension library that gives developers API access to a restricted set of functions. Examples of the kind of information available to extension library developers include data regarding compile time and run time objects [Blu10]. The BlueJ extension API has three packages that can be used by extension developers to gain access to core data [Blu10]. These three packages are:

- **`bluej.extension`** – this is the core package of the BlueJ extension API. This package allows developers to access to data that BlueJ users create. This data includes classes, methods and source code.
- **`bluej.extension.editor`** – this package allows developers access to users' current active editor window. The editor window is a part of the user interface that programmers use to write Java source code. This API lets a BlueJ extension retrieve source code text, make changes to that text, and determine which class the user is currently editing and where in a particular class they are editing.
- **`bluej.extension.event`** – BlueJ extensions can register themselves with the BlueJ event system. This means that the extension receives events as they are generated while the user writes their software. Example events that the extension can listen for include: when the user compiles their code; when the user invokes a method in their software; or when the user begins changing a class.

Developers have used the BlueJ extension API to create a range of extensions. For instance, the extended version of BlueJ discussed in 3.1.3 made extensive use of BlueJ's event API during its implementation [Jad05].

In considering the issues, outlined in Chapter 3, the BlueJ API provides good support to extension developers to gain access to data and functions that are useful in

designing a tool to support learning. Furthermore, using the BlueJ extension library has the additional benefit that BlueJ can remain as the tool used in the design of the course. A student's familiarity with this environment could increase the likelihood that the TEDS tool would not be a barrier to adopting it in their learning.

It is not appropriate to map the issues from Chapter 3 directly to functions of the BlueJ APIs. The TEDS system provides a set of features that use the BlueJ API, so discussion of this type of mapping is, instead, included in the descriptions of the remaining design investigations. Instead, the creative process of design resulted in a set of prototype features that were directly included. The features that were derived from BlueJ API capability were:

Feature A: Report of students' last compile i.e. success/fail

The **`bluej.extension.event`** API allows extension developers access to the compile events created by BlueJ. If the event is a compiler error the result includes:

- The location of the compiler error within the code i.e. the class and the line of code where the compilation error occurred
- The type of compiler error i.e. 'missing ;'

If the compiler is successfully able to complete a compilation process over the student's code base, a confirmation of this is given to the student. However, a limitation of BlueJ is that it only outputs the first compilation error that is found by the compiler [Jad05].

Feature B: Report of student's last method invocation

The **`bluej.extension.event`** package allows an extension developer access to events generated when BlueJ users invoke their program. The invocation events include whether the method invocation was successful or unsuccessful, and the location of the run method in the BlueJ user's code.

A limitation of the method invocation API is that certain runtime errors are not recorded. An example of this would be if the method runs but returns an incorrect result. Another limitation is that it does not give a developer access to the details of a

runtime error; e.g. it does not make any report on what runtime errors occurred during execution of the student's software.

Feature C: A snapshot of the student's code that is sent to the teachers

The two packages `bluej.extension` and `bluej.extension.editor` give extension developers access and the ability to copy a BlueJ users source code. This then means that the extension could, for example, replicate or transfer their source code. Snapshots could then be taken of a student's source code over a period of time and these could be used to view code development.

4.2.2. Focus groups and interviews

In February 2008, a focus group was arranged to present TEDS. The audience consisted of academics and postgraduates from the Computer Science department at Durham University. A description of TEDS was given and a demonstration of an early prototype. The intention of the demonstration was to give the participants an introduction to the TEDS system, including a full review of the proposed feature set. During the session the focus group had the opportunity to use the prototype in the role of a student, and a small task was set for them to complete.

The prototype of TEDS presented was a basic system consisting of:

- Feature A – Compiler Errors
- Feature B – Method Invocation

The prototype was networked so that the data collected by TEDS could be collated and displayed to the members of the focus group. This provided the audience with an overview of what a teacher, using TEDS, would be able to see in a practical.

The participants all taught within the department either as course leaders, lecturers or teachers.

After the short presentation and demonstration of TEDS, the focus group was consulted to collect their views on the current feature set. The objective of this

consultation was to see if any changes should be made to the system. In total 14 suggestions were made. These suggestions were:

1. Seeing the context of compiler error, i.e. which class the error is in.
2. Compiler errors by frequency, errors grouped together by type.
3. Help button, for students to ask for help.
4. See the student's current activity while they type.
5. Chart of a student's compiles over the period of a practical.
6. Maintain *practical* data by task.
7. Store student records in a database.
8. Provide information to group students by shared difficulties.
9. 'You are not alone' provide students with data on the progress of their peers.
10. Remote editing by teachers of students code.
11. Sending files and tasks to the students.
12. Provide teachers data on student's semantic errors.
13. Provide teachers data on student's runtime errors.
14. Introduce unit tests.

These observations were thematically analysed to view their suitability to the four design principles of TEDS:

Minimal change to the students' methods of working.

Provide a maximum amount of data to the teacher.

Ensure data is accessible.

Facilitate improved communication.

When thematically analysed, seven of the suggestions were found to be incompatible. The incompatibilities were due to Design Principles 1 and 3.

Design Principle 1 seeks to make TEDS adjust to a student's learning, rather than the other way around. The themes of five of the suggestions from the focus group were incompatible, as they would change the ways *practicals* are operated. An example of this was to add J-Unit tests to TEDS. J-Unit is a method of passing variables to a piece of code and running it to test if it works. The use of J-Unit tests are very valuable, but require students to be relatively rigid in the way they complete coding tasks to a specification. TEDS should not be designed for this purpose rather its focus should be on monitoring students and informing teachers if they require assistance.

J-Unit (14) is incompatible with Design Principle 1 because at Durham University, students engage in open, flexible coding projects, which could mean that all the students could be working on different projects within the same practical [Cut07]. This makes it impractical and time inefficient to create a J-unit test for each student. In light of this J-Unit tests were not implemented in TEDS.

Another instance of a suggestion that was analysed as being incompatible with Design Principle 1 was to use data collected by TEDS to group similar students together (8). This would be a change for the students, as they are not currently grouped together based on assessment. Furthermore, generating groups based on similar problems would require a form of partnering algorithm, which is not the focus of this work.

Design Principle 3 rendered four of the suggestions incompatible (4, 9, 10, 12). Design Principle 3 is concerned with ensuring that any data is presented in an accessible manner to the teachers, but this is not always practical. In one of the suggestions made by the focus group, they commented that it might be beneficial to see details of what a student was working on. Through revealing to the teacher what line and class a student is working on, they could see if the student is making any errors that can be rectified before they worsen. In principle this is a good idea but it is impractical to display all of that data to a teacher especially taking into account that a practical could consist of up to 40 students. Due to this reason it was concluded that the suggestions were incompatible with Design Principle 3.

Seven of the suggestions were found to be compatible with the Design Principles (1, 2, 3, 5, 6, 7, 13).

Design Principle 4 is concerned with ensuring that TEDS improves communication. One of the suggestions was the idea of a "Help Me" button (3), which would enable students to ask for help in a more discreet way rather than by putting their hands up or by shouting to attract a teacher's attention. It was anticipated that the feature would increase the likelihood of shy students asking for help. In response to this suggestion, Feature D – Help Button and Feature E – Short Message functionality were implemented.

The six other suggestions from the focus group that were implemented all corresponded with Design Principles 2 and 3. These two principles are concerned with enabling teachers to view useful data in an accessible way. Three of the suggestions related to how data that was currently available was displayed to the students (1, 2, 5). For instance, it was suggested that displaying a student's compiler status over the period of a whole practical in graphical form would be useful (5). This suggestion matches both Design Principle 2 and Design Principle 3 as a graph can be easily generated from the data collected from the students and displayed in a useful manner to the teachers.

Two other suggestions (6, 7) were concerned with integrating a database into TEDS. The suggestion was that with a persistent record of a student's data, a teacher could view how a student progresses over a number of weeks. The main focus of this thesis and consequently TEDS however, is to provide teachers with live data on a student's status during the course of one practical. Despite this there are benefits to integrating a database into TEDS. For example, it would maintain a record of data that can be viewed interactively after the session, and with this data a lecturer could observe if a student cohort is making common errors. The main advantage of using a database is that it maintains a permanent record of data, which provides a fail safe if a client has network problems. The data, as long as the server receives it, is saved in the database, this ensures that the data is accessible to the teachers matching Design Principle 3. In light of these perceived benefits a database was added to TEDS.

4.2.3. During Use

TEDS was tested in both a departmental showcase and during a practical. These were carried out to test the features of TEDS, and also to get feedback from the intended users of the system. As a result of showing TEDS to the users some new features were suggested.

Showcase

The showcase was an event where researchers systems were presented to students and took place towards the end of the 2007-2008 academic year. The suggestions for features for TEDS generated in the focus group (Feature D – Help Button and Feature E – Short Message Functionality), had been implemented for the prototype. The showcase became a test of the features developed so far as the visitors were able to take part in a short demonstration of TEDS.

The difference between the showcase and the focus group was that the attendees were students. Students along with the teaching staff are the intended users of TEDS. Therefore, the feedback from the students was useful, as feedback from the focus group had already provided feedback from the teachers.

A further benefit of the showcase was that it enabled TEDS to be tested to examine if it could perform satisfactorily with up to 8 student users.

The students did not make any new suggestions for features, but nevertheless they gave positive feedback, which was useful as it gave indication that TEDS's features were sufficient to allow progression to the next phase – full investigations.

First Investigation Practical

In June 2008, TEDS was tested in the first investigation practical, which took place in a ‘real-life’ educational environment. The first investigation practical examined the performance of the software with a high number of users, as well as its practicality as a teaching tool.

The investigation practical explored the use of the combined features from the focus group and the BlueJ API and *practicali*ty of how the data was displayed to the teachers.

During this practical, data was collected and the results are presented in Section 7.1.

While taking part in the exercise one of the teachers commented that they felt it would be beneficial if TEDS could be used to draw images for a student, and then to send these to them for reference. Some of the teachers noted that often it was important to raise the level of abstraction when teaching novice programmers to that of design. Since TEDS did not support graphics at the time of *practicals*, that was not possible. A new feature, Feature G was implemented in response to this idea.

After *practicals* teachers commentated that they used a number of windows more than others and would prefer those windows to be larger. Modifications later made in response to this suggestion enabled the teachers, to view the window displaying students' requests for help and their current compiler status easier than before. The final version of the teacher console is displayed in Figure 4.3.

4.3. Features Identified

As a result of these investigations, a set of seven features was determined. Table 4.1 presents the seven features along with the investigation in which the feature was determined.

	BlueJ Extensions	Focus Group	During Use
Feature A: Report of a student's last compile i.e. success/fail	√		
Feature B: Report of a student's last method invocation i.e. success/fail	√		
Feature C: A snapshot of a student's code that is sent to the teachers	√		
Feature D: Ability for a student to request help electronically		√	
Feature E: Ability for teachers to reply remotely to a student via Short Messages		√	
Feature F: Objective setting functionality allowing clear objectives to be set in *practicals*			√
Feature G: Ability for teachers to draw and send images to a student's to help them understand concepts			√

Table 4.1: Presenting link between features and investigations

Feature F was not devised through any of the three investigations directly, but rather was influenced by the questionnaires and feedback from both the students and the teachers. Each feature is explained in detail below.

4.4. The Features of TEDS

Seven features were generated out of the investigations presented in section 4.1 and these features can be related back to the issues highlighted in Chapter 3. These were:

Issue 1. Students' are afraid to initiate dialog with teachers;

Issue 2. Communication difficulties between:

i. Teachers and students

ii. Students and teachers

Issue 3. Students' with a range of different skill levels

Issue 4. The independence of learning in *practicals* leading to students not knowing what to do

Issue 5. Visibility of a student's work status

In summary the features that were generated for TEDS and are explained in this section are:

Feature A. Report of a student's last compile i.e. success/fail;

Feature B. Report of a student's last method invocation i.e. success/fail;

Feature C. A snapshot of a student's code that is sent to the teachers;

Feature D. Ability for a student to request help electronically;

Feature E. Ability for teachers to reply remotely to a student's via Short Messages;

Feature F. Objective setting functionality allowing clear objectives to be set in *practicals*.

Feature G. Ability for teachers to draw and send images to the students to help them understand concepts;

The five Issues (identified above) and the seven Features (identified above) can be cross-referenced. These relationships are presented in Table 4.2, where a letter represents each function and each issue has a row.

In the Table 4.2 a tick (√) indicates where it is anticipated a specific feature could ease a particular issue. For instance, Features D and F are anticipated to ease the impact of Issue 4.

				Features				
		A	B	C	D	E	F	G
	1	√	√		√	√		
	2	√		√	√	√	√	√
Number	3	√		√		√	√	√
Issue	4				√		√	
	5	√	√	√				

Table 4.2: Map of features to issues

In total seven features were implemented and each are looked at individually in this chapter in their own subsection. In relation to each feature these three statements are considered:

- The role of the feature in overcoming the issues identified in Chapter 3.
- The process of developing each feature.
- What the feature does.

4.4.1. Teacher Client

TEDS is required to show enough data that it is useful to the teachers but not too much that it becomes too complicated to use within a live practical. Robins et al support this [Rob03, pp 164], writing that data needs to be presented in a way, where

teachers can use it successfully during a practical. Before the final teacher console was decided upon, two other versions were used in design phase. These can be viewed in Figure 4.1 and Figure 4.2.

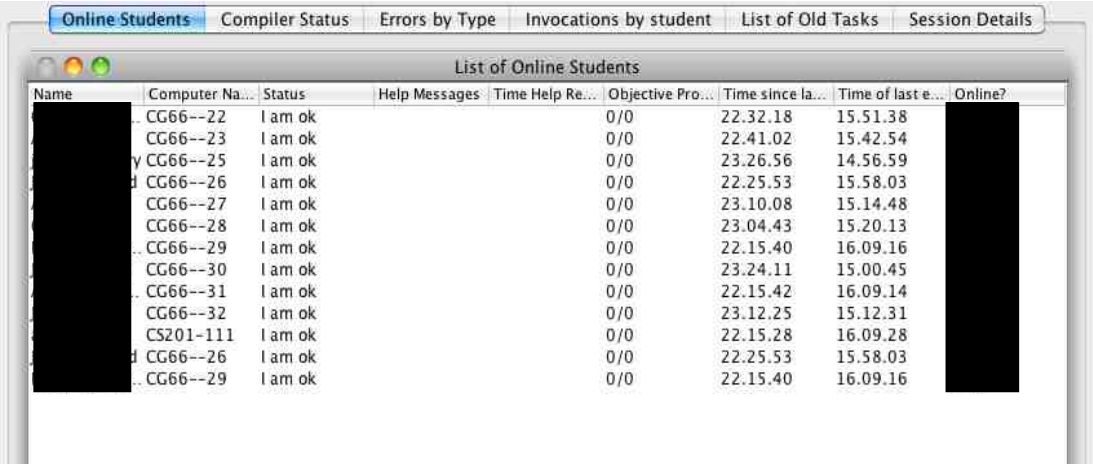


Figure 4.1: Teacher Console Version 1

Figure 4.1 presents version 1, which is a tabbed view that by cycling through the tabs the teacher can view the students' data grouped together under different tabs. For example the compiler status tab gives a teacher access to all of the students' compiler data.

The benefit of version 1 is that data is not overwhelming as the screen only presents a small amount of the whole data collected by TEDS at one time. This is also a negative effect of the version 1 teacher console, as to view all of the activity updates that TEDS collects from the students during the *practical*, at least the Compiler Status and the Method Invocation Status data needs to be viewable at any one time.

After these negative observations Teacher Console Version 2 was implemented for the 'during use' component of the design of TEDS presented in 4.2.3. This can be seen in Figure 4.2.

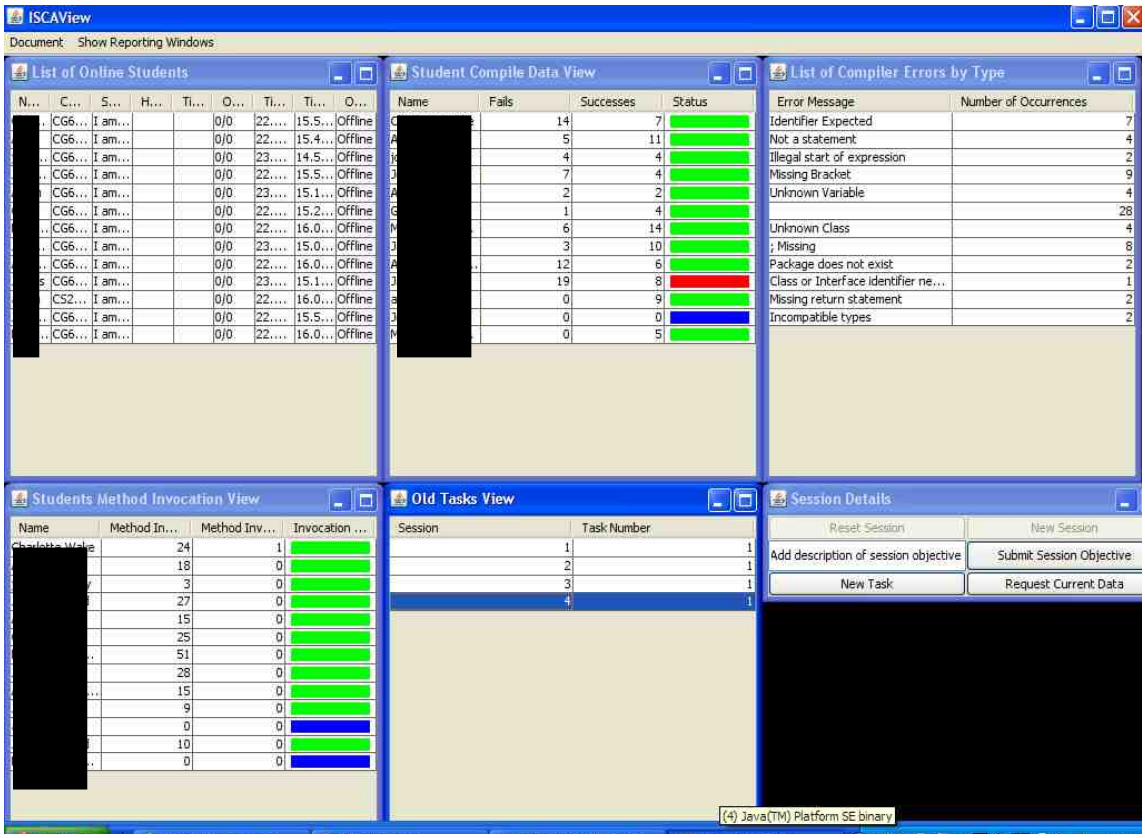


Figure 4.2: Teacher Console Version 2

Figure 4.2 is a large console window with moveable and resizable sub-windows so the teachers can adapt the layout to their own requirements. This version presented more data at one time than version 1, but the teachers requested for some windows to be combined to make it easier for them to access and interpret the data. The teachers especially asked for the 'List of Online Students' window to be combined with the compile and method invocation windows.

In response to factors concerning the displaying of data and feedback from users of the teacher client the layout shown in Figure 4.3 was developed.

In the figure, window **1** is the view of each individual student's status during *practicals*. It presents data such as the time since last event, if the student is offline or online, compiler and method invocation status. In the lower half of the main teacher view the windows are from left to right:

- Window **2** is the cohorts collated compiler errors which are looked further at in Figure 4.7
- Window **3** is the old task overview which allows teachers to view a student's progress over a number of *practicals*
- Window **4** allows the teacher to interact with a student. This feature is explored further in section 4.4.7.

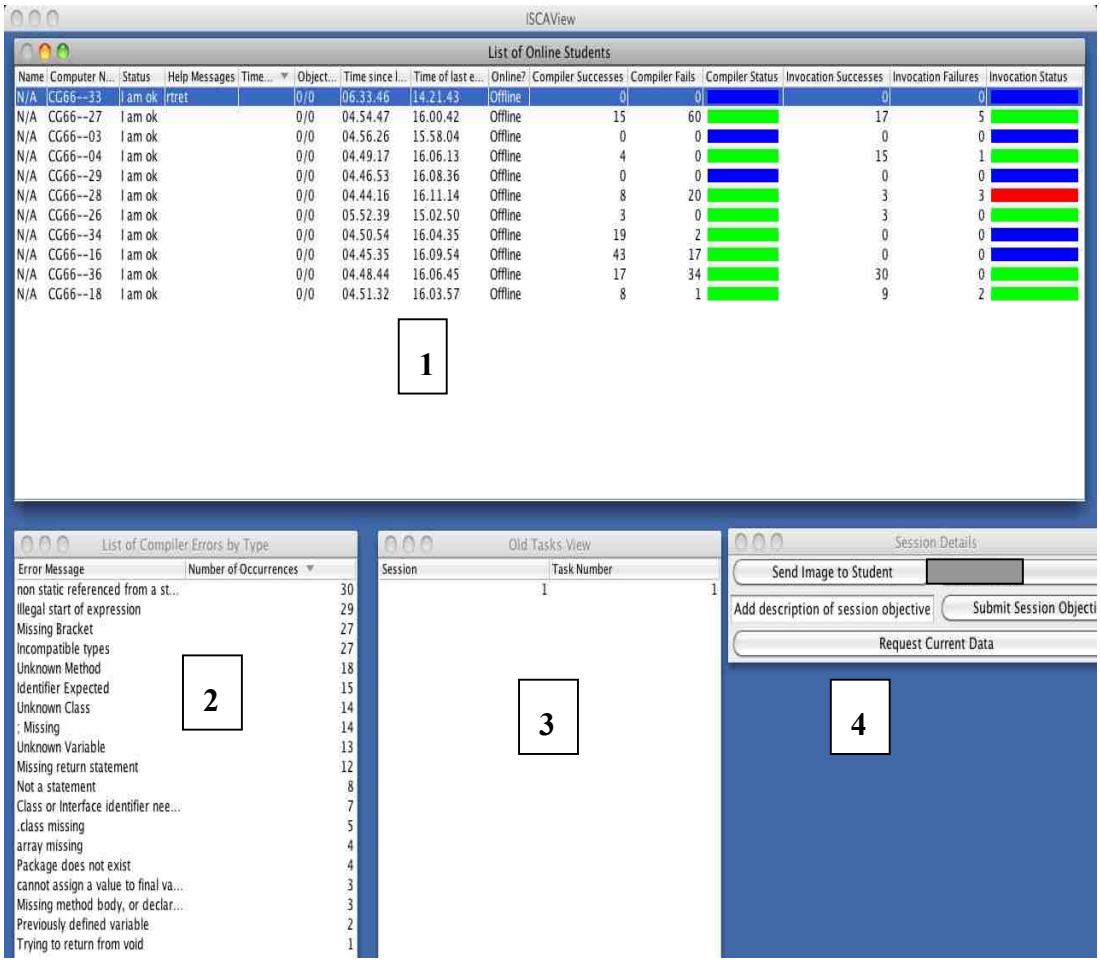


Figure 4.3: Teacher client

4.4.2. Feature A - Compiler Errors

Rationale

The rationale of Feature A – Compiler Errors, is to explore the opportunities for using compiler errors to help a teacher monitor the current status of the students. The two reasons for Feature A – Compiler Errors are:

- To see if the students are actively engaging with the set task
- To see how successful the students are with compiling their code

These reasons are both important as they address Issue 2 – Communication Difficulties. In typical laboratory conditions teachers cannot clearly view the students' screen to assess how well a student is progressing. Feature A – Compiler Errors enables teachers to observe compiler errors remotely, so they can assess what the current status of a student. Feature A – Compiler Errors would allow a teacher to view, for instance, if the student was making common novice compiler errors such as 'missing ;' [Jad05]. Feature A could also reveal errors created by a student attempting a more challenging task, for example trying to use methods from a Java class without creating an instance of that class. A compiler error like this could reveal that the student is trying to create more complicated systems than the task requires. Thus the error indicates the individual student's status but could also highlight to teachers that a student is trying to work on a problem that is too advanced for them at the current time. Therefore, TEDS can reveal which students are working at the different levels by the nature of the errors that they are making.

The compiler error data can be used and displayed in a number of different ways, which enable the teachers to observe individual student's status. The compiler error data can present errors that are common in a cohort as a whole. In addition, since the data is stored within a database the lecturer has the opportunity to review the data and use it to tailor future lectures. By tailoring lectures to those errors committed by that particular set of students, could make class content more attentive to students as they could directly use this to guide their work in *practicals*.

As Table 4.2 highlights, Feature A – Compiler Errors, is aims to counter some of the communication difficulties (issue 2 on Table 4.2). The elements of the feature that assist include the following:

- Recording errors. TEDS indicates that the students are having problems.
- Displaying the committed errors. This has the benefit that the teachers could have an indication about the kind of errors that the students' have before they interact with them student.

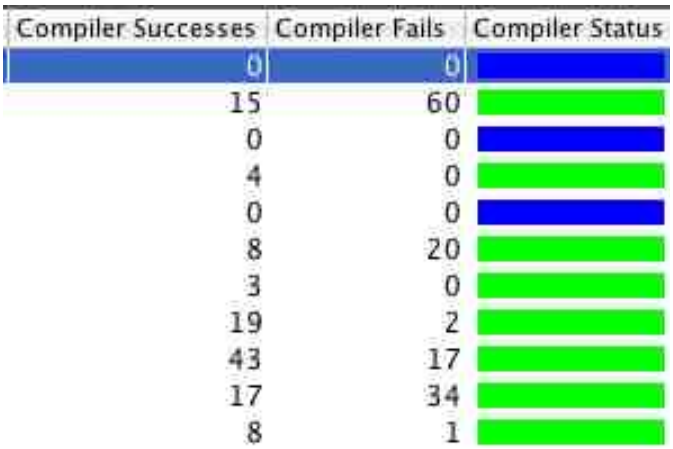
There are a number of limitations when the teachers use the compiler error data collected by the BlueJ extension to track student status. The three main limitations are listed here:

1. The data is only collected when a student's software is compiled.
2. If the student's code has multiple compiler errors, only the first compiler error the compiler finds is collected.
3. The compiler error description is not always correct i.e. a missing bracket compiler error could return the compiler error description 'illegal start of type'

These limitations mean that the teachers cannot depend completely on the compiler errors that TEDS gives them, and the teacher would have to further investigate the students code to see the true cause of the compiler error. The feature does indicate to the teachers that a student has a compiler error and that indicator is the features main benefit to TEDS.

Design and Implementation

This subsection presents example screenshots of TEDS in relation to Feature A – Compiler Errors.



Compiler Successes	Compiler Fails	Compiler Status
0	0	
15	60	
0	0	
4	0	
0	0	
8	20	
3	0	
19	2	
43	17	
17	34	
8	1	

Figure 4.4: View of compiler status

Figure 4.4 shows how a teacher can monitor a student's compiler status. The first column (Compiler Successes) is the quantity of successful compiles made by each student, the second column (Compiler Fails) is the quantity of their compiler fails and the third column (Compiler Status) has a colour-coded view of the outcome of last compile. The colours indicate:

- Blue no compile in the lesson
- Green a successful compile
- Red an unsuccessful one.

