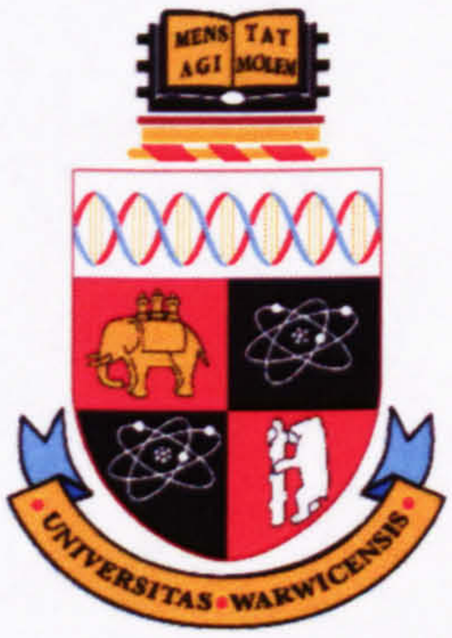
THE UNIVERSITY OF
WARWICK



An investigation into novel software tools for enhancing students' higher cognitive skills in computer programming

by

Jirarat Sitthiworachart

Thesis

Submitted in partial fulfilment of
the requirements for the degree of
Doctor of Philosophy in Computer Science

Department of Computer Science
University of Warwick

June 2005

Contents

List of Figures

List of Tables

Acknowledgements

I am indebted to a number of people who have helped me in various ways during the writing of this thesis.

First and foremost, I would like to thank my supervisor Mike Joy for his support throughout the preparation of this thesis. I am very grateful for all the comments, discussions, encouragement and feedback. This thesis would not have been completed without his assistance.

Thanks also to Ashley Ward for his friendship, discussions and technical knowledge. Further thanks to Steve Matthews and Meurig Beynon for stimulating discussions and constructive feedback.

Last, but certainly by no means least, I would like to thank my family and friends for all the support that they have given me during the course of this research. It means more to me than I could ever write down.

Declarations

This thesis is presented in accordance with the regulations for the degree of Doctor of Philosophy. It has been composed by myself and has not been submitted in any previous application for any degree. The work in this thesis has been undertaken by myself, except the anonymous communication tool is developed by Jonathan Pallant (with my design).

Seven published articles arose from work on the thesis.

1. *The Evaluation of Students' Marking in Web-Based Peer Assessment of Learning Computer Programming,* International Conference on Computers in Education (ICCE2004), Melbourne, Australia, November 30-December 3, 2004, pp. 1153-1163.
2. *Web-based Peer Assessment System with an Anonymous Communication Tool,* The 4th IEEE International Conference on Advanced Learning Technologies (ICALT2004), Joensuu, Finland, August 30-September 1, 2004, pp. 918-919.
3. *Using Web-based Peer Assessment in Fostering Deep Learning in Computer Programming,* International Conference on Education and Information Systems: Technologies and Applications (EISTA'04), Orlando, Florida, USA, July 21-25, 2004, pp. 231-236.
4. *Effective Peer Assessment for Learning Computer Programming,* The 9th Annual Conference on Innovation and Technology in Computer Science Education (ITiCSE2004), University of Leeds, Leeds,UK, June 28-30, 2004, pp. 122-126.
5. *Aspects of Web-based Peer Assessment Systems for Teaching and Learning Computer Programming,* 3rd IASTED International Conference on WEB-BASED EDUCATION (WBE2004), Innsbruck, Austria, February 16-18, 2004, pp. 292-297.
6. *Deepening Computer Programming Skills By Using Web-based Peer Assessment,* The 4th Annual Conference of the LTSN Subject Centre for

Information and Computer Sciences, National University of Ireland, Galway, Ireland, August 26-28, 2003, pp. 152-157.

7. *Web-based Peer Assessment in Learning Computer Programming,* The 3rd IEEE International Conference on Advanced Learning Technologies (ICALT2003), Athens, Greece, July 9-11, 2003, pp. 180-184.

Abstract

Active learning is considered by many academics as an important and effective learning strategy. Students can improve the quality of their work by developing their higher cognitive skills through reflection on their own ideas, and through practice of analytic and evaluative skills. Assessment is a tool for learning, but traditional assessment methods often encourage surface learning, rather than deep learning which is an approach to developing higher cognitive skills. Peer assessment is one of the successful approaches, which can be used to enhance deep learning. It is a method of motivating students, involving students discussing, marking and providing feedback on other students' work. Although it is often used in the context of essays, it has seldom been applied to computer programming courses. The skill of writing good software includes understanding different approaches to the task, and stylistic and related considerations - these can be developed by evaluation of other programmers' solutions.

As part of a study investigating the extent that peer assessment can promote deep learning to develop the higher cognitive skills in a programming course, a novel web-based peer assessment tool has been developed.

- The process used is novel, since students are engaged not only in marking each other's work, but also in evaluating the quality of marking of their peers.
- This system is designed to provide anonymity for the whole process, in order to ensure that the process is fair, and to encourage students to discuss without embarrassment by using an anonymous communication device (ACD) in a variety of roles (script authors, marker, and feedback marker).

In this thesis, we describe and compare the learning theory and tools, which are relevant in learning computer programming. Deep learning, which can be described using the six categories of learning in Bloom's taxonomy, is discussed. Other peer assessment software tools are compared and discussed. The design

and implementation of a novel web-based peer assessment system (with anonymous communication device) are described, and set in the context of the learning theories. The results of evaluating the tools through several experiments involving large programming classes and an essay writing module are reported. In this thesis, we also propose a new variation of Bloom's taxonomy, which is appropriate to describe the skills required for tasks such as programming.

The results indicate that this approach to web-based peer assessment has successfully helped students to develop their higher cognitive skills in learning computer programming, and peer assessment is an accurate assessment method in a programming course.

Abbreviations

ACD	Anonymous Communication Device
BOSS	BOSS Online Submission System
JVM	Java Virtual Machine
OASYS	On-line Assessment SYStem
PBL	Problem Based Learning
RMI	Remote Method Invocation
SD	Standard Deviation
SMS	Short Message Service
SQL	Structured Query Language

Chapter 1 Introduction

1.1 Motivation for the thesis

Learning is increasing knowledge. Some students see learning as a matter of memorising and comprehension of knowledge only to cope with course requirements, which are strategies in surface learning (lower level) [Entwistle2001]. Others see learning as a way to satisfy their own requirements to develop new skills by relating previous knowledge with experiences, which are strategies in deep learning (higher level) [Sandberg1997, Entwistle2001]. Encouragement of creative thinking in science and technology studies is particularly difficult, as the lower level of learning is often emphasised, focusing on instructor teaching [Robins2003, Barak2005]. Students are taught facts and asked to repeat them in various ways [Newcomb1987]. Barak and Jonassen [Jonassen2000, Barak2005] note that education technology should enhance the teaching and learning process, with the emphasis on higher levels of learning, but that many higher education courses use the technology to support only surface learning, such as digital sliding blackboards [Robling2004], automated marking environments [Blumenstein2004], and online tutorials [Yue2004].

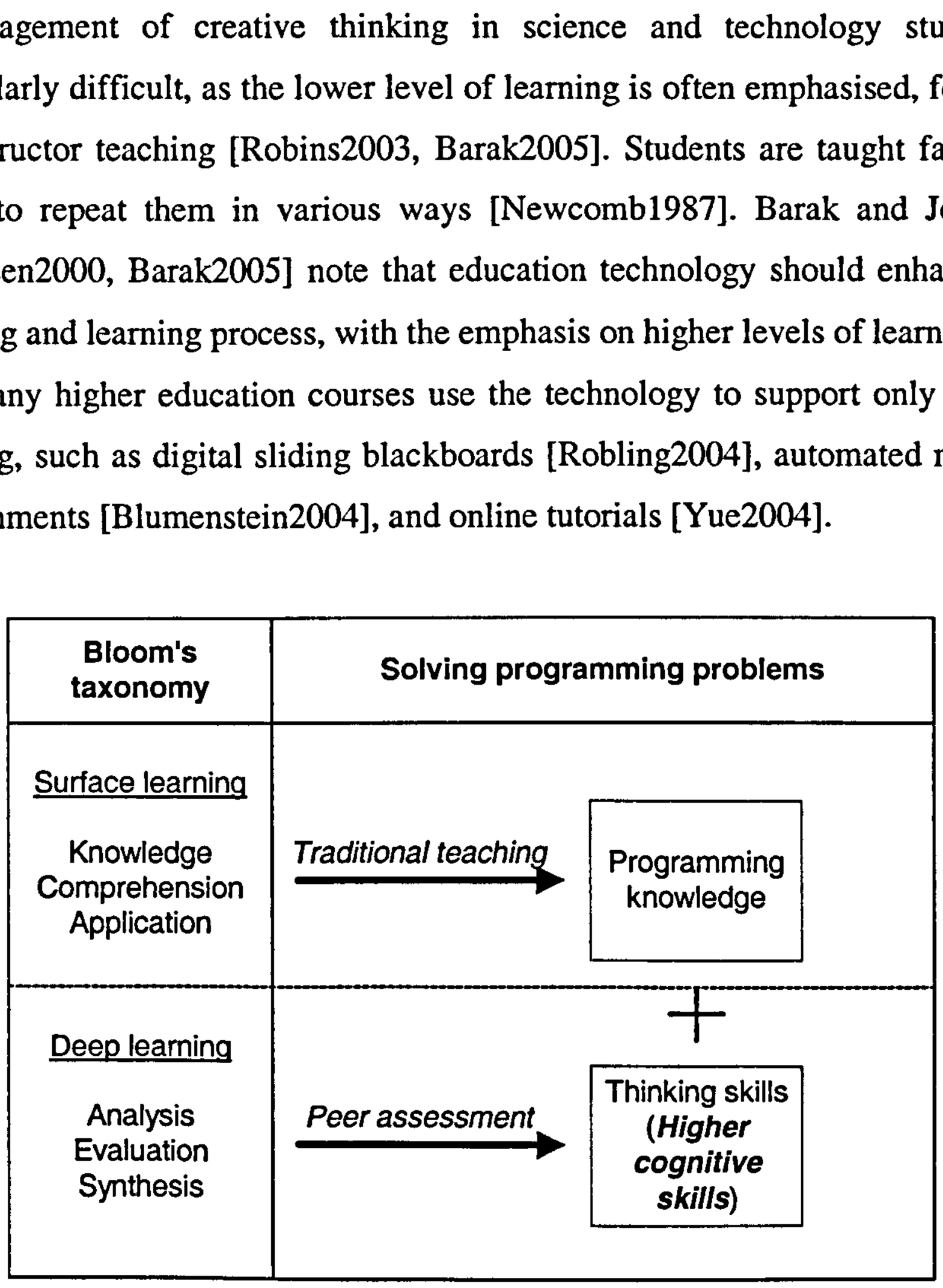


Figure 1 Bloom's taxonomy and peer assessment in learning programming

When learning computer programming, students need to understand the programming language (knowledge) in the first stage, then develop thinking skills (e.g. problem solving, creative thinking) required to solve the programming problems [Baldwin2001, Robins2003, Diwan2004]. However traditional teaching in programming courses usually emphasises the surface learning level (as defined in Bloom's taxonomy), but does not emphasise explicit cognitive development [Lister2000, Robins2003] (as illustrate in Figure 1). Therefore teaching approaches should be reconsidered, and the learning theory of cognition should be applied [Clear1997].

The six categories of Bloom's taxonomy match clearly and easily the progressive development of learning skills in computer programming [Scott2003, Oliver2004], especially the final three levels of Bloom's taxonomy (analysis, synthesis, and evaluation) that are combined into a problem-solving skills category [Cox1998]. Higher cognitive skills, such as problem solving, creative thinking, self evaluation, and critical judgement skills, could be developed through the deep learning approach, to develop thinking skills and increase proficiency in programming [Pea1984, Fowler1996, Cox1998, Fox2003].

When designing tools to encourage students in learning computer programming, the development of higher cognitive skills should be considered [Johnson1997, Deek1998]. Many tools have been created to help students in learning, especially for novice programmers in an introductory programming course. However most tools are appropriate for surface learning, such as helping in program construction, compilation, testing and debugging [Fercheri1994, Deek1998]. There are not many tools that are well designed for deep learning to enhance students' higher cognitive skills. Peer assessment, the deep learning approach that is often used in the context of essays, is investigated to discover whether it can be used to enhance students' higher cognitive skills in a computer programming course.

1.2 Research questions

Many studies in peer assessment emphasise different issues, such as investigation of plagiarism in essay assignments [Davies2000], and contributions of individual grades in group work [Lejk2002]. Our research focuses on enhancing higher cognitive skills, instead of the traditional teaching of programming that emphasises surface learning [Lister2000, Robins2003]. We are interested in finding out how peer assessment enhances students' higher cognitive skills in computer programming. In addition, many studies about peer assessment report that students think peers' marking may not be as accurate as tutors' marking [Orsmond1996, Davies2000, Topping2000, Ballantyne2002]. Therefore the accuracy of the peer assessment method in a programming course will be assessed. The research questions are illustrated below (Figure 2).

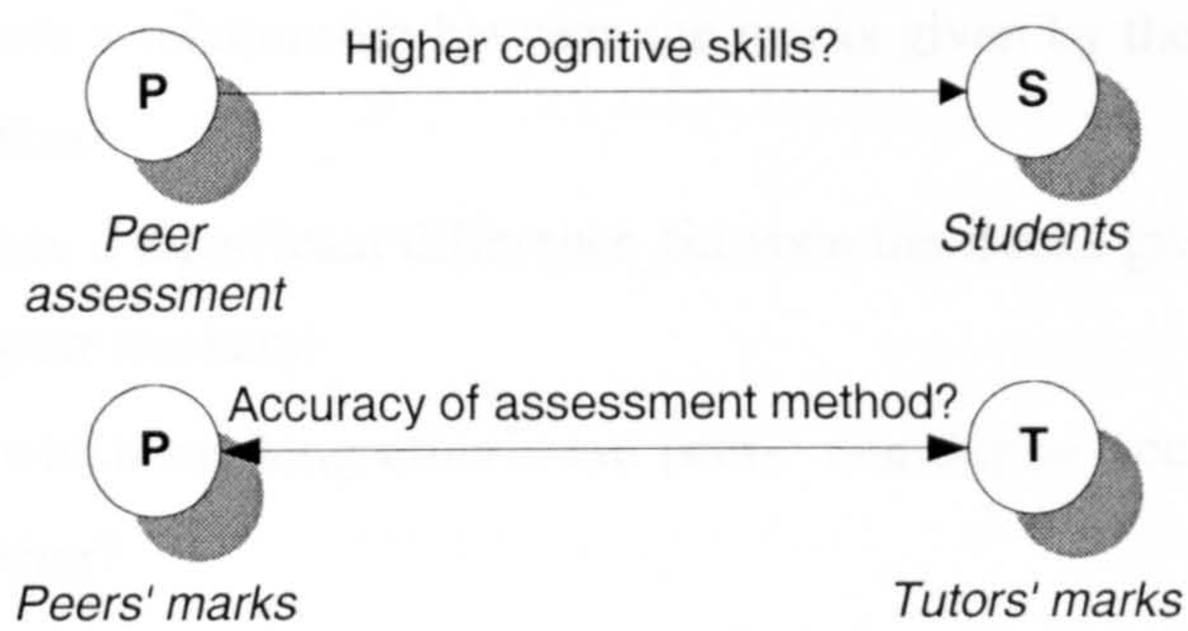


Figure 2 Research questions

Research question 1: Does peer assessment enhance higher cognitive skills in a programming course?

We are interested in how peer assessment encourages students to develop higher cognitive skills, which include deep learning (analysis, evaluation and synthesis) and critical judgement skills. The following sub research questions are considered.

- *Analysis*: Does peer assessment help students to understand and think more about programming?
- *Evaluation*: What is the link between peer and self assessment?

- *Synthesis*: Does the performance of the students improve in subsequent assignments after performing peer assessment?
- *Higher cognitive skills*: Does peer assessment encourage students to develop higher cognitive skills?
- *Critical judgement skills*: Do students display critical judgement skills on evaluation of the initial marking?

Research question 2: Can peer assessment be an accurate assessment method in a programming course?

Tutors' marking is usually accepted as reliable, but student peers' marking in a peer assessment process is suspect. Comparisons between peers' and tutors' marking are examined. The following sub research questions are considered.