The advantage of colour coding is to enable teachers, at a glance, to quickly assess individual student's current compile status. Thereby, such a feature makes it easier for teachers to identify which students are currently successfully compiling code and those that are not.

An additional column, which is useful for more than just Feature A – Compiler Errors, is displayed in Figure 4.5. The column named 'Computer Name' has the ID of the computer, indicating where the student is seated. The ID provides the location of a student's computer (from Figure 4.5 the text before the '--' is the classroom where the student is located and after the '--' the number refers to the student's computer ID). Therefore, the teacher can locate where the student is sitting to provide assistance.

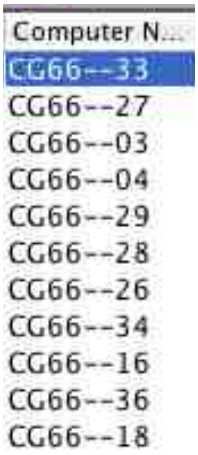


Figure 4.5: Students' computer ID's

Figure 4.6 presents a view where a chart is displayed showing a student's compiles. In the Figure the graph shows the student's percentage of compiler success rate during *practicals* in the form of a line graph. The Y-axis is the percentage of compiler success and the X-axis is the time that the events take place. The view enables the teacher to identify prolonged periods of unsuccessful compiles, which might indicate extra support should be directed towards that student. For instance, a teacher may dedicate additional support to those students who have recently recorded the lowest percentage of compile successes.

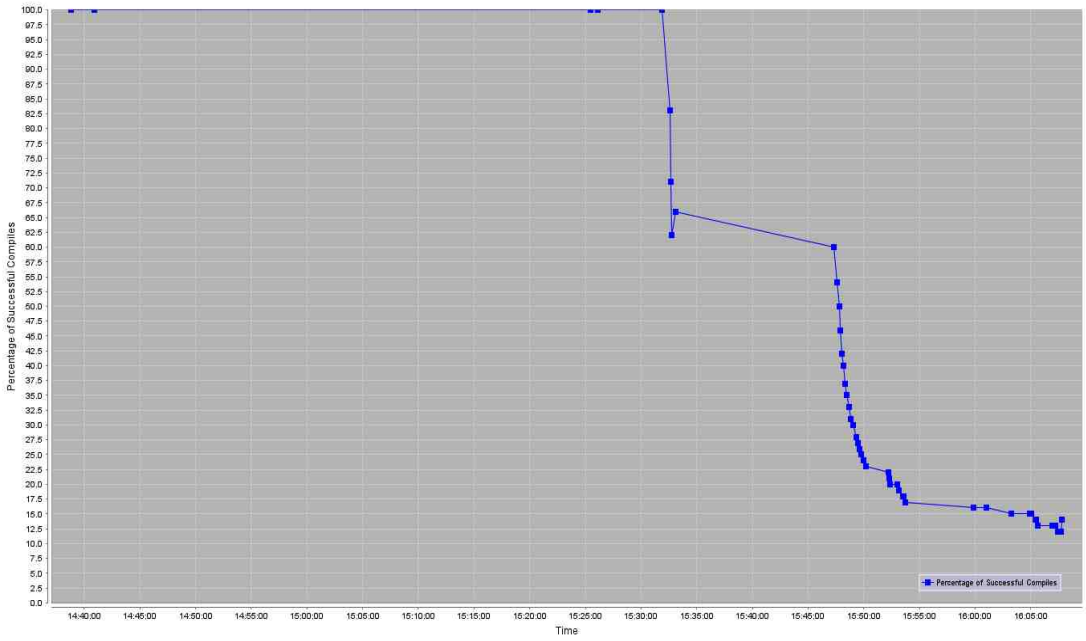


Figure 4.6: Graph to show a student's percentage of successful compiles

Figure 4.7 presents how TEDS collates errors by type across a cohort of students. This feature presents a rich set of data to course designers who may use it to examine frequent sources of errors evident in student programming practices in order to redesign future iterations of the course.

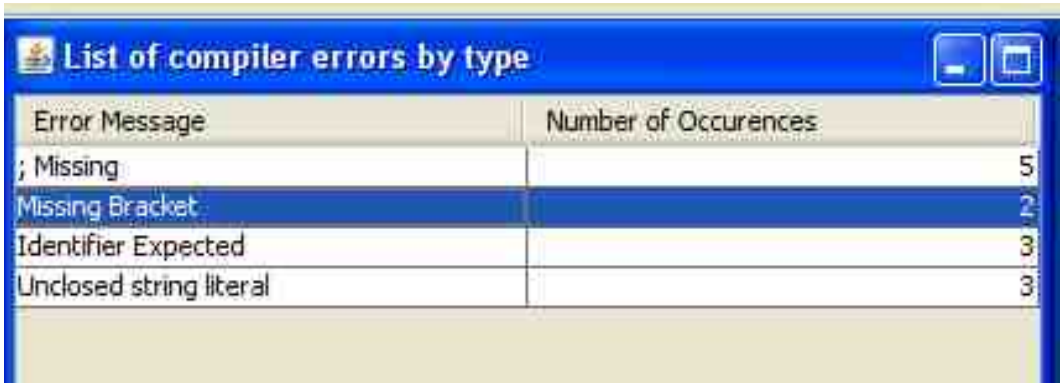


Figure 4.7: List of groups of compiler errors

Figure 4.8 shows an alternate view of the information presented in Figure 4.7 and is available for consultation by the teachers using TEDS in a practical.

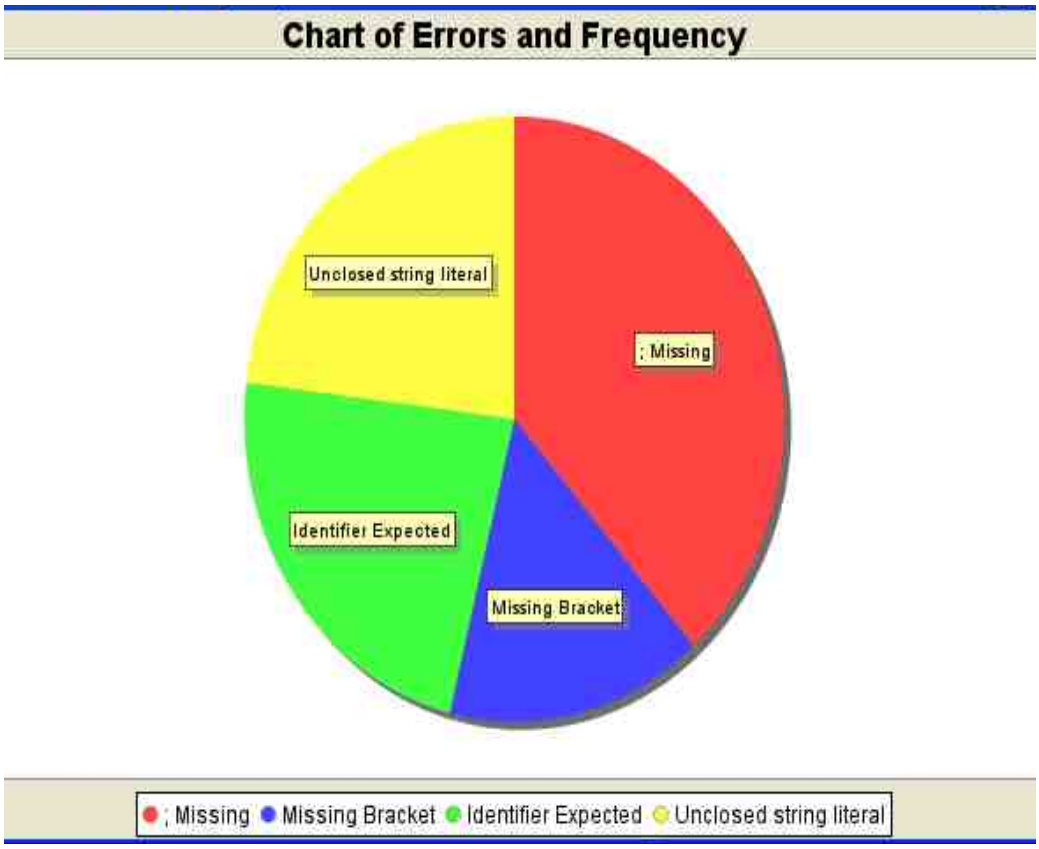


Figure 4.8: Pie chart of combined groups compiler errors

4.4.3. Feature B - Method Invocation

Rationale

The aim of Feature B – Method Invocation is to deal with a number of issues that were identified in Chapter 3 and the links are shown in Table 4.2. The ability to get method invocation data is facilitated by the BlueJ extension library, which creates an event when the student tries to run their code. Feature B gives an indicator if the code ran successfully or not and also details on the method that was run.

The main benefit of Feature B – Method Invocation, is that it provides the teachers with information on how a student's status is at a particular point of *practicals*. It does this by revealing whether a student is successfully running their code. By doing this it enables teachers to judge whether a student should be encouraged to carry on with their work or if they should be offered assistance.

A second benefit of Feature B – Method Invocation, is how it supports the students. This feature reduces the need for a student to alert a teacher, something that observations described in Chapter three identify that students avoid doing. Feature B helps as TEDS indicates to the teachers the students who need support. The teachers using this feature can use the knowledge it provides to be proactive in offering advice. This could ultimately enable teachers to more effectively allocate their time to those students who are most in need of support.

A limitation of Feature B – Method Invocation is that it is not possible to record runtime errors, which return an incorrect result, but still execute. An example is shown in the code snippet below:

```
int i = 0;
int total = badArray.size();
while(i < total){
        String currentWord = badArray.get(i);
        System.out.println(currentWord);
}
```

In this example of a while loop the code compiles and the method runs but gets stuck in an infinite loop due to the lack of ever adding 1 to variable 'i' in the loop.

In this case Feature B – Method Invocation would not report a method invocation error although the student's code does create one.

Implementation

Figure 4.9 presents the data that the teachers would have to with TEDS in relation to Feature B – Method Invocation. The data for Feature B – Method Invocation, is very similar to that displayed for Feature A – Compiler Errors.

- The first column (Invocation Successes) presents the number of successful method invocations

- The second column (Invocation Failures) presents the number of unsuccessful method invocations
- The third column (Invocation Status) presents the status of the last method invocation.

The third column uses the same colour coded indicators as Feature A – Compiler Errors; with the aim of showing teachers at a glance the status of a student's last method invocation.

- Blue indicating no method invocations
- Red indicating unsuccessful method invocation
- Green a successful method invocation.

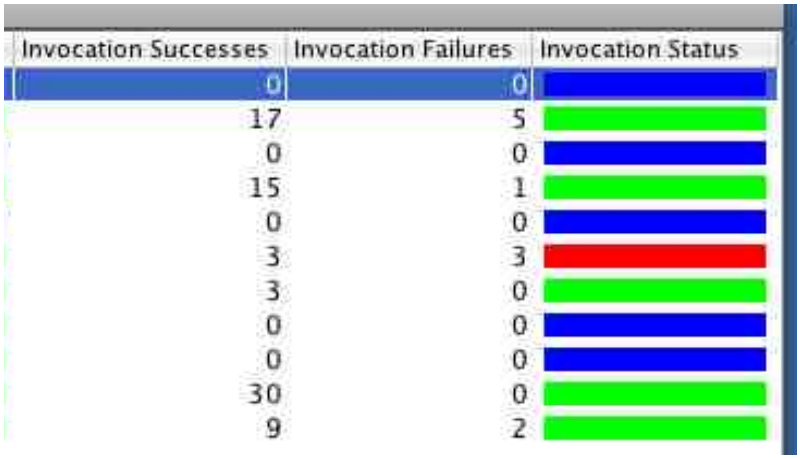


Figure 4.9: View of method invocation status

4.4.4. Feature C - Code Snapshot

Rationale

The aim of Feature C – Code Snapshot is to get a snapshot of a student's code every time they compile it, thereby collecting a profile of the student's code progression through *practicals*. The BlueJ extension library allows a developer to get the:

- Classes of the active BlueJ project,
- Methods that are within the classes

- Student's code from the classes.

Feature C – Code Snapshot exploits this component of the BlueJ extension library.

The main benefit of Feature C – Code Snapshot, is the way it enables the teachers to get a copy of the student's code every time that they compile. The teachers consequently are able to view this code to help the student to find a solution. As Chapter 3 discusses, it is a challenge for the teachers to help their students as they usually provide little context when formulating questions for help. The students are more likely to indicate that "it doesn't work" or "why doesn't it work?" By having the code and also the compiler error, the teacher now has sufficient information about what the error is. Therefore, this feature supports improved communication between teachers and students.

As was the case with Feature A – Compiler Error, by being able to view the code that the student creates the teachers are able to see the coding styles of the different students. This provides information as to the level the students are working at and the teacher has the opportunity to explore ways to improve how students implement certain elements of their software. For example, a student that was familiar with using more procedural or functional programming languages before progressing to Java may be attempting to translate traditional programming techniques into the object orientated paradigm. Through observing this, teachers could suggest more appropriate ways of writing code.

Finally with these snapshots of the student's code collected over the course of a practical, it is possible to view a novice programmer's coding skills development. For example, the techniques they use to try and overcome compiler errors.

Design and Implementation

Figure 4.10 shows the view of a student's code that the teacher would be able to access during the experimental *practicals*. This view is accessible by the teacher, as long as the student has compiled their software at least once during *practicals*. The view on the right is the student's code and on the left side is the student's project hierarchy. The project hierarchy is structured like this:

- Project
    - Package
        - Class
            - Method

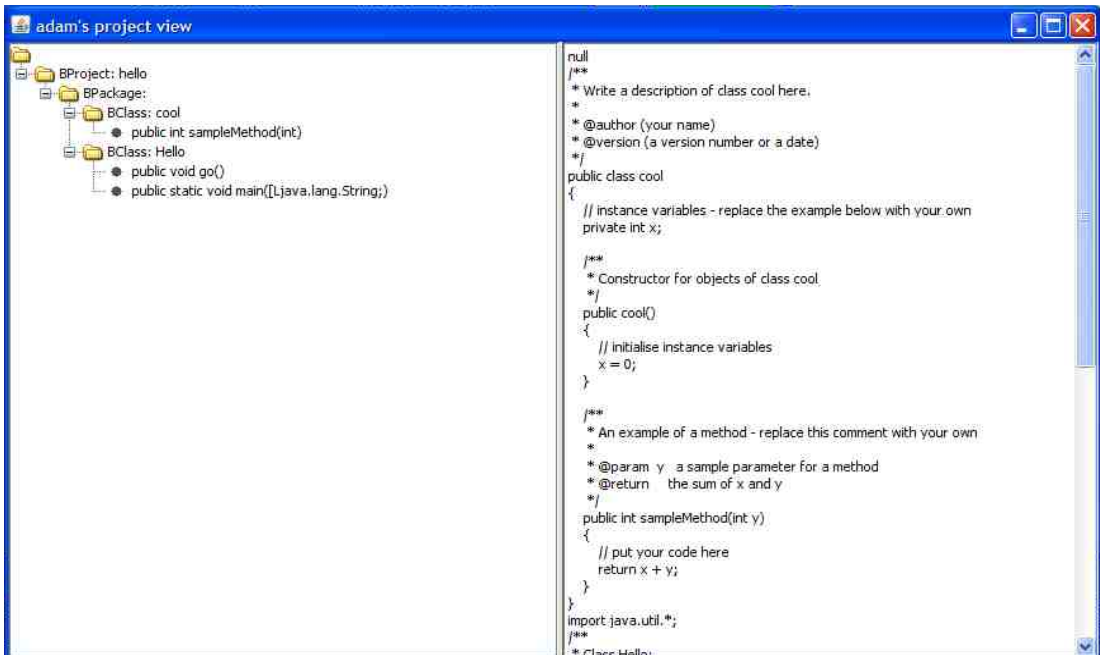


Figure 4.10: Example of student code view

The student's code on the right hand side is a direct replication of the work so a teacher can check a student's coding style. For instance to examine their approach to indentation and commentating.

4.4.5. Feature D - Help Button

Rationale

Feature D – Help Button is primarily concerned with the communication difficulties that exist in the existing *practicals* setting. Its location is presented in Table 4.2.

Feature D – Help Button is simple in that it allows the student to click a button and request help from the teachers. The help request notifies the teacher and the students

can additionally choose to submit information on their problem. This allows teachers to think about the students' problem before they go over to help.

The main reason for this feature is to provide students with an alternate way of alerting the teachers other than via verbal requests. As Chapter 3 highlighted only having the option of verbal interaction could result in students being less willing to ask for help.

Implementation

Figure 4.11 presents the student view of Feature D – Help Button. At point 1 there is a box for a student to send a text based question to a teacher. Point 2 indicates the radio button where a student can set their status to either, "I am Ok" or "I need help".

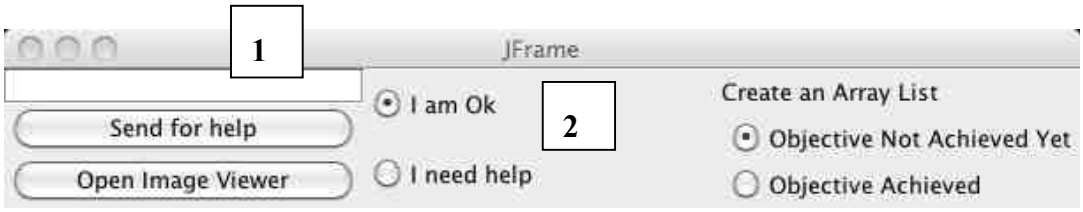


Figure 4.11: View of the student help box

The teachers then receive an update to their view as shown in Figure 4.12. Column 1 represents the status of each student and Column 2 is where a student's text questions are displayed.

Status	Help Messages	Time Help...
I am ok		
I need help	I need help with Arrays	11.18.16

Figure 4.12: View of the teachers view rows associated with the 'help button" feature

The third column on the diagram is the time that the student requested help. The time that help was requested enables the teacher to see how long the student has been waiting for assistance.

4.4.6. Feature E - Short Message Functionality

Rational

Feature E – Short Message Functionality was designed to deal with the communication difficulties that exist within *practicals* setting. The feature works in conjunction with Feature D – Help Button. Feature E – Short Message functionality like the Help Button is intended to facilitate communication between teachers and their students. It enables students, like the help button, to ask for help electronically. The students use Feature E – Short Message Functionality by typing a short message into their TEDS student window (shown in figure 4.14). This message, when submitted, is sent to the teachers console for them to address.

The short message feature deals with communication problems between the student and teacher in two ways. Firstly Feature E – Short Message Functionality, enables the students to ask for help without putting their hands up, so increasing the likelihood they may ask for help. Secondly by enabling the student to describe and contextualise their problem, which could lead to them gaining a better understanding of their problem or even to find their own solution.

The ability for teachers to be able to communicate remotely with the students lessens some of the room design difficulties, for instance where it is difficult for the teachers to walk around the classroom to talk to a student. By enabling the teacher to be able to solve a student's issues remotely could support reactive teachers in becoming more proactive and willing to help the students.

One limitation of Feature E – Short Message Functionality is that some teachers may consider it much easier to just go over to the student rather than to type in replies. So it may be observed during the investigations of TEDS that the student may make more use of this feature but that the teachers will continue to favour face-to-face communication in providing support.

4.4.7. Feature F - Objective Setting

Rationale

Feature F – Objective Setting was not directly discussed during any of the investigations, but its general idea was one of the main topics of discussion in the focus group. This feature has elements of enabling the dialogic method of teaching that Laurillard [Lau06] extols. Laurillard's method was described in detail in the Chapter 2 of this thesis.

Feature F – Objective Setting, allows teachers to set objectives for the students to complete over the course of *practicals*. When the student has completed the task they can define the objective as complete. This allows the teacher to track a student's opinion of their status. Feature F – Objective Setting, supports a Laurillardian conversation by allowing a student to set an objective as completed, which begins the conversation. The teacher is notified and has the option to either:

- Verify the completed objective, ending the conversation,
- Or if the student has not completed the objective, the teacher can explain what they need to do to complete their work.

A further benefit of Feature F – Objective Setting, is that it helps with the problem that came to light in the questionnaires, which is that sometimes the students do not always know what task they have been set. With objectives being a set and viewable by the students, they will be better informed on their objectives for *practicals*.

A limitation to Feature F – Objective Setting, is due to the openness of the course it is difficult to set objectives for the whole group. Despite this the students have certain features of Java that they must use, for example ArrayList. This is necessary so that the students can show the teachers that they can use the typical features of Java. In this example an objective would work as when the student has implemented the ArrayList in their project, they could mark this objective as complete.

Implementation

Figure 4.13 presents the screen showing how the teacher can set the objective (by typing in the objective in the text box) and then submitting it by clicking on the button indicated by the red number 1.

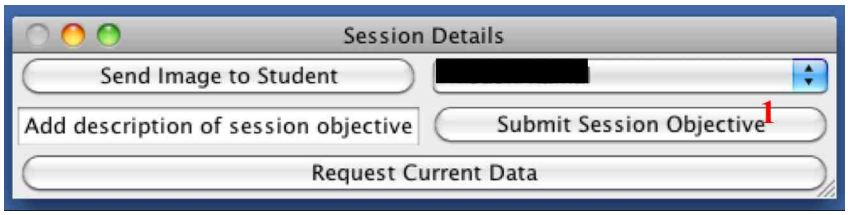


Figure 4.13: Teacher objective setting window

The objective then is sent to the student and displayed as in Figure 4.14.

Figure 4.14: Student view of the set objective

The student then can click on the radio button most suited to their status on the objective. When there is a status change the teachers are notified.

4.4.8. Feature G – Image Sending

Rationale

During traditional programming *practicals* often a teacher uses a pad of paper to draw diagrams to assist the students. This feature can be recreated by using an existing graphics package such as MS paint and a tablet PC. However, it would be helpful to a teacher if there was an easy way of passing this image to the student without either getting the student to copy the image or to e-mail the image later.

Feature G – Image Sending allows teachers to create an image and send this through TEDS to a student. This image can then be exported as a JPEG and saved for later reference.

Feature G – Image Sending could help with 2 of the Issues identified in Chapter 3, specifically Issue 2 – Communication Difficulties and Issue 3 – Range of Students Abilities. This feature gives the teacher another way of communicating to the student and also allows them to provide explanations on programming concepts tailored to a student's ability. The teachers can already give vocal feedback, but with this feature

they are able to send the students other forms of written feedback. It enables students who may have failed to understand a concept presented in a lecture to get the teacher to re-explain the concept in a different way, such as by using visualisations or design notations.

Implementation

The implemented version of Feature G – Image Sending is displayed in Figure 4.15 where the teacher can draw an image and then to send the image to a specific student. The student receives a copy of this that they can then save as a .jpeg on their computer.

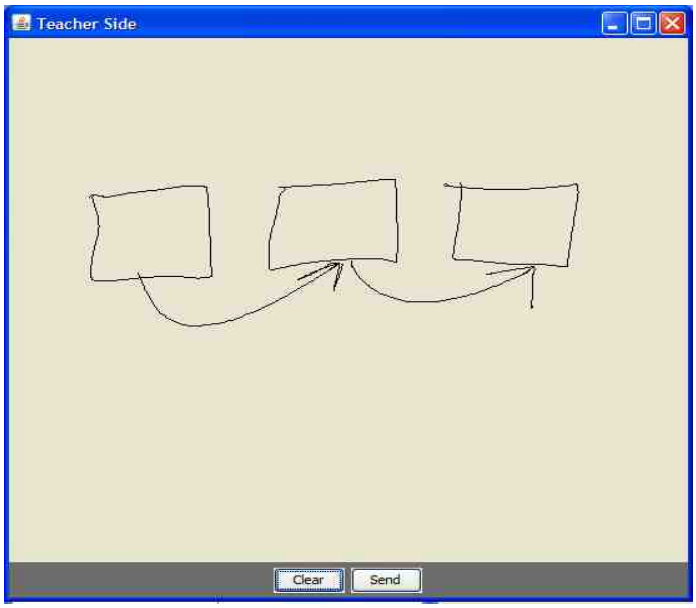


Figure 4.15: Image creating tool

4.5. Summary

TEDS or Technologically Enhanced Demonstrator Support was the suite of tools that was developed to help lessen the impact of the issues identified in response to Research Phase (i) The main aim of TEDS was to provide teachers a view of any student's current status during the course of a practical, so allowing the potential for teachers to view which students are having problems despite; the room and the

student not being willing to ask for help. The features developed to overcome these issues are:

Feature A. Report of a student's last compile i.e. success/fail;

Feature B. Report of a student's last method invocation i.e. success/fail;

Feature C. A snapshot of the student's code that is sent to the teachers;

Feature D. Ability for a student to request help electronically;

Feature E. Ability for teachers to reply remotely to a student via Short Messages;

Feature F. Objective setting functionality allowing clear objectives to be set in *practicals*.

Feature G. Ability for teachers to draw and send images to a student to help them understand concepts;

Chapter 5 explores some of considerations, related to the implementation of TEDS. These considerations include the server architecture and the hardware the teachers and students use during the investigations into using TEDS.

5. Implementation

This chapter looks at the architecture and design principles used to develop TEDS to enable it to deal with the issues outlined in Chapter 3. Important technical considerations are discussed, especially in regards to server architecture. The environment in which TEDS was used is also considered, in particular what effect it had on the implementation.

The chapter highlights some of design issues which need to be considered for TEDS in section 5.1, section 5.2 looks at the server architecture used in the design of TEDS, finally section 5.3 looks at the design considerations for TEDS user clients.

5.1. Design Issues

In the development of TEDS there are a number of design issues, which need to be considered. They included:

- The classroom
- The network

Chapter 3 noted that a system, which would allow for better communication between the student and the teachers, would be helpful in improving the learning experience for students within *practicals*.

In Chapter 3 the computer laboratory that is currently used for *practicals* was shown in Figure 3.5. In the computer laboratory individual students have their own computer to complete their work. TEDS can utilise the IT infrastructure to facilitate the communication between students and teachers. TEDS is not concerned with student-to-student communication, as this is open for abuse as well as not being a focus of this thesis. In an experiment by Lavery [Lav08] it was observed that the students used student-to-student communications functionality within a lecture to spend more time on social communication rather than focusing on the work.

Chapter 3 Issue 2 – Communication Difficulties, identified that there are communication difficulties between students and teachers within laboratory *practicals*. In response to this issue TEDS's primary concern is with improving communication by reporting information automatically collected from the student client, and communicating this to the teachers in a form of status reports.

The University IT infrastructure is such that the use of client server architecture for the TEDS system is technically feasible. This facilitates the use of a distributed server, which can be used to host the system. Using the University network is beneficial in a number of ways, including, that the network is fast, typically operating at around 100 Mbps and reliable. The University network enables multiple rooms to connect to the same session when cohorts are too large to just use one room.

The client/server architecture is best suited for TEDS with the server being the centre for data processing. The server receives data from the different clients and is tasked with the storage and redistribution of this data to the intended recipients. The benefit of having a server as a central location for data means that, if there are any network issues, for instance, should any of the clients lose connection or encounter network issues, and then the data is safely retained.

Using TEDS the students used a wired link to the network so there connection was reliable. However, when using TEDS teachers were required to use the typically less reliable wireless network, as they needed to be mobile in the classroom, to talk personally with the students. With the teachers using the wireless network means that if they lose the link to the server then a method of reconnecting and getting the current data is required.

The usual number of student-client machines within a practical can range from 15 to 30 students. The number of teachers and the size of the room dictate the overall numbers. The number of clients may impact on the server's ability to support the class. In a similar example [Sch05] a system created for interactive lectures, with a server written in Java and running on a notebook PC, was found to comfortably

operate with 300 clients. So the numbers of students is not considered to be a problem in the use of TEDS.

5.2. Server Architecture

The TEDS server has the task of receiving connections from different clients and processing the data that is required for each service.

The server is separated into 4 components or managers that deal with its functions. These are:

1. The **Connection Manager**. This component receives connections from both types of clients.
2. The **User Manager**. This has the task of checking what kind of user the client is (teacher or student). Through this information it can be identified what type of data and reports need to be sent.
3. The **Functions Manager**. This manages all of the server side processing of the data. The Functions Manager's tasks consist of collecting data received from the clients and then judging whether the teacher needs to be informed about this data.
4. The **Database Manager**. This manages the storage of the data.

Figure 5.1 presents a diagrammatical view of how the server managers interact:

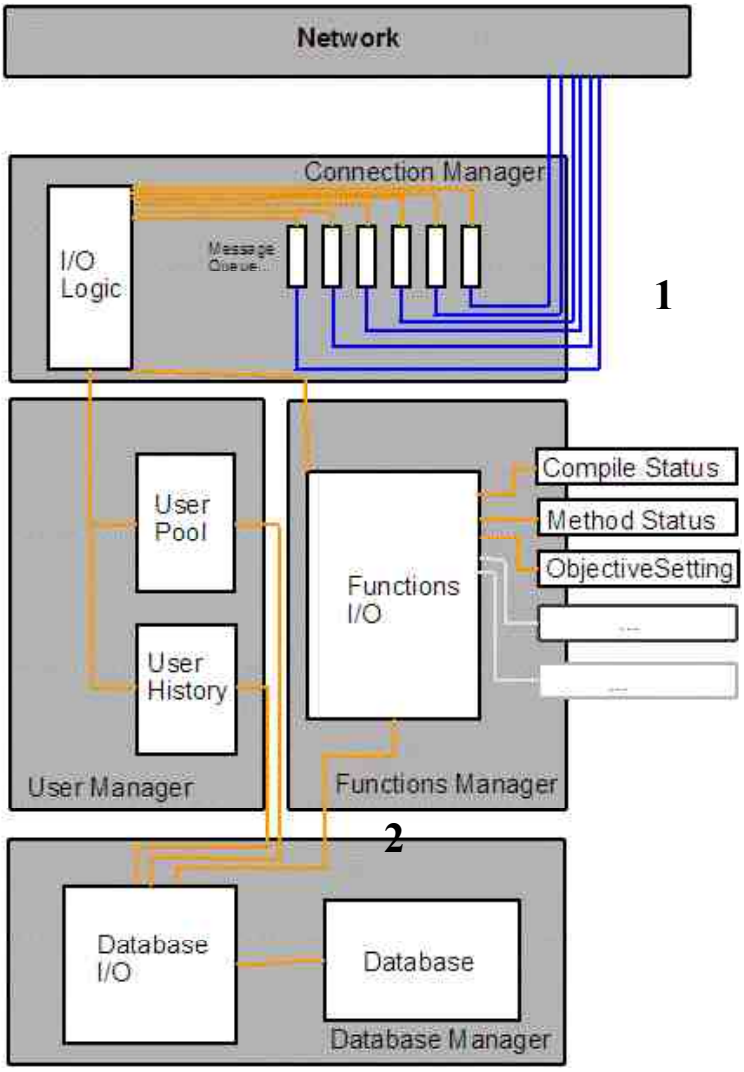


Figure 5.1: Server Managers

Figure 5.1 shows the different routes a message, sent to the server, can take before it is stored in the database. The numbers on Figure 5.1 relate to two tasks of the server, number **1** is the connections from the clients reaching the server and number **2** is the persistence of this data. These two tasks are defined below:

1. The Connection Manager first receives the message. The message is then processed and passed onto two managers:
    a. The User Manager. This is where the personal record (for example, name, current compiler/method invocation status) of the student is kept and updated according to the message that is sent.

b. The Functions Manager. This is where the message is processed to see if the teachers need to receive an update from the server on a student's status.

2. After stage 1 both the Functions Manager and the User Manager store the details of the message into the database.

Sections 5.2.1 through 5.2.4 further describe each of the manager's tasks within TEDS.

5.2.1. Connection Manager

The Connection Manager has the function of processing any connections that are received by the server.

The Connection Manager uses a socket-based approach where each client has a connection to the server through which they receive and send data. The main benefit of using this approach is speed, it operates by allowing the client to have a designated socket connection to the server ensuring the connection is fast.

A disadvantage with this approach is that if a connection is lost the Connection Manager needs to be able to catch this and have a fail-safe. The fail-safe would have to protect messages that the clients try to send until they are able to re-establish their connection.

The Connection Manager also maintains a reference of the type of client a connection is from. This data is sent to the User Manager to see if any previous data is kept on the students from past laboratory *practicals*.

5.2.2. User Manager

The User Manager is responsible for identifying which client has sent a message to the server. The design for TEDS has two types of users; teachers and students. By identifying what type of user has sent a message to the server, the User Manager keeps a record of which clients are online and also which type of user they are. For data protection purposes the errors made by the students should not be sent to an

unintended user, to ensure no student receives data on another student's status on the work.

The User Manager as presented in Figure 5.1, is connected to the database as that keeps a record of the data of student progress over a number of *practicals*.

5.2.3. Functions Manager

The Functions Manager processes all of the messages that are received from the students and the teachers that relate to a particular feature of TEDS.

The functions that are managed in the Function Manager were described in detail in Chapter 4. Chapter 4 illustrated the links between the functions that have been developed and how they seek to alleviate the issues found in Chapter 3. Table 4.2 presented the links diagrammatically.

5.2.4. Database Manager

The TEDS server holds past data from previous *practicals* that the students have participated in so that it can be accessed in the future.

Storing the data into a database also provides a safety mechanism. The benefit of this is that in the event of a server failure, the database will still have the data recorded. With this record the server can be restored with the current *practicals* data intact.

Storing a record in a database allows a teacher to access student data from previous *practicals*. The record could highlight if students have reoccurring problems with specific aspects of the course. For example, if a student scored poorly in an exam then the lecturer could look at the past data for that student, and formulate help for the students, based on the consistent errors that TEDS had recorded.

The implementation of the Database Manager uses an object-orientated database. Using Java in an object-orientated way allows data to be sent as objects. The data objects that the Functions Manager uses to send data to the clients can be taken directly out of the database and then sent to the user. This same method also works

for the input of the data, as the data received from the clients, can be saved directly as an object into the database.

5.3. Clients

The clients that use TEDS are split into two different types, teachers and students. Subsections 5.3.1 and 5.3.2 present an overview of these two clients and the factors that need to be considered in their design and implementation.

5.3.1. Teacher Client

The teacher client has to be able to present a high volume of data to the teachers. This is so the teachers can view any difficulties that the students may be encountering.

Interviews with teachers and observations in *practicals* classes highlighted that within the setting of the classroom it is preferable that teachers can move around to interact with the students on a one-by-one basis to see each individual student in the class. Chapter 3 Issue 2 – Communication Difficulties discussed the advantages of proactive over reactive teachers. Chapter 3 also discussed the reasons for the communication difficulties such as:

- Room layout
- Students unwillingness to ask for help
- And how within the current practical setting the teachers have problems tracking student current status due to the classroom environment.

With these findings considered it is a requirement of TEDS that the teacher client is able to run on a portable device.

A number of portable computer devices are available which allow teachers this mobility whilst enabling them to access data through TEDS. Three possible options are:

- Laptops

- Personal Digital Assistant (PDA's)
- Tablet Personal Computers (Tablet PC's)

All three have individual advantages and disadvantages with their use within a practical. Table 5.1 presents the advantages and disadvantages of each option.

Portable Computer Options	Advantages	Disadvantages
Laptops	• Full interactivity – with Keyboard • They usually have a large screen which are suitable for reading easily	• Not very mobile or easy to use when moving around
PDA's	• Very portable due to the handheld size.	• Not very easy to interact with for example, to modify code • Small screen which does not allow much data to be presented at any one time
Tablet PC's	• Good interactivity using touch screen • Good mobility, it is designed to be easy to hold and walk with it • Larger screen which is easy to read and allows a lot of data to be displayed	• No off screen keyboard results in a difficult to use user interface.

Table 5.1: Advantages and Disadvantages of different portable computer options

Table 5.1 highlights that despite the advantage of the PDA's portability, its small screen counteracts these benefits. This would have an impact on the ability of the teachers to provide learning support to the students. For example, TEDS frequently transmitted status updates to the teachers, which needed to be presented in a way that

is easily accessible. This would be less of an issue if the teachers had laptops as this would allow them to have a bigger screen and view a lot of data. Also with a laptop the teachers could interact with the data using tracker pads and a full keyboard. Unfortunately, laptops are not very easy to use whilst moving from desk to desk. Therefore, considering the disadvantages of Laptops and PDA's, Tablet PC's have the advantage as they have a big screen that can present a lot of data, whilst also maintaining mobility. Tablet PC's can be easily held in one hand, leaving the teacher's other hand free to interact with the device.

Considering all the factors above, Tablet PC's were identified as the best option to enable teachers to be able to support students proactively, taking into consideration the classroom environment and networking constraints.

5.3.2. Student Client

The main design constraint to be considered in the design of the student client is that the client should be unintrusive to the students. This is to limit the impact of the Hawthorne Effect [Lan58] within the experiment. The Hawthorn Effect is where a subject of observation acts differently to how they would in a situation where they are not being observed [Lan58].

Two approaches were taken to make the student client of TEDS less intrusive. They were:

- Using BlueJ so that the students appeared to be using the same software as they usually would in any other practical.
- Making the windows relating to TEDS on the student client as small and unintrusive as possible.

The Hawthorn Effect could still cause difficulties for the investigation and these are considered in the following Chapter Section 6.4.

5.4. Summary

In summary Chapter 5 presented the factors that were taken into account in the implementation of TEDS. It specifically looked at the general design issues, related to TEDS, the server architecture and the clients. Furthermore, it considered the reason behind choosing Tablet PC's as the teacher client hardware for the investigations.

Within association with Chapter 4, this chapter has presented the steps taken in the implementation of TEDS's range of tools designed for Research Phase 2 – Develop tools to overcome issues in *practicals*.

Chapter 6 will now present the design of the case studies where TEDS was used.

6. Case Studies Design

Research Phase (iii), is concerned with investigating whether TEDS can be successful at resolving the issues found during Research Phase (i). Chapter 6 is focuses on describing the case studies to test and evaluate TEDS and is structured as follows:

- The methods used during the case studies
- The format of the case studies
- The sample selection
- The limitations of the case studies

The primary aim of the case studies, was to determine whether TEDS has an impact on the way that teachers support *practicals*.

6.1. Methods

There are a number of methods that can be used to determine if there was an impact with using TEDS in *practicals*. There are some inherent limitations with these methods and these are addressed in section 6.4.

In this thesis a combination of methods are used to enable triangulation to evaluate TEDS. Triangulation was used to judge whether data collected from a number of different sources, could be combined to answer the research questions. By the use of triangulation it is possible to check the validity of the results of one set of data by using two other sets of data [Fli92]. In the theory of triangulation there are usually three different types of data collection. Within this work three methods were used:

- User Feedback (both student and teacher) and teacher interviews to get quantitative and qualitative views from the users
- Observations of the *practicals* by the researcher
- Using TEDS to record quantitative data.

In the collection of this data both qualitative and quantitative will be collected and these are summarised in the table 6.1

6.1.1. User Feedback

Aim

The aim of user feedback was to discover user experiences of TEDS in the context of programming *practicals*.

Experimental Design

User feedback was collected for two reasons (i) To discover the students and teachers views on TEDS and (ii) To discover if students and teachers are willing to accept using TEDS during *practicals*.

The user feedback in this case study (See Appendix 3 and 4) took the form of questionnaires and semi-structured interviews with the teachers.

The questionnaire used for this case study was based on one used by Scheele [Sch05]. Although Scheele's research is focused on lectures, some topics of his questions are based on the acceptance of using a new system. This topic was also relevant in this thesis in judging user acceptance of TEDS.

The use of questionnaires is common within literature of learning tool assessment including the work of Lloyd [Llo68]. Lloyd used questionnaires and tests to view attention levels and knowledge assimilation in lectures. Lloyd used three methods in his experiments:

1. Testing the students on the content of the lecture, after the lecture, to view attainment.
2. A survey of the students to see how they felt they had learnt during the lecture
3. An interview with the lecturer to see how they felt the lecture went and how they thought the students learned in the lecture.

With these three methods Lloyd could use triangulation to judge the validity of each source of data.

For the case studies in this thesis, questionnaires were given to the students and the teachers. The questionnaires were designed to get a view of staff and students' acceptance and experiences of using TEDS.

The questionnaire takes the form of a statement where the respondents can either:

- Strongly Agree
- Agree
- Disagree
- Strongly Disagree.

With a four point scale students were not given the opportunity to answer neutrally, to promote greater levels of decision [Kal80]. Kalton et al's research revealed that when presented with an opinion question with a middle option, that responders would be likely to select that, rather than an anti or pro opinion response. Kalton et al further argued that without providing respondents with a middle option, enables researchers to view the leanings of the respondents rather than definitive answers, where leanings are an acceptable result.

In addition to the closed answer questions a set of open-ended questions were presented. This was to facilitate more in depth answers to get a more detailed understanding of how the students and the teachers experienced the use of TEDS. This approach was also used by Scheele [Sch05]. Examples of the questionnaires used during this thesis are in appendices 1, 2, 3 and 4.

After this data was collected, semi-structured interviews were carried out with teachers to follow up on some of their responses. Another element of the interviews was that some of the questions used resulted from notable observations taken by the researcher during the practical class observations. This approach was used see if the teachers could triangulate any of the notable observations made by the researcher.

During the interviews questions examined if the teachers could provide reasons why they or students acted in the observed behaviours made during the *practicals*.

Expected Outputs

The expected output of the questionnaires was to collect data on how the students opinions of using the system and their overall acceptance of it. Also via open responses, qualitative feedback was collected from the users.

The outputs from the teacher interviews provided additional qualitative data about its use and any perceived benefits.

These outputs were valuable when triangulated with the observations discussed in subsection 6.1.2.

6.1.2. Observations

Aim

The aim of taking observations, when the students and teachers were using TEDS, was to be able to compare the data collected to the observations in Chapter 3. The observations are concerned with who started the interaction and the length of interaction.

Experimental Design

Participant observations are one-way observations. Participant observation is where the observer is part of the event that they are observing [Kem01]. In this case the researcher would be sat in the class, available to the students as a teacher if the other teachers are occupied. Thus the researcher appeared as if they are working within the class. This technique aims to minimise change to the learning environment and prevent students from being conscious of their being observed. Thus this approach has the intention of keeping student behaviour within the class as natural as possible.

The observations reported in Chapter 3 consisted of one observer doing simple coded records of a practical. These observations primarily recorded who instigated interactions and for how long these interactions lasted.

In this new case study the same kind of observations were performed to get comparative data, to see if there were any significant differences when using TEDS. The differences could consist of:

- Who instigated the interaction.
- The duration of interactions.

In the observations for the case studies with TEDS, TEDS provides students with additional technological possibilities for requesting help. When students use TEDS to get assistance these events are recorded by the system. An example of the sheet used to record the observations can be found in Appendix 5.

The observation method used in these case studies created a set of challenges to the researcher in performing the observations. In some cases the observer could not see if the interaction was initiated via the tool. This is because TEDS reported to the teachers any student who was having problems with their work thereby initiating potential interactions. An instance of this could be if a student was having a long run of compiler errors. Situations like this may not be noted as being instigated by TEDS as there is no way to differentiate this from any other interaction. Potential ways of limiting this could include asking the teacher if TEDS was a factor in making them interact with the student.

The accuracy of the researcher's qualitative observations can be given validity by the experience that the researcher has in these practical laboratories. The researcher has three years of experience with practical teaching. Kemp identifies experience as a benefit with the observer being familiar with the situation that they are observing [Kem01]. The researcher was also known to the majority of the students through teaching them on another course, so students are less likely to be distracted as the individual was familiar. This enabled a form of covert observation [Kem01]. Covert observations are where the subject of observation does not feel like they are being observed, as the observer is part of the environment.

Making the observer part of the environment is not easy to achieve. Kemp suggests using devices such as one-way mirrors. Since the classes took place in teaching rooms such an approach is not possible for this case study.

A limitation of the researcher being acquainted with the students could result in biased observations, but an awareness of this potential limitation would help to mitigate the issue.

Expected Outputs

The expected outputs was a set of observations, which recorded the number of interactions between the teachers and the students. Specifically the observations recorded:

- How long the interaction lasted
- Who started it:
    - Teacher
        - Through enquiring on current student work status
        - Through the system revealing problems
    - Student
        - Through putting their hand up to request interaction
        - Through requesting help through the system

The researcher also took qualitative observations of any notable events during the observed *practicals*. An instance, of a notable event could be any behaviour exhibited by students or teachers that could be considered unusual and linked to the TEDS. An example, could be a teacher who was seen as being reactive during the earlier observations, becoming more proactive within the case study practical.

6.1.3. Automated Data Collection

Aim

The aim of using TEDS to automatically capture data was to log the way users interact with TEDS.

Experimental Design

TEDS has recording facilities that have the ability to record many of the different events that a student triggers during the course of a practical (See Chapter 4). Logged data was stored in a database and the data was analysed after the practicals.

Expected Outputs

The expected outputs from this data collection was a mixture of different reports that the tool was programmed to create. Every time a user, both teacher and student, uses TEDS it sends an update to the server. These events were recorded from a number of sources for later analysis:

- Server log – This is a text file of everything sent to the server recorded in chronological order.
- The Database – This stores all the data sent to the server. These can then be accessed and analysed by the researcher.

6.2. Case Studies

The combination of the three methods of collecting data, described in section 6.1, provided data from the case study *practicals*. It also revealed if there were any differences between specific *practicals* and students behaviour.

Case Study Type	Type 1 - Teachers not using TEDS (data type collected in method)	Type 2 - Teachers using TEDS (data collected in method)
Group A	TEDS – (Quantitative)	TEDS – (Quantitative) Questionnaires – (Quantitative + Qualitative) Observations – (Quantitative + Qualitative) Teacher Interviews – (Qualitative)
Group B	TEDS – (Quantitative)	TEDS – (Quantitative) Questionnaires - (Quantitative + Qualitative) Observations – (Qualitative) Teacher Interviews - (Qualitative)

Table 6.1: Table of different case studies by groups

Table 6.1 shows the two types of case studies that were used in the evaluation of this tool. There are two groups being used in the case studies in this thesis:

- Group A which was the larger group which ranges from 25 to 30 students depending on attendance
- Group B which was a smaller group which ranges from 14 to 16 students

Both groups had three teachers to support the students.

The first case study uses the technology to gather information to see how the students work within a class. In type 1 – Teachers not using TEDS (Table 6.1), the reporting systems are not sending status reports to the teachers. The data is recorded as a

background process. In the type 2 – Teachers using TEDS, case study the teachers use the technology to help them support the students.

Observations were also carried out for Group A. During the observations taken for both groups in Chapter 3 it was found that despite the difference in the differing number of students in each group, that the observations were similar. Table 3.1, which evaluated the earlier observations of both groups using T-Tests, revealed that there was no significant difference in any of the observations. So it was decided in the case studies using TEDS that only Group A would be formally observed.

The design of the case studies was to ensure that the *practicals* would appear to the students as not being significantly different to the ones they usually attend. This was to ensure the students did not act differently due to them being observed and aware that they were part of a case study. Such an effect is referred to as the Hawthorne effect [Lan58]. However some changes had to be made. Students were required to sign consent forms and in case study type 2 – Teachers using TEDS, the teachers held tablet PC's with the reporting systems running on them. It must therefore be acknowledged that the students were aware of the case study, but otherwise design measures were taken to minimise the intrusiveness. With regards to the teachers, however, changes in the actions were unavoidable, as they were required to act differently. However, it was hoped that with repeated use of TEDS the impact of the effect would be lessened [Cla83]. Also this was the material to be collected in the the teacher questionnaires to evaluate TEDS usage.

As a conclusion the underlying principle of the case studies was to make it appear as similar to a standard practical class as possible, as to limit the impact of the Hawthorn Effect [Lan58].

This particular set of data collection methods still had some limitations, which are discussed in section 6.4.

6.3. Sample Selection

The subjects for this study were selected through using a convenience sampling method. It was a convenience sample for three reasons:

- All of the students study the introductory programming course, which TEDS had been designed for.
- The lecturers allowed the experiment to be used on their course.
- It takes place in the department where the researcher was based.

The particular *practicals* which were chosen for the study were chosen for three reasons:

- They consisted of two thirds of the students on the course.
- The chosen *practicals* were the biggest group and smallest group in that current year's cohort. This would therefore provide data from the extremes of group sizes in that cohort.
- The groups ran one after the other in the same laboratory. This meant that it is more convenient with less need to move hardware between experiments.

6.4. Limitations

There are five limitations with the case studies as described in this chapter. Many of the limitations were created due to the attempt to limit the impact of the Hawthorne Effect [Lan58].

1. Within the experiment it was not possible to record all the different interactions between the teachers and the students. This was due to the attempt to maintain a low level of intrusion, which means it was only possible to see that an interaction took place and who began it. This means that useful data may have been missed, such as the subject of the interactions.

   The data could have been collected covertly to avoid the Hawthorn effect with hidden cameras and Dictaphones, yet this would have many ethical issues. Also

the devices are not certain to be completely hidden from the students and the teachers. With covert recordings the teachers would have to be complicit to hide the voice recorders and then they could then be less likely to act as they would usually. The experiment could use, open cameras and voice recorders, but these potentially would have been more likely to cause both the students and teachers to act differently.

In conclusion it was important to consider, how many observations can be taken before the subjects of observation begin to act differently. So it is hoped observations take place in this experiment in a way that the impact of the Hawthorne Effect was lessened whilst still maintaining a substantial amount of collected observational data.

2. With new technology there is always a risk of novelty creating false results. The novelty of the system could cause the users to act differently to how they would after using a system over an extended amount of time [Cla83].

   TEDS has been designed to ensure that it collects data in the background. For example, a student's client of BlueJ, with TEDS, appears the same as the BlueJ they use every week but with a little window that they can use to ask for help. So it was hoped that with this small change to the usual system that the novelty effect was minimal.

   Conversely, for the teachers with TEDS they had access to a lot more data than they usually would, so for teachers the novelty factor is unavoidable. The teachers were shown the tool before the experiment and used the system over four, two hour practicals. It was hoped that the more frequently the teachers use the tool that this will lessen the impact of the novelty factor [Cla83].

3. A third limitation was that there was not enough time for longitudinal studies on TEDS effectiveness. Which would of allowed comparisons if the students have learnt more successfully in the practical when the teachers have access to TEDS. This is due to two reasons:

   1. The openness of the *practicals*

2. The limited time scale of this experiment

Ideally assessment tasks could be used to examine how much the students have learnt within the specific practical [Sch05] [Llo68]. But in the case of this project the students in *practicals* learn at their own pace. So creating an assessment that for a student cohort to examine their learning during the practical would have been problematic.

The limited time scale of the experiment meant that it cannot reported if prolonged use of the system could allow the students to increase their learning during *practicals*. While it would have been beneficial to carry out a study of a semester of the traditional *practicals* and compare the outcomes to a similar time period with the use of TEDS, similarities between different cohorts could not be guaranteed. These would all impact on the reliability of the test**.**

4. The fourth limitation was created through the necessity of using the Durham Universities centrally managed computers in the case studies.

   The University manages the open access computers that the students can use in the laboratories in Durham. The University therefore controls the software that can be used including that of the BlueJ installation. To extend BlueJ with the TEDS client, a user needs access to the extensions library of BlueJ, which the University does not allow. The University does allow users to temporarily install software onto the computers local drive for the duration of a session that they are logged on. This allows the workaround by installing a local version of BlueJ complete with the TEDS client at the beginning of each case study practical.

   This workaround is a limitation as it left the potential for students to have two instances of BlueJ running during the case study session, one recording data as a TEDS's client and the other just being a standard instance of BlueJ. This limitation could have resulted in students accidentally using the standard version of BlueJ and therefore appearing on the TEDS reports as being a student who was inactive, whereas they could be compiling and running code intensively just on the wrong BlueJ installation.

The teachers were alerted to this issue and therefore with this awareness they prompted students to use the TEDS BlueJ installation.

5. The fifth limitation was the ordering of the case studies. The case studies are ordered so that Type 2 just recorded data from the students. Type 3 presented the recorded data to the teachers so that they could use it to support the students.

   This design was a limitation as it was not possible to see if factors such as increasing difficulty of the work or increasing knowledge impacted on the results. These factors could potentially impact on students or staff evaluations of TEDS.

   A cross over design may have mitigated against this issue and with two groups it would be possible to order the case studies differently, for instance during Case Study Two, Group A's teachers could use the data collected by TEDS, and in Case Study Three Group B's teachers. This would allow the experience and work difficulty data to be compared with a group where the teachers use TEDS and one where they do not.

   This ordering was not adopted, as the observations taken for Chapter 3 showed that the total number of students in each group impacts on how teachers interact with the students. For example, Group B has a higher ratio of teachers to students, so the teachers were more inclined to interact in a one to one manner with the students.

   It was noted in section 6.2 in the reference to table 3.1, that the difference between Group A and Group B observations was qualitative and that it was not reflected by the quantitative data collected, as t-tests taken of the observation data showed no significant difference between the two. However, the qualitative data did seem to emphasise that the groups were different enough to make the data collected from the two groups incomparable.

   Considering the differences in the way interactions were conducted depending on the teacher to student ratio, it was decided to adopt the order of experiment outlined in Table 6.1.

6.5. Summary

Chapter 6 describes the design of the case studies that were used to measure TEDS effectiveness at overcoming the issues identified in Chapter 3. This chapter has acknowledged that some limitations exist in the proposed methods, but where feasible procedures have been put in place to lessen their effects.

The results from these case studies are presented in the next chapter along with an evaluation of the findings.

7. Case Studies

Chapter 7 look at the three case studies performed during the evaluation of TEDS. Each case study has its own section: Section 7.1 is on Case Study One and presents a small preliminary case study using TEDS. Section 7.2 presents Case Study Two where TEDS was run on the students' computers but without the teachers having access to the data. Section 7.3 finally presents Case Study Three, which was where TEDS was used to assist teachers to support students within *practicals*.

Each case study is split into three subsections:

1. The context of the case study
2. The results collected in the case study
3. The evaluation of the case study and the main findings.