- Is there a relationship between the marks given by the tutor and peer markers?
- Is there a significant difference between the marks given by the tutor and peer markers?
- For which marking criteria are peers' marking as accurate as tutors' marking?
- Do peers comment on similar or different issues compared to tutors?
- Are marks within a group of students consistent?
- Are the comments from students with a wide range of abilities different?

1.3 Research methodology

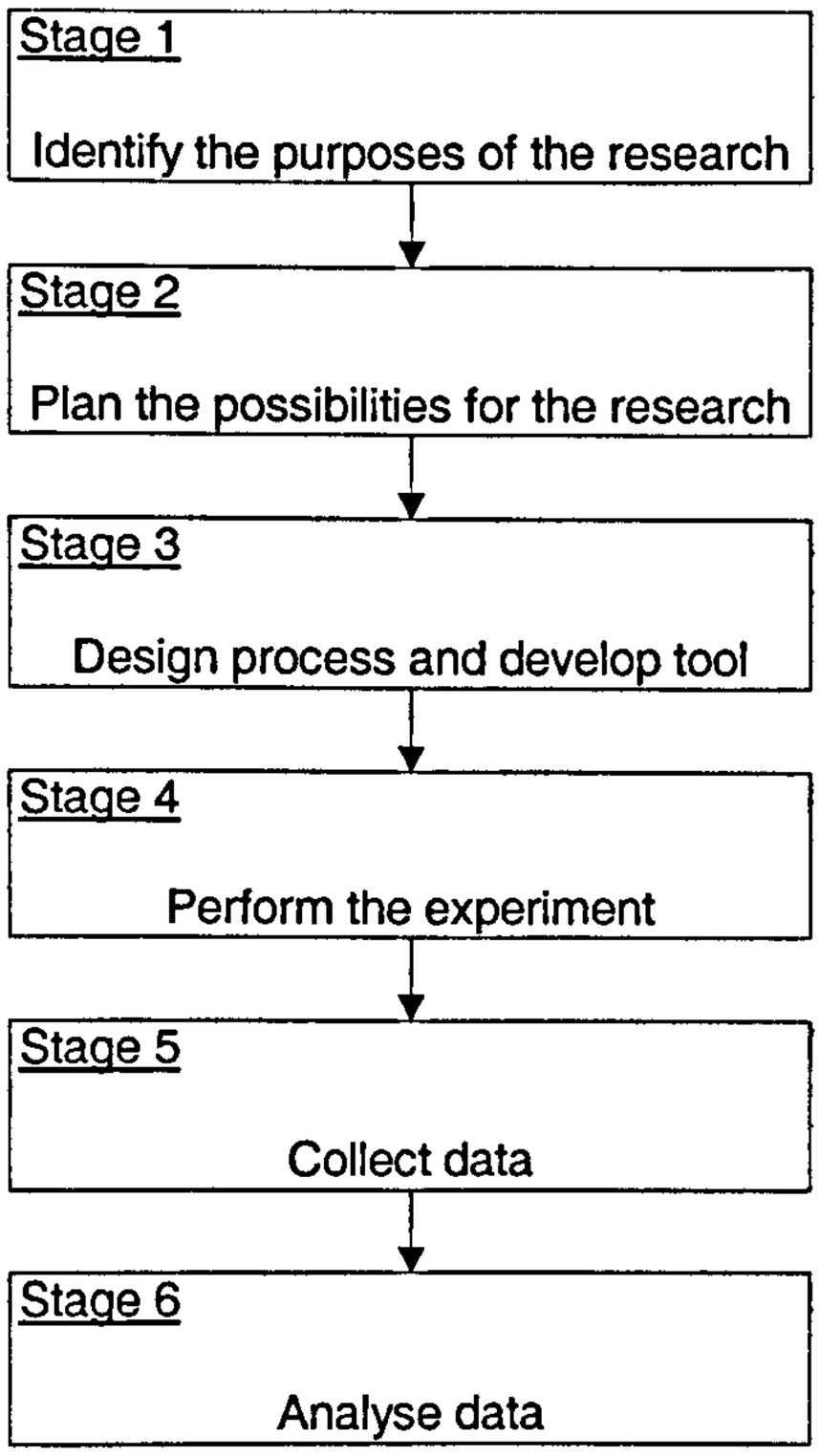


Figure 3 Research methodology

We designed the research methodology by following the standard of research methods in Education [Cohen2000]. Figure 3 illustrates six stages in our research methodology.

Stage 1: Identify the purpose of the research

Our main purpose of this research is to invent novel software tools for enhancing students' higher cognitive skills in computer programming for novice programmers in higher education. From our literature review in this area we found that most current software tools support only surface learning, whereas well-designed software tools to support deep learning in computer programming are quite rare. In addition, peer assessment is a well known technique for the deep learning approach, but there is no evidence that it works well with a computer programming course. Therefore we have developed the novel web-based peer assessment system to support this investigation

to check whether peer assessment does, in fact, enhance higher cognitive skills.

Stage 2: Plan the possibilities for the research

With large programming classes, tools and techniques are required to enhance the teaching and learning process. This research is possible for the following reasons.

- *Valid data:* a large number of first year undergraduate students in programming course are enough evidence for data analysis.
- *Experiment:* performing the experiment on first year undergraduate students is possible, since they are enthusiastic in learning the new technology and can accept new teaching and learning processes easily.
- *Collecting data:* data can be collected from two main sources (students and tutors), firstly, the students' marking in peer assessment exercises, and their opinions about this exercise via online questionnaires and interviews, and secondly, data from the double marking by tutors.
- *Data analysis:* SPSS and NVivo are appropriate software tools to help in analysing the large amount of data, both quantitative and qualitative.

Stage 3: Design process and develop tool

We have designed a new peer assessment process for a programming course, based on deep learning as defined in Bloom's taxonomy. We designed a web-based peer assessment system to provide anonymity for the whole process, in order to ensure the process is fair, to encourage students to discuss without embarrassment, and to allow the process to be closely monitored and supervised. The interface is simple, therefore it can be used easily. Web-based peer assessment with an anonymous communication tool has been developed, and a simplified version of this system with an annotation tool has been developed for an experiment with a computer programming course and an essay-writing course, respectively.

Stage 4: Perform the experiments

We performed three experiments, involving a large programming class (UNIX shell programming module), and an essay writing class (professional aspects of computing module) in the Computer Science department at the University of Warwick. In the first two experiments in programming courses, the assignments were independently double-marked by module tutors, to provide an expert reference against which the marks awarded through the peer assessment process can be compared.

- For *a pilot study* in 2002, the second of three programming assignments was marked using a peer assessment process, and a pre-test and a post-test were provided to measure the students' analysis skills as an effect of the peer assessment exercise. The purpose of this pilot study is to investigate whether peer assessment tool works well with the computer programming course.
- For *a main study* in 2003, we have extended the system by developing an anonymous communication tool to encourage interaction with others, which is a key element in fostering deep learning [Biggs2001]. This second experiment was performed on *all* of three assignments in the same programming module, in order to investigate whether peer assessment promotes higher cognitive skills and to measure the accuracy of this tool (see more detail in section 1.2).
- In the *third experiment*, peer assessment for essays was performed on students who have used peer assessment for programming before. Therefore we can make a comparison with their experience with the programming module to find out the difference between using peer assessment in programming and essay modules for computer science students.

Stage 5: Collect data

Online questionnaires and interviews are appropriate and widely used instruments for data collection [Sapsford1996, Cohen2000,

Bowling2002]. Both methods are required to obtain valid and reliable data, especially with interviews that are helpful for obtaining honest and detailed information from students [Sapsford1996]. The research objectives are translated into questions. The construction of questionnaire and interview are illustrated below.

- *Questionnaire:* we designed questionnaires, by ordering from the factual questions (e.g. gender, first language, experience), then moving to more complex questions (using multiple choices with short answers), and ending with open-ended questions (seekly opinions, attitudes, perceptions and views, together with reasons) [Cohen2000]. Several types of question are applied to encourage more responses. In order to check for reliable answers, we ask the same question in a different form of words, and also use related questions. In addition, to encourage students' replies, we inform them of the questionnaire purpose and assure them of confidentiality.
- *Interview:* face to face standardized open-ended interviews have been chosen, as students are asked the same questions in the same order [Cohen2000]. The questions start with students' background in learning programming, then move to peer assessment exercise (e.g. tool, process, results, benefits, and problems), and end with the suggestions for improving this exercise. Each interview takes around 30-45 minutes, and is recorded, with the students' permission in order to review the answers again later.

Stage 6: Analyse data

In order to answer the research questions, students' marking and tutors' marking are compared (using statistical tests), and the responses from online questionnaires and interviews are analysed and compared. We start from validating data by checking for the completeness, accuracy, and uniformity [Cohen2000]. Software is used for helping in quantitative and qualitative analysis of the large number of data.

- *Quantitative analysis* with SPSS (a statistical analysis and data management system). Descriptive statistics (e.g. dispersion, correlation, tests of significant different) are calculated and interpreted.
- *Qualitative analysis* with NVivo (a qualitative data analysis software). Students' responses from questionnaires and interviews are classified and coded in NVivo. The software helps us to link the ideas, then we can search and explore the patterns of data and ideas easier to analyse them [Gibbs2002].

1.3.1 Ethics

Following advice in the human research ethics handbook [NHMRC2001], the option was available for students not to take part in this research, and their assignments would be marked by a tutor using the traditional assessment. In addition, to be fair to all students, we did not classify them into two groups (controlled and uncontrolled group) for the experiments in order to compare group data.

1.4 Outline of the thesis

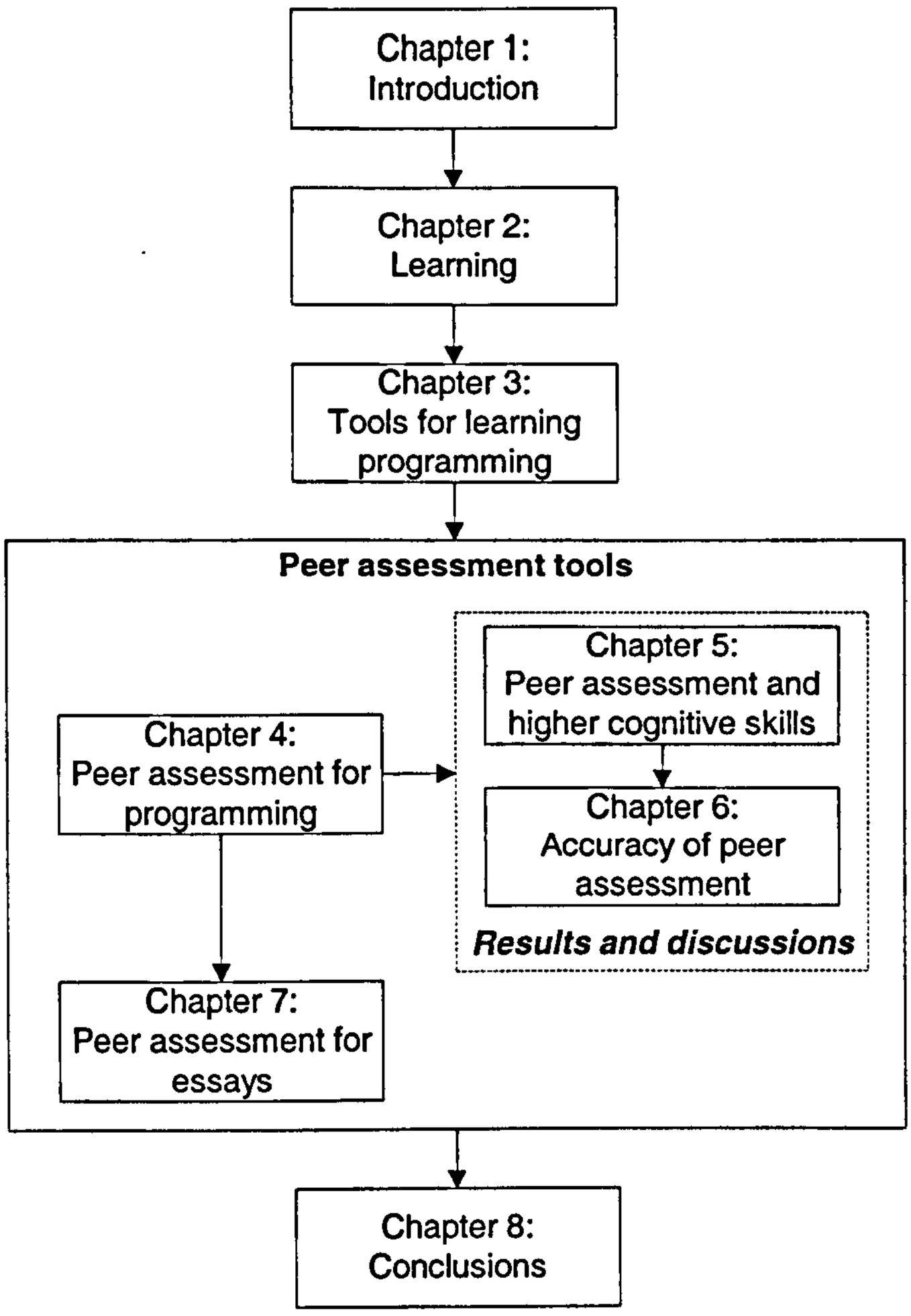


Figure 4 Simplified thesis outline

This thesis is divided into three main parts: learning theory, other tools in learning programming, and my tool (web-based peer assessment). The thesis begins with *introduction* chapter (chapter 1) that gives information about the background of the research, then followed by a *learning* chapter (chapter 2) that explains the learning theory (especially the appropriate learning theory for computer programming), and pedagogy of learning computer programming. We examine *tools for learning programming* (chapter 3), which are divided into surface and deep learning tools.

The design and implementation of our novel web-based peer assessment system is described in the *peer assessment for programming* chapter (chapter 4). The results and discussions of experiments involving large programming classes are reported in chapter 5 and chapter 6.

- Chapter 5 (*peer assessment and higher cognitive skills*) investigates whether peer assessment enhances higher cognitive skills in a programming course.
- Chapter 6 (*accuracy of peer assessment*) examines whether peer assessment can be an accurate assessment method in a programming course.

The comparison of using peer assessment in programming and essays is discussed in the *peer assessment for essays* chapter (chapter 7). The thesis ends with a *conclusions* chapter (chapter 8) that summarises whether peer assessment works well in a computer programming course, and gives suggestions for future work. The summary of each individual chapter follows:

Chapter 1: **Introduction**, we report the research background, including motivation for the thesis, research questions, research methodology, and outline of the thesis.

Chapter 2: **Learning**, in this pedagogical chapter, learning theories are described, with the emphasis on Bloom's taxonomy and deep learning, followed by an overview of learning computer programming. We discuss what should be taught and learned in a computer programming course, and what are the important skills in learning programming. Finally, we propose a new learning theory in the context of programming, which is based on the six categories of Bloom's taxonomy.

Chapter 3: **Tools for learning programming**, we report surface and learning tools for learning programming, with the emphasis on peer assessment, and compare peer assessment methods and tools.

Chapter 4: **Peer assessment for programming,** we describe the methodology, the design of our web-based peer assessment tool, results and discussions for both the pilot (prototype architecture) and the main study (modified architecture).

Chapter 5: **Peer assessment and higher cognitive skills,** we discuss the results of the peer assessment for programming experiment, focusing on how peer assessment encourages students to develop deep learning skills, which is based on our new learning theory in context of programming that is refined from Bloom's taxonomy. The other higher cognitive skills that students could develop from peer assessment are explored. Finally the evaluation of 'quality of marking' that helped students to develop their critical judgement skills is investigated.

Chapter 6: **Accuracy of peer assessment,** this chapter investigates whether peer assessment can be an accurate assessment method in a programming course. Therefore the first investigation is the comparison of peer vs. tutor marking. Then the consistent marking within a group (peers) is investigated, since each group consists of students with a wide range of abilities. Finally the students' response on the accuracy of peer assessment is reported.

Chapter 7: **Peer assessment for essays,** we describe the peer assessment for essay method, and the design of a web-based peer assessment for essay system, followed by the results and discussions. The differences between using peer assessment in programming and essays are examined.

Chapter 8: **Conclusions,** we summarise the thesis, together with proposing a checklist for designing a peer assessment system, followed by the discussion of future work.