In this chapter the term *event* is used. In the context of this chapter an *event* covers both compiles and method invocations.

7.1. Case Study One

The first set of case studies took place in June 2008 and consisted of two revision *practicals* designed to prepare first year students for making the transition into the second year.

The main aims of the case study were to:

- Ensure the tool operated in a practical,
- Get feedback from the teachers on the data collected by TEDS
- Get feedback from the students on using TEDS.

The *practicals* were optional and were aimed at students who had difficulty completing a task they had been set over the summer break.

Teachers and the course leader supported the *practicals* so that the students could get expert advice if they had specific problems with their work.

As Chapter 6 highlighted, the data collected within this set of case studies was data from the system and questionnaires. Unfortunately, the two *practicals* were not well attended. Out of the two *practicals* only a combined total of nine students attended which is around 15% of that cohort. Despite the low attendance qualitative observations, feedback and the testing of the TEDS were still possible.

7.1.1. Results

Table 7.1 shows the compiler and invocation success and failure counts, which was data collected by TEDS from the students. The different columns in Table 7.1 are:

i. Students – this is the anonymised id of the students in the group

ii. Compiler Success – Shows a count of the amount of successful compiles

iii. Compiler Fails – Shows a count of the amount of compiler fails

iv. Invocation Success – Shows a count of the amount of successful method invocations

v. Invocation Fails – Shows a count of the amount of failed method invocations

Student ID	Compiler Success	Compiler Fails	Invocation Success	Invocation Fails
1	14	10	96	2
2	8	15	0	0
3	8	15	0	0
4	5	7	15	2
5	0	0	0	0
6	2	2	14	1
7	0	2	0	0
8	6	1	7	0
9	0	0	0	0
Totals	43	52	132	5

Table 7.1: Case Study One session data

During Case Study One, nine students agreed to use TEDS, producing a total of 95 compiles and 137 method invocations.

The breakdown of the compiles is that 43 were successful, which is 45.26% of the total compiles. Students 2 and 3 each contributed the highest frequency of compiles, 23 (16.79% of total) of which eight were successful (34.78% of their total). Also they both contributed no method invocations.

The breakdown of the method invocations is that 132 of the total were successful which is 96.35% of the total method invocations. Student 1 alone contributed the highest frequency of method invocations, 98 (71.53% of the total), of which 96 were successful. The high frequency of successful method invocations hides the fact that

the code was not returning the result that the student wanted. This is discussed in further detail in section 7.1.2.2.

Excluding students 1, 2 and 3 the other six students contributed 26.32% of the total compiles (25/95) and 28.47% of the total method invocations (39/137). This includes students 5 and 9 who both contributed no compiles or method invocations during the case study. A possible explanation for the results of these two students could be that they just showed their problem to the teachers and were helped to a solution that did not require them to compile or run their code. Another potential explanation could be that they are using another instance of BlueJ that is not running the TEDS student client.

Table 7.2 shows the collated compiler errors that were made by the students. The left hand column is a description of the error type and the right hand column is the number of times the error occurred in the case study.

Error Message	Number of Occurrences	Percentage of Total
Cannot find symbol	16	30.77
Incompatible types	8	15.38
Unexpected type	3	5.77
Missing Bracket	3	5.77
; Missing	3	5.77
Missing return statement	3	5.77
Illegal start of expression	2	3.85
Variable not initialized	2	3.85
Class or Interface identifier needed	1	1.92
Previously defined variable	1	1.92
Identifier Expected	1	1.92

Table 7.2: Case Study One, grouped compiler errors

Table 7.2 presents the compiler errors collected from the students who participated in Case Study One. In Case Study One, 52 compiler errors occurred and can be classified into 11 different types.

The highest occurring compiler error type was “Cannot find symbol” with 16 occurrences (30.77% of the total errors). The second highest compiler error type was “Incompatible types” with 8 (15.38% of total) occurrences.

Finally, in Case Study One there were 5 compiler error types with 2 or fewer occurrences.

7.1.2. Evaluation

The findings from Case Study One are now discussed. The section highlights a number of the benefits of using a system like TEDS in a practical situation.

The three findings from Case Study One are:

i. Interaction is aided in smaller groups with a high ratio of teachers to students.

ii. Successful method invocations do not always mean successfully running code.

iii. Teachers prefer to see data at a glance rather than having to interact with the system to get to it.

The following subsection looks at these three findings in more detail.

7.1.2.1. The benefits of smaller groups

Case Study One, unlike the two other case studies described in sections 7.2 and 7.3, had almost one teacher to every one student and this dramatically changed the way that the students interacted with the teachers. The qualitative observations taken by the researcher showed that the students were quick to ask for help and make use of the one-to-one assistance to overcome the problems that they were facing.

A number of limiting factors must be taken into account when considering the behaviour exhibited by the students in this case study. A significant factor is that the students participating are those who have identified a problem with the set task and have come to the class to discuss this problem with the teacher. This could lead to bias as the students who attended the *practicals* may have been those more likely to ask for help in the usual practical settings anyway.

The impact of having almost one-to-one teacher support was highly advantageous for the students. It enabled Laurillardian [Lau06] dialogue within the practical setting without the teachers being concerned that they were leaving students waiting with problems as discussed in section 2.2.

The use of smaller groups and higher ratio's of teachers to students has much literature promoting its benefits at all levels of education, yet resource constraints mean that such a favourable staff – student ratio is rarely possible. This is discussed in section 2.2.

In the later case studies there is a higher ratio of students to teachers, which is a more realistic practical setting. This puts more focus on TEDS's potential to assist teachers in keeping track of a higher number of students than during Case Study One.

7.1.2.2. Runtime Errors

During Case Study One, partly due to the ratio of students to teachers, the teachers did not use the reporting functions of TEDS, although the collected data was analysed by the researcher. One particularly notable result was that one student completed 96 successful method invocations. Further analysis of this identified that although the student's code was running successfully it was not working as was intended. Such a runtime error is not something that TEDS alone can identify in the information that it records.

The fact that TEDS does not reveal some runtime errors was not so relevant in Case Study One as the students had a teacher assisting them at all times. It is, however, important to note that TEDS has the potential to reveal false positives to teachers when they become more reliant to the data it reports. For instance, it may lead to the assumption that students are running code successfully where they are in fact not.

A potential solution to TEDS not being able to detect a hidden method invocation would be to have a dialog box asking the student if the method returned the correct result. However, this may make students more self conscious of the teachers watching them and thereby subverting a principle design feature of TEDS's student client. This design feature was for the student client to be relatively unobtrusive and to allow the students to work as usual. Students may become frustrated with TEDS if they were asked each time they ran their code to confirm if the method had returned the correct result.

However, the method invocation data does highlight to teachers that the students are active at any given time, which in itself is useful.

7.1.2.3. Teachers Behaviours Using TEDS

Despite the teachers not using TEDS extensively it was still possible to identify that one of TEDS features was unlikely to be useful in a class situation. It was possible to identify this by a mixture of teachers' feedback and also through the observations by the researcher of how the teachers used TEDS. Feature C – Code Snapshot was the feature that was not that useful in Case Study One within the classroom setting as it was shown to be too impractical to use whilst trying to interact with students. It was found that the teachers would be more likely to just ask the students to show them their code on the student's computers.

The teachers requested that they be presented with an overall view of the class and the individual student's number of compiles and method invocations, rather than having to interact with the software to locate this data. These design considerations were added to TEDS and used for the final two case studies.

7.2. Case Study Two

The second set of case studies was carried out during the first term of the 2008 – 2009 academic year. In Case Study Two the students' had TEDS student client running on their computers but the data collected from the students was only viewable by the researcher and not the teachers.

At this point of the academic year the students' had varying levels of programming ability. The students ranged from complete novices who had never programmed before to professional programmers who have been employed to complete programming projects. This issue of varying ability was discussed in Chapter 3 Section 3.3.3 as being an issue that teachers have to deal with in *practicals*.

Within the *practicals* in Case Study Two the students were using Java with the BlueJ IDE, to develop simple systems to assist them with understanding object orientated programming.

The aim of Case Study Two was to explore the data TEDS collects from the students and to consider the possibilities of using this data to support them, when the teachers use the live data collected by TEDS in Case Study Three. Case Study Two is a comparison group that was created to compare to Case Study Three, where the teachers used the data to support the students.

7.2.1. Results

During Case Study Two, three *practicals* were used to record data. Each practical-class is referred to as a session within the context of the results and the evaluation. Two were taken from Group A (Group A Session (i) and Group A Session (ii)) and one was taken from Group B (Group B Session (i). Group A is the larger group in terms of number of students (25 to 14).

The results are split into sessions to differentiate between groups and also to consider how the level of student learning effects the errors they create. This is necessary as during Case Study Two, Group A Session (i) took place four weeks before Group A Session (ii) and Group B Session (i), so the compiler errors collected in both sets of sessions could reflect the progress in level of the students' study.

7.2.1.1. Results from Group A Session (i)

This subsection presents the results collected from Group A Session (i). Table 7.3 presents the students' compiler successes and failures and method invocation data.

Student ID	Compiler Success	Compiler Fails	Invocation Success	Invocation Fails
1	7	14	24	1
2	11	5	18	0
3	4	4	3	0
4	4	7	37	0
5	2	2	15	0
6	4	1	25	0
7	19	6	51	0
8	10	3	28	0
9	6	12	15	0
10	8	19	9	0
Total	75	73	225	1
Mean	7.5	7.3	22.5	0.1
Standard Deviation	4.95	5.85	13.97	0.32

Table 7.3: Group A Session (i), student data

The data presented in Table 7.3 is from the ten students who used TEDS from Group A Session (i). Together they produced a total of 148 compiles and 226 method invocations.

The 148 compiles consisted of 75 successful compiles. This is 50.68% of the total compiles. Student 10 was the highest contributor of compiles with 27 compiles (18.24% of total) of which 8 were successful (29.63% of their total).

The 226 method invocations almost entirely consisted of successful method invocations where only one method invocation was unsuccessful. Student 7 contributed the most method invocations with 51, which is 22.57% of the total.

The other eight students contributed 71.62% of the total compiles (106) and 73.45% of the total method invocations (166). These results demonstrate the students displaying an equal level of engagement with the task.

Table 7.4 shows the groups collated compiler errors.

Error Message	Number of Occurrences	Percentage of total
Missing return statement	12	16.44
Integer number too large	11	15.07
Missing Bracket	9	12.33
; Missing	8	10.96
Cannot find symbol	8	10.96
Identifier Expected	7	9.59
Not a statement	5	6.85
.class missing	2	2.74
Illegal start of expression	2	2.74
Incompatible types	2	2.74
Package does not exist	2	2.74
Possible loss of precision	2	2.74
Variable not initialized	2	2.74
Class or Interface identifier needed	1	1.37
Trying to return from void	1	1.37

Table 7.4: Group A Session (i), grouped compiler errors

Table 7.4 presents the students' grouped compiler errors. In this session there were 73 compiler errors consisting of 15 different types.

The highest occurring compiler error type was "Missing return statement" with 12 occurrences. The second highest occurring compiler error type was "Integer number too large" with 11 occurrences.

Out of the 15 types of compiler errors, six of these occurred only two times and two errors occurred only once.

7.2.1.2. Results from Group A Session (ii)

Table 7.5 shows the compiler and invocation success and failure counts from Group A Session (ii).

Student ID	Compiler Success	Compiler Fails	Invocation Success	Invocation Fails
1	35	2	123	5
2	18	9	23	1
3	3	7	18	1
4	6	13	1	0
5	7	5	8	0
6	11	22	11	0
7	112	56	52	1
Total	192	114	236	8
Mean	27.43	16.29	33.71	1.14
Standard Deviation	38.82	18.67	42.65	1.77

Table 7.5: Group A Session (ii), student data

Table 7.5 shows the data collected from the seven students who used TEDS in this practical. Together they contributed 306 compiles and 244 method invocations.

Considering the breakdown of the compiles, 192 were successful which is 62.75% of the total compiles. The highest contributor of compiles was student 7 who made a total of 168 (54.9% of total) compiles of which 112 were successful (66.67% of their total). This student was also a high contributor with regards to method invocations with 53 (21.72% of total).

Method invocations in Group A Session (ii) were dominated by successful method invocations as with Group A Session (i). In this session 236 of the total method invocations were successful which is 96.72% of the total. Student 1 was the highest contributor of method invocations 128 (52.46% of total).

The other five students contributed 33.01% of the total compiles (101) and 25.82% of the total method invocations (63). This result highlights that in this group, the compiles and methods invocations were not as evenly distributed between the students as the previous group.

Table 7.6 presents the different types of errors and the frequency over the course of the Group A Session (ii).

Error Message	Number of Occurrences	Percentage of Total
Cannot find symbol	54	47.37
Incompatible types	12	10.53
Missing Bracket	8	7.02
; Missing	6	5.26
Private Access Violation	4	3.51
Previously defined variable	4	3.51
Package does not exist	4	3.51
Missing return statement	4	3.51
Variable not initialized	3	2.63
Possible loss of precision	3	2.63
Identifier Expected	2	1.75
Class or Interface identifier needed	2	1.75
Not a statement	2	1.75
Illegal start of expression	2	1.75
Trying to override abstract methods	1	0.88
Trying to return from void	1	0.88
Void can't be used here	1	0.88
Illegal escape character	1	0.88

Table 7.6: Group A Session (ii), grouped compiler errors

Table 7.6 presents the collated students' compiler errors from Group A Session (ii). In this session there were 114 compiler errors consisting of 18 different types of errors.

In Group A Session (ii), 54 of the errors came from the same compiler error type, which was “Cannot find symbol”. This is opposed to 12 “Missing return statement” compiler errors in Group A Session (i). The only other type with over ten occurrences was the compiler error type “Incompatible types” with 12 occurrences, which is 10.53% of the total.

7.2.1.3. Results from Group B Session (i)

Table 7.7 presents the compiler success and failure rate and the invocation success and failure counts from Group B Session (i).

Student ID	Compiler Success	Compiler Fails	Invocation Success	Invocation Fails
1	1	0	2	0
2	54	22	35	1
3	7	4	0	0
4	6	11	7	0
5	2	1	0	0
6	15	28	0	0
7	35	34	55	1
8	0	0	0	0
9	16	26	17	0
Total Group	136	126	116	2
Mean	15.11	14	12.89	0.22
Standard Deviation	18.22	13.57	19.68	0.44

Table 7.7: Group B Session (i), student data

Table 7.7 presents the data collected from the nine students that used TEDS in Group B Session (i). In total the students produced 262 compiles and 118 method invocations.

The 262 compiles consisted of 136 successful compiles, which is 51.91% of the total compiles. Student 2 was the highest contributor of compiles with 76 (29.01% of total) of which 54 were successful (71.05% of their total).

The 118 method invocations were almost entirely successful, with just two unsuccessful. Student 7 was the highest contributor with 56 (47.46% of total) method invocations.

The other seven students contributed 117 compiles (44.66% of the total) and 22.03% of the total method invocations (26). Four of these students did not contribute any method invocations during the session with one of these also not making any compiles.

Table 7.8 presents the compiler errors and the number of the errors that occurred in Group B Session (i).

Error Message	Number of Occurrences	Percentage of the total
Cannot find symbol	35	27.78
; Missing	20	15.87
Illegal start of expression	17	13.49
Missing return statement	13	10.32
Incompatible types	12	9.52
Missing Bracket	12	9.52
Private Access Violation	4	3.17
Missing method body, or declare abstract	4	3.17
Identifier Expected	2	1.59
Trying to Override abstract methods	2	1.59
Class or Interface Identifier needed	2	1.59
Unexpected type	1	0.79
Variable not initialized	1	0.79
Void can't be used here	1	0.79

Table 7.8: Group B Session (i), groups combined compiler errors

Table 7.8 presents the combined compiler errors from Group B Session (i). In the group there were 126 compiler errors that consisted of 14 different types.

The highest type of compiler error was, as in Group A Session (ii), “Cannot find symbol” with 35 occurrences (27.78% of the total). The second highest occurring compiler error type was “; missing” with 20 occurrences (15.87% of total).

In Group B Session (i) there were eight error types with four or fewer occurrences and three with just one occurrence.

7.2.2. Evaluation

The aim of Case Study Two was to evaluate if TEDS collects data that has the potential to be used by teachers to support students more effectively.

The four findings listed below did highlight that TEDS did record data that could potentially be useful to teachers. The findings were that:

i. The majority of students' did compile/run code

ii. Students do not always run their code

iii. Types of compiler errors are common across the cohort

iv. Some students did not run or compile code

These four issues are now discussed in the remainder of this subsection.

7.2.2.1. The majority of students' did compile/run code

To be an effective teacher support tool TEDS requires students' to compile and run their code. This thesis has identified that the majority of students in the case studies did, perform these actions.

The three *practicals*, that Case Study Two consisted of, had a high number of compiles and method invocations. The totals are:

- 26 students participated in the case studies with a combined total of 1304 events (An event is a compile or method invocation)
- 403 successful compiles
- 313 unsuccessful compiles
- 577 successful method invocations
- 11 unsuccessful method invocations

The total of 1304 events over a six hour period is substantial amount of data for the teachers to use to track current status and highlights that the students are at least working and running code in *practicals*. The volume of the events alone causes one

of the TEDS features to take more prominence, the presentation of data. The data is presented so the teachers were not overwhelmed and could miss important events.

In summary, it was revealed that the students do create the events that TEDS presents to the teachers. It is highlighted again in the evaluation of Case Study Three (section 7.3.2), how TEDS provides these activity measures, which can be used by the teachers to judge the activity levels of the students.

7.2.2.2. Some students do not execute their code

The data collected by TEDS identified that some students created a significant number of compiles during the practical but at no point during that time did they execute the code to see if it was functioning correctly. In industry the programmers work on very big projects where it is not necessary or even possible to run code very often, due to modules relying on other modules to do their tasks.

The results collected by TEDS revealed that some of the students who did not run their code still compiled it. Compiling the code checks the syntactical correctness of their code. TEDS revealed the majority of the students did execute the code that they had written.

Out of the 26 students who participated in Case Study Two, four did not execute their code during the class. With this data, the teachers could reiterate the benefits to students of executing their code at suitable points. For instance, if a student's code has runtime errors, it could be possible that they made a lot of changes without fixing the original problem, and actually create more runtime errors elsewhere. A better strategy could be to use test driven development (TDD). TDD is where a programmer first creates a test case and uses this to 'drive' the development of their software [Edw03].

Debugging and testing code is a vital skill for programming, and some higher education institutions put great focus on these skills on their courses [Ahm05]. An example is from Edwards [Edw03] who used TDD to prompt students into creating test cases and then generating their systems using the test case as a guide.

In summary TEDS reveals those students that are not executing their code. It must be noted that the case studies took place early in an introductory programming course where some students have problems in generating code. So the main focus of lectures and the tasks that the students have to complete are to generate code. The course focus does not move to testing until the end of the academic year, which was after Case Study Three. So it is understandable that some students would not be aware of such testing strategies.

A number of students who participated in Case study 2, are aware of debugging and executing as *issue 3 (Students range of programming experience)* revealed, most students have programming experience prior to coming to University (25 students out of 35 students surveyed). This could be a possible explanation for why TEDS has recorded the majority of students executing their code, as they may be capable of generating code and be inquisitive if it successfully executes.

7.2.2.3. Types of compiler errors are common across the cohort

During Case Study Two it was found that two of the three *practicals* there was one error, which dominated the overall total. In case studies Group A Session (ii) it was "Cannot find symbol" (47.37% of total, Rank 1) and once again in Group B Session (i) it was "Cannot find symbol" (27.78% of total, Rank 1).

The "Cannot find symbol" compiler error occurs when a programmer tries to call a class, method or variable that does not exist in the scope from which that they attempt to access it. Two factors could lead to this compiler error and both had instances during Case study 2:

- That the programmer spelt the called class, method or variable wrong. For example Class *Tast* rather than Class *Test*.
- The class, method or variable is not in the scope of where it is being accessed. This could occur due to two reasons:

- The programmer is calling the wrong class. For example that the method run() is in class A, but the programmer attempts call the method in the class B.
- The programmer attempts to access something that does not exist. For example they try and call the method run() in the class A, but they have not written it yet.

Instances from Case study 2 include, 'cannot find symbol – method get()' and also 'cannot find symbol – constructor Artist()'.

The "Cannot find symbol" compiler error was common. One reason for this could be the way the BlueJ does not have an auto complete functionality, which is a function in some IDE's, for instance Eclipse [Ecp10]. Auto complete functionality allows a developer to create an instance of a class and then the IDE checks what methods are available in that class, it then gives a list of the potential methods that can be called on that class. In providing this list of the potential methods it avoids the programmer from trying to reference a method that does not exist.

The error types recorded by TEDS during Case Study Two highlights the progress made by students over a number of *practicals*. As the course develops the compiler errors evolve as well. For example, as was mentioned in section 7.2.1, Group A Session (i) was a practical early in the academic year where some students were unfamiliar with programming. This could indicate why "Missing return statement" was the highest occurring error type in that session. In comparison to Group A, Session (i) was approximately four weeks later than Group A Session (ii) and Group B Session (i). At this point in the academic year students have gained some experience in programming and so were more competent. The lack of experience of the students in Group A Session (i) could be a potential reason for why these students were creating a high proportion of missing return statements. The high frequency of missing return statements could be due to the students not being used to ensuring in Java that each method that is not a void must return something. The results seem to suggest that the students are getting more confident in avoiding this

error in the later sessions. Although they still do make this mistake it is no longer the highest occurring error.

The data recorded by TEDS highlights that the students within a group do create clusters of errors. Knowledge of these clusters is useful for lecturers as they can reveal miscomprehension in certain cohorts. Teachers can use the compiler error data in *practicals* to help them to support the students.

7.2.2.4. Some students did not run or compile code

As noted earlier, for TEDS to be successful it needs the students to use it and 20 out of 39 students (51.28%) used the tool in this case study. One student had TEDS installed and did not perform any compiles or method invocations. The students that did not adopt TEDS were thereby excluded from the case study. Their failure to adopt TEDS could have been for four reasons:

- Not attending the practical.
- Turning up late to the practical and missing the briefing.
- Running multiple instances of BlueJ, and using primarily the one without TEDS's student client.
- Choosing not to use the tool. Formally no feedback was collected from students who fell into this grouping as to why they did not install the tool. One reason why they chose not to use TEDS could be that they were concerned of someone being able to see their work, or that their productivity in the lesson has been low.

Feedback from the teachers commented that some students do not actually program within the *practicals* but rather read the textbook and ask questions of the teachers. Which may explain the number of students who ran TEDS but did not generate many compiles or method invocations.

7.3. Case Study Three

Case Study Three took place at the end of the first term (December 2008) and ran through to the beginning of the second term (January 2009). In Case Study Three the students once again had the student client of TEDS but now the teachers now had tablet PC's with the teacher client of TEDS.

The context of the cohort at the time of Case Study Three is that they are approaching the end of using BlueJ and moving to more professional IDE's, for example Eclipse. Despite this, the majority of the students were still using BlueJ at the time of the case study and 18 students used TEDS during the two *practicals*.

The main aim of Case Study Three was to view the how teachers use the data collected by TEDS to assist students. With the overall aim in mind, a number of different sources of data were collected:

- Observations for comparison of those taken of the current practical setting presented in Chapter 3
- Questionnaires and semi structured interviews to receive students and teachers opinions of using TEDS
- Student and teachers interactions with the system through the practical. These interactions are collected by the TEDS system.

These three sources of data are presented in the section 7.3.1 and evaluated in section 7.3.2 to view the potential benefits of using TEDS within *practicals*.

7.3.1. Results

As in Case Study Two this section is split into sessions, but Case Study Three has only two sessions one from Group A and one from Group B. The groups in Case Study Three are the same as in Case Study Two.

7.3.1.1. Results from Group A Session (iii)

During Group A Session (iii) observations were taken to see if there were any differences in the way the students and the teachers interacted, when TEDS was used in the session. Some differences were found and these are described in section 7.3.2.

To add a further dimension to the data presented in this section some of the students did not choose to use TEDS in this group. Although these two groups were not intentional it does allow comparisons to be made between the teacher's interactions with students using TEDS and the students not using TEDS.

Figure 7.1, Figure 7.2 and Figure 7.3 present the interaction data collected from Group A Session (iii), including students using TEDS and students not using TEDS.

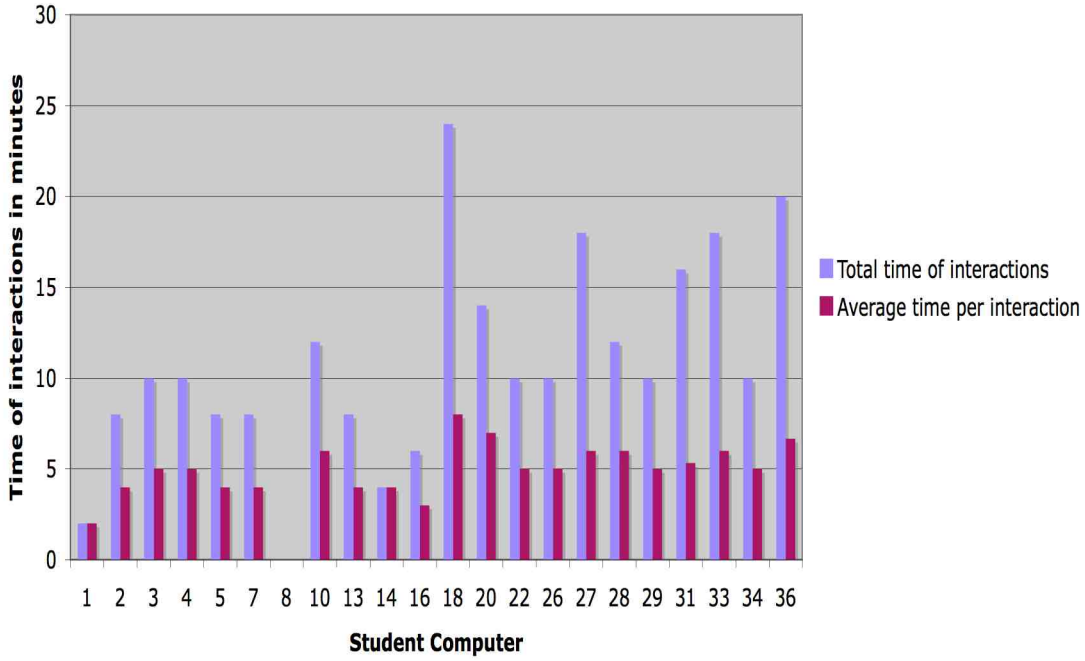


Figure 7.1: Comparison between students' total time of interactions with teachers and average time per interaction in a practical

Figure 7.1 displays the data regarding the student's interactions with the teachers. The Figure includes the length in minutes of the student's total interactions with the students in the practical and the average time per interaction for each student with the teachers.

Despite there being fewer students in Group A Session (iii) there was still a rise in the total amount in minutes of interactions between the teachers and the students (238 minutes), in comparison to observations of Group A collected for Chapter 3 (215 minutes). The highest total time of interactions between the students and the teachers are analysed further in the next section, but factors like Feature G – Image Sending could have caused longer interactions with the teachers taking time to create diagrams for the students. Another potential reason for the longer interactions in Group A Session (iii) is that with a smaller group the teachers may have felt that they could spend more time with each student.

Figure 7.2 and Figure 7.3 present how the interactions between the students and teachers began during Group A Session (iii). An interaction can either begin through the teacher or the student. Figure 7.2 displays a breakdown of how each individual students interactions were initiated and Figure 7.3 presents a pie chart of the breakdown of how the whole groups interactions were initiated.

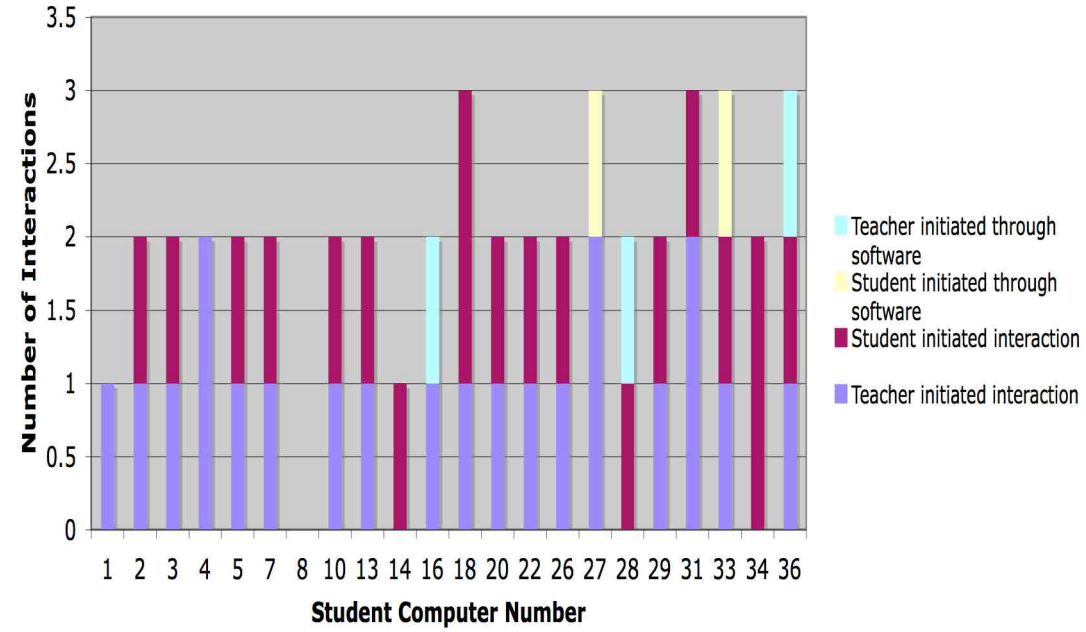


Figure 7.2: The amount of interactions by students and who initiated the interactions

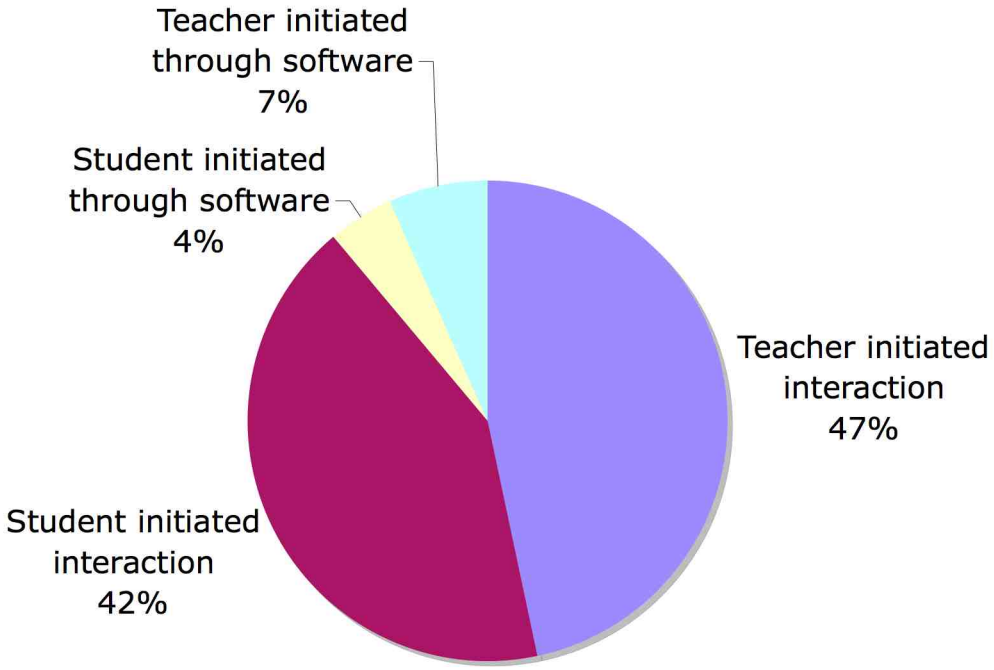


Figure 7.3: Showing the total groups breakdown of how interactions began

Figure 7.2 and Figure 7.3 reveal that the percentage of "Teacher initiated interaction" interactions rose significantly from the case studies taken of Group A for Chapter 3 from 28.57% to 46.67%. This is an increase of 18.1%.

As a result of the "Teacher initiated interaction" interactions rising, the "Student initiated interaction" fell in the TEDS practical from 71.43% (35/49) to 42.22% (19/45)

Finally, during Case Study Three observations there are two additional routes for initiating interactions through the software from the students using the help box and through the software revealing student problems. The interactions initiated through the software accounted for five of the 45 interactions in Group A Session (iii), 'Help requested through TEDS' 4.44% of the total interactions (2/45) and 'issues revealed through TEDS' 6.67% of the total interactions (3/45). With the observation method used it was difficult to define how the interactions really began, though it was possible to observe whether it was a student or teacher who began the interaction. For instance, a number of observations recorded as "Teacher initiated interaction"

could have been started from the teachers being alerted by TEDS to a student having problems through a high number of compiler errors. Unfortunately, through the vantage position of the observer, it was impossible to differentiate between these and the ones where the teacher would just like to make sure the student was working.

Figure 7.4, Figure 7.5 and Figure 7.6 present the data recorded through the observations excluding the students who were using TEDS. Figure 7.4 presents the student's total time of interactions and also their average time per interaction.

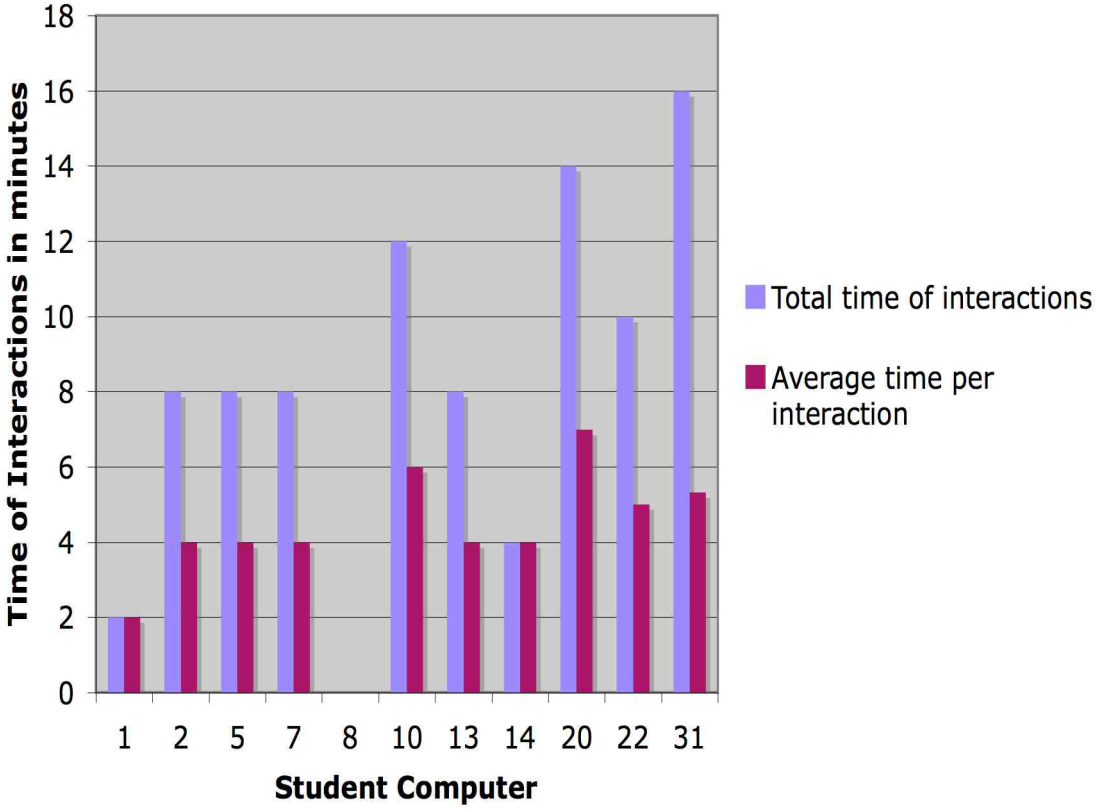


Figure 7.4: Comparison between student's total time of interactions and average time per interaction in a practical. (Students not using TEDS)

When the data presented in Figure 7.4 is compared to the data in Figure 7.7 the students not using TEDS had a lower average time per interaction to the students using TEDS. Students not using TEDS averaged 8.18 minutes per interaction (with a standard deviation of 4.85), compared to 13.25 minutes for the students using TEDS (with a standard deviation of 5.59). This disparity is discussed in more detail in the

evaluation subsection 7.3.2. One reason for the disparity is that without TEDS recording the students' current status the teachers may feel a greater need to check the status of students not using TEDS, which may result in a greater number of interactions, some of which concluded quickly if the individual was making good progress.

Figure 7.5 and Figure 7.6 present data collected during the observations on how the interactions began between the teachers and the students not using TEDS.

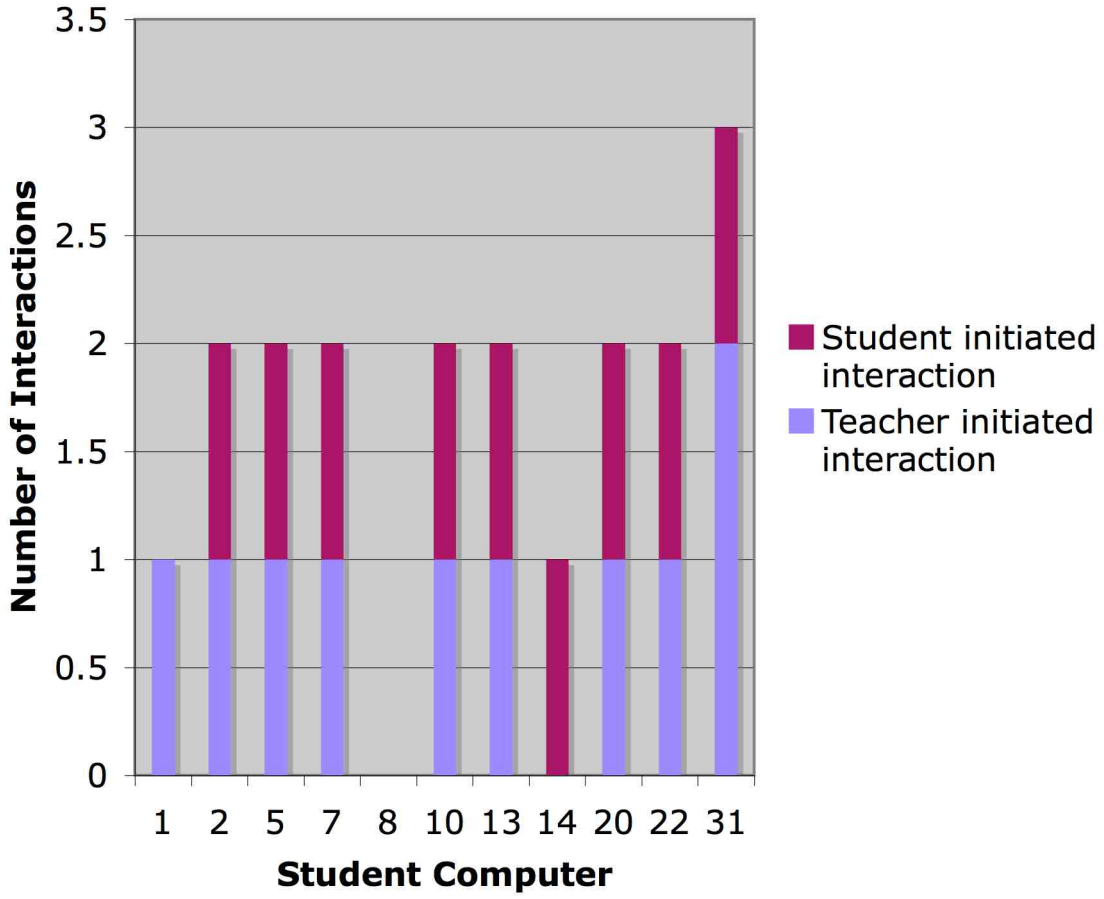


Figure 7.5: The amount of interactions by students and how these interactions began (Students not using TEDS)

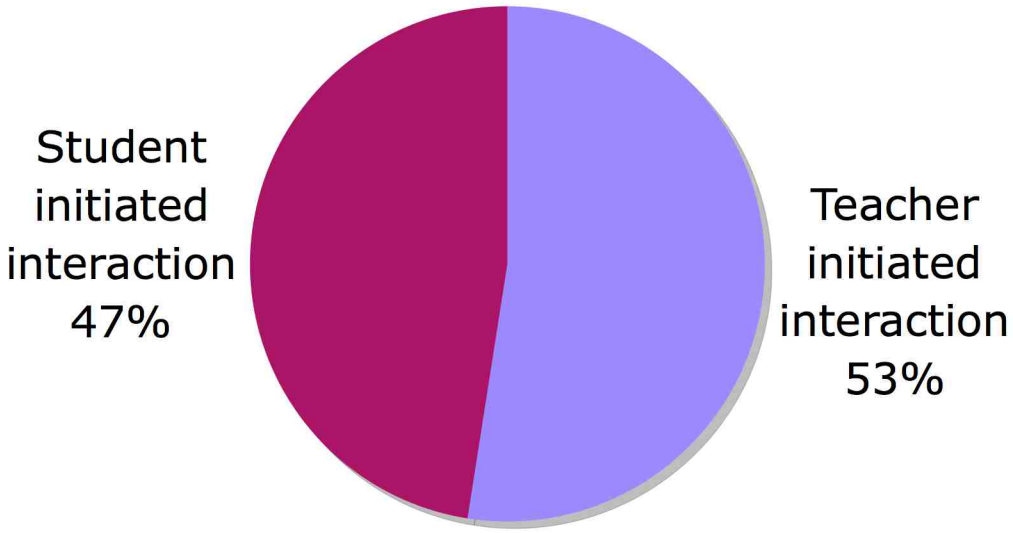


Figure 7.6: Showing the breakdown of how interactions began (Students not using TEDS)

As Figure 7.5 and Figure 7.6 highlighted "Teacher initiated interaction" interactions were slightly higher in this cohort by 6% to "Student initiated interaction". This outcome reinforces the reasoning that without TEDS supplying information to teachers on their students' current status they would need to use another approach to check how they are progressing i.e. verbally enquiring.

Figure 7.5 also shows that one student did not have an interaction with the teachers. This student was seated at computer 8. Computer 8 was noted in chapter 3 as being on a row that is difficult to reach due to the classroom layout, although other factors may also have resulted in the outcome of the student having no interactions. For instance, the teacher could know the student was a strong programmer or a teacher may have simply observed that they were making good progress.

The next three figures in this subsection are Figure 7.7, Figure 7.8 and Figure 7.9. These present the data collected from the students using TEDS in the practical. Figure 7.7 shows the students' data with regards to how long, in total, their interactions lasted and also the average time per interaction. As was mentioned

previously the students using TEDS had a higher average time per interactions than the students without TEDS.

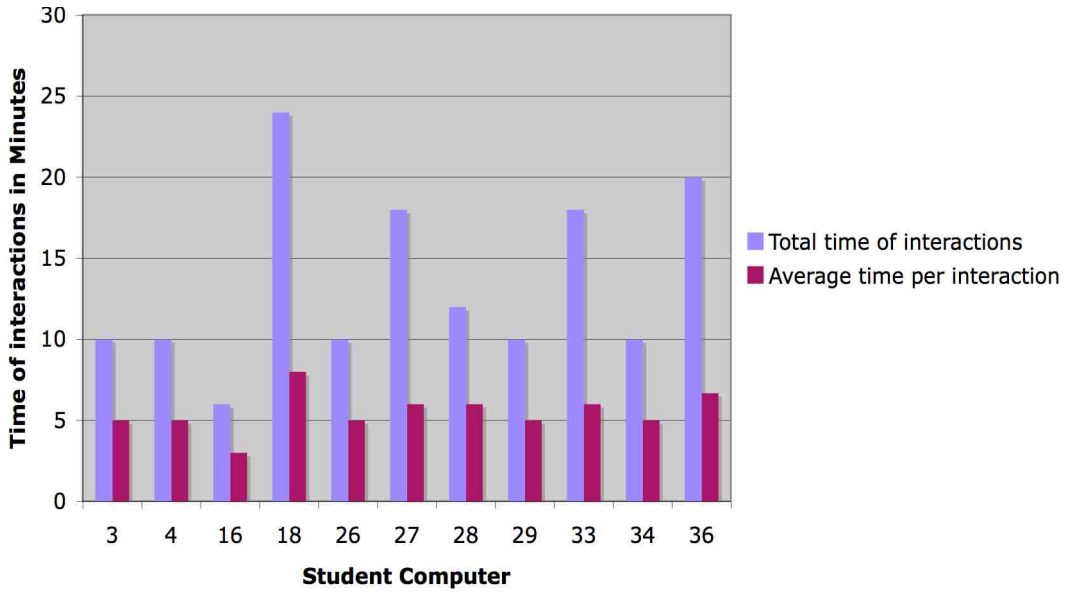


Figure 7.7: Comparison between students' total time of interactions and average time per interaction in a practical. (Students using TEDS)

Figure 7.8 and Figure 7.9 focus on the particular interactions that took place within the practical. Figure 7.8 presents how each individual student using TEDS, interactions were initiated and Figure 7.9 shows the breakdown of how the whole cohort who used TEDS interactions with the teachers were initiated.

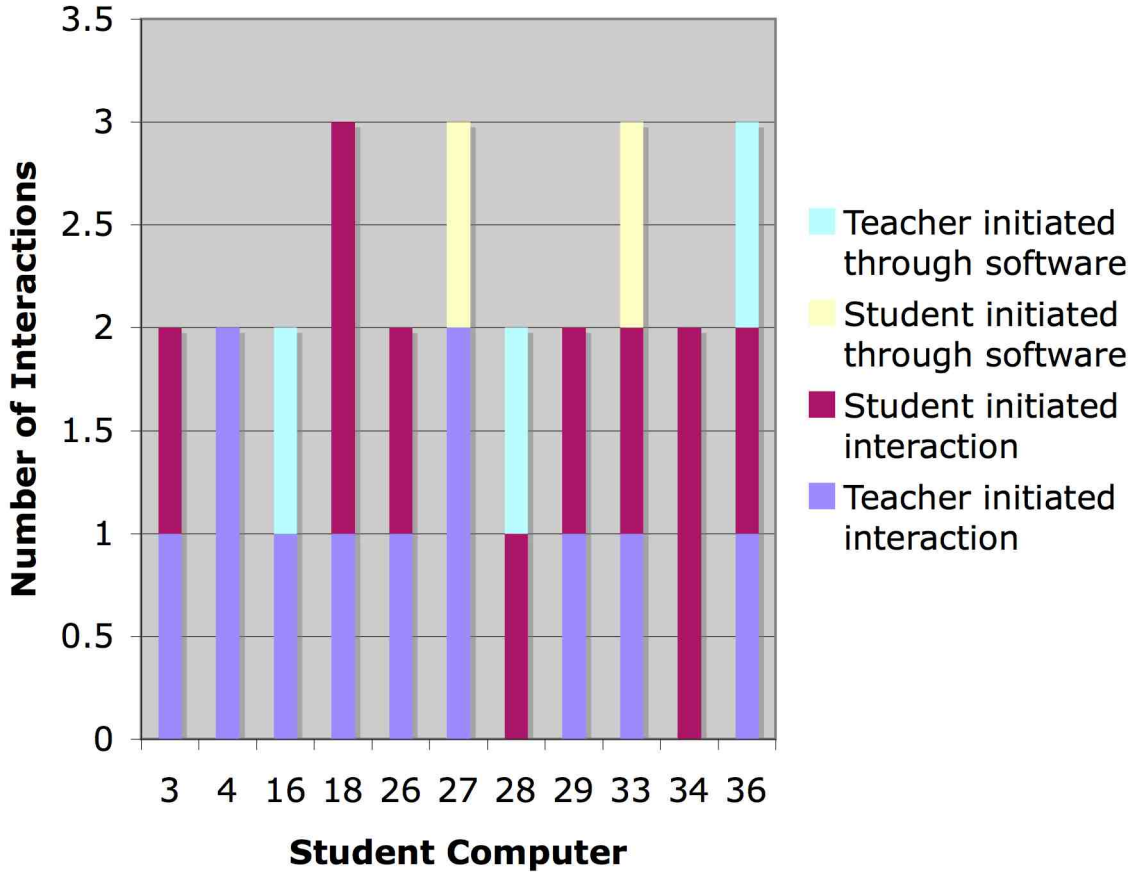


Figure 7.8: The amount of interactions by students and how these interactions began (Students using TEDS)

Figure 7.8 shows that five of the interactions began through just the use of TEDS, instances of this are students 27 and 33. These students explicitly asked for help using TEDS (as shown on Figure 7.8 with interactions labelled as "Student initiated through software") and students 16, 28, and 36 were where the teachers were altered through TEDS that they needed help (as shown on Figure 7.8 with interactions labelled as "Teacher initiated through software").

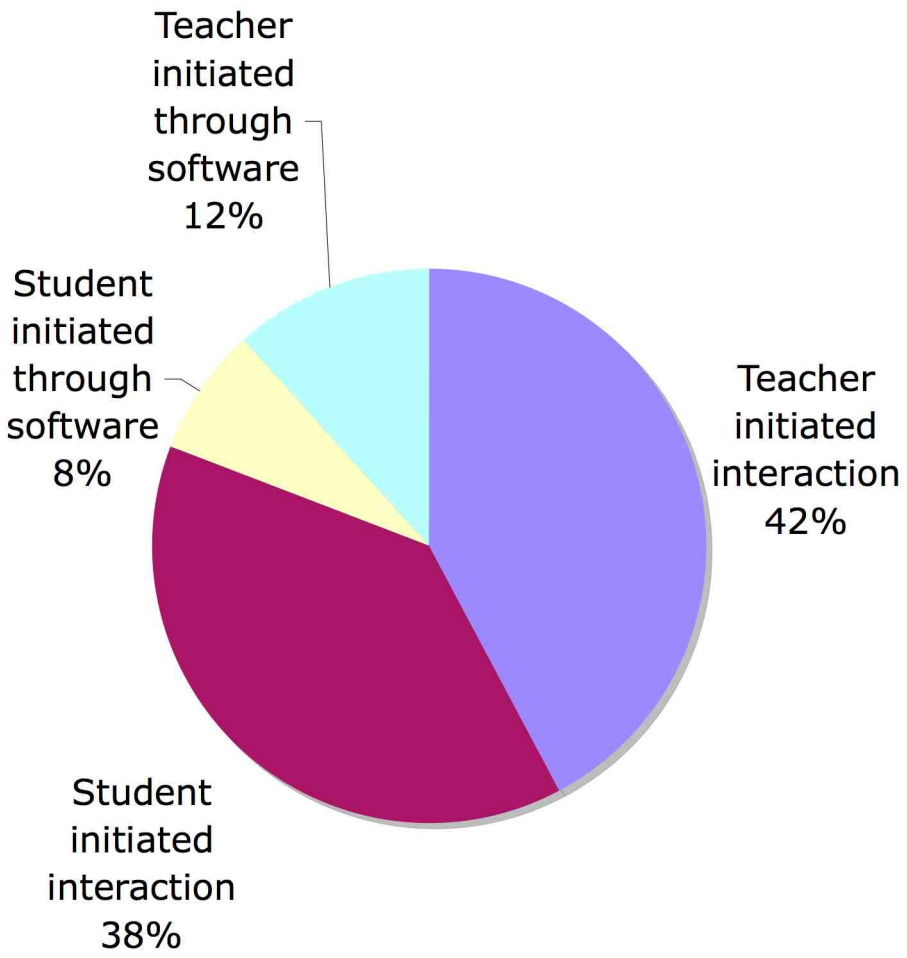


Figure 7.9: Showing the breakdown of how interactions began (Students using TEDS

Figure 7.9 shows that the percentage of "Teacher initiated interaction" was similar to that of "Student initiated interaction".

The mode by which the interactions were initiated across the whole of Group A Session (iii) is listed below. The data highlights that the results for the students using TEDS and without TEDS were similar. The reasons for this are explored in section 7.3.2:

- Teacher initiated interaction; without tool = 10, with tool = 11;
- Student initiated interaction; without tool = 9, with tool = 10.
- And five extra interactions that began through the tool.

A final observation is that the students not using TEDS accounted for only 19 interactions and the students using TEDS had in total 26 interactions.

Group A Session (iii) data collected by TEDS

This section focuses on the successful and unsuccessful compiler and method invocations collected by TEDS during Group A Session (iii). The computer number corresponds to the computer that the student was sat at.

Student ID	Computer Number	Compiler Success	Compiler Fails	Invocation Success	Invocation Fails
A1	27	15	60	17	5
A2	3	2	1	32	0
A3	28	17	30	5	7
A4	34	19	2	0	0
A5	18	10	1	11	2
A8	33	0	0	0	0
A9	4	4	0	15	1
A10	29	7	42	22	1
A11	26	3	0	7	1
A12	16	126	66	0	0
A13	36	17	34	30	0
	Total	220	236	139	17
	Mean	20	21.45	12.64	1.55
	Standard Deviation	35.80	25.92	11.67	2.34

Table 7.9: Group A Session (iii), student data

Table 7.9 presents the data collected from the 11 students who used TEDS in Group A Session (iii). Table 7.9 also presents the computer number where each

student sat which relates to the observation data presented in section In Group A Session (iii) a total of 456 compiles and 156 method invocations were made.

The breakdown of the compiles includes 220 successful ones; 48.25% of the total compiles. Student A12 provided 42.11% of the total compiles without any method invocations. The fact that student A12 did not run their code is a factor that is discussed in the evaluation and matches some of the findings from Case Study Two section 7.2.2.2.

With regard to method invocations, as with much of the data collected over the three case studies, successful ones dominated unsuccessful ones. In this practical, there were 139 successful and only 17 unsuccessful ones. No single student dominated the total amount of method invocations.