Chapter 2 Learning

2.1 Overview of the chapter

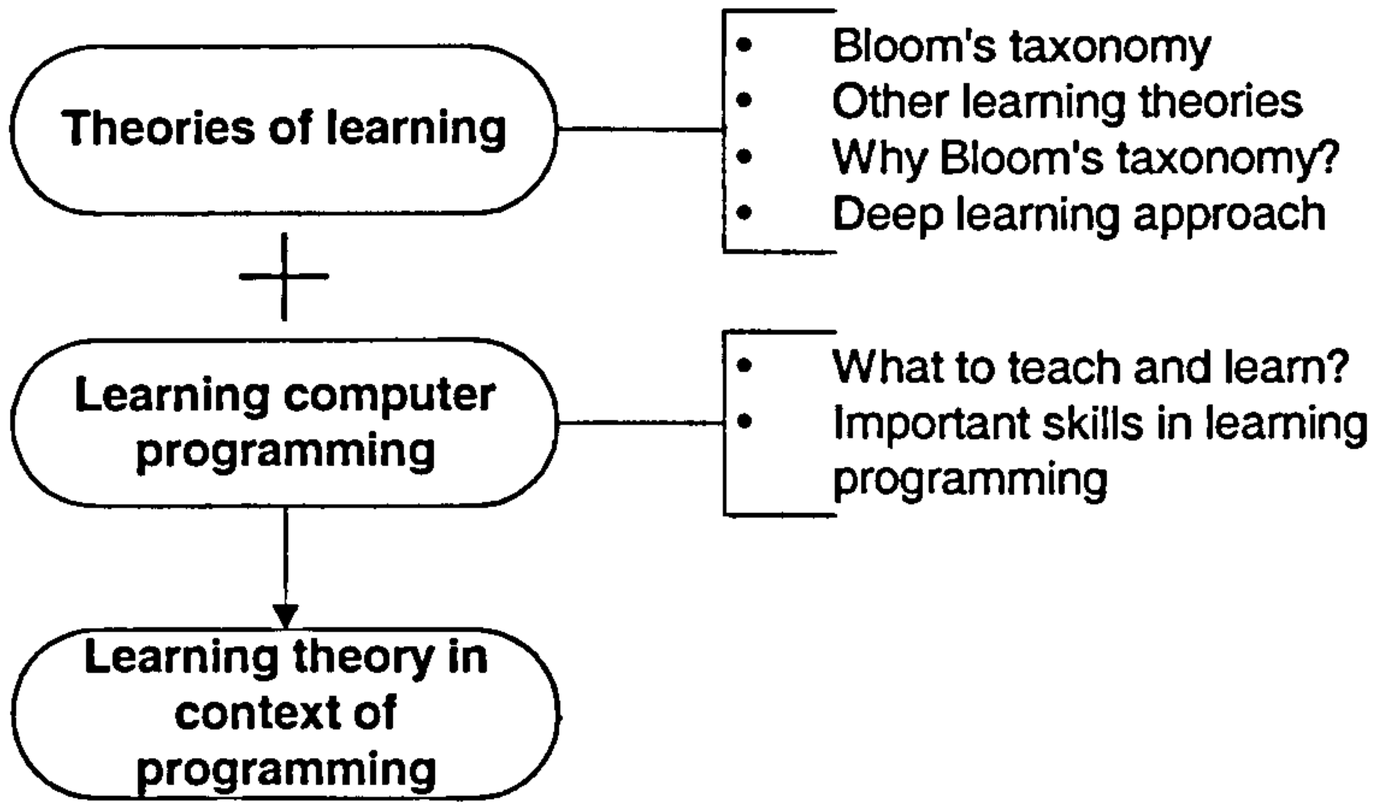


Figure 5 Outline of 'learning' chapter

This is the pedagogical chapter. Learning theories are described in the first part, which compares Bloom's taxonomy with other learning theories. Since this research emphasises higher cognitive skills, a deep learning approach is considered. In the second part, learning computer programming, we discuss what should be taught and learned in a computer programming course, and what are the important skills in learning programming. Finally, we present a learning theory in one context of programming, which is based on the six categories of Bloom's taxonomy.

2.2 Theories of learning

Learning is increasing knowledge, which results in the development of a variety of skills. Brown, Bull, and Pendlebury [Brown1997] defined learning as "changes in knowledge, understanding, skills and attitudes brought about by experience and reflection upon that experience". They suggested that graduates are expected to have two main skills:

- *"knowledge skills*: graduates should have a body of knowledge in the field studied, and be able to apply theory to practice in familiar and unfamiliar situations;

- *thinking skills*: graduates should be able to exercise critical judgement, be capable of rigorous and independent thinking, adopt a problem solving approach, and be creative and imaginative thinkers."

Many academics define learning theory with a different emphasis, such as "condition of learning" by Robert Gagne [Gagne1965], "child cognitive development" by Jean Piaget [Funders2001], "stimulus-response" (S-R) pattern by B. F. Skinner [OLTC1996], etc. Bloom's taxonomy, which is the most widely used, and the other learning theories are described in this section. Then reasons why Bloom's taxonomy is chosen to be applied to learning computer programming instead of other learning theories are discussed. Finally the deep learning approach is examined.

2.2.1 Bloom's taxonomy

In order to develop students' intellectual abilities, Benjamin Bloom [Bloom1956] created a taxonomy to categorize levels of learning in orders of complexity, which was aimed at higher education [Anderson2001]. Six categories of learning in Bloom's taxonomy can be divided into two levels of learning: surface learning and deep learning [Bloom1956]. *Surface learning* consists of the first three categories of Bloom's taxonomy, namely knowledge, comprehension, and application, which emphasise recall, understand and solving trivial problems. The final three categories of Bloom's taxonomy - analysis, synthesis, and evaluation - are combined into problem-solving skills and critical thinking skills or *deep learning* [Cox1998, Fowler1996]. However each succeeding category assumes competence at an earlier category. For example, students should have knowledge and understanding of information in order to apply their knowledge to solve problems. A brief description of the six categories of learning in Bloom's Taxonomy is given in Table 1.

Surface Learning	1. Knowledge *(remember)*	ability to remember and recall information
	2. Comprehension *(understand)*	ability to understand the meaning of material by describing or reviewing material in one's own words
	3. Application *(apply)*	ability to apply material in new situations to solve problems
Deep Learning	4. Analysis *(analyse)*	ability to break down material into its components by identifying parts and analyse their relationships
	5. Synthesis *(create)*	ability to combine parts together to form a new whole
	6. Evaluation *(evaluate)*	ability to judge the value of material for a given purpose or make a decision based on appropriate criteria

Table 1 Six categories of learning in Bloom's Taxonomy

A simplification of Blooms' taxonomic definition of individual categories is summarised below, including examples and key words [Armstrong1998, Clark1999, Pelley1999, and TEP2001].

1. *Knowledge:* remember specific facts

   For example: - recalling facts
   - recalling definitions
   - knowing basic concepts

   Key words: define, describe, identify, label, list, name, etc.

2. *Comprehension:* interpret or explain facts

   For example: - giving descriptions
   - stating main ideas
   - making simple comparisons
   - interpreting charts and graphs

   Key words: classify, cite, discuss, explain, give example, interpret, etc.

3. *Application:* demonstrate the correct usage of a method

   For example: - applying techniques, concepts, rules, and theories to solve problems.
   - constructing graphs and charts

   Key words: apply, demonstrate, show, solve, use, utilise, etc.

4. *Analysis:* analyse the relevancy of data
   For example: - identifying causes
   - distinguishing between facts and inferences
   - relating evidence to conclusions
   Key words: analyse, point out, differentiate, distinguish, discriminate, compare, etc.

5. *Synthesis:* integrate material from different areas into a plan for solving a problem
   For example: - creating new patterns or structures
   - revising process to improve the outcome
   Key words: create, design, plan, organise, generate, write, etc.

6. *Evaluation:* judge whether conclusions are supported by the data
   For example: - giving opinions
   - judging the validity of ideas
   - judging the efficiency of solutions
   Key words: appraise, critique, judge, weigh, evaluate, select, etc.

Beattie et al. [Beattie1997] suggest that the difference between surface and deep learning is relevant in "analysing student learning intentions, learning styles, learning approaches adopted and learning outcomes". In surface learning, students learn simply by memorising facts. They accept the information without query [Buckland2001]. On the other hand, in deep learning, students seek to understand the contents and think critically [Entwistle2001]. Therefore surface learning can be taught using a *passive learning* method, but deep learning requires action or involvement by the learners, which is called *active learning* [Clark1999, Entwistle2001]. The examples of surface and deep learners' behaviours [CELT2001 and Entwistle2001] are shown in Table 2.

Surface learners	Deep learners
- concentrate purely on assessment requirements - accept information and ideas passively - memorize facts and carry out procedures routinely - find difficulty in making sense of new ideas presented - see little value or meaning in either courses or tasks - study without reflecting on either purpose or strategy	- endeavour to understand material for themselves - relate ideas to previous knowledge and experience - use organising principles to integrate ideas - relate evidence to conclusions - examine logic and argument cautiously and critically - become actively interested in the course content

Table 2 Surface and deep learners' behaviours

However there are a few weakness in the original Bloom's taxonomy, such as emphasising the major six categories (not much detail on subcategories) [Anderson2001], and that six categories do not form a cumulative hierarchy [Anderson2001, Huitt2004, Seddon1978].

2.2.1.1 A revision of Bloom's taxonomy

In 2001, Anderson et al. [Anderson2001] revised Bloom's taxonomy in order to make it clearer and more meaningful to a wider audience. For example, changing the names of the six major categories, describing more details of subcategories, reframing the knowledge category to include four types of knowledge, and emphasising how it can be used by teachers at all grade levels. The revised framework is called *Taxonomy framework,* consisting of both knowledge (i.e. factual, conceptual, procedural and metacognitive knowledge: see Table 3), and cognitive process (Bloom's taxonomy in verb form: remember, understand, apply, analyse, evaluate and create). It can be presented in a two dimensional table: the rows contain categories of knowledge, and the columns contain categories of cognitive processes. They hypothesised that this new taxonomy framework helped to spot educational objectives, with a cognitive emphasis on cells of the table, where the knowledge and cognitive process dimensions intersect. The major changes from the original Bloom's taxonomy are summarised below.

- The revision emphasises teachers at all grade levels. Since the original Bloom's taxonomy was aimed at higher education, the revision is designed to include teachers in elementary and secondary education.
- The revision emphasises the subcategories in more detail, as the initial version emphasised only the six major categories.
- Major category titles were changed to verb, which made for clearer understanding. The original terms were renamed from noun: knowledge, comprehension, application, analysis, evaluation and synthesis to verb: remember, understand, apply, analyse, evaluate and create, respectively.
- The knowledge subcategories were reorganised as four types of knowledge, including factual, conceptual, procedural and metacognitive knowledge (see Table 3).
- The taxonomy framework is created with a new design of a two dimensional table. Six major categories are the verb form, which describes the action in individual four types of knowledge (noun form). It makes their relationships more explicit.

Major types and subtypes	Examples
A. **Factual knowledge** – the basic elements students must know to be acquainted with a discipline or solve problems in it	
Aa. Knowledge of terminology	Technical vocabulary, musical symbols
Ab. Knowledge of specific details and elements	Major natural resources, reliable sources of information
B. **Conceptual knowledge** – The interrelationships among the basic elements within a larger structure that enable them to function together	
Ba. Knowledge of classifications and categories	Period of geological time, forms of business ownership
Bb. Knowledge of principles and generalisations	Law of supply and demand
Bc. Knowledge of theories, models, and structures	Theory of evolution, structure of Congress
C. **Procedure knowledge** – How to do something, methods of inquiry, and criteria for using skills, algorithms, techniques, and methods	
Ca. Knowledge of subject-specific skills and algorithms	Skills used in painting with watercolours, whole-number division algorithm
Cb. Knowledge of subject-specific	Interviewing techniques, scientific method

Major types and subtypes	Examples
techniques and methods	
Cc. Knowledge of criteria for determining when to use appropriate procedures	Criteria used to determine when to apply a procedure involving Newton's second law, criteria used to judge the feasibility of using a particular method to estimate business costs
D. **Metacognitive knowledge** – Knowledge of cognition in general as well as awareness and knowledge of one's own cognition	
Da. Strategic knowledge	Knowledge of outlining as a means of capturing the structure of a unit of subject matter in a textbook, knowledge of the use of heuristics
Db. Knowledge about cognitive tasks, including appropriate contextual and conditional knowledge	Knowledge of the types of tests particular teachers administer, knowledge of the cognitive demands of different tasks
Dc. Self-knowledge	Knowledge that critiquing essays is a personal strength, whereas writing essays is a personal weakness; awareness of one's own knowledge level

Table 3 The knowledge dimension by Anderson et al.

Anderson et al. [Anderson2001] remark that analysis is an extension of comprehension and an introduction to evaluation and synthesis. They also recommend that synthesis (the most complex step) should be placed after the evaluation step, since the cognitive process should start from simple to complex (remember to create). However Huitt [Huitt2004] suggests that synthesis and evaluation should be at the same level because both of these steps depend on the analysis, which is the foundational process. Therefore analysis is the most important step in developing from a surface to a deep learning skill. Based on the material in Anderson et al., Figure 6 illustrates the summary of structural changes from the original Bloom's taxonomy to the revised framework.

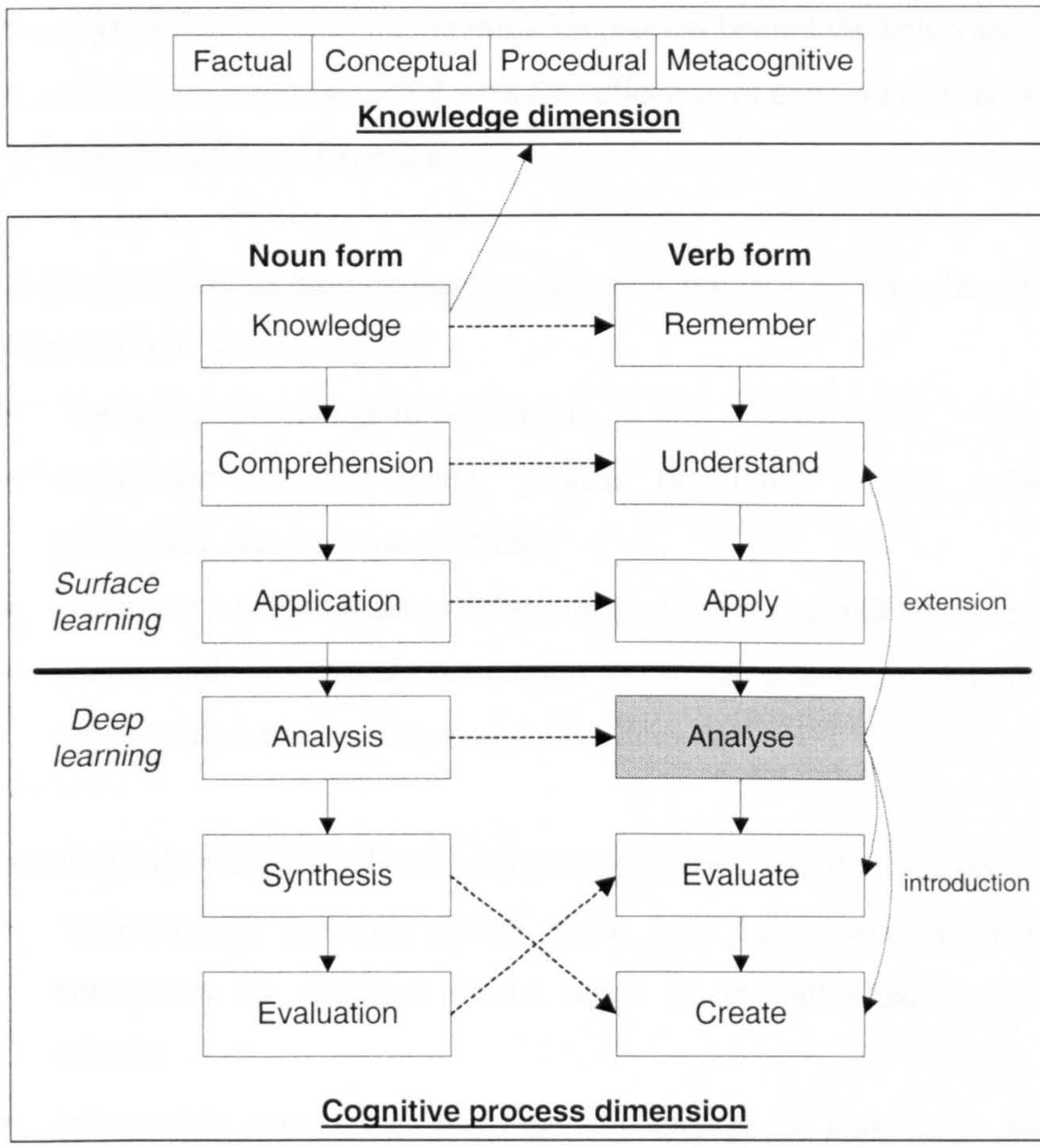


Figure 6 Revision of Bloom's taxonomy by Anderson et al.

Since the revision of Bloom's taxonomy is complicated, arguably difficult to follow and is not targeted at higher education, the original Bloom's taxonomy with the place of synthesis reordered is considered in this thesis. However the original Bloom's taxonomy is quite broad for all subjects. A subject specific learning taxonomy in the context of programming, which is based on the Bloom's taxonomy, has been designed (see more detail in the following section).

2.2.2 Other learning theories

Besides Bloom's taxonomy, there are the other interesting learning theories, such as those of Gagne, Pavlov, Skinner, Piaget and Vygotsky. Mergel, Smith and Carbonell [Mergel1998, Smith2003, Carbonell2004] have classified the learning theories into different groups. According to Mergel [Mergel1998], the basic learning theories can be classified into three groups:

- *"behaviourism*: concentrates on the study of changes in behaviour;

- *cognitivism*: focuses on the thinking process behind the behaviour;
- *constructivism*: associated with the reflection of experience (knowledge is constructed from experience)."

Smith [Smith2003] focused on learning as a process: how or why change occurs. Four orientations to learning are:

- "*behaviourist*: change in behaviour;
- *cognitivist*: internal mental process (including insight, information processing, memory perception);
- *humanist*: a personal act to fulfil potential – self directed learning;
- *social and situational*: interaction in social contexts - learning is a relationship between people and environment."

Carbonell [Carbonell2004] classified the learning theories into three groups:

- "*behavioural*: learning occurs when new behaviours or changes in behaviours are acquired as the result of an individual's response to stimuli;
- *information processing*: learning is a change in knowledge stored in memory;
- *constructivist*: learning is the process where individuals construct new ideas or concepts based on prior knowledge and/or experience."

These learning theory classifications are not much different, and in particular Mergel and Carbonell have identified three similar groups (with essentially the same definition). Behaviourist principles are widely used in the classroom [Elliott1996], and Gagne, Pavlov, and Skinner are the leading theorists in the study of behaviour.

2.2.2.1 Well-known learning theorists

Gagne: In 1965, Gagne [Gagne1965] published 'the conditions of learning', which outlined the relation of learning objectives to appropriate instructional designs. The external and internal conditions that support learning outcomes are teacher and learner skills, respectively [Perry2002], and the behaviour outcome is

emphasised [Kruse2004]. Different instruction is required for different learning outcomes [TIP2004, OLTC1996]. The theory outlines nine instructional events and mental processes as presented in Table 4. These instructional events provide the necessary conditions for learning and serve as the basis for designing instruction and selecting appropriate media [Gagne1979].

Instructional event	Internal mental process
1. Gaining attention (reception)	Stimuli activates receptors
2. Informing learners of the objective (expectancy)	Creates level of expectation for learning
3. Stimulating recall of prior learning (retrieval)	Retrieval and activation of short-term memory
4. Presenting the content (selective perception)	Selective perception of content
5. Providing learning guidance (semantic encoding)	Semantic encoding for storage long-term memory
6. Eliciting performance (responding)	Responds to questions to enhance encoding and verification
7. Providing feedback (reinforcement)	Reinforcement and assessment of correct performance
8. Assessing performance (retrieval)	Retrieval and reinforcement of content as final evaluation
9. Enhancing retention and transfer (generalization)	Retrieval and generalisation of learned skill to new situation

Table 4 Gagne's instruction events

Pavlov: Pavlov is best known for classical conditioning or stimulus substitution theory [Mergel1998], and his work is associated with dog behaviour: interaction with food and bell [Jarvis2003, Mergel1998]. It can be applied to human learning behaviour, which associates with "the conditioned stimulus, unconditioned stimulus, and the production of the same outcome" [Jarvis2003]. John Watson, another behaviourist, has used Pavlov's ideas as the basis for more extensive claims in behaviourist learning theory in psychology [Jarvis2003, PBS2005]. However Jarvis [Jarvis2003] argues that this classical conditioning is only reflexive, it is not really learning. A summary of Pavlov's experiments about stimulus and response items with dogs is displayed in Table 5 [Mergel1998].

Food Salivation	Unconditioned stimulus Unconditioned response (natural, not learned)
Bell Salivation	Conditioned stimulus Conditioned response (to bell)

Table 5 Pavlov's stimulus and response items

Skinner: This theory believes in the stimulus-response (S-R) pattern of conditioned behaviour. Skinner's theory provides behavioural explanations for a wide range of cognitive phenomena [OLTC1996, Good1990]. He suggests that learning is a function of change in behaviour [OLTC1996]. Behaviour that is positively reinforced by a stimulus will reoccur [TIP2004, OLTC1996, Jarvis2003]. A positive reinforcement (reward) can be used to get the expected response, whilst a negative reinforcement (punishment) can be used to prevent unwanted responses [Jarvis2003]. Skinner's operant conditioning mechanisms are presented in Table 6 [Good1990].

Operant conditioning	Description	Examples
Positive reinforcement or reward	Responses that are rewarded are likely to be repeated.	Good grades reinforce careful study.
Negative reinforcement	Responses that allow escape from painful or undesirable situations are likely to be repeated.	Being excused from writing a final because of good term work.
Extinction or non-reinforcement	Responses that are not reinforced are not likely to be repeated.	Ignoring student misbehaviour should extinguish that behaviour.
Punishment	Responses that bring painful or undesirable consequences will be suppressed, but may reappear if reinforcement contingencies change.	Penalizing late students by withdrawing privileges should stop their lateness.

Table 6 Skinner's operant conditioning

Piaget: this theory is based on the development of child cognitive structures [Funders2001]. It is called genetic epistemology because "the primary interest is in how knowledge develops in human organisms" [OLTC1996]. The appropriate

learning materials and activities involve the level of mental operations, for example teachers should avoid asking students to perform tasks that are beyond their current cognitive capabilities [TIP2004, OLTC1996]. Piaget's theory identifies four development stages characterising the process of child progression (see Table 7). Although Piaget's work is limited to children of age up to 15, other cognitive development theorists have extended his work, such as Kohlberg and Fowler [Jarvis2003].

Stages	Age	Process
1. Sensorimotor	*birth - 2 years old*	The child, through physical interaction with his or her environment, builds a set of concepts about reality and how it works. This is the stage where a child does not know that physical objects remain in existence even when out of sight.
2. Preoperational	*ages 2-7*	The child is not yet able to conceptualise abstractly and needs concrete physical situations.
3. Concrete operations	*ages 7-11*	As physical experience accumulates, the child starts to conceptualise, creating logical structures that explain his or her physical experiences. Abstract problem solving is also possible at this stage. For example, arithmetic equations can be solved with numbers, not just with objects.
4. Formal operations	*beginning at ages 11-15*	By this point, the child's cognitive structures are like those of an adult and include conceptual reasoning.

Table 7 Piaget's genetic epistemology

Vygotsky: The major idea of Vygotsky is "social interaction plays a fundamental role in the development of cognition" [TIP2004]. His original work was in the context of language learning in children: thinking development is determined by language [Vygotsky1962]. Then in 1978, he published *Mind in Society*; "every function in the child's cultural development appears twice: first, on the social level – between people, and later, on the individual level - inside the child"

[Vygotsky1978]. Table 8 displays Vygotsky's idea about how culture teaches children both what to think and how to think [Funders2001].

What to think	*Through culture* children acquire much of the content of their thinking (knowledge).
How to think	The *surrounding culture* provides a child with the processes or means of their thinking (intellectual adaptation tools).

Table 8 Vygotsky: social cognition learning

It can be concluded that Gagne, Pavlov and Skinner focus on studying changing behaviour, whilst Piaget and Vygotsky concentrate on cognitive development. These learning theories can be applied in teaching and learning by selecting the suitable stimulus for the expected response (students' behaviour), and being aware of the level of cognitive development matching the students' ages and capabilities.

2.2.2.2 Other learning theorists

Other learning theorists are summarised in Table 9, including types and scopes of individual learning theories [Crain1985, OLTC1996, Mergel1998, Funders2001, Jarvis2003, Aggelia2004].

Learning theorists	Types	Scope
Bandura	Social learning theory	Observational learning: observing and modeling the behaviors, attitudes, and emotional reactions of others. Observers like to adopt a modeled behaviour, if: • the model is similar to the observer • the behaviour has functional value
Bransford	Anchored instruction	Technology based learning: development of interactive videodisc tool to encourage students in solving problems.
Bruner	Constructivist theory	Learners construct new ideas based on their knowledge and experience. Teacher should encourage students to discover the solution by themselves (willing to learn).

Learning theorists	Types	Scope
DeBono	Lateral thinking	Human problem solving: problems require a different perspective to be solved successfully.
Fowler	Religious faith development	Another cognitive development based on Piaget and Kohlberg's ideas, with addition of pre-stage (infancy). This theory is for religious faith development.
Gardner	Multiple intelligences	Gardner identified the distinct seven intelligent ways to resolve problems. Starting from the ability to use words and language, to self-reflection, and awareness.
Guthrie	Contiguity theory	Changing in learning behaviour related to stimulus and response. His framework was applied to personality disorders.
Kohlberg	Moral development	Children's moral development: how they develop a sense of right, wrong, and justice, is related to moral thinking, not moral action.
Lave	Situated learning	Learning requires social interaction and collaboration (activity).
Miller	Information processing theory	Miller proposed two theoretical ideas: chunking concept with short-term memory, and using computers as a model for human learning.
Paivio	Dual coding theory	Paivio's theory concerns verbal and non-verbal processing. It has been applied to cognitive phenomena (e.g. problem solving, concept of learning and language).
Rogers	Experiential learning	There are two types of learning: cognitive (meaningless) and experiential (significant). Personal students' interests/wants address significant learning.
Thorndike	Connectionism	A law of effect that specifies responses to a situation, which are followed by: • satisfaction will be strengthened; • discomfort will be weakened.
Watson	Emotional conditioning	Watson's work presented the role of conditioning in development of emotional responses (i.e. fears, phobia, and prejudices) to certain stimuli.

Table 9 Summary of learning theory

The above learning theories can be summarised in the following categories:

- learning theories presenting stimulus-response in learning (e.g. Pavlov, Skinner, Thorndike, Guthrie), and cognitive learning development (e.g. Piaget, Kohlberg, Fowler);
- learning theories observing animal behaviour, which could apply to human beings (e.g. Pavlov, Thorndike, Guthrie), and research on child development, which could apply to all ages (e.g. Piaget, Vygotsky);
- learning theories which are adaptations of other theories, such as Pavlov-Watson, Piaget-Kohlberg, etc.

2.2.3 Why Bloom's taxonomy?

Bloom's taxonomy is a categorisation of knowledge and thinking skills, which are the most widely accepted practical taxonomies for educational objectives [Eliott1996, Brown1997, Clark1999]. Six categories of learning in Bloom's taxonomy have been translated into more than twenty languages [Anderson2001]. It is an appropriate learning taxonomy focussing on higher education, in order to develop students' learning skills (from surface to deep learning). Other reasons why Bloom's taxonomy is chosen to be applied in learning computer programming include:

- it is considered as a benchmark to measure a student's level of understanding in a particular subject [Bell1995, Imrie1995];
- it is a technique for analysing the higher cognitive skills of a given task [Oliver2004];
- it is clearly and easily matched to the development of learning skills in computer programming [Scott2003, Oliver2004];
- it also has been applied in many computer programming courses. The following publications indicate Bloom's taxonomy is an appropriate pedagogy.
  - The first year programming subject [Lister2000]: MCQ and Bloom's taxonomy (emphasising the surface level).
  - Bloom rating in IT course [Oliver2004]: planning 3 year programming courses according to Bloom's taxonomy.

- Computer science classes testing [Scott2003]: proposing testing examples for different levels in Bloom's taxonomy.
- JkarelRobot in introductory programming courses [Buck2001]: considering Bloom's taxonomy as guiding principles for designing programming exercises.

The comparison between Bloom's taxonomy and the other well-known learning theories is displayed in Table 10 [OLTC1996, Mergel1998, Funders2001, Jarvis2003, TIP2004].

Theorists	Types	Scope	Learning
Gagne	Condition of learning	Military training: Intellectual skills (behaviour)	There are different types of learning and each type requires specific instruction.
Pavlov	Classical conditioning or stimulus substitution	Dog behaviour: Conditioned/ unconditioned stimulus-response	Teacher decides an appropriate stimulus for the expected students' responses.
Skinner	Operant conditioning	Stimulus-response (S-R) pattern	
Piaget	Genetic Epistemology	Children's cognitive development	Content to be learnt should match children's ages/capabilities.
Vygotsky	Social cognition	Culture contribute to a child's intellectual development	Culture teaches children both what to think and how to think.
Bloom's taxonomy	Knowledge and thinking skills	Cognitive taxonomy (higher education): surface and deep learning	Hierarchy of learning skills starts from knowledge to evaluation.

Table 10 Bloom's taxonomy against the leading learning theories

The Gagne, Pavlov and Skinner theories concentrate on changing behaviour, which can be applied to students' learning behaviour in programming courses. Piaget's and Vygotsky's theories focus on the development of cognitive skills, which is the focus of this thesis. However, learning computer programming is a kind of problem solving, which also requires knowledge and thinking skills

[Brown1997, Baldwin2001, Robins2003]. Therefore in order to investigate the development of students' higher cognitive skills in the context of computer programming, six categories (especially the deep learning level) in Bloom's taxonomy are an appropriate and practical learning taxonomy.

2.2.4 Deep learning approaches

According to Biggs [Biggs2001], deep learning involves explaining, arguing, reflecting, and applying knowledge to new problems, then relating new problems to established principles, and hypothesizing. Cox and Clark [Cox1998] describe deep learning as "the capacity to use concepts creatively, and leads to the development of ability to think about problem situations and create new solutions to those problems." Rosie [Rosie2000] argues that deep learning is a strategy that students can adopt. He suggests deep learning requires "higher cognitive skills, meaningful engagement in and enjoyment of learning, and a desire to think conceptually rather than amass detail".

The deep learning approach is associated with *intrinsic motivation* [Chin2000], which is "the primary motivator of a deep interest in content for its own sake" [Jenkins2001]. In the deep learning approach, students attempt to relate parts to each other, previous knowledge and experiences to new ideas [Chin2000]. In contrast, the surface learning approach is based on *extrinsic motivation* ("the primary motivator is the career and associated rewards that will follow from the successful completion of the course" [Jenkins2001]). The students who use a surface learning approach perceive the task as a demand to be met through routine learning [Chin2000]. Thus motivation is a main factor for both surface and deep learning.

The design of teaching and assessment methods can be used to encourage deep learning [Brown1997]. Many academics have suggested a variety of approaches to help students achieve deep learning. For example,

- Entwistle [Entwistle2001] states that deep learning can be promoted through curriculum design, teaching, and assessment;
- Grauerholz [Grauerholz2001] proposes teaching holistically to help students achieve deep learning;

- Nine strategies for fostering a deep learning approach have been published in the AAHE Bulletin [AAHE1993].

According to ***Entwistle*** [Entwistle2001], the three approaches to promote deep learning are elaborated in Table 11.

Curriculum design	• identifying generative, open topics; • using aims to emphasize understanding; • incorporating authentic, relevant topics; • defining 'essential' information; and • selecting appropriate textbooks.
Teaching	• analysing the derivation of new terms; • emphasizing principles and concepts; • conveying information effectively (through clarity, level, pace, and structure); and • evoking a deep response (through explanation, enthusiasm, and empathy).
Assessment	• focusing on understanding performance, using tasks to develop and demonstrate understanding, and feedback to clarify and stress understanding; • using techniques to tap understanding, including more open-ended questions and less reliance on multiple-choice questions; and • grading in relation to levels of understanding, using qualitative criteria to boost validity

Table 11 Three approaches to promote deep learning by Entwistle

Grauerholz [Grauerholz2001] proposes that one of the best ways to help students achieve deep learning is to teach holistically. He defines holistic teaching as "pedagogical approaches that consciously attempt to:

- promote student learning and growth on levels beyond the cognitive;
- incorporate diverse methods that engage students in personal exploration and help them connect course material to their own lives; and
- help students clarify their own values and their sense of responsibility to others and to society."