Finally in this session one student did not contribute any compiles or method invocations and four other students contributed fewer than 10 events. Events are either compiles or method invocations.

Table 7.10 shows the common error types from Group A Session (iii) and the number of times that these errors occurred.

Error Message	Number of Occurrences	Percentage of Total
Non static referenced from a static context	30	12.71
Illegal start of expression	29	12.29
Missing Bracket	27	11.44
Incompatible types	27	11.44
Unknown Method	18	7.63
Identifier Expected	15	6.36
Unknown Class	14	5.93
; Missing	14	5.93
Unknown Variable	13	5.51
Missing return statement	12	5.08
Not a statement	8	3.39
Class or Interface identifier needed	7	2.97
.class missing	5	2.12
array missing	4	1.69
Package does not exist	4	1.69
cannot assign a value to final variable	3	1.27
Missing method body, or declare abstract	3	1.27
Previously defined variable	2	0.85
Trying to return from void	1	0.43

Table 7.10: Group A Session (iii), collated groups compiler errors

Table 7.10 presents the collated compiler errors collected from the students in Group A Session (iii). In this session 236 compiler errors occurred and were collected into 19 different types of errors.

The error frequency is more spread than in Group A Session (i), with four compiler error types having between 27 and 30 occurrences and only one error type ("Trying to return from void") having just one occurrence.

The highest occurring type was 'non static referenced from a static context' (30 occurrences, 12.71% of the total). It should be noted that using static variables was a recent lecture topic before the practical. So the high volume of these errors could occur from students' relative unfamiliarity with the construct.

Group A Session (iii) is the only practical where the "Cannot find symbol" compiler error type is not found within the top two most frequently occurring errors.

7.3.1.2. Results from Group B Session (ii)

In Group B Session (ii) the teachers used TEDS to support all seven of the students that used the system. Observation data was not collected from this group, the rationale for this decision is in section 6.2.

Table 7.11 shows the compiler and method invocation success and failure counts for Group B Session (ii).

Name	Compiler Success	Compiler Fails	Invocation Success	Invocation Fails
B1	26	39	31	1
B2	19	13	5	17
B3	0	3	2	0
B4	0	0	0	0
B5	15	18	7	2
B6	45	35	2	0
B7	6	4	0	0
Total Group	111	112	47	20
Mean	15.86	16	6.71	2.86
Standard Deviation	16.14	15.66	11.01	6.28

Table 7.11: Group B Session (ii), student data

In Group B Session (ii) the seven students produced a total of 223 compiles and 67 method invocations.

In this session there was almost an equal distribution between the number of successful and unsuccessful compiles, with just one more unsuccessful compile (112) than successful compiles (111). Student B6 was the highest contributor with 80 (35.87% of total) compiles, 45 of which were successful (56.25% of their total).

Method invocations were not so evenly split in this session with 47 successful method invocations and 20 unsuccessful ones. This is a higher proportion of unsuccessful method invocation than recorded during the other case studies. The amount of these unsuccessful method invocations are unevenly distributed with student B2 making 17 of the 20. Student B1 dominated the proportion of method

invocations with 32 (47.76% of total) method invocations of which 31 (96.88% of their total) were successful.

Two students contributed fewer than 10 compiles with one not contributing any compiles or method invocations.

Table 7.12 presents the types of errors committed by the students and the frequency they occurred. ‘Cannot find symbol’ was the most frequently occurring error type, as it was for both Group A Session (ii) and Group B Session (i).

Error Message	Number of Occurrences	Percentage of Total
Cannot find symbol	25	22.32
Missing Bracket	19	16.96
; Missing	10	8.93
Incompatible types	9	8.04
Illegal start of expression	6	5.36
Missing return statement	4	3.57
non static referenced from a static context	4	3.57
Not a statement	3	2.68
Else without if	3	2.68
Unexpected type	3	2.68
Possible loss of precision	2	1.79
Package does not exist	1	0.89
Previously defined variable	1	0.89
Unreachable statement	1	0.89
Identifier Expected	1	0.89
Class or Interface identifier needed	1	0.89

Table 7.12: Group B Session (ii), collated compiler errors

Table 7.12 highlights the compiler errors the students in Group B committed in session (ii). There were 112 compiler errors in 16 different types of errors.

A total of 25 (22.32%) of the compiler errors belong to the same group ("Cannot find symbol"). The second highest occurring compiler error type was "Missing Bracket" with 19 (16.96% of total) occurrences.

The other 14 compiler error types each accounted for ten or fewer occurrences.

This data highlights some notable results collected by the system with the different types of errors that the students seem to have and also the frequencies of these errors. It presents a relationship that some students in the cohort generate similar errors, which can be used to reflect on teaching within lectures. This data is further analysed in section 7.3.2.6.

7.3.1.3. Aggregated Student Errors

Table 7.13 presents the combined errors made by the students over the course of case studies two and three, where TEDS was used.

Error Message	Number of Occurrences	Percentage of total
Cannot find symbol	138	20.12
Missing Bracket	78	11.37
Incompatible types	70	10.20
; Missing	61	8.89
Illegal start of expression	58	8.45
Missing return statement	48	7.00
non static referenced from a static context	34	4.96
Identifier Expected	28	4.08

Unknown Class	18	2.62
Not a statement	18	2.62
Unknown Method	14	2.04
Class or Interface identifier needed	14	2.04
Unknown Variable	13	1.90
Package does not exist	11	1.60
Integer number too large	11	1.60
Private Access Violation	8	1.17
Previously defined variable	8	1.17
Variable not initialized	8	1.17
Missing method body, or declare abstract	7	1.02
Possible loss of precision	7	1.02
.class missing	7	1.02
Unexpected type	7	1.02
array missing	4	0.58
Trying to override abstract methods	3	0.44
Else without if	3	0.44
cannot assign a value to final variable	3	0.44
Trying to return from void	3	0.44
Void can't be used here	2	0.29
Illegal escape character	1	0.15
Unreachable statement	1	0.15

Table 7.13: Whole set of compiler errors collected by TEDS over the three case studies

Table 7.13 highlights some notable findings from the case studies. A total of 686 errors were committed by the students and can be collected together into 30 different types of errors. The five most common errors accounted for 59.03% of the overall errors.

The highest occurring error type was “Cannot find symbol” which accounted for 20.12% (138 errors) of the overall number of errors. The second highest error types were “Missing Bracket” with 11.37% (78 errors) of the overall errors.

Finally there were eight errors with fewer than five occurrences and accounting for 20 of the total errors (3.43%).

7.3.1.4. Student Questionnaires

Students were asked to complete questionnaires during Case Study Three to discover their opinions on TEDS and also any differences that they felt they experienced in Case Study Three in comparison to a usual practical.

The results to the six questions asked to the students in the questionnaires are reported. 12 students replied out of 18 students (66.67%) who used the tool in Case Study Three.

The format of this section is that the six questions are grouped by themes based on the topic. The two groups are:

- The students’ opinions on how the teachers interacted within the class
- The students use and opinions of the help box

Was there any difference in teacher behaviour during Case Study Three?

Four questions were used to investigate the students’ perceptions of how the teachers acted within Case Study Three. The main focus of the questions were to find out perceptions regarding if the teachers interacted more with the students when they have TEDS and if the quality of their advice is higher with the additional information they get on the students’ current status.

As the teachers in a practical in Durham University are known as demonstrators the questions used the term ‘demonstrators’ to avoid confusion, but in the analysis they are referred to as teachers.

The first question asked to the students was if they: “felt more supported in the case study practical than usual”. The students were unanimous with all 12 agreeing with the statement. Therefore, it is reasonable to suggest that the students saw a positive impact in teacher support when TEDS was used.

The 100% agreement from the students in their responses to that question is reflected by the responses given to the second question: “The demonstrator spent less time with you today in the practical than usual”, where all 12 students disagreed with the statement (ten students disagreed and two students strongly disagreed). From these responses it can be concluded that the students felt that they spent equal or more time with the teachers and therefore one could reasonably assume that the students felt equally or better supported.

One observation presented in section 7.3.1.1 supported the students’ opinions on the amount of time that the teachers spent with them. The students using TEDS averaged 5.69 minutes per interaction in the Case Study Three observations, this compares with the observations collected in the preliminary case studies from Chapter 3 where the average time per interaction from the same group was 4.39 minutes. This is an increase of one minute per interaction.

The students were also asked whether they: “…felt more watched by the demonstrators in this practical than usual”. This question was asked because of the researchers concern that the students might feel more watched using TEDS, and would work differently to how they usually would in the case study practical. The students answered with eight agreeing to the statement (two strongly agreeing) and four disagreeing with the statement.

The next question the students were asked was on how well they felt supported in the case study practical. 85.71% either agreed (eight) or strongly agreed (two) with the statement “The demonstrators in this practical gave more valuable advice than usual”

with two strongly agreeing with the statement. The results would suggest that the students did feel more supported during the case study.

Students' opinions on the help box

During Case Study Three, students had the option to ask the teachers for help from their desks using TEDS. Two questions were asked in the questionnaire to find out the students' opinions on the help box.

In response to the first question: "Would you rather use the help box then putting your hand up", the results were mixed: eight students agreed that they would prefer to use it and four disagreed.

The students were also asked: "Did you use the help box?" in response to this only two students admitted to using it, which were only 16.67% of the respondents. This means that although the students like the idea of the help box, in practise they still prefer to use traditional means of asking for help.

7.3.2. Evaluation

Case Study Three consisted of TEDS being used by the teachers in live *practicals*, monitoring students who were using TEDS's student client. Seven main findings were discovered during Case Study Three.

These findings are:

i. TEDS makes it easier for teachers to view student status

ii. TEDS reveals to teachers the student characteristic which may be classified as, 'stoppers', 'movers' or 'extreme movers'

iii. TEDS enabled teachers and students to communicate more effectively

iv. The diagramming tool makes it easier for the teacher to communicate with some students

v. Teachers did not choose to make use all of the functions of TEDS

vi. The results of student's compiler errors highlighted that groups of students had similar errors.

vii. Questionnaires and interviews revealed that users accepted TEDS

Section 7.3.2 now looks into these findings in more detail referring to the quantitative results presented in section 7.3.1 in combination with the qualitative results that the researcher collected during this round of case studies.

7.3.2.1. TEDS shows teachers students' status

The first finding is that TEDS seems to have been successful in highlighting the students' status to the teachers.

The majority of TEDS's functions allow teachers to view how a student is progressing with their work and also to see what they were doing at certain points throughout the practical.

TEDS features in particular Feature C – Code Snapshot (in Case Study One), Feature A – Compiler Errors and Feature B – Method Invocation are all aimed at giving teachers a view of students' status. The intentions of these features are to reveal to teachers if a student is active without the need for them to oversee each student's screen watching them work.

Figure 7.10 presents student A's compile status during Group A Session (iii). Student A was chosen as an example as the student used a number of the features during the session, that they had available with TEDS. Also the teachers informed the researcher that the student's compiler status had prompted them to interact with the student.

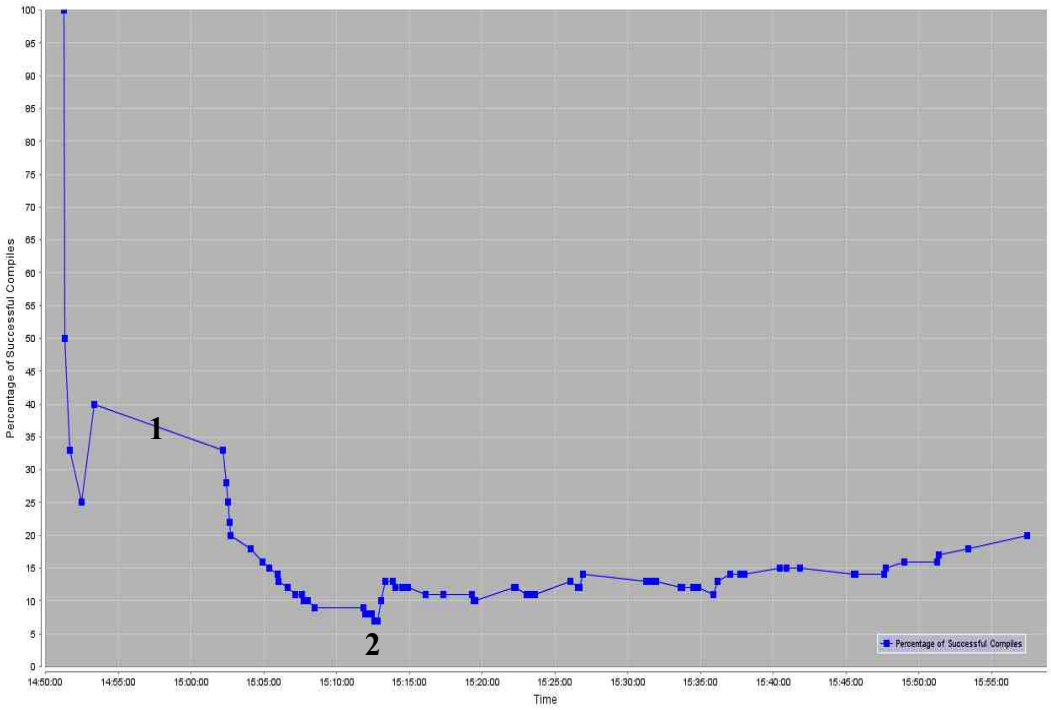


Figure 7.10: Student A's compile success rate

Figure 7.10 is graphical representation of student A's compile success rate over the course of Group A Session (iii). The X-axis is the time during the practical, and the Y-axis is the percentage of compiler success rate. The blue line is student A's percentage of successful compiles after each of their compiles during the session, and the blocks represent each a compile. A descending line in Figure 7.10 reflects a subsequent unsuccessful compile, while an ascending line represents a subsequent successful compile. The bold numbers in Figure 7.10 highlight notable occurrences. At point **1** there is a 10-minute gap where student A is not doing any compiles or method invocations after a successful compile, which could indicate that the student is trying to overcome a problem with their work. Point **2** shows where student A, used the help box to ask for help after a period of unsuccessful compiles, which was noted in the observations and through server logs. The researcher knew that student A asked for help through their version of the teacher client, which the researcher ran during the session. A teacher responded to the request and the student then began to compile successfully again.

Figure 7.11 compares student A's compiler success rates within the practical, with the student's method invocation success rates.

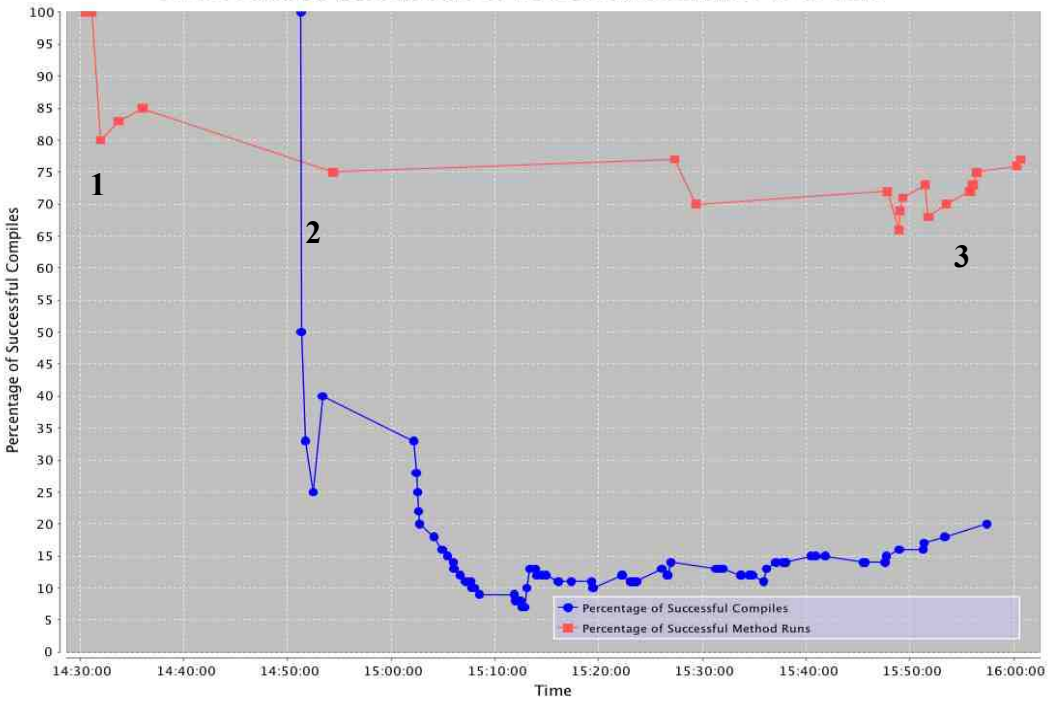


Figure 7.11: Student A's compile success rate and method invocations

Figure 7.11 presents a red line that charts the percentage of successful method invocations. The red line reveals that the student came with code that would compile but had some runtime errors (Point **1**). The compiler status updates when combined with the method invocation status updates, get more contextual meaning with the method invocation status included as well, for instance, the gap marked as **1** on Figure 7.10 is now shown as a response to point **2** on Figure 7.11. Point **1** highlights that the students code is not running, so the gap at point **2** could be viewed as an attempt to solve this runtime error. The succession of compiler errors could show unsuccessful attempts at solving the error. The help provided by the teachers enable to student to progress beyond their compiler problems and then led onto point **3** where the student code seemed to be working more successfully.

The statistics did not reveal any significant improvement in compiler and method invocation success rates between Case Study Three (where the teachers used TEDS)

and Case Study Two (where the teachers did not use TEDS). Independent T-Tests were performed on the compiler and method invocation results collected from Case Study Two and Case Study Three.

	N	Mean	Sd	T	P - Value
Successful Compiles (Teachers Not Using TEDS)	26	15.50	23.29	-0.35	Not Significant
Success Compiles (Teachers Using TEDS)	18	18.39	29.16		
Unsuccessful Compiles (Teachers Not Using TEDS)	26	12.04	13.05	-1.26	Not Significant
Unsuccessful Complies (Teachers Using TEDS)	18	19.33	22.12		
Successful Method Invocations (Teachers Not Using TEDS)	26	22.19	26.44	2.03	0.049
Successful Method Invocations (Teachers Using TEDS)	18	10.33	11.47		
Unsuccessful Method Invocations (Teachers Not Using TEDS)	26	0.42	1.03	-1.62	Not Significant
Unsuccessful Method Invocations (Teachers Using TEDS)	18	2.06	4.19		

Table 7.14: T-Tests on data collected by TEDS during Case studies two and three

As Table 7.14 presents only one of the T-Tests showed any significant difference between the data collected for Case Study Two (Teachers Not Using TEDS) and

Case Study Three (Teachers Using TEDS). The significant result was found through the comparison of successful method invocations. The T-Test found that there were significantly more successful method invocations during Case Study Two. The result probably reflects that the work has increased in difficulty between the two case studies and therefore the students ran their code successfully on fewer occasions. The fact that there is no significance between the other three data sets reveals that the students had improved at a similar rate to the increasing difficulty of the work. These statistics do not show that TEDS assisted in the students' ability to cope with the work.

The remainder of this subsection looks at other data sources that can assist in getting closer to a conclusion.

The merits of TEDS Feature B – Method Invocations was initially seen as being limited, due to the inability to view the types of runtime errors that are produced. Yet, the indication that an unsuccessful method invocation is recorded can highlight to the teachers that a student is encountering difficulties. Figure 7.10 and Figure 7.11 presented a situation where combined compiler and method invocation data can create a snapshot of a student's status at a certain time in the practical.

Figure 7.12 presents how the recording of both a student's compiler and method invocation data also can spotlight a student's bad programming practice.

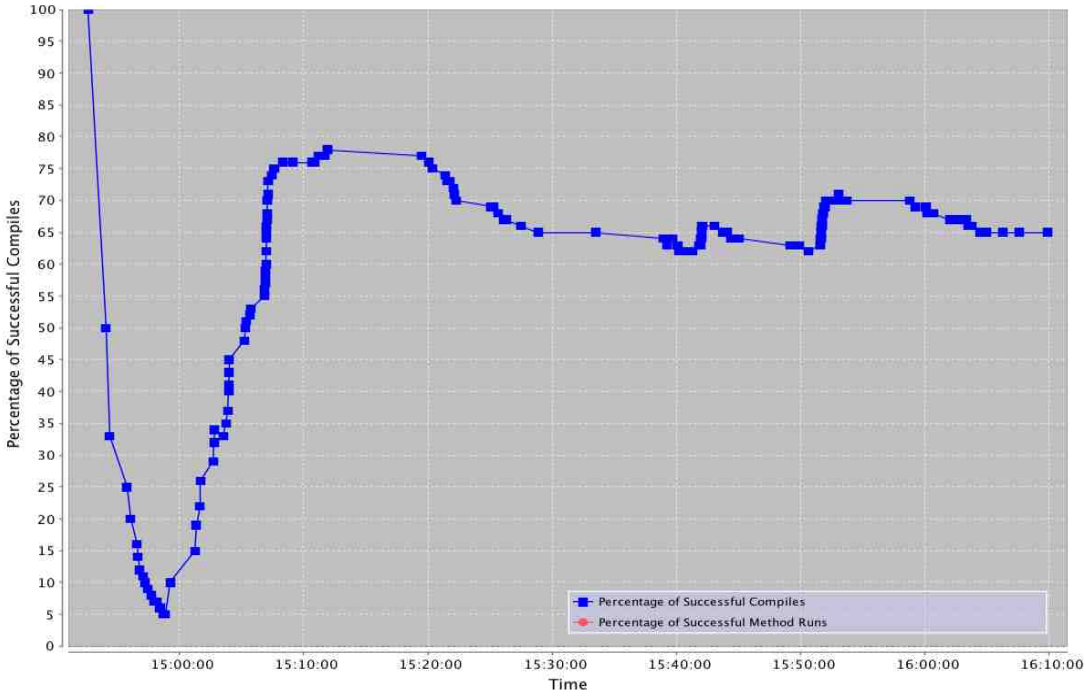


Figure 7.12: Student B's compile success rate

Figure 7.12 shows a student who, despite compiling their code a large number of times, did not run their code at any point during the laboratory class.

Figure 7.11 highlighted that by viewing the data collected by TEDS that some of the students do understand that they need to be able to have code that works, but TEDS also seems to provide some evidence of students not understanding the value of running their code to examine it for runtime errors. For instance, Figure 7.12 presents a student that either does not know about testing their code or is creating methods, which are not complete enough to test within a two hour practical. Student A (Figure 7.11) finds runtime errors and then work towards solving them. Student B (Figure 7.12) is using a development strategy where they are just fixing compilation errors whilst generating their code. An issue with this kind of strategy is that the code may compile successfully yet the code may not run, or not run as expected. For example, consider the code snippet below:

```
public void runTimeError()
    {
        ArrayList<String> test = new ArrayList<String>();
        test.get(-1);
    }
```

The code compiles successfully as it is syntactically correct yet when run the code creates the error: “java.lang.ArrayIndexOutOfBoundsException: -1”. This highlights the kind of problem student B may face when they do run their code. If a student contrives to make additional changes then the complexity of identifying the source of the error increases, as the change set is larger.

The teachers can use TEDS to alert them to instances such as the student B where they can use their experience to improve the students’ software development techniques. Although teachers would most likely have advised students of such strategies without using TEDS but TEDS provides immediate evidence to the teachers, so that teachers can identify and help these students earlier and well before bad practice becomes too ingrained in a student’s usual programming practise.

The observations, presented in section 7.3.1.1, suggest that the teachers by having access to more information increases the average time the teachers interact with the students. Group A with the whole group not using TEDS averaged at 4.39 minutes per interaction (section 3.3.1). In the Case Study Three observations, the students who did not use TEDS in the practical had a similar average of 4.74 minutes per interaction. These are both lower than in comparison to the average time of the students using TEDS who averaged at 5.69 minutes per interaction. T-Tests performed on this data did not reveal a significant difference between the observations taken during the preliminary case studies and the ones taken for Case Study Three. This highlights that despite the identified higher average time per interaction across the two data sets there was not a statistically significant difference.

As well as the observations described above, observations were taken in the class between students who used TEDS during Case Study Two and Case Study Three. T-Tests were then performed to see if there was any significance between the two groups:

	N	Mean	Sd	T	P - Value
Total time of interactions per student (Students With TEDS)	11	13.45	5.59	-2.36	0.029
Total time interactions per student (Students Without TEDS)	11	8.18	4.85		
Total frequency of interactions per student (Students With TEDS)	11	2.36	0.50	-2.26	0.037
Total frequency of interactions per student (Students Without TEDS)	11	1.73	0.79		
Frequency of teacher initiated interactions (Students With TEDS)	11	1.27	0.65	-1.43	Not Significant
Frequency of teacher initiated interactions (Students Without TEDS)	11	0.91	0.54		
Frequency of student initiated interactions (Students With TEDS)	11	1.09	0.70	0.28	Not Significant
Frequency of student initiated interactions (Students Without TEDS)	11	0.82	0.40		

Table 7.15: T-Tests on observation data taken during Case Study Three

As Table 7.15 shows there was a significance difference between the frequency of interactions and the amount of time the students with and without TEDS spent interacting with the teachers. Four potential reasons for these results are listed below:

- The students using TEDS could have been the weaker students in the class
- Having a record of what compiler errors the students make could enable the teachers to provide more in depth support and advice to the students
- Teachers who begun the interactions with the students using TEDS could have been prompted to inquire through viewing concerning behaviour in the students status updates i.e. high percentage of unsuccessful compiles. Whereas students without TEDS may be asked about their current status, which could be going quite well or well in the student's opinion, so this would result in more but quite short interactions.
- The actual process of drawing images to send to the students could take more time than a traditional interaction

These points present an outcome that was not anticipated when using TEDS. It was predicted that an interaction would be quicker, but as described previously this was not the case. The initial prediction was based on TEDS reducing the amount of time a teacher needs to understand the student's problem by alleviating the need for a student to describe it in detail. As TEDS reporting systems give teachers more information on the student's current status before they interact with them, just the solution or nudging in the right direction would be necessary in the interaction, therefore the lower interaction time.

Despite the amount of time and frequency of total interactions being significantly different, there was no significance between interactions initiated by the teachers and the ones by the students. These T-Test results reveal that that TEDS, in statistical terms, was not a significant factor in impacting on increases or decreases in students or teachers initiating interactions.

The teachers in their questionnaires noted that they liked the fact that they could see the student's current status via the frequency of their compiles and method invocations. There was a hundred percent agreement to the statement: "The technology made it easier for me to see which students were struggling". This highlights that the teachers did agree that the data on compile frequency and method invocations acted as a suitable indicator of a student's status at a snapshot of time during the practical. A second question was asked to garner if the teachers with the extra data provided by the systems went on to help the students more. The results were more mixed as one disagreed with this statement. The particular teacher's view on another question could explain why they felt that they helped less in the practical than they usually do: "I could tell which students weren't working and not those using the new version of BlueJ". This highlights how with TEDS the teacher could view if a student was progressing, so they did not have to request updates from students. This finding presents another benefit of TEDS in that the students who are working well can be left to their own devices (or be given positive reinforcement), leaving the teachers to focus on the students with problems or who are not engaging with their work.