Grauerholz [Grauerholz2001] believes that holistic approaches lead to deep learning more than traditional approaches for several reasons: "first, holistic teaching consciously attempts to acknowledge and address students' emotional, moral spiritual, and intellectual concerns and struggles; second, holistic teachers view students as multifaceted people who have very active lives, rich backgrounds, and multiple intelligences that are all integral to the learning and teaching process; third, holistic teaching seeks to provide a safe environment for students to express their ideas and feelings openly".

From ***AAHE*** research [AAHE1993], nine strategies for improving the quality of student learning and fostering deep learning are:

- encouraging independent learning;
- supporting personal development;
- presenting problems;
- encouraging reflection;
- using independent group work;
- learning by doing;
- developing learning skills;
- setting projects; and
- fine tuning.

Results from AAHE research [AAHE1993] suggest that the most important criterion encouraging a deep learning approach is a reflection on learning. Setting projects, which involves the application of knowledge to new situations, is a most common strategy used for fostering a deep learning approach in higher education. Peer assessment is another interesting method that can be used to encourage deep learning [AAHE1993, Biggs2001]. Peer assessment is an approach, which, although often used in the context of essays, has seldom been applied to computer programming courses. Therefore we will investigate whether peer assessment can be used to enhance students' higher cognitive skills in computer programming. A comparison of existing peer assessment tools and the other assessment tools will be discussed in Chapter 3.

2.3 Learning computer programming

"Programming is learned by programming, not from books."

[Jenkins2002]

Computer programming has been defined as "the set of activities involved in developing a reusable product consisting of a series of written instructions that make a computer accomplish some task" [Pea1983]. When thinking about the task of learning to program, most students think of learning the syntax of a programming language [Pea1983]. What the teachers should teach and students should learn in a programming course are discussed, followed by the necessary skills in learning computer programming.

2.3.1 What to teach and learn?

> *"At the end of the module, a student will have an understanding of: Data types; Variables, identifiers and scope; Program control structures; Recursion and iteration; Objects and classes; Instance and class definitions; Parameter passing by reference and by value; Array handling; Class inheritance; Error handling; Program design, construction and testing."*
>
> [Java programming: Jarvis2004]

This thesis focuses on an introductory programming course, which is fundamental for the other advanced programming courses. The purpose of an introductory programming course is to teach students to write programs, emphasising correctness, clarity, efficiency, and maintainability of programs [Johnston1985, Jenkins2002, Jarvis2004], as in the above example of expected learning outcomes of a Java programming course. Deek [Deek1998] suggests that a model for teaching and learning programming should combine "the problem solving methods, the language, and the instructional methodology in a comprehensive system", to support the problem solving and program development process.

When learning to program, Johnston [Johnston1985] suggests that students should start from:

- learning to read and understand programs (e.g. syntax, comments and procedures);
- then they should learn more about program design, encouraging the use of subprograms;
- as people always make mistakes, programming errors should be dealt with;
- finally some attention needs to be paid to program efficiency, in order to save time (work fast) and save storage space.

Fincher and Kolling [Fincher1999, Kolling2001] propose that a strategy for teaching programming should emphasise reading good examples, which leads to writing a better program. Baldwin [Baldwin2001] also suggested, in order to write a good program, a student needs to:

- "design program architecture, break large or complex tasks into smaller or simpler and more manageable tasks, and
- choose appropriate data structures and algorithms."

Moreover, Oliver [Oliver2004] proposed a programming stream over three years structured according to Bloom's taxonomy:

- "First year: reading and comprehending code
- Second year: writing code fragments for a defined context
- Third year: writing complete non-trivial programs"

One of the important tasks in teaching programming is changing the negative view of learners about learning programming, since programming is reputed to be a difficult and complex subject [Baldwin2001, Jenkins2002]. Educators have created many tools to help students in learning programming (discussed in the next chapter). Examples of techniques that have helped students to learn to program are:

- *immersion, reading and writing* [Campbell2002]: questions of interfaces, architecture and design are emphasised, instead of the syntactic details.

The author's introductory programming course started with reading, modifying and writing 200 lines of bank ATM simulation program.

- *blended learning* [Boyle2003]: proposes a number of changes in module organisation, tutorial support and online resources, such as integrating traditional face to face approach with online delivery, providing tutorial support through e-learning materials; monitoring students' performances through continuous online assessment.
- *problem based learning* [Ellis1998]: provides a range of resources (e.g. collaborative drawing tools and editor, electronic whiteboards, and multi-user virtual environments) to help students to solve problems using pair work. This strategy has been used in teaching a first year undergraduate course in Java. Students can ask questions to teachers via newsgroups. The expectation of this technique is that students can improve their skills in working methods, document writing, teamwork, and project management.
- *problem transformations* [Azalov2003]: a set of programming problems was used to teach the C++ programming language. Students learned to *analyse* and *synthesise* the new solutions for those problems. A set of similar sequence programming problems would start from simple to complicated problems (systems of problems) to help students gradually develop their programming skills.

Fincher [Fincher1999] argues that teaching programming is not only teaching students how to get the computer to do something, but also developing their transferable skills. Student learning should be focused on rather than instructor teaching, in order to foster deep learning and create independent, reflective learners [Kirkwood2000, Robins2003].

2.3.2 Important skills in learning programming

A computer is only a tool, writing a good program is based on the programmers' skill and creative thinking [Baldwin2001]. Baldwin, Byrne and Jenkins [Baldwin2001, Byrne2001, Jenkins2002] state that problem solving and mathematical abilities are essential skills for learning programming. However

Pea [Pea1984] argues that there is no evidence for the relationship between mathematical ability and programming skills. He proposes that the prerequisites skills for learning computer programming are:

- *"processing capacity* – good memory or a great concentration are required in following what the program does;
- *analogical reasoning* – students may have background knowledge and capacities relevant to programming. Solving programming problems may depend on these analogical thinking skills;
- *conditional reasoning* – students have the ability to understand the data flow through conditional statements (e.g. loop, selection statements);
- *procedural thinking* – everyday thought may influence students to understand the flow of programming easily, including following complex instructions."

Deek [Deek1999] suggests that programmers must develop skills, which include: "learning the language, composing new and comprehending existing programs, testing and debugging solutions, and documenting and modifying the programs they write." McGill [McGill1997] also suggests three necessary knowledge skills that students should have in order to solve programming problems:

- *"Syntactic*: specific facts of programming languages to write programs (e.g. its rules);
- *Conceptual*: constructs and principles of computer programming to design solution for simple problem;
- *Strategic*: recognition and decomposition of a problem (problem solving skills)."

Baldwin and Pea [Baldwin2001, Pea1983] advise that learning computer programming requires higher cognitive skills (e.g. planning, reasoning, creative thinking, and problem solving), and efforts to discover new techniques. Figure 7 displays the activities of problem solving processes in programming that are required for both novice and expert programmers [Pea1983].

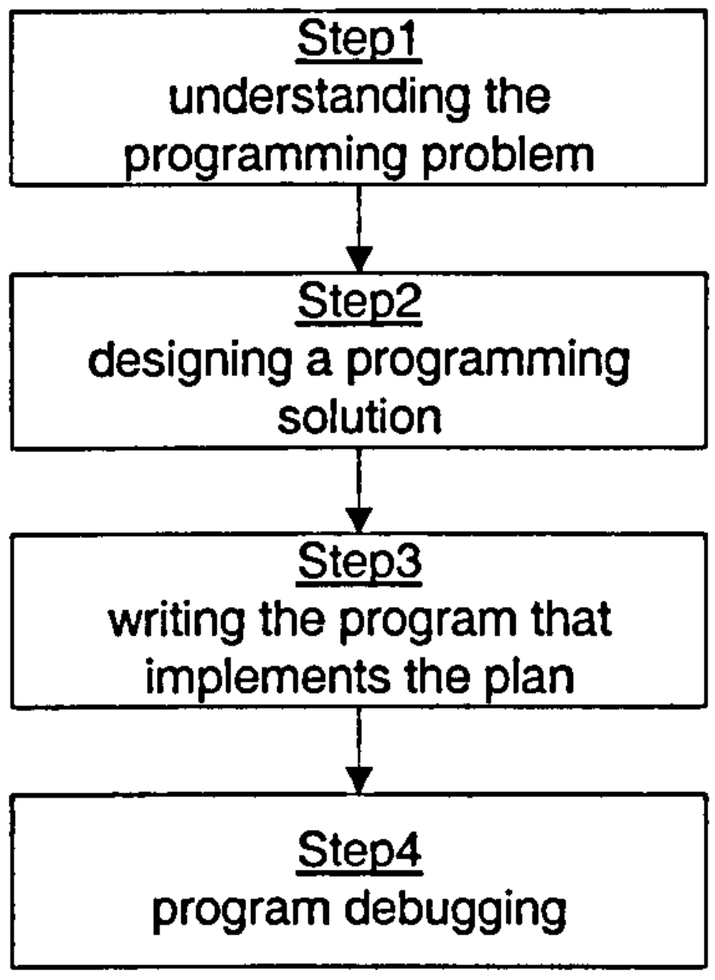


Figure 7 Problem solving process in programming

Peden and Deek [Peden1994, Deek1998] suggest that the improvement of students' critical thinking skill should be concentrated in an introductory programming course, as critical thinking ability will enable students to solve programming problems effectively [Kiper1996, Deek1998]. Robins [Robins2003] also quoted that to write a good program, besides requiring programming knowledge, students need problem solving skills. Critical thinking and creative thinking skills have influenced the improvement of problem solving skills [Kirkwood2000]. However these two skills (critical and creative thinking) can be developed from analysis and evaluation of others' and of their own programs. A peer assessment tool, which can be used to develop these higher cognitive skills, is discussed later.

2.4 Learning theory in the context of programming

Learning computer programming can match the cognitive skill in Bloom's taxonomy, as knowledge (surface learning) and thinking skills (deep learning) are required [Baldwin2001, Robins2003]. However the definition and explanation of Bloom's taxonomy in each category are for general education. Many examples of cognitive processes are in an area, which is not relevant to programming, for example history, painting, language, mathematics, essays, business, etc. A definition of learning taxonomy in the context of programming, which is based on Bloom's taxonomy is presented in Table 12. The order of the synthesis step is moved to after the evaluation step because creative thinking

depends on analysis and evaluation. This idea is also supported by Anderson et al. [Anderson2001] as synthesis is the most complex category, therefore it should be placed last in a cumulative hierarchy.

In the higher level (analysis, evaluation, synthesis), students think more deeply about a program, and figuring out the best solution. However to achieve deep learning, students need a base knowledge and understanding of programming from the lower level (knowledge, comprehension, application). The discrimination between surface and deep learning is the level of difficulty and thinking. We propose a new definition of a learning taxonomy in the context of programming (with examples) (see Table 12). This learning taxonomy in the context of programming will be used in this research.

Surface Learning	**Knowledge** (remember)	The ability to remember programming syntax and structure.
		For example: for/do/while repetition, if/else selection, case/switch multiple-selection, and the structure of each programming language.
	Comprehension (understand)	The ability to grasp the meaning of the basic concepts of programming and to interpret a program.
		For example: understand what the program does; understand different data types and conditional statement; understand the concepts of array, methods/functions, error handling techniques.
	Application (apply)	The ability to write code fragments, solve programming problems by applying simple techniques, methods, and programming concepts.
		For example: write a program to print out a thousand statements with the for/do/while repetition.
Deep Learning	**Analysis** (analyse)	The ability to identify the parts of the program and analyse their relationship, compare programming utilities.
		For example: identify the causes/faults of the program, identifying appropriate utilities.
	Evaluation (evaluate)	The ability to judge the quality of a program, determine how well the program works (i.e. efficiency/stability/consistency).

		For example: analyse the positive/negative features, judge which of two methods is the most effective solution.
	Synthesis (create)	The ability to rewrite a program using creative thinking (other possible/different solutions) in order to improve it.
		For example: write the method to reduce redundancy of repeated code within the program in order to simplify the structure, which makes the whole program easier to follow and maintain.

Table 12 Learning taxonomy in context of programming

2.5 Summary of the chapter

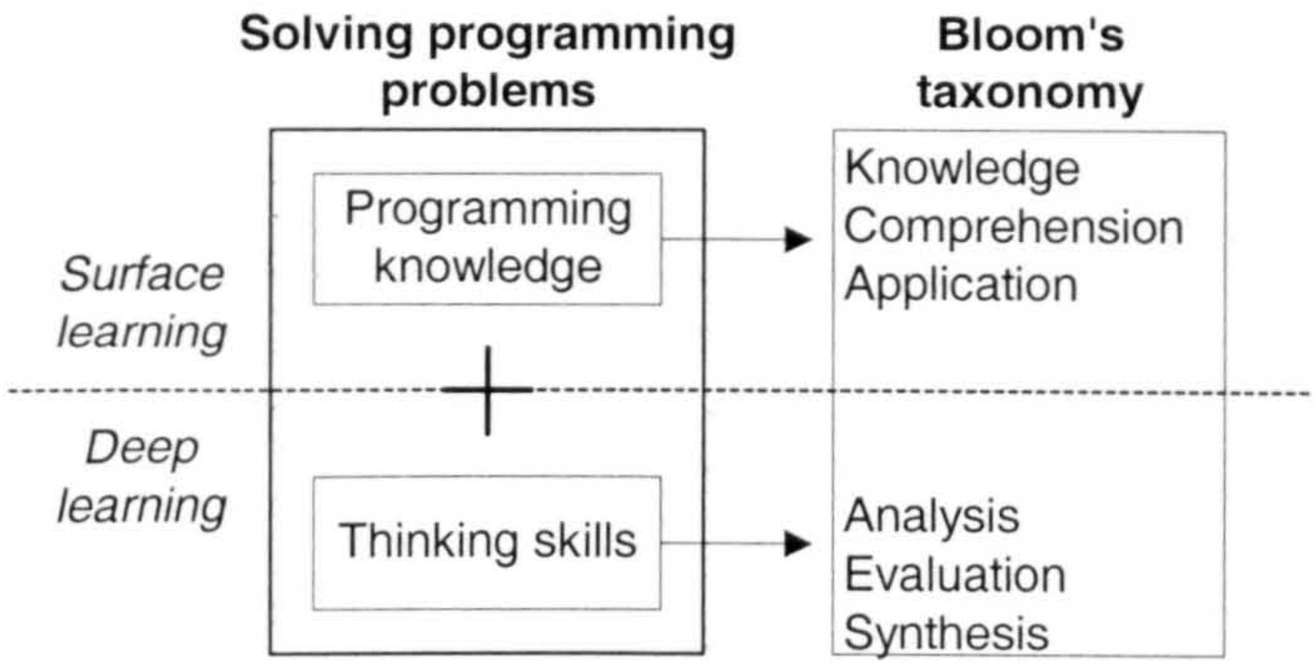


Figure 8 Bloom's taxonomy and learning programming

We have described theories of learning, and the pedagogy of learning computer programming. A new learning theory in the context of programming is proposed, which is a variation of Bloom's taxonomy map to learning programming. Although other learning theories can apply to learning programming, six categories of Bloom's taxonomy match well the development of learning skills in computer programming.

Solving programming problems requires knowledge (surface learning level), and thinking skills (deep learning level) (see Figure 8). In order to write effective programs, besides the programming knowledge, higher cognitive skills (e.g. problem solving, creative thinking, critical thinking) are required. Peer assessment is a technique using a deep learning approach that can be used to encourage the development of these higher cognitive skills (which will be

investigated in Chapter 5). Existing tools and strategies for learning computer programming are examined in the next chapter.

Chapter 3 Tools for learning programming

3.1 Overview of the chapter

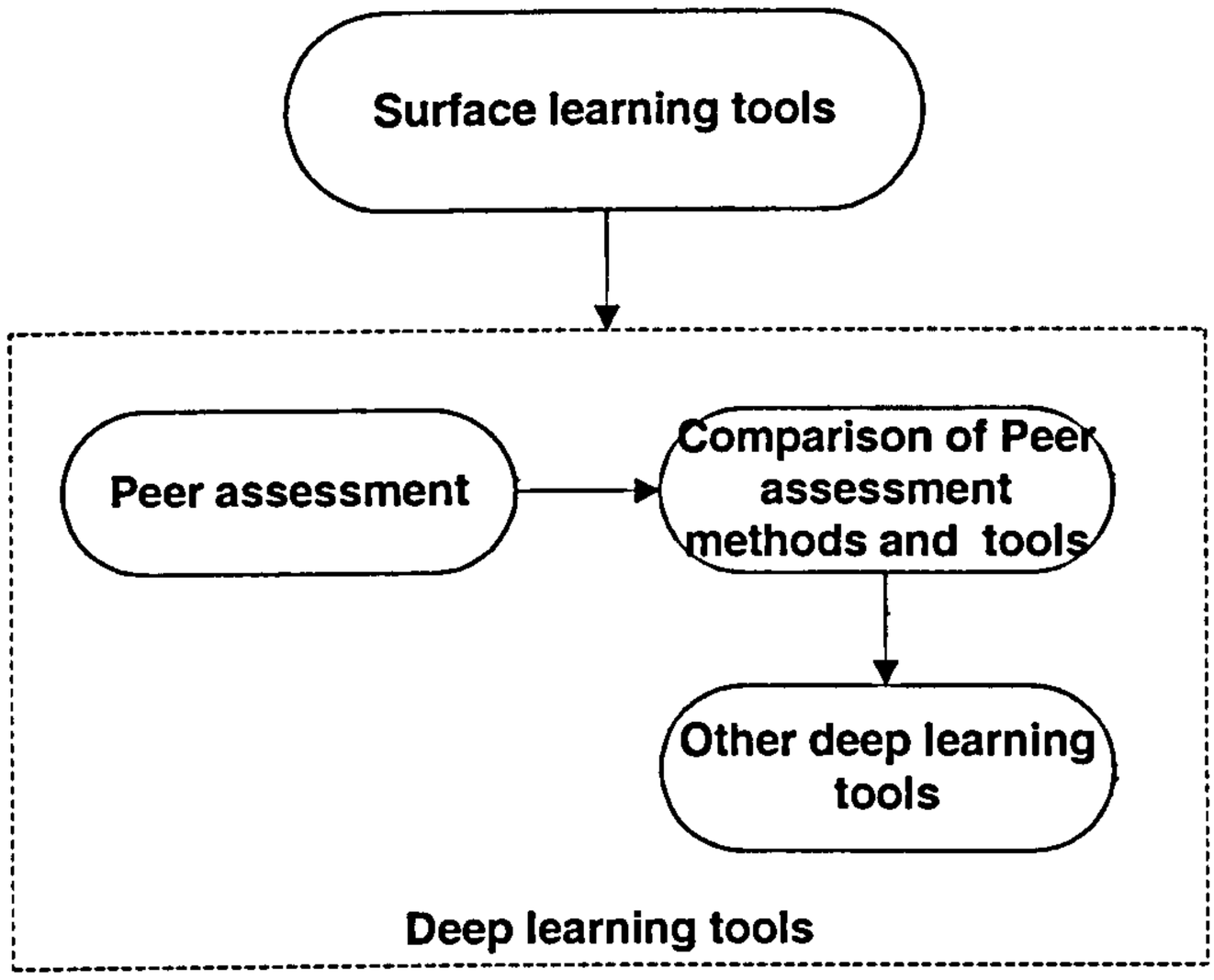


Figure 9 Outline of 'tools for learning programming' chapter

Computer programming is well known for being a difficult subject. Many tools have been created to help students in learning, especially novice programmers in an introductory programming course. However most of the tools were designed for surface learning, such as helping in program construction, compilation, testing and debugging [Forcheri1994, Deek1998]. There were not many tools well designed for deep learning to enhance students' higher cognitive skills. Surface and deep learning tools are examined, with the emphasis on peer assessment (Figure 9).

3.2 Surface learning tools

Many tools have been developed to help students in learning to program, emphasising program correctness, such as testing programs, debugging programs, interpreting errors, online tutorials, and automated marking tools. These tools can be used to encourage students at the surface learning level (programming knowledge), especially as most tools were developed for online tutorials, online submission and automated marking for large classes. Table 13

illustrates examples of surface learning tools that help in learning programming (e.g. Pascal, C++, Java, SQL).

Type	Tools	Description
Testing programs	TRY system [Reek1996]	A software package for the UNIX operating system that tests students' programs.
Debugging program	UWPI [Henry1990]	UWPI provides a graphical illustration of data structures and program source code.
	Lens [Mukherjea1994]	Lens helps students to find logical mistakes from incorrect coding, by graphically executing the code.
	VINCE [Rowe1999]	VINCE is an online tutorial tool for teaching introductory C programming. Students can see what is happening during program execution.
	Jeliot 3 [Kannusmaki2004]	Jeliot 3 is an animation tool for novices to learn Java programming, algorithms, and data structures.
Interpreting errors	Compiler error messages interpreter [Coull2003]	Common compiler error messages and solutions are provided for Java programming in a first year computing course.
Online tutorial	Capra [Verdejo1993]	Capra is an intelligent programming environment. Tutoring is based on students' knowledge level.
	Hyperex [Altamura1995]	Hyperex is an intelligent tutoring system that provides help and guidance during the learning process in a Pascal programming course.
	CUTE [Churcher1998]	CUTE provided tools to support the software development cycle, such as writing programs, submitting work for marking, and running sample assignment solutions.
	CBL [Rowe1999]	This system provided tutorials and animated demonstrations via the web.
	RoboProf [Daly1999]	RoboProf is an online teaching system, which based on solving small programming problems, relevant to computer games.
	SQLator [Sadiq2004]	SQLator is a web-based interactive tool for learning SQL (Structured Query Language). It provides multimedia tutorial material.

Type	Tools	Description
Automated marking	Ceilidh [Notting1992]	Ceilidh is an assessment and management system for programming courses.
	Get [Reek1996]	Get is an automated file retrieval system (assignment files). The online submission and automated grading programs were developed to mark programming assignments (e.g. C, C++).
	BOSS [Joy1998]	BOSS is an online submission and assessment of programming assignments system. It is suitable for summative assessment.
	CourseMarker [Higgins2003]	CourseMarker is a computer based assessment system. It provides automatic marking and detailed feedback of programming assignments.
	Submit [Venables2003]	Submit is an online submission program. It provides automated feedback for Java programming assignments.
	Tutor board system [Heaney2004]	This web-based feedback system provides a number of facilities to help tutors in marking Java programming assignment.

Table 13 Surface learning tools in learning programming

Testing program

TRY system [Reek1989]: it is an automated program tester, which allows students to run their programs against the teacher's test data. If a test fails, students are informed about the problems, then they can correct the errors and try to test again. As this system allows unlimited attempts, teachers can see the result of each attempt from the log file, which helps in assigning grades. It can be used with any programming language and any type of project. The author reported that this system might encourage students to design programs against the test data only, rather than respond to general specification.

Debugging program

UWPI [Henry1990]: UWPI stands for the University of Washington Illustrating Compiler. It was developed to help in teaching a basic programming course for first year computer science students. UWPI provides a graphical display to illustrate the data structures and program source code,

and also helps in debugging programs, therefore students can learn how a program works. "UWPI solves program illustration using compile-time pattern matching and type inference to link anticipated execution events to display events, rather than relying on user assistance or specialized programming techniques."

Lens [Mukherjea1994]: Lens was used to teach the C language. It is a visual debugging system, and also helps students to develop algorithms in animation style, without requiring any graphics knowledge or using any text coding. Lens could also be used to build rapid animated presentations of programs.

VINCE [Rowe2000]: VINCE stands for Visual Instruction for Novices in a C Environment. This tutorial tool provides a graphical environment to help in learning C programming. Students can follow the working of a C program step by step, as the tool allows programmers to view the execution of a program in detail. For example, "they can see memory being allocated when a variable is declared; watch the exact sequence of operations; observe the transfer of control to a called function." The system allows students to edit the code in a tutorial to produce their own experiments, or write their own programs from scratch and then run them using the online tutorial tool.

Jeliot 3 [Kannusmaki2004]: Jeliot 3 is a program visualization application. It is an animation tool for novices to learn Java programming, algorithms, and data structures. Students can see both loop animation and the current states of methods, variables, and objects when they execute Java programs. They can code a program, visually debug and test it by using Jeliot 3. The authors claimed that Jeliot 3 provides many benefits to learning and teaching programming, such as teaching programming concepts, helping students in doing their assignments, practising writing programming, and is very helpful in a distance learning program.

Interpreting errors

Compiler error messages interpreter [Coull2003]: This tool helps novice programmers to interpret both compiler error messages and solve problems, since it provides solutions and suggests causes of the error messages. The research revealed that most common errors are 'files not added, and incorrect case', which were recorded only when students requested help from tutors. This application is aimed at the first year undergraduate students on a Java programming course.

Online tutorial

CAPRA [Verdejo1993]: Capra is used to verify correctness of students' programs using model solutions [Deek1998]. It is an intelligent programming environment, which consists of the tutor module – checking students' understanding, the knowledge based debugger – monitoring students' activities, and the interface module – managing multi window interfaces. This system does not support creative thinking, since student's solutions are compared with the model solutions that are store in a knowledge base only [Deek1998].

Hyperex [Altamura1995]: Hyperex is an intelligent tutoring hypertext system for learning Pascal programming in a first year programming course. It consists of four features: domain representation (storing Pascal programming syntax and exercises), the student model (recording students' activities), the tutor module (determining direction), and the student interface (presenting exercises with a given difficulty level). The research revealed that Heperex is of less benefit for students who lack self organisation skills.

CUTE [Churcher1998]: CUTE stands for Canterbury University Teaching Environment. It is a software development tool with a simple graphical user interface. The main features of CUTE are file management, compiler, program executor, editor, email, and online help. The environment for teaching programming was focused to save time for learning fundamental computer use operations (e.g. file management, compilers and editors) in a UNIX environment. The author claims that CUTE not only provides benefits

to students who have no prior computing experience, but also helps them to work more effectively in the software development cycle.

CBL [Rowe1999]: CBL stands for Computer-Based Learning, and this system was implemented on the web. It provided question-answer tutorials, programming tutorials, and animated demonstrations for teaching first and second year computing courses (i.e. Java, objected oriented programming using C++). Three main features are HTML tutorial pages (short question-answer), programming tutorials (modify or add code, compile and run code, feedback to students via Web browser), and HTML image maps (animation algorithms).

RoboProf [Daly1999]: teaching introductory programming through computer games. This online teaching system provides small programming problems with gradually increasing levels of difficulty. Students receive automatic feedback and are able to resubmit a program after correcting it. This system focuses on helping students in learning the syntax and semantics of a programming language. The author reported that the main problem areas in learning programming are control flow (if/for/while) and arrays.

SQLator [Sadiq2004]: SQLator was developed to help students in learning SQL. The system provides multimedia tutorials, databases for query practices, SQL query execution, monitoring activities, status reports, feedback and interaction with the teacher through the tool. SQLator emphasises the evaluation of SQL query correctness. This system also benefits assessment and plagiarism detection. The author reported SQLator was successfully deployed in providing feedback to students.

Automated marking

Ceilidh [Notting1992]: Ceilidh stands for Computer Environment for Interactive Learning in Diverse Habitats. This system consists of two main parts, including course management and automated assessment facilities. Course management involves developing, setting up, and monitoring the running of the course. Automated assessment supports both computer

programs and multiple-choice questions. The important feature of Ceilidh system is automated assessment of programming assignments. The system provides automated feedback on the correctness and quality of programs.

Get [Reek1996]: the systems support file retrieval (Get), submission of work, and grading students' work. Get is an automated file retrieval system, which allow students to retrieve a set of assignment files with a single command. After students submit their programming assignments, the submission system checks for the validity and correctness of programs. Then tutors check for design and style of the programs. The author claims that this system saved time for routine activities, such as program testing, therefore tutors could spend more time on checking the quality of the program.

BOSS [Joy1998]: BOSS is a online submission, assessment, and course management system. BOSS consists of two parts, including students' and teachers' interfaces. Students can submit their programming assignments through BOSS, and test their programs against sets of testing data provided by teachers. Teachers or tutors can mark the style of programs and provide feedback for students' work. The new version of BOSS also includes plagiarism detection facilities for programming assignments.

CourseMarker [Higgins2003]: CourseMarker (previously call CourseMaster) developed from Ceilidh with improvements including a better user interface and more detailed for student feedback. It supports marking of programming code and diagrams. Students can develop and run their programs within the system. CourseMarker provides automatic marking, instant feedback for the programming assignments, and also detects student plagiarism.

Submit [Venables2003]: 'submit' is an online submission system, which allows students to submit their programs several times, aimed at improving their programs at each attempt. 'Submit' also provides immediate feedback on Java programming assignments, including style and correctness. For the programming style, 'submit' analyses comments and lengths of methods (should not more than 50 lines long), then 'submit' checks program

correctness by running the program against the sample input from the teacher. Beside these automatic feedbacks, students can view marks and comments from the tutors later.

Tutor board system [Heaney2004]: Tutor board is a web-based feedback system, which helps tutors in marking Java programming assignments. This system catches common programming errors, using keyword highlighting for clarity on a whiteboard. Teachers can also store assignment details and monitor the tutors' marking. Research reveals that this system provides a lot of benefit to students, especially the weak students, since the quality and rapid feedback increase students' motivation and confidence.

3.3 Deep learning tools

There are many tools that were developed to support software development only, whilst solving problems, critical thinking and other transferable skills in learning programming are ignored [Forcheri1994]. Deep learning approaches (e.g. self assessment, peer assessment, collaborative learning, problem based learning) can be used to enhance students' learning in higher education. Peer assessment is emphasised in this thesis (see motivation for the thesis in Chapter 1). We discuss peer assessment and the comparison of peer assessment methods and tools (for varieties of subjects), followed by the other deep learning tools for learning programming.

3.3.1 Peer assessment

Peer assessment involves students in marking and providing feedback on each other's work [Falchikov2001, Somervell1993]. Deep learning, such as creating new ideas, and critical judgement of a student's work, can be encouraged by the use of peer assessment [AAHE1993, Topping1998, Bhalerao2001, Fiore2001]. When students evaluate each others' work, they see how others tackle problems, learn to criticise constructively, hence developing critical thinking skills [Sluijsmans1999, Davies2000, Tsai2002].

In the peer assessment process, students are involved both in the learning and in the assessment process. Peer assessment is a tool for learning, and students can

learn through marking [Brown1997, Davies2000]. Dochy and McDowell [Dochy1997] remarked "peer assessment is not only a tool to provide a peer with constructive feedback which is understood by the peer. Above all, peer assessment is a tool for the learner himself." Lin et al [Lin2001] also state that receiving many and frequent peer feedbacks helps in better learning outcomes. From many studies of peer assessment in a variety of subjects [AAHE1993, Somervell1993, Falchikov1995, Brown1997, Brindley1998, Topping1998, Davies2000, Bhalerao2001, Fiore2001], peer assessment provides many benefits to enhance students' learning, including the following:

- encouragement of deep learning skills by making judgements and providing feedback on other student's work;
- opportunities to compare and discuss about what constituted a good or poor piece of work;
- when marking, students realise mistakes that they had made in their own work (self assessment);
- development of reflective learning, when providing feedback on other work; and
- deepening of students' understanding of the assessment process, and what is required to be achieved.

However several problems with peer assessment were reported [Falchikov1995, Orsmond1996, Davies2000, Topping2000, Ballantyne2002], including:

- students prefer feedback from experts rather than from their classmates who may be not qualified;
- students complained that peer assessment is a time consuming process;
- students do not fully understand marking criteria;
- students may be biased when marking, and may collude together;
- students who are inexperienced markers are not confident in marking other students' work.

Therefore careful design of peer assessment methods, including "assessment tasks, assessment criteria, anonymity, procedural guidelines, distribution

systems, marking procedures, and tutor re-marking" [Ballantyne2002] should be considered.

3.3.2 Comparison of peer assessment methods and tools

Peer assessment has been studied in a variety of subjects, but rarely for learning programming. In addition, most peer assessment methods are paper based. We report the peer assessment methods (paper based) and tools (electronic systems), which are deployed in a variety of subjects.