Some of the qualitative responses that the teachers made in response to the question: "Did the technology reveal anything surprising?" highlighted some more positive comments towards TEDS and especially what TEDS reveals about student behaviour. Two of the teachers commented surprise at the amount of compiles that the students did. One said: "A lot more students than I thought seem to use the compiler button as a check for errors in their code" and another commented that: "When students got frustrated or bored they clicked the button rapidly." Both of these comments highlight that TEDS records useful events that enable teachers to effectively judge where students need support.

7.3.2.2. TEDS reveals 'movers, 'stoppers and 'extreme movers'

'Stoppers', 'Movers' and 'extreme movers' are three types of novice programmer behaviour exhibited when they are faced with problems. These are behaviours

identified by Perkins [Per85] and discussed in detail in section 3.1.1. To summarise each behaviour;

- 'Movers' when they are faced with a problem carry on looking for a solution until they finally discover the best route to take,
- 'Stoppers' when faced with a problem stops and cannot think of a solution or even really try to find one,
- 'Extreme Movers' take moving too far and change too much between compiles and without any real thought between changes where on reflection would have told them that the change would never had been successful.

TEDS reporting systems revealed students who exhibited these kinds of behaviour through the compiler and method invocation data. Especially the record of time between events is an important indicator of "stoppers" and "extreme movers".

The next three subsections, present examples and the ways that TEDS revealed each of Perkins three types of novice programmers that occurred.

"Movers"

"Movers" are the students who are able to explore their own routes for overcoming programming difficulties. TEDS reveals these students by the way that they are solving compiler errors and having successful method calls regularly.

Figure 7.11 presented an example of how TEDS can reveal students who could be considered a "mover". The compiles and method invocation success rates reveal that the student did create compiler and method invocation errors but they overcame these issues and could continue to progress with their work.

TEDS revealing these "movers" allows the teachers to be satisfied with a student's current status and more open to concentrate on the "stoppers" and "extreme movers". Naturally the students who through TEDS could appear to be "movers" could be extreme movers or stuck with issues, which they might feel they can solve themselves. This means that a teacher would still need to check on the student's status during the class, but TEDS does allow teachers to prioritise how they support

students. They can prioritise based on a range of factors: frequency of compilers errors, frequency of unsuccessful method invocation and extended period time since last event.

It was noted in section 7.1.2 that the student data collected by TEDS can be misleading such as the successful method invocations, so this further reinforces that the teachers cannot leave a student completely without interacting with them.

"Stoppers"

A "stopper" is a student who has a problem and does not have the ability or the will to progress. Students could present "stopper" behaviour by not compiling or running their code or by having large gaps of time between trying to fix errors. TEDS can highlight these behaviours in two ways to the teachers:

- Through the amounts of compiles and method invocations
- Through the data on the time since last event

During the Case Study two and Case Study three the students exhibited both of these "stopper" behaviours. In Case Study Three, seven students were shown to have made fewer than ten compiles over the two hour *practicals*. A lack of compiles does not necessarily mean that the students are in "stopper" situations as they could be making progress without actually compiling their code, but this would also mean that they are not running their code as BlueJ forces students to compile their code before they run it. However, it could mean that they are making progress without compiling which may lead to the other extreme of an "extreme mover". Both of these examples are not good programming practise for students at this level [Per85] still trying to master the semantics and syntax of the language, as the compiler can work as a form of spell checker for the student seeing if the student has made any errors such as missing a semi-colon. So even though they may not be "stoppers" they would still be worth getting attention from the teacher if they were not compiling.

The second way a teacher using TEDS could see that a student is in "stopper" situations is by a student committing an unsuccessful compile or method invocation,

which is then followed by a prolonged period of time before they compile again. This was seen a number of times during the third set of case studies. Figure 7.13 presents a student's status throughout a practical where this was seen.

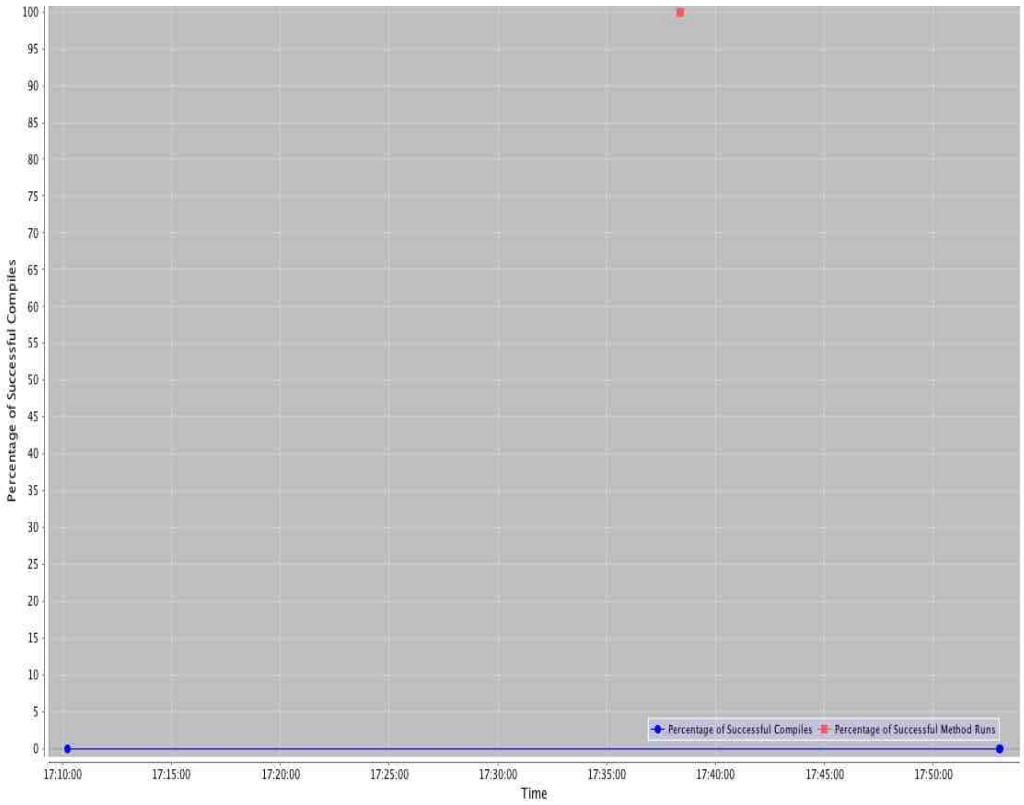


Figure 7.13: Example of a stopper

As Figure 7.13 presents the student only did two compiles for the whole of the practical, though they did make one successful method invocation. The student could be identified as a "stopper" by not finding a solution to any compiler errors they have and also by the amount of time taken between changes. The data for Figure 7.13 was taken from a practical in Case Study Two which explains why a teacher did not see it to be able to respond to the student's behaviour.

In later *practicals* conducted when the teachers did have access to the student's status reports, all of the teachers felt that TEDS gave them a good overview of which students are having problems. Furthermore, one teacher commented that: "I could tell which students weren't working." This is a rather negative interpretation from

the teacher in assume that the student is not working rather than that student being a "Stopper" and struggling to get over a problem. Either way the teacher can see this inactivity and can choose to interact with the student, and the ability to present this to the teachers is one of the aims of TEDS.

It could be the case that the "stopper" had never started and further features could have been added to TEDS to view activity. One way that could potentially have revealed student activity could have been a key logger to generate an event every time a key is pressed. This key logger could send data to a teacher when a student is neither compiling nor running their code, but still working.

"Extreme Movers"

"Extreme Movers", are students who try lots of different ways of fixing an issue without considering the merits of the solutions, when faced with a problem. During his study Perkins [Per85] identified "Extreme Stoppers" and "Extreme Movers" as the two largest groups of students. These two behaviours were present in the TEDS three case studies, with some students even showing both behaviours, with extended periods of inactivity followed by periods of high activity. Figure 7.14 presents one such student.

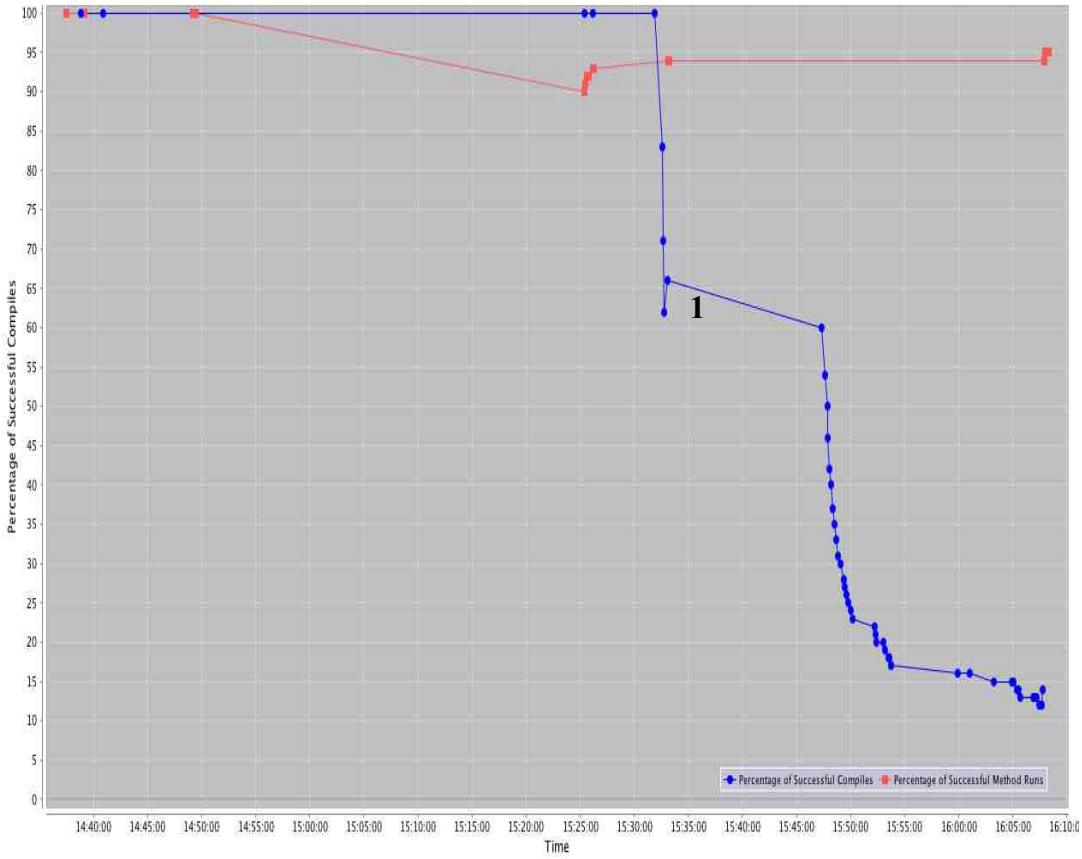


Figure 7.14: Example of "Extreme Mover"

Figure 7.14 shows a student with a period of inactivity of 20 minutes at point **1** on the figure, which could be seen as “stopper” behaviour where the teacher could want to see how a student is progressing. After the 20 minutes they then began programming and completed a high frequency of compiler errors over a 10-minute period. This would suggest that they were not reflecting on their approach to fixing their code before attempting it resulting in more compiler errors.

The data collected by TEDS highlights students who may be exhibiting “Stopper” or “Extreme Mover” behaviour, and in a way that the teachers can use whilst in a live practical. In his case studies Perkins [Per85] required students to be observed personally by a researcher to see if they exhibited any of the concerning novice programmer behaviours, whereas with TEDS teachers can get an overview of a number of student’s statuses at any point during the lesson. A lesson where each student has their own personal teacher would be beneficial for students as was noted

in the evaluation of Case Study One, but this is impossible within most universities budgets. TEDS also allows students to feel less watched so potentially to act more naturally with TEDS only listening to the events rather than a teacher watching everything they do.

7.3.2.3. TEDS enabled better teacher to student communication

Communication problems exist within the existing practical setting as section 3.3.2 highlighted, creating a number of different issues. These issues range from the teachers and the students having problems communicating with each other, to the room layout being ill suited as a learning environment.

The first feature that is designed to address this issue is Feature E – Short Message Functionality, which allows students to request help via TEDS from the teachers. The students did not make much use of this functionality within the case study *practicals*. Within Case Study Three the functionality was used 10 times. Several of the students who used the system seemed to come to the laboratory *practicals* with issues. Examples messages that the students submitted are:

- "Arrays hate me!"
- "Help me!"

Both of these examples are useful for the teacher. Especially the first comment, as the teacher at least knows that the student has an issue with Array's before they go over to assist.

Two students used Feature D – Help Button to get help. One used the feature to its potential as when they faced problems during the software development cycle they asked for help after two periods of difficulties. This example, was described in detail in section 7.3.2.1 and the student's data shown in Figure 7.11. After the student used the feature they then used the feature a second time and both times resulted in successful compiles and also method runs. This particular student in the questionnaires responded that they felt their use of the help box was: "good". Although this does not say too much about how they felt the system worked for

them, it does highlight that they viewed it positively. The positive view is further supported by the students repeated use of the Feature D – Help Button.

The second set of tools that were hoped to improve communication within the laboratory *practicals* were; Feature A – Compiler Errors and Feature B – Method Invocation. It was expected that a persistent error made by a student over an extended period of time would alert a teacher to a student being in difficulties. These functions were designed to counter the behaviour exhibited by some teachers, where they react to students asking for help rather than proactively seeing if they require help. The status updates provided by TEDS were intended to alert teachers to students with problems in a similar way to them putting their hands up. Functions A and B did seem to work, especially for some of the more reactive teachers.

For instance, on one occasion it was observed that a teacher who during the observations taken for Chapter 3 was judged to be reactive became proactive, when they had access to the data provided by TEDS. The behaviour viewed by the researcher was that the teacher would wait for the student to compile to see how they were progressing at any particular time during the *practical*.

The researcher feared during the observations that the way the teacher, who was more reactive in previous observations, had become too proactive and maybe overzealous in the way that they interacted with the student. In response to this concern the students were asked in the questionnaire about their opinions over the ways that the teachers interacted with them. Two questions were asked to see how they accepted the potential intrusiveness of TEDS. The first question was: “I felt more watched by the teachers in this practical than usual”. In response to this 66.67% agreed with this statement. The second question “I felt better supported in this practical than usual” had a 100% agreement. These responses suggest that the students felt that they were more watched in the practical and that this led to them feeling more supported within the practical. This presents the positive way that the students viewed the software although further information needs to be gathered to

explain more fully their perceptions of being ‘more watched’. That the students felt better supported is good as this is one of the main aims of TEDS and the questionnaire responses from them suggest that they felt better with a teacher keeping an eye on them. For instance, by TEDS enabling teachers to see more information thereby enabling them to better support the students.

Laurillard [Lau06] is interested in communication between a learner and teacher in classroom conditions and Laurillard’s research influenced many of TEDS’s features. TEDS was in particular designed to improve steps one and three of Laurillard’s conversational framework presented in Figure 3.10. Step one is the teacher communicating a theory to a student, and step three is the student communicating their grasp of the theory back to the teacher, and this process is repeated until the student completes step 3 appropriately as judged by the teacher. The design for these case studies by not recording the communications between the students and teachers, means that it is impossible to evaluate the success definitively of TEDS during stages one and three. Qualitative questionnaire responses from the students and the teachers were favourable towards TEDS and instances where TEDS was used successfully were also presented throughout this section. Yet further investigations would have to be done on the actual conversations between the students and teachers to view TEDS potential in fully assisting in the conversational framework.

One teacher (during unstructured interviews), who was noted as being more proactive during the observations taken when not using TEDS, felt that they became more reactive when using TEDS. The teacher further noted that they began to prioritise their interactions with the students who were shown on the system as having a negative trend of events. The problem with this impact of TEDS is that the stronger students could have fewer interactions when the teachers use TEDS. As a result of the stronger students having fewer interactions they could lose any positive reinforcement feedback they may have received from the teachers. Another further negative impact with communication based purely on the data collected by TEDS, is that the students with positive status trends are not necessarily working

successfully. So when using TEDS teachers would still need to interact with the students personally to check progress.

That particular teacher's reflection on using TEDS, suggests that the system could make all teachers reactive in its method of supplying status updates to the teachers.

7.3.2.4. The diagramming tool is beneficial for supporting the students

Object orientated languages use the concept of objects as their basis. Some novice programmers have problems visualising these objects at the start. Learning tools, for example BlueJ [Kol03], try and help novice programmers with this problem. BlueJ uses UML like representations of Java classes to demonstrate how objects interact. Despite using learning tools like BlueJ some students still have a gap in their understanding.

During Case Study Three, the teachers had Feature G – Image Sending available to them, which they used to try and fill gaps in the students understanding. In one example, the researcher observed where the image sending feature was used, to assist a student who had difficulty understanding nested loops. In this example, the teacher did not draw a diagram but rather wrote down a solution for the student. The fact that the teachers were able to use tablet PC's enabled them to add text to any images they drew. By having the solution written down it allows the student to have the solution for that occasion and further on during the course.

Figure 7.15 shows another image used in a practical, to help a student with Arrays:

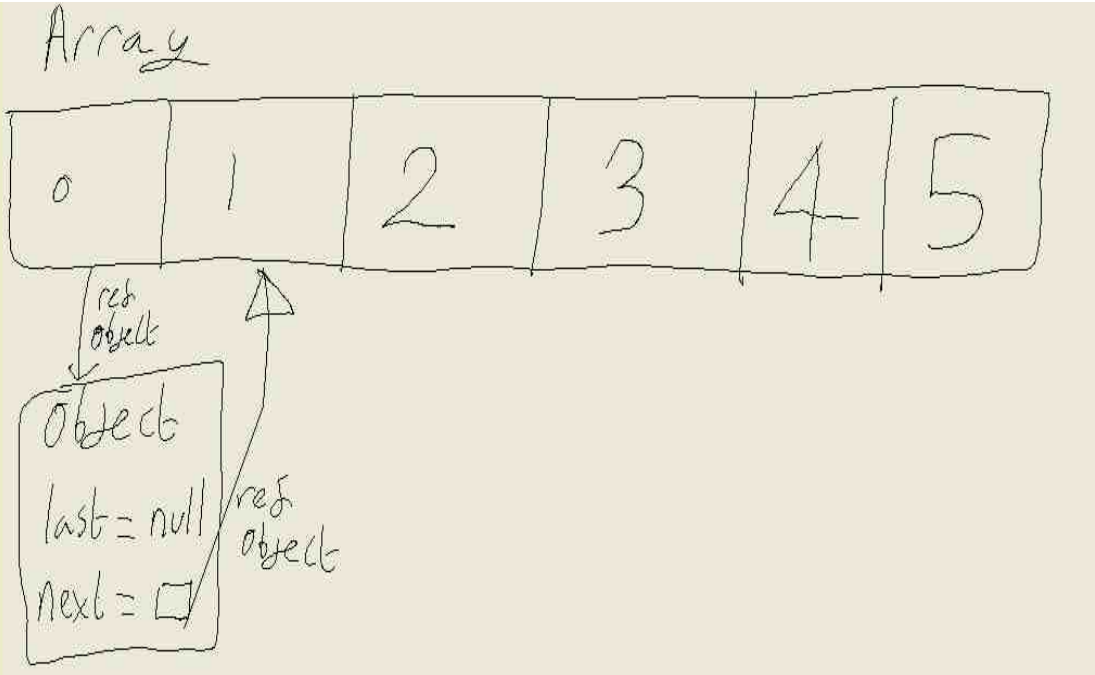


Figure 7.15: Screenshot of Image for student

Figure 7.15 shows how TEDS allows text to be included in images to further contextualise images.

The teachers who used the diagramming tool said that the students did "gratefully receive" diagrams from them. One teacher added that one student asked for the diagram to be sent to them for further reference.

The feedback from the students was a little less informative. They were asked if the teacher drew them an image and how the valued the image they were sent. Two students answered that the teacher sent images to them, but they did not further clarify their feelings positively or negatively towards it. With these responses further research would be required to see to what level they thought it was useful.

7.3.2.5. Teachers did not use all of the functions

As Chapter 4 presented there are seven different functions that combine to make TEDS. These functions were demonstrated to the teachers before the case studies and also the design considerations behind the development of the functions. The teachers were not required to use any of the functions.

It was found that the teachers did use the data and they saw the benefits of using the data, yet some of the functions were not used. These were functions that the teachers had to actually interact with the system to use such as Feature C – Code Snapshot and Feature F – Objective Setting. The functions they did use were the ones where the data was just presented without them having to interact with TEDS. These include Feature A – Compiler Errors and Feature B – Method Invocations.

Research by Robins et al on the possibility of using a tool in a practical situation may explain one reason why the additional functions were not used that much: "Any diagnostic tool to be used in actual laboratory situation will need to be rich enough to be useful, but simple enough to be manageable" [Rob03 pp 164]. So with regards to TEDS it could be that the main screen of the TEDS could be considered to match Robin's requirements with the data provided in a manageable and simple way. Conversely the additional functions seem to be too much for the teachers to carry out along with their role of supporting the students. Although with more training and more experience using TEDS the teachers could have used more of the features.

7.3.2.6. Students' compiler errors were grouped together

As was discussed in the evaluation of Case Study Two the students did commit similar compiler errors. This behaviour was repeated in Case Study Three as well where "Cannot find symbol" was the highest occurring error in Group B Session (ii) and in Group A Session (iii) it was "non static referenced from a static context".

By looking at the errors that are created, they reflect the increased complexity in the material taught in lectures. This was specifically in regards to the highest error in the second case study "non static referenced from a static context". Only four repeat occurrences of this error happened outside of that practical and they were in Group B Session (ii), which was also in Case Study Three. Using static methods and classes was covered during the lectures, which took place whilst Case Study Three was carried out.

TEDS recording that students are committing these types of errors is useful as it:

- Allows teachers to use this data straight away to help students overcome and avoid errors in the future
- Records an overall view of which compiler errors students are committing which can be used by both lecturers and teachers for helping students in this cohort and future cohorts

The data collected by TEDS from this cohort of students over the three case studies is combined in the Table 7.13 and is given certain reliability by its similarity to data collected by Jadud [Jad05]. Jadud's work was discussed in section 2.3 and studied novice compiler behaviour, which included recording compiler errors they committed, and analysing the data after the laboratory class.

7.3.2.7. Positive Responses in the questionnaires

A final outcome that can be noted in the evaluation of Case Study Three is that the questionnaire replies from the students and the teachers were both positive in regards to TEDS. Unfortunately, due to the small amount of participants in the case study who filled out questionnaires there are not enough replies to draw any statistical significance.

The questionnaires from both teachers and students are both positive towards the four particular aims of TEDS:

- The positive experiences that the students and the teachers had with using Feature G – Image Sending.
- The benefits that the teachers felt with using TEDS to view a student's current status.
- The benefits that the students and teachers saw in using TEDS to overcome communication difficulties.
- The final positive is new and still needs to be explored, that is both the students and to an extent the teachers see the benefits of using TEDS in a lecture situation.

The first three of these questionnaire responses have already been looked at in this evaluation section, which just leaves the final point.

This thesis began with identifying the potential of using TEDS within the context of a lecture to introduce active elements. In the questionnaires given to the students and the teachers it was asked what their thoughts were on introducing actual programming elements into lectures. The responses to the questionnaires from the students highlighted this where 75% agreed with the statement that: "I believe lectures on Java would be more interesting if the theory that is being taught is supported by practical programming components where you could program in the lecture". The agreement with this statement reveals that the students would accept the programming elements or at least see the benefits.

The teacher's responses also noted that lecturers could be improved with programming elements. Although they warned that it could be too time consuming getting the students and the tasks organised within the strict time constraints of the timetabled lecture.

The questionnaire responses suggest that the students would accept a form of programming in lectures but that the teachers understand that currently would be difficult to run it in the existing lecture setting.

7.4. Summary

This chapter presented and analysed the results collected from the three case studies that were carried out using TEDS. The case study was concerned with Research Phase 3 – Can technology be used to improve a teacher's ability to support students in *practicals*.

The introduction to this thesis linked the research phases to two research questions, which are explored in the rest of this summary.

7.4.1. Research Question 4 - In what areas can TEDS change the way that teachers track student status?

Research Question 4 was concerned with evaluating how successful TEDS is at assisting teachers. The teacher did have access to more data during the case studies and they to some extent used this data to view the current status of the students.

During Case Study Three, there were instances where the teacher used the data to see that a student was not making progress. The teachers then used this evidence to help the students. Especially during Case Study Three results collected from the students by TEDS potentially revealed instances of all three of Perkins [Per85] three types of novice programmer behaviours. With this categorisation reported to teacher they could then decide with how to proceed in order to support the students.

TEDS was also found to have a negative impact by the way that the teachers used the data collected by TEDS. The teachers admitted that they trusted the data too much and were therefore less inclined to interact with those students with 'positive' status updates (no compiler errors or method invocation failures). As was noted in sections 4.4.2 and 4.4.3, the compiler and method invocation data may show a positive trend (i.e. successful compiles and method invocations) when the teachers used TEDS, while the student may have difficulties with their code. This means that a teacher by trusting this data could result in them not interacting with students who may actually have difficulties.

A further impact is that formerly proactive teachers felt that through using TEDS they became more reactive.

7.4.2. Research Question 5 - In what areas can TEDS change the way that teachers and students interact?

TEDS providing the teachers with additional data on the status of the students changed the way that some interactions started. Some teachers and students did use the features and this combined with repeated use and feedback from the students,

illustrates the positive opinions on these features and TEDS potential to improve communication. In addition to this both the students and teachers in the questionnaires agreed that they perceived that TEDS improved communication.

Despite the positive questionnaire responses the features designed specifically to assist directly with communication were not widely used. Features that were not widely used but that the students and teachers liked in theory were Feature D – Help Button, Feature E – Short Message Functionality and Feature G – Image Sending.

A further area where TEDS impacted on interactions is through the teachers relying on TEDS to track progress. Feedback from the teachers noted that the teachers were more inclined to interact with students showing a negative status (i.e. compiler errors and unsuccessful method invocations). This reliance on the status updates meant that the stronger students did not receive positive reinforcement, which is a useful component of teacher to student interactions.

However, it must be further investigated to see how using TEDS in the long term could improve communication as teachers and students get more familiar with using its features.

8. Conclusions

This work has sought to evaluate whether technology can be used to improve a teacher's ability to support students during *practicals*. This chapter outlines what was discovered during each of the three research phases. The chapter also considers any limitations of the thesis and areas for further work.

8.1. Research Phases

The research phases conducted are:

Research Phase i. Analyse existing methods of teaching programming and identify issues with the current methods

Research Phase ii. Develop technologies to overcome these issues

Research Phase iii. Carry out case studies to see the potential of the technologies to overcome any issues discovered.

Five research questions were addressed during the research phases. The questions are reproduced below together the findings from the case studies.