3.3.2.1 Peer assessment methods

- ***presentation*** in marketing modules [Brindley1998]. In the first step, students are told of the purpose of peer assessment, and the marking scheme is explained. Peer groups assess other groups' presentations (no individual marks are awarded). Final marks consist of marks from peers (40%) and tutors (60%).
- ***academic writing*** in postgraduate educational psychology [Topping2000]. This process is not anonymous; pairs of students who are interested in the same topic assess each others' reports, by filling in an assessment feedback form. Face to face discussion between pairs is arranged, after receiving feedback from partners.
- ***written and oral presentations*** in group work [Sivan2000]. This study proposed two types of peer assessment: intra-group (assess work of group members), and inter-group (assess work between groups). Peers assess a group project report and presentation by answering criteria questions, provided by the teacher.
- ***oral presentation*** in a communications module [Magin2001]. Marks are a combination of peers' and teachers' marks. This peer assessment focuses on the comparative reliability of peers' and teachers' marks. Individual oral presentations are each assessed by 5 students and 3-7 teachers. A4 mark sheets (questions with a 10-point scale) are provided and forwarded to the student presenters at the end.
- ***posters*** in Psychology module [Smith2002]. Setting the marking scheme by students is followed by individual marking of three posters. Mark

agreements are discussed within a group in the next step. The average mark of each poster is based on 10-14 students.

- ***group project*** [Lejk2002]. This method is a combination of self assessment and peer assessment. Students were asked to award marks for themselves and peers, in order to apply individual weightings to the group marks. This type of peer assessment aims at identifying the contributions of individual grades in group work.
- ***poster presentation and written assignment*** in Education, Business and Health faculties [Ballantyne2002]. One assignment is marked by three students. Students were responsible for rotating the assignments. Final marks for each piece of work are discussed within the group (written comments on peer-assessed criteria sheets). Then students assess their own work by answering questions on self-assessed criteria sheets. The quality of peers' marking is assessed by tutors.

3.3.2.2 Peer assessment tools

Table 14 displays the comparison of peer assessment systems, including our peer assessment system (more detail is reported in Chapter 4). These following peer assessment systems are used in essays/reports, except our peer assessment system, which is used in programming. The uniqueness of our peer assessment system is marks for 'the quality of marking' awarded by peers instead of tutors, aimed at encouragement of critical judgement skills. Group discussion is emphasised by providing an anonymous communication tool.

Tools	Assess	Distinct aspects	Features
Netpeas [Lin2001]	Project work (essay)	2 rounds of peers' marking with chances to modify work after each round	- Online submission - Assignment modifying - Assessment monitoring
CPR [Fiore2001]	Short essays	Self assessment after peer assessment, and rating each peers' documents after marking	- Online submission - Assignment-authoring tool (text editor) - Assignment library

Tools	Assess	Distinct aspects	Features
CAP [Davies2000]	Essays/reports in computing	Identify plagiarism by comparing at least given 8 web references within essay	- Web link for plagiarism purpose
Our peer assessment system	Programming assignment	Extra step – mark the initial marking by peers, aim at critical judgement	- Anonymous communication tool - Assessment monitoring

Table 14 Comparison of peer assessment systems

Netpeas [Lin2001]: Netpeas stands for Networked Peer Assessment System. It was introduced to help students in learning operating systems in computer science, and the other writing courses (e.g. English as a Second Language: ESL writing course [Knoy2001], and thesis writing course [Lin2002]). 58 students were asked to write a survey paper for their project work, and then submit it through the Netpeas system. This peer assessment process consists of two rounds of peers' marking (work is assessed by different peer groups in each round), and allows students to edit their work, based on the peers' feedback in each round. Feedback quality marks for each round are award by tutors. Final marks are calculated from both peers' marking (quality of original work) and tutors' marking (quality of feedback). Students perform this assessment in their own time, with an anonymous process. Marks are awarded for quality of feedback encouraging students to take assessor roles seriously.

CPR [Fiore2001]: CPR stands for Calibrated Peer Review. It is a web-based writing and peer review system. CPR aims at assessment of short essays for which students can import their documents to the system, or create them within the system by an assignment-authoring tool. It is an anonymous peer review system. One essay is marked and rated by three students in the first step, then they assess their own work (self assessment). CPR offers two useful features to support frequent writing assignments:

- *"assignment-authoring tool* enables the instructor to control the components that form the learning environment;

- *assignment library* provides a set of existing units so the instructor may give writing assignments without increasing his or her workload."

CAP [Davies2000]: CAP stands for Computerised Assessment with Plagiarism. This system helps students in learning through investigation of plagiarism. Students assess each others' work, and also report plagiarism by other students by checking every source (references in the essay). Students assess each others' work in their own time, and it is individual work (no discussion). The whole process is anonymous, starting from students submitting essays via email. Then they mark at least 10 essays by comparing with the given web references within essays (at least eight references) to identify plagiarism. The quality of marking is awarded by tutors. The author reports that this system could prevent cheating, since students know that CAP is used for detecting plagiarism.

The above peer assessment methods suggest that peer assessment is used for different purposes besides helping learning, such as plagiarism, individual grading in group work, and saving marking time. The features of the above peer assessment methods are summarised below.

1. paper based/electronic system
2. group/pair/individual marking
3. anonymous/non-anonymous process
4. discussion online/face to face
5. discussion before marking/after receiving feedback
6. the quality of marking assessed by tutors
7. modify work after receiving feedback and resubmit
8. multiple rounds of marking
9. marking guidelines are provided
10. marking criteria are set by teachers or students
11. practise marking
12. combine with self assessment in a group project
13. final marks are awarded by peers (students) and tutors

3.3.3 Other deep learning tools

Collaborative projects form a major strategy used in higher education to enhance students' higher cognitive skills [AAHE1993]. Students apply their knowledge and experience to solve the problem together. The following deep learning strategies are frequently found in computer science.

- *Collaborative learning* involves students working in a group to solve a problem together. It is not only developing knowledge and deep learning skills, but also cooperative methods of learning [Hartley1996, Kreijns2003].
- *Problem based learning (PBL)* is a teaching technique that uses real-world problems (large set of problems) as the stimulus, and focuses on student activity [Boud1997, Barg2000].
- *Self assessment* involves students evaluating their own studies, particularly their achievements and the learning outcomes [Boud1989].

Table 15 displays the examples of deep learning tools in learning programming with more detail below.

Type	Tools	Description
Collaborative learning	SCOOT [Craighill1994]	SCOOT is a synchronous multimedia collaboration system, which helps in developing software applications.
	PL-Detective [Diwan2004]	PL-Detective is a tool to encourage students to collaborate on assignments.
Problem based learning	SPIMbot [Zilles2005]	SPIMbot is a problem based learning system to help students in learning assembly programming with concrete tasks and a challenge robot-programming contest.
	Raptor [Carlisle2005]	Raptor is a visual programming environment for teaching algorithmic problem solving with a set of programming tasks.
Self assessment	CAP [Schorsch1995]	CAP is an automated self assessment tool to check for syntax, logic and style errors in Pascal programs.

Table 15 Deep learning tools in learning programming

SCOOT [Craighill1994]: SCOOT stands for the Synchronous Collaborative Object-Oriented Toolkit. It provides real time multimedia collaboration for developing software applications in C++, and supports group interactions. SCOOT offers shared tool control (i.e. replicating events to multiple applications), shared presentations, and process trace (changes and progresses). An activity-based locking mechanism is provided to ensure that only one programmer can modify a shared object.

PL-Detective [Diwan2004]: PL-Detective was used in teaching programming language concepts by getting students to collaborate in working on assignments. This system is for building assignments and course demonstrations. Students discuss about the language analysis and the language design assignments, with the support of PL-Detective that provides the compiler and run-time system. In the first assignment, students can run the given programs and observe the results. In the second assignment, students can select an implementation for the semantic interfaces from PL-Detective. Since, there is no single solution for the programming problem, collaboration is important. With the support of the PL-Detective system, students can discuss, reflect on their own ideas, practise critical thinking, giving the reasons within group work.

SPIMbot [Zilles2005]: SPIMbot was used to teach assembly programming by providing the environment to encourage students to program a robot. A set of *concrete tasks* is used to motivate students in learning. It starts from small to large structure assignments, then the SPIMbot tournament, a competition between the students' programs, is set with the presentation of a problem solving process ("a top-down design, followed by a bottom-up implementation"). Students are encouraged to brainstorm in solving the contest task, for which there are multiple approaches.

Raptor [Carlisle2005]: Raptor is a tool that helps students to develop algorithms without worrying about programming syntax. Students can also execute their algorithms within the Raptor environment. In this study, students use Raptor to solve three programming problems with increasing levels of difficulty. The

authors report that Raptor helps students to develop problem solving skills. Their programs are much more structured and easier to read.

CAP [Schorsch1995]: CAP stands for Code Analyzer for Pascal. It aims at helping students to analyse their own programs, since CAP provides feedback that identifies a problem and how to fix it. Model answers are also included, and diagnostics of programming errors (i.e. syntax, logic, and style errors) are focused. After analysing a program, CAP displays the number of errors, incorrect source code, and error annotation. The authors report that the quality of program, especially programming style, has increased by using CAP system. CAP also helps in saving time for marking programming assignments.

3.4 Summary of the chapter

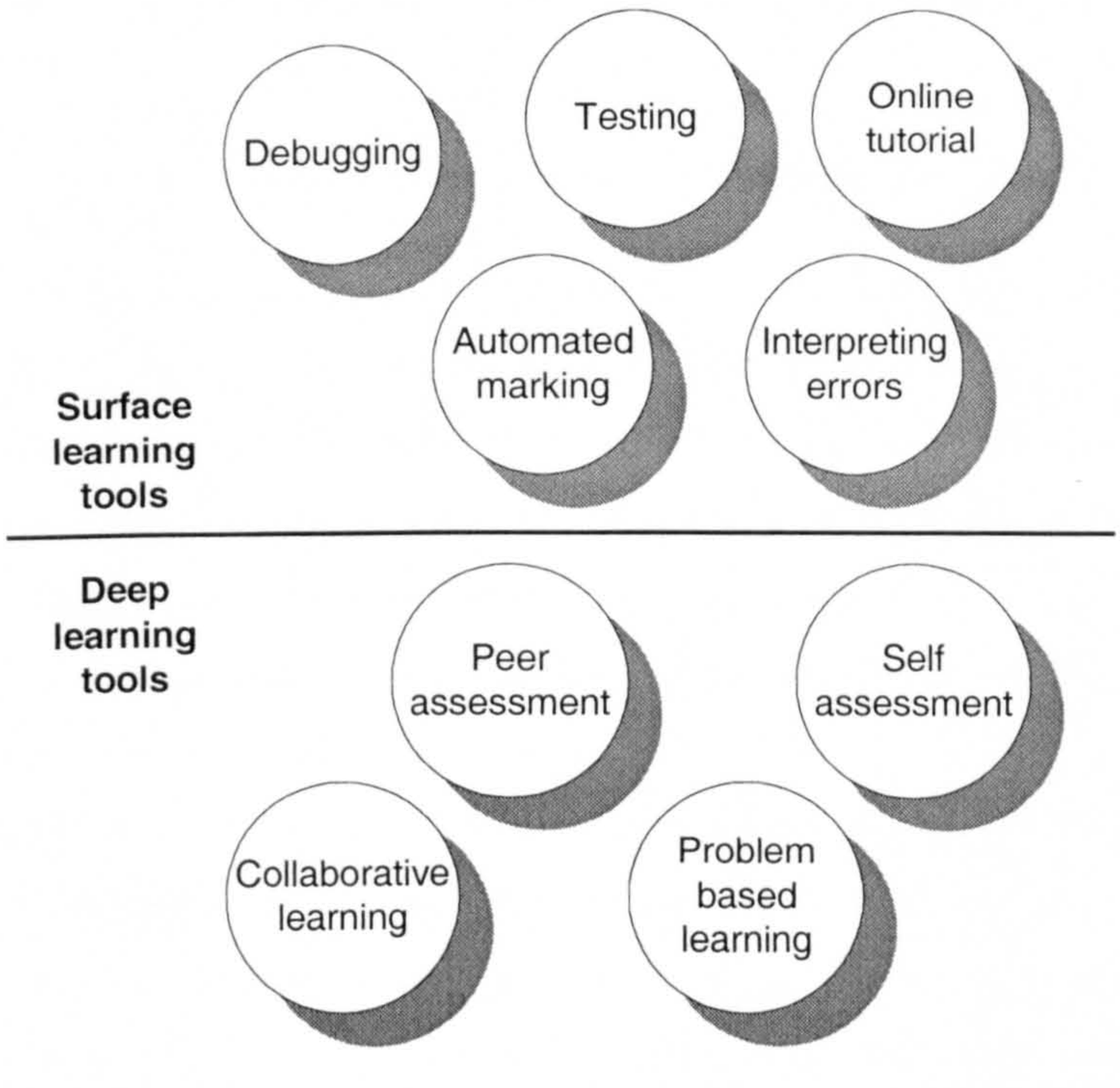


Figure 10 Surface and deep learning tools in learning programming

We have described tools for learning computer programming, with the emphasis on peer assessment. Peer assessment methods and tools in a variety of subjects are discussed. Tools for learning computer programming can be classified as surface and deep learning tools, based on their purpose: software development, or encouragement of creative thinking, problem solving and other transferable

skills (see Figure 10). Most learning tools support only surface learning, rarely deep learning. Peer assessment can be used to encourage deep learning, but most existing tools support essays and reports (e.g. Netpeas, CPR, CAP). Therefore peer assessment for learning programming is investigated.

Chapter 4 Peer assessment for programming

4.1 Overview of the chapter

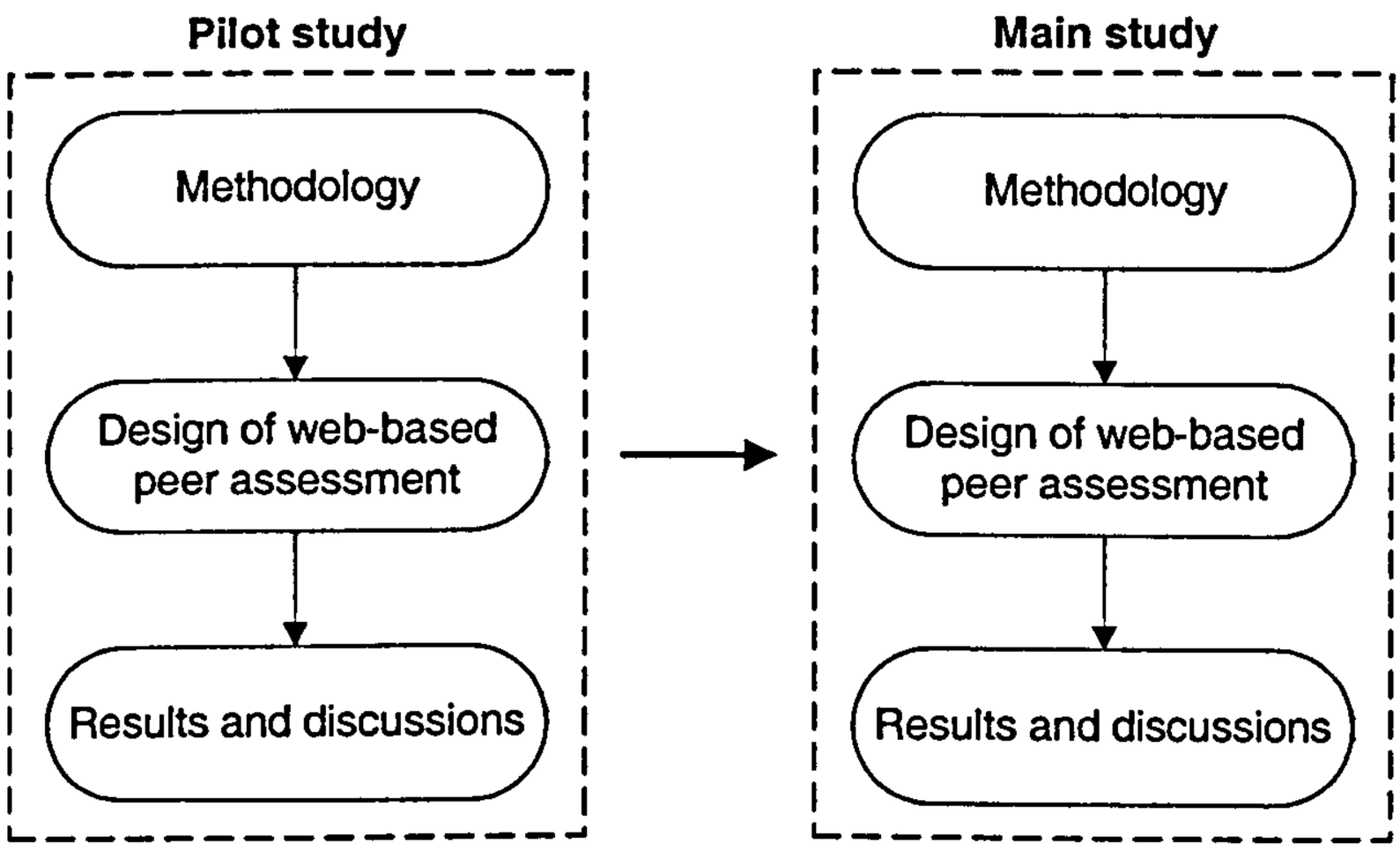


Figure 11 Outline of 'peer assessment for programming' chapter

The peer assessment experiments were performed on first year undergraduate students. The UNIX shell programming module (CS120) in the Computer Science department at the University of Warwick was chosen for this investigation. This module aims to give students a basic understanding of the UNIX operating system, and competence in programming using a UNIX shell. There are three programming assignments in this module, which students submit via the department's "BOSS" online submission system [Joy1998]. For a pilot study in 2002, the second of the three assignments was marked using a peer assessment process. After modification of the web-based peer assessment in 2003, the second experiment was performed on *all* three assignments in this module. This peer assessment was used in summative assessment, which could encourage students to take their assessors roles seriously (Magin2001, Segers2001, Smith2002).