8.1.1. Research Question 1 – What are the students' and teachers' opinions on the pedagogic value of practicals?

Research Question 1 was concerned with evaluating how the students and teachers felt about *practicals* within the context of an introductory programming module. The students appreciated the pedagogic value of lectures as a mode of knowledge transfer, but they value *practicals* more highly. The students noted, in the questionnaires, that they especially value the way that they can apply the knowledge taught from lectures, whilst being supported by teachers.

The majority of students viewed themselves as active learners. From the literature, presented in Chapter 2, active learners are usually more suited to *practicals*, this

could provide an explanation as to why the students perceive *practicals* as having a higher pedagogic value to lectures in their learning.

8.1.2. Research Question 2 – How do students begin interactions with teachers in the existing practical setting?

Students were found to begin interactions with the teachers in two ways:

1. Raising their hand and attracting the attention of the teacher
2. Getting the attention of the teacher as they patrolled the classroom

The first method is a typical way of attracting attention and is effective in the *practical* also. The second method is more problematic and usually took the form of a student attracting the attention of a teacher, when the teacher walks behind or near to them and subtly ask for help.

A minority of the teachers were reactive in the classroom and preferred to wait for students to ask for help, rather than actively inquiring on current status or patrolling the classroom. These 'reactive' teachers created some communication difficulties, as the students who *covertly asked* for help, it was observed, tended not to have interactions with these teachers.

Communication in general, was observed as being difficult in existing *practicals*. This was due to two principle reasons. The first reason was the room layout, which made it physically difficult for the teachers to get to the students to interact with them. It was observed that a minority of the teachers avoided interacting with students in difficult physical locations. The second reason occurred during the interactions between the students and teachers. The teachers reported that the students had difficulties explaining problems encountered with their work and these led to some teachers having difficulties explaining solutions to the students.

8.1.3. Research Question 3 – To what extent can teachers perceive student status in the existing practical setting?

The teachers who took part in this research valued the ability to be able to track a student's progress so that they can view the level each student is working at and divide support according to this progress. The majority of teachers said they found it easy to track the weaker and stronger students in the cohort, but the average students are more difficult to track. These average students can vary between eventually becoming a strong performer on the course, or they could fail, based on teacher feedback.

Problems were identified with communication between teachers and students. One factor involved in this problem is room layout, which made it difficult to see what a student is doing and also to get to the student to interact with them. Difficult communication has a direct impact on the ability of a teacher to track the progress of a student, as without effective communication, tracking cannot be effective.

Furthermore, some of the students remarked that they find it difficult in some case to effectively communicate their progress to teachers.

8.1.4. Research Question 4 – In what areas can TEDS change the way that teachers track student status?

TEDS was developed during Research Phase (ii). TEDS aims to improve the teachers' ability to support students in programming *practicals*. Case studies were carried out using TEDS to determine whether it enables teachers to provide better support to the students.

The case studies generated both quantitative and qualitative results. The quantitative observations alone did not provide any evidence that TEDS improved the teachers' ability to track student status. The qualitative observations revealed that the teachers, in a small number of cases, used the data collected by TEDS, to support the students. These teachers used TEDS as a form of alert system to indicate which students needed help.

The teachers commented that having the student status data provided by TEDS allowed them to prioritise the students based on their current status i.e. more help for students with more errors. But the teachers also noted that this resulted in students with a positive status trend not be interacted with. This has the twofold problem of the students who are doing well not receiving positive feedback, and also if a student has a problem they may miss having a teacher coming to interact with them.

8.1.5. Research Question 5 – In what areas can TEDS change the way that teachers and students interact?

TEDS did not change the way that the students and teachers interacted, although it did have some impact on the way that the interactions were initiated. The teachers noted that they felt having access to the data collected by TEDS, that they could prioritise student support by the students who had errors.

During the case studies, observations revealed that the students using TEDS had statistically longer interactions with the teachers than the ones not using TEDS with a difference of 13.45 minutes (Students With TEDS) to 8.18 minutes (Students without TEDS).

In regard to whether students perceived an improvement in the interactions, the qualitative feedback from the questionnaires revealed that the students did perceive improved communication in *practicals* using TEDS. They felt that TEDS enabled the teachers to support them more in depth than they usually did. The response rate from the questionnaires although was not great enough to draw any statistically accurate results from these responses.

The feedback from the students was positive in regards to TEDS. The students responded that that they did not see it as being too intrusive and they valued the support that they received from the teachers using TEDS.

8.2. Limitations

The case studies revealed that TEDS has the potential to overcome some of the issues that exist within *practicals*, more extended use of the tool would be required to see if the potential can be realised over different cohorts, over extended periods and with different configurations of teachers. It could be that the tool would be more beneficial in *practicals* with a higher proportion of reactive teachers where the reporting systems by revealing students with issues, prompt teachers to 'react' to these stimuli.

The results of this work may not be generalisable outside the context presented, because the case study was not extended outside of the single participating institution. The results collected during this research may not occur if the same experiments were run in courses related to those other than programming. Further work would need to be carried out to explore the transferability of the approach across institutions or disciplines.

8.3. Further Work

There are two areas where the work presented in this work can be furthered. The first is by extending the case study and the second is by using TEDS in lectures.

Extending the case study would help to discover, with more certainty, if TEDS has a positive impact on the teachers ability to support students in *practicals*. The main way of extending the case study would be to make it longer. The case studies could be extended to include a full year of the teachers and students using TEDS. Although this would increase the amount of data collected it would still provide difficulties in judging whether TEDS is an improvement over the current methods. Control groups would be required with students and teachers of similar abilities either using or not using TEDS.

A further way of extending the case studies would be to put more focus on the diagramming tool. During this work the potential of the feature was seen both

through the use of it and from feedback. A further case study just using the diagramming tool feature would be useful to further see if it has potential to assist teachers in supporting students.

Another way of extending the case study would be to use a different programming language. This would present another challenge as languages such as Java require to be compiled which enables some of the important functionality of TEDS in presenting to the teachers how a student is currently proceeding with the work. Other languages such as PHP are not compiled so would not allow TEDS to collect these messages. The challenge with languages like this would be to find other ways to track status/activity, such as by creating key loggers, which would record how many characters are being typed by the students. With this data it can be viewed if the students are not coding and therefore could be having difficulties or that they are coding. Also with non-compliable languages more focus could be put onto the features designed to facilitate communication, as teachers would no longer be able to use just the compiles and compiler errors to see how a student was progressing.

The second area would be to investigate if TEDS can assist students to assimilate more knowledge in lectures. TEDS's features allow students to compile or run there code and a lecturer could potentially view the success of this on a console. This would allow lecturers to teach theory and then the students can apply it. TEDS could then provide the teachers with an indication of the success of a student's application of the theory in terms of their compiler and method invocation events.

8.4. Summary

In summary this work has revealed that tools can be developed in a way that are both practical and useful for teachers to use in *practicals* element of introductory programming courses. The tools can be used to highlight which students may need help and also have the potential to improve communication in classes. The notable findings from research are that:

- There are proactive and reactive teachers.

- Students prefer to covertly ask for help.
- Teachers noted that they used compiler and method invocation status updates to prompt interactions with students with problems.
- TEDS can help to overcome the difficulties in supervising students in typical computer laboratories.

Despite these findings, some of the features of TEDS were not widely used by the teachers. These were typically the features that required the teacher to interact with the system.

The use of TEDS in its current form could also be perceived as having a negative impact in some cases on how the teachers supported the students in the *practical*. This is especially the case for more proactive teachers, who used the data to prioritise their support for the students with the most errors. This resulted in two negative impacts:

1. Some stronger students missing out on positive feedback.
2. Some of the data collected by TEDS is not always accurate, the teachers may not have interacted with a student based on a false interpretation of the data.

These negative impacts of using TEDS do show that TEDS on its own cannot replace the valuable one–to-one communication between a teacher and a student, but could be an extension to their methods of support. TEDS could especially be useful where teachers are more reactive and wait for stimuli before they interact with students .The tool, in its present form, can alert teachers to students who may need help and can provide further context on their problems. Although further work could be done on interpreting the data that is collected and how this is displayed to the teachers to make TEDS more effective.

Appendix 1 – Teacher Preliminary Case Study Questionnaire

Question Statements	Strongly Agree	Agree	Neutral	Disagree	Strongly Disagree
Practicals are a good way of teaching programming					
Students don't seem to remember much of what they are taught in lectures					
BlueJ is the most appropriate tool for teaching novice programmers Java					
Students are usually willing to ask for help if they need it					
Students are good at describing the problems that they have with their code					
I generally find it easy to help students with their problems					
You find the majority of the students are open to talking to you if you ask them questions					
I am a proactive demonstrator (ask the students questions) rather than a reactive demonstrator (wait for the students to ask me)					

If you had the option to change anything about *practicals* what would it be?

What do you see as the positives and the negatives of a practical?

Do you think in the current practical setting it is difficult to judge how well one student is doing?

"Students are just interested in the mark" What do you think?

Do you have any other general comments on *practicals*:

Appendix 2 – Student Preliminary Case study Questionnaire

Dear Student

This questionnaire is to see what your opinions are about the way IP/PDS is taught and also about what type of learner that you think you are.

This is part of a project to see if communication and feedback in IP/PDS *practicals* can be improved by the use of a technology that I have developed.

Question Statements	Strongly Agree	Agree	Disagree	Strongly Disagree
1. I think lectures are useful as a catalyst of learning				
2. I prefer practicals for learning rather than lectures				
3. I usually achieve what I am set to do in practicals				
4. I find it easy to approach the demonstrators for help in practicals				
5. I find it difficult to explain my problems to the demonstrators				
6. I have an outgoing character				
7. I am more of an active learner (learning by doing) than a passive learner (learning by listening)				

Prior to University have you had any programming experience?

How does a University practical compare to a secondary school lesson, in terms of support, independent learning, difficulty etc?

Any other comments on *practicals*:

Appendix 3 – Teacher Post Main Case Study Questionnaire

Question Statements	Strongly Agree	Agree	Disagree	Strongly Disagree
The technology improved my ability to help students				
You were a more proactive demonstrator today than a practical without the technology				
The technology was difficult to use				
The technology made it easier for me to see which students were struggling				
You helped more students in the lesson due to this software				
Students asked for help more often in this practical in comparison to a normal practical				
You see the benefit of this system and would like a system like this to be introduced to programming practicals on a regular basis				

Did the technology reveal any things that you found surprising? I.e. in regards to student compiler behaviour, particular students success/failure rates etc

Did you use the diagramming tool today in *practicals*? If yes do you think it improved your ability to help the students?

Would you like any more functions to be added to the system?

Do you think this technology could be used in lectures? For example by using small programming tasks to reinforce theory and using the reporting systems to show the lecturer if students understand what is being taught?

Any other Comments:

Appendix 4 – Student Post Main Case Study Questionnaire

Question Statements	Strongly Agree	Agree	Disagree	Strongly Disagree
The demonstrators in this practical gave more valuable advice than usual				
I felt that the reporting systems were not too obtrusive				
I am more likely to use the help message box than asking the demonstrator directly for help (for example by putting my hand up)				
I did not see any noticeable difference between the usability of the normal BlueJ and the extended BlueJ				
I believe lectures on Java would be more interesting if the theory that is being taught is supported by practical programming components(needs rephrasing)				

Did you use the help box to request help from the demonstrators in this practical? If yes how was it?

Did the demonstrator use the diagramming tool to help to explain something to you? If yes do you think that the diagram helped you to understand the problem better?

Did you feel that with the demonstrators seeing how your code was progressing in *practicals* that you were more conscious of making mistakes?

Did the demonstrators spend more or less time with you in this practical?

Did you have a feeling of being continuously supported by the demonstrators?

Did you feel continuously watched by the demonstrators?

Any other comments/ suggestions for improvements to the software:

Appendix 5 – Observation Tally Chart Sheet

	General		Types Of Interactions					
Computer	Student Name	Tally of Interactions (2 min per stick)	Inquire on progress	Help provided at request	Mark work	Help requested through software	Students help requirement revealed by software	Social
1								
2								
3								
4								
5								
6								
7								
8								
9								
10								

* Scaled down from A3 to A4

References

[Ahm05] M. Ahmadzadeh, D. Elliman, and C. Higgins. **An Analysis of Patterns of Debugging Among Novice Computer Science Students.** ITiCSE ’05, June 27-29, Monte de Caparica, Portugal. Pp 84 - 88. 2005.

[AM00] G. D. Abowd. E. D. Mynatt. **Charting Past, Present, and Future Research in Ubiquitous Computing.** ACM Transactions on Computer-Human Interaction, Vol. 7, No. 1, March 2000, Pp 29–58.

[Ban06] D. A. Banks. **Audience Response Systems in Higher Education.** Information Science Publishing, London, UK, 2006.

[Bar32] F.C. Bartlett. **Remembering: A study in experimental and social psychology.** Cambridge University Press, London, U.K., 1932.

[Bar05] L. Barkhuus. **Bring Your Own Laptop Unless You Want to Follow the Lecture: Alternative Communication in the Classroom.** GROUP’05, November 6-9, 2005, Sanibel Island,, FL, USA. Pages 140 - 143

[Bas68] M. Bassey. **Learning methods in tertiary education.** Nottingham Regional College of Technology, 1968.

[Ben01] M. Ben-Ari. **Constructivism in Computer Science Education.** Journal of Computers in Mathematics and Science Teaching, 20(1), pp 45-73. 2001.

[Big02] J. Biggs. **Teaching for Quality Learning at University.** Open University Press, Buckingham UK, 2002.

[Bli98] D.A. Bligh. **What's the use of lectures?** Intellect, Exeter UK, 1998.

[Blu10] BlueJ, www.bluej.org. WWW, last visited: March 2010

[Bon03] C. C. Bonwell. **Active Learning: Creating Excitement in the Classroom.** Active Learning Workshop, 2003.

[Bor80] A. Bork. **Interactive Learning.** In The computer in school: Tutor, tool, tutee pages 53-66, New York: Teachers College Press, 1980.

[Bor03] A. Bork. **Interactive Learning 20 years later.** Contemporary Issues in Technology and Teacher Education, 2(4) pp 608 – 614, 2003.

[Bow06] J. Bown. **Disciplinary Commons Submission.** WWW. www.disciplinarycommons.org/ last accessed March 2009.

[Boy90] E. L. Boyer. **Scholarship Reconsidered.** The Carnegie Foundation for the Advancement of Teaching, New York, 1990.

[Car01] L. Carson. **Teaching Power.** In H. Edwards, B. Smith, and G.Webb, editors, *Lecturing. Case studies, experience and practice*. Kogan Page Limited, London, U.K., 2001.

[Cla83] R. E. Clark, **Reconsidering Research on Learning from Media**. Review of Educational Research, Vol. 53, No. 4, Pp. 445-459. Winter, 1983.

[Cof04] F. Coffield. D. Moseley. E. Hall. K. Ecclestone. **Learning styles and pedagogy in post-16 learning.** Learning & Skills research centre, 2004.

[Cow81] L. Cowan. **Suggestions for a modified lecture programme Educational Methods Unit Occasional Paper**. Oxford Polytechnic, 1981.

[Cut01] Q. Cutts. **Engaging a Large First Year Class.** In M. Walker, edited, *Reconstructing Professionalism in University Teaching*, SRHE series of the Open University Press, 2001, pp105-128.

[Cut05] Q. Cutts. G. E. Kennedy. **Connecting Learning Environments Using Electronic Voting Systems.** In A. Young & D. Tolhurst, edited, Australiasian Computing Education Conference 2005, Conferences in Research and Practice in Information Technology, Vol.42, 2005. Newcastle, Australia.

[Cut07] Q. I. Cutts. S. Jamieson. **Comparing Two Free Programming Projects Used In Introductory Programming Courses.** 8th Conference of the Subject Centre for Information and Computer Sciences, Southampton, England. 2007.

[Daw03] P. Dawabi, L. Dietz, A. Fernandez, and M.Wessner. **Concert-Studeo: Using PDAs to support face-to-face learning.** In Proceedings of the International Conference on Computer Supported Collaborative Learning (CSCL'03), pp 235–237, Bergen, Norway, Jun. 2003.

[Die03] L. Dietz, P. Dawabi, A. Ferndandez, and M. Wessner. **Improving Face-to-Face Learning with ConcertStudeo.** Learning Technology, IEEE Computer Society, Vol. 5(No. 2), pp 51–57, 2003.

[Dij89] E. W. Dijkstra. **On the cruelty of really teaching computer science.** Communications of the ACM, 32, pp 1398 – 1404, 1989.

[Dra10] S. Draper, http://www.psy.gla.ac.uk/~steve/ilig/people.html. WWW, last visited: March. 2010.

[Dre86] H. Dreyfus and S. Dreyfus. **Mind Over Machine: The Power of Intuition and Expertise in the Era of the Computer.** New York: Free Press. 1986.

[Duf96] R. J. Dufresne. W. J. Gerace. W. J. Leonard. J. P. Mestre. L. Wenk. **Classtalk: A Classroom Communication System for Active**

Learning. Journal of Computing in Higher Education, 7, 3-47, (1996).

[Dun05] D. Duncan. **Clickers in the Classroom.** Addison-Wesley, US. 2005.

[Dun06] D. Duncan. **Clickers: A New Teaching Aid with Exceptional Promise.** Astronomy Education Review, Volume 5, Apr 2006 - Oct 2006, pages 70 – 88

[Dur10] Durham University, www.durham.ac.uk. WWW, last visited: September. 2010

[Ecp10] Eclipse IDE, www.eclipse.org. WWW, last visited: September. 2010

[Edw03] S. Edwards. **Improving Student Performance by Evaluating How Well Students Test Their Own Programs.** ACM Journal of Educational Resources in Computing, Vol. 3, No. 3, Article 01, September 2003.

[Eng06] J. English. **The Checkpoint Automated Assessment System.** ITiCSE ’06, June 26-28, Bologna, Italy. Pp 337. 2006.

[Eng09] J. English, & T. Rosenthal. **Evaluating Students’ Programs Using Automated Assessment – A Case Study.** ITiCSE ’09, July 6-9, Paris, France. Pp 371. 2009.

[Fin06] S. Fincher Ed. **Some Good Ideas From The Disciplinary Commons.** 7th Conference of the Subject Centre for Information and Computer Sciences. 2006.

[Fli92] U. Flick. **Triangulation Revisited: Strategy of Validation or Alternative?** Journal for the Theory of Social Behaviour, 22:2. 1992.

[Her89] N. Herrmann. **The creative brain.** Brain Books,
The Ned Hermann Group, North Carolina, 1989.

[Hme04] C. E. Hmelo – Silver. **Problem-Based Learning: What and How Do Students Learn?** Educational Psychology Review, Vol.16, No.3, September 2004.

[Gag65] R.M. Gagne. **The conditions of Learning.** Holt, Rinehart & Winston, 1965.

[Gal91] K. B. Gallagher, & J. R. Lyle. **Using Program Slicing in Software Maintenance.** 1991

[GB93] N.L. Gage and D.C. Berliner. **Educational Psychology**.
Houghton, Mifflin, Boston, Massachusetts, 5th edition, 1993.

[Gib81] G. Gibbs. **Twenty terrible reasons for lecturing**. SCED Occasional Paper No. 8, Birmingham. 1981.

[Gib92] G. Gibbs. **Improving the quality of student learning.** Technical and Educational Services Ltd, Bristol, UK. 1992.

[Int07] InterWrite Learning. www.interwritelearning.com. WWW, last visited: June. 2007.

[Jac02] N. Jackson. **QAA: CHAMPION FOR CONSTRUCTIVE ALIGNMENT!** LTSN Generic Centre, University of Surrey, November 2002.

[Jad05] M. C. Jadud. **A First Look at Novice Compilation Behaviour Using BlueJ.** Computer Science Education, Vol. 15, No. 1, pp 25 – 40. March 2005.

[Jam06] S. Jameson. **Disciplinary Commons Submission.** WWW. www.disciplinarycommons.org/ last accessed March 2009.

[JISC06] JISC. **Designing Spaces for Effective Learning.** Higher Education Funding Council for England, Bristol. 2006.

[Jon23] H.E. Jones. **Experimental studies of college teaching**. Archives of Psychology, Vol. 68, 1923.

[Kal80] G. Kalton, J. Roberts, & D. Holt. **The Effects of Offering a Middle Response Option with Opinion Questions.** The Statistician, Vol. 29, No. 1. Pp. 65-78. 1980

[Kem01] E. Kemp. **Observing Practise as Participant Observation – Linking Theory to Practise.** Social Work Education, Vol. 20, No. 5, pp 527 – 538. 2001.

[Kin91] A. King. **Effects of students' self-questioning, summarizing, and note taking-review on immediate and delayed lecture comprehension.** Paper presented at the meeting of the American Psychological Association, San Francisco, August 1991.

[Kir06] P. A. Kirschner, J. Sweller & R. E. Clark. **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), pp 75 – 86. 2006.

[Kol03] M. Kölling, B. Quig, A. Patterson, & J. Rosenberg. **The BlueJ System and its Pedagogy.** Journal of Computer Science Education, Special Issue on Learning and Teaching Object Technology. Vol. 13, No. 4. Dec 2003.

[Lan58] H. A. Landsberger. **Hawthorne Revisited: Management and the Worker, Its Critics, and Developments in Human Relations in**

Industry. Distribution Center, NYS School of Industrial and Labor Relations, Cornell University, Ithaca, New York. 1958

[Lau06] D. Laurillard. **Rethinking University Teaching: a framework for the effective use of learning technologies. 2nd Edition.** Routledge Falmer, Cornwall UK. 2006.

[Lav08] J. Lavery, & A. Low. **Concept Mapping in Lectures.** Proceedings of 9th Conference of the Subject Centre for Information and Computer Sciences, Liverpool, England. 2008.

[Llo68] D.M. Lloyd. **A concept of improvement of learning response in the taught lesson.** Visual Education, pages 23–25, Oct. 1968.

[LDK+07] D. Lindquist. T. Denning. M. Kelly. R. Malani. W. G. Griswold. B. Simon. **Exploring the Potential of Mobile Phones for Active Learning in the Classroom.** SIGCSE'07, March 7-10, 2007, Covington, Kentucky, USA.

[Mac70] J.M. Trenaman. *The Length of a Talk.* In McLeish: *The Psychology of Teaching Methods* (1976), 1951.

[McL76] J. McLeish. **The Lecture Methods.** In N.L. Gage, editor, *The psychology of teaching methods*, pages 252–301. The University of Chicago Press, Chicago, IL, U.S.A., 1976.

[Mil07] A. J. Milne. **Entering the Interaction Age Today.** In *Educause review.* January/ February 2007.

[ML03] M. McCabe and I. Lucas. **Engagement with Mathematics in an Interactive Classroom.** In Proceedings of the 6th International Conference on Technology in Mathematics Teaching (ICTMT6), Volos, Greece, Oct. 2003.

[Mye95] B. A Myers. **User Interface Software Tools.** ACM Transactions on Computer-Human Interaction, Vol.2, No 1, March 1995, pages 64-103

[Nov84] G. Novak. **Learning technologies should be designed to increase, and not to reduce, the amount of personal contact between students and faculty on intellectual issues.** Study Group on the Conditions of Excellence in American Higher Education, 1984.

[Opt07] Option Technologies. www.optiontechnologies.com. WWW, last visited: June. 2007.

[Par05a] R.P. Pargas. **MessageGrid: Providing Interactivity in a Technology-Rich Classroom.** Two Page Overview, Clemson University, 2005.

[Par05b] R. P. Pargas. **MessageGrid: Providing Interactivity in a Technology-Rich Classroom (Presentation).** Microsoft Research, Faculty Summit 2005.

[Par06] R. P. Pargas. **Reducing Lecture and Increasing Student Activity in Large Computer Science Courses.** ITiCSE '06, June 26-28, 2006, Bolgna, Italy.

[Per85] D. N. Perkins, C. Hancock, R. Hobbs, F. Martin & R. Simmons. **Conditions of Learning in Novice Programmers. Concept Paper.** Educational Technology Centre, ETC-85-13, April 1985.

[Qwi07] Qwizdom. www.qwizdom.com. WWW, last accessed: June. 2007.

[Rav00] A. Ravenscroft and M.P. Matheson. **Developing and evaluating dialogue games for collaborative e-learning.** Journal of Computer Assisted Learning (2002) Vol. **18,** 93-101

[Rob03] A. Robins, J. Rountree & N. Rountree. **Learning and Teaching Programming: A Review and Discussion.** Computer Science Education, Vol. 13, No. 2, pp. 137-172. 2003.

[Ros71a] B. V. Rosenshine. **Objectively measured behavioral predictors of effectiveness in explaining.** In I. D. Westbury & A. A. Bellack, editors, *Research into classroom processes* (pp. 51-98). Teachers College Press, New York, 1971.

[Ros71b] B. V Rosenshine. **Teaching behaviours and student achievement.** London, National Foundation for Educational Research in England

and Wales. *Brings together, clusters, and summarizes about fifty studies.*

[Ruh90] K. L. Ruhl. C. A. Hughes. & A. H. Gajar. **Efficacy of the Pause Procedure for Enhancing Learning Disabled and Nondisabled College Students' Long- and Short-Term Recall of Facts Presented through Lecture.** *Learning Disability Quarterly*, Vol. 13, No. 1. (Winter, 1990), pp. 55-64.

[Sch05] N. Scheele. **The Interactive Lecture: A new Teaching Paradigm based on Pervasive Computing.** PhD thesis, University of Mannheim, Faculty of Computer Science, 2005.

[Sch70] J.R. Schoen. **Use of Consciousness Sampling to Study Teaching Methods.** Journal of Educational Research, Vol. 63, pages 387–390, May 1970.

[Smi01] B. Smith. **Just give us the right answer.** In H. Edwards, B. Smith, and G.Webb, editors, *Lecturing. Case studies, experience and practice*. Kogan Page Limited, London, U.K., 2001.

[SS69] H. W. Stevenson. & A. Siegel. **Effects of instruction and age on retention of filmed content.** Journal of Educational Psychology, Vol. 68, pages 71 – 74, 1969.

[Ste99] R. J. Sternberg. **Thinking styles.** Cambridge University Press, Cambridge, 1999.

[Tre51] J.M. Trenaman. **The Length of a Talk.** In McLeish: *The Psychology of Teaching Methods (*1976), 1951.

[Ver92] J. D. Vermunt. **Learning styles and directed learning processes in higher education: towards a process-oriented instruction in independent thinking.** Swets and Zeitlinger, Lisse, Netherlands, 1992.

[WFU07] Wake Forest University. http://classinhand.wfu.edu/. WWW, last visited: June. 2007.

[Win96] L. E. Winslow. **Programming Pedagogy – A Psychological Overview.** SIGCSE Bulletin, 28. Pp 17 – 22. 1996.

[Wit03] E. Wit. **Who wants to be … The Use of a Personal Response System in Statistics Teaching.** In MSOR Connections Vol. 3 No 2 May 2003, pp 14 – 20.

[You92] M. F. Young, & J. M. Kulikowich. **Anchored Instruction and Anchored Assessment: An Ecological Approach to Measuring Situated Learning.** Paper presented at the Annual Meeting of the American Educational Research Association, April 1992.