We describe our methodology, the design of the web-based peer assessment tool, results and discussions about both the pilot and the main study (Figure 11).

4.2 Peer assessment exercise (pilot study)

The peer assessment experiment was performed on 215 first year undergraduate students (189 male and 26 female) of whom 153 students' first language is English and 62 students who are not native English speakers. In the pilot study, the second of the three assignments in the UNIX shell programming module was marked using a peer assessment process. Students learn how to design and develop programs in the shell, which is a programming language that allows programs to be written in many styles. The purpose of this pilot study is to investigate whether the peer assessment tool works well with the computer programming course.

4.2.1 Methodology

We describe the three simple stages in the peer assessment process, and the mark scheme. Each assignment is marked by an anonymised group of 3 students. Segers and Dochy [Segers2001] reported that three was the appropriate number of students on which to base the mark for a peer assessment.

4.2.1.1 Process

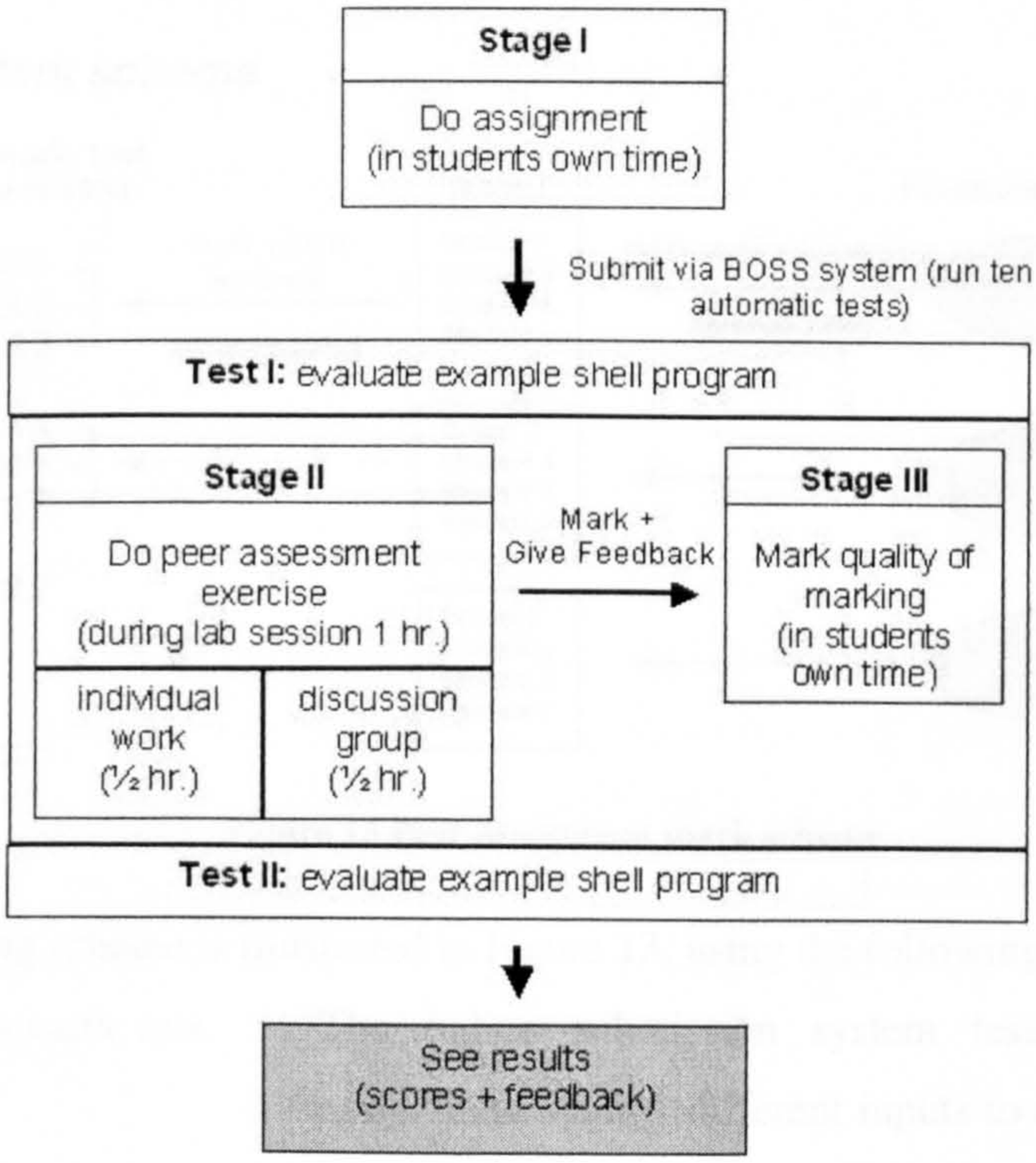


Figure 12 Peer assessment process (pilot study)

This peer assessment exercise was divided into three separate stages, as shown in Figure 12. Test I and test II (see appendix) were provided in order to measure the students' analysis skills, before and after the peer assessment, and this took place over one week. Students analyse and evaluate short example shell programs in test I and test II, which are similar in content but cosmetically different.

Stage I: Students do the assignment in their own time. Then they submit the assignment via the online submission system. Ten automatic tests are then run on the submitted programs.

Stage II: Students were divided into the small groups (three students per group). Each group consisted of students with a range of ability. Each student was assigned three other students' assignments to mark during the first half hour of a lab session. Then they discussed their marking with the other students in their group, who marked the same assignments.

Stage III: In their own time, each student marked the quality of three markers' marking. This additional stage aims to develop critical judgement and make students take marking more seriously during the previous stage.

4.2.1.2 Mark scheme

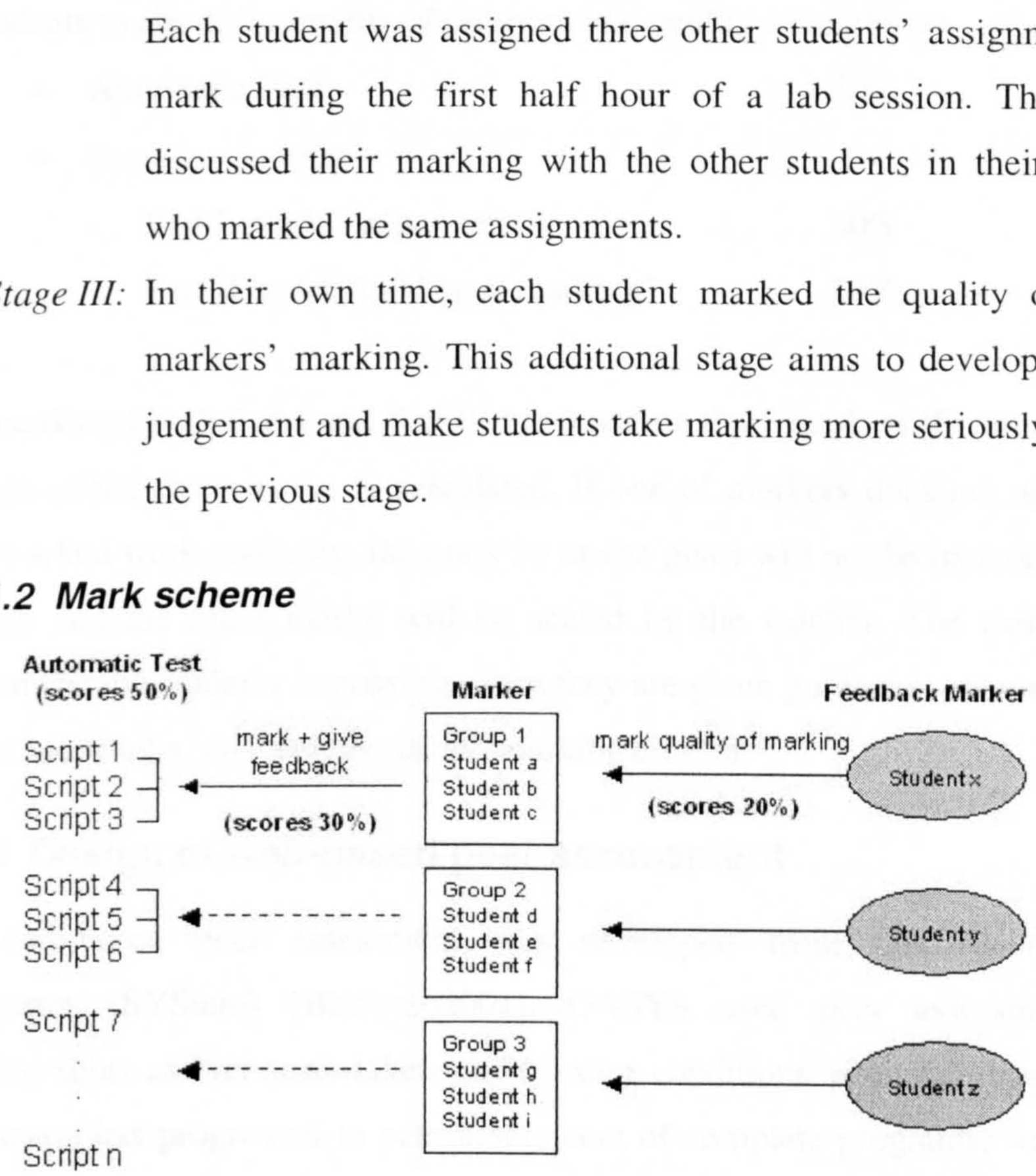


Figure 13 Peer assessment mark scheme

The marking scheme is illustrated in Figure 13, using the following definitions.

Automatic test: The online submission system tests a student's assignment against different inputs to check whether it functions correctly. Ten tests are used.

Marker:	Student marker who marks assignments.
Feedback marker:	Student marker who reports on the quality of the marking given by the three student markers.
Script:	Assignment that students submit via the online submission system.

The marks weighting depends on the purpose of assessment. In this peer assessment process, we have chosen as a mean value that 50% of the marks are awarded by some *automatic tests* (designed by the teacher to check for program correctness) and the remaining 50% are awarded by *peer assessment* (assessed by students to check for quality of program and quality of marking).

- Automatic Test 50%
- Peer Assessment
  - Part I: mark assignment 30%
  - Part II: mark 'quality of marking' 20%

Peer marks (of both Part I and Part II) are based on three markers (Figure 13); the average of the three marks is calculated. If one of markers does not appear to have marked work seriously, the mark he or she gives will not be included in the average and the other marks will be scaled by the teacher. The marking of assignments by students is possible since they are given guidance, automatic test scores and results, and well explained marking criteria.

4.2.2 Design of web-based peer assessment

The web-based peer assessment was developed from OASYS (On-line Assessment SYStem) [Bhalerao2001]. OASYS used peer assessment for marking short answer tests taken under exam conditions. Our web-based peer assessment has progressed to peer assessment of complete programs, written in the students' own time (see appendix: Difference between OASYS and our web-based peer assessment). In this section, we describe the architecture of the web-based peer assessment system, database, and web pages (both for students and the administrator).

4.2.2.1 Architecture

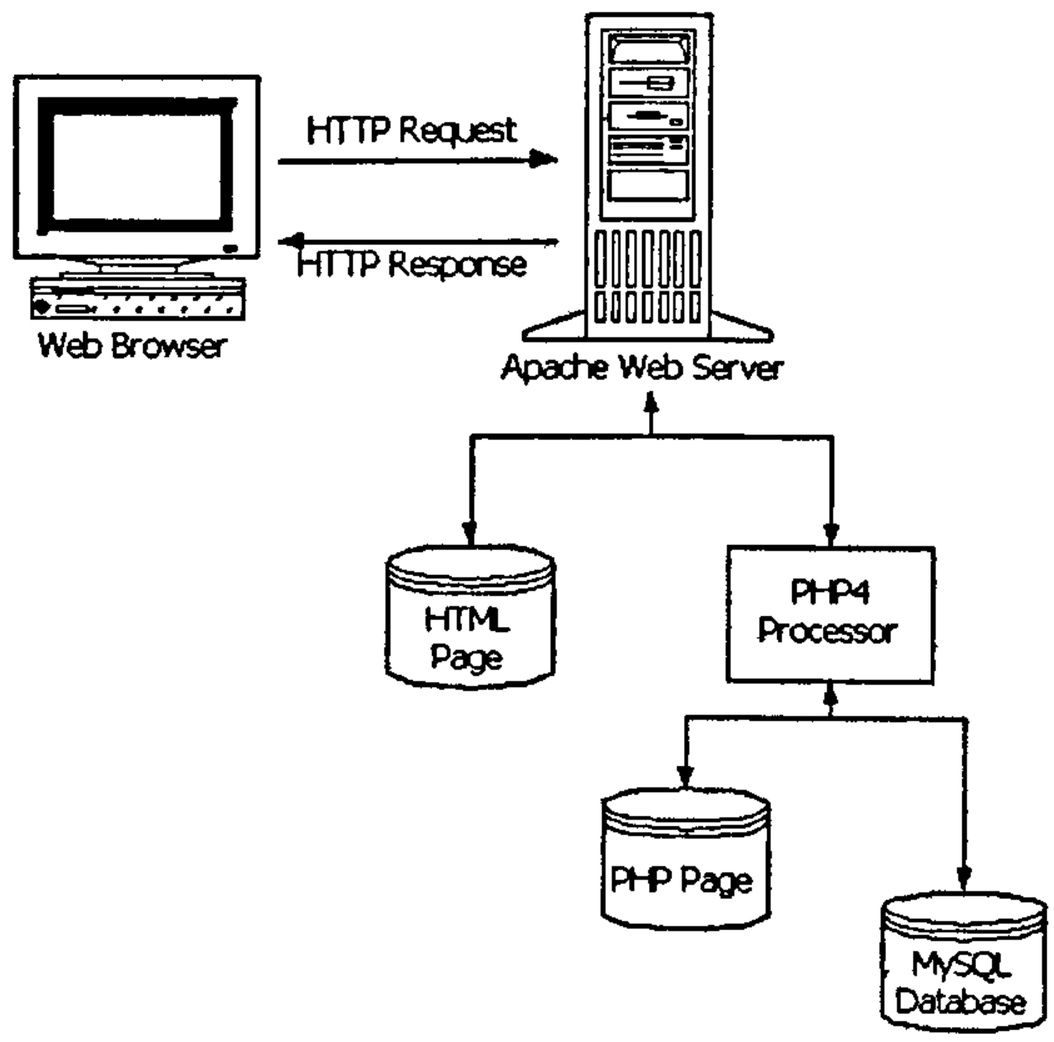


Figure 14 Architecture of the web-based peer assessment system (pilot study)

The web-based peer assessment software uses the standard combination of Apache web server, the PHP4 programming language, and a MySQL database running on a Linux platform. MySQL is one of the most popular databases on the web. PHP is also a popular programming language for dynamic web pages. Therefore both of MySQL and PHP are often considered for developing web pages. This architecture is illustrated in Figure 14. Dynamic web pages are written in PHP4 and static web pages are written in HTML.

4.2.2.2 Database

The database consists of four main parts (see Figure 15), including:

- students' assignments and their automatic test results (table: `assignment, automatic_test`);
- user details and assigned marker (table: `user, script, marker`);
- marking criteria and students' marking in both steps (table: `question, mark, mark_feedback`); and
- questionnaire (table: `questionnaire_question, questionnaire_answer`).

The `Error_log` table records all errors that occur during execution. Script id is used instead of student number for security reasons.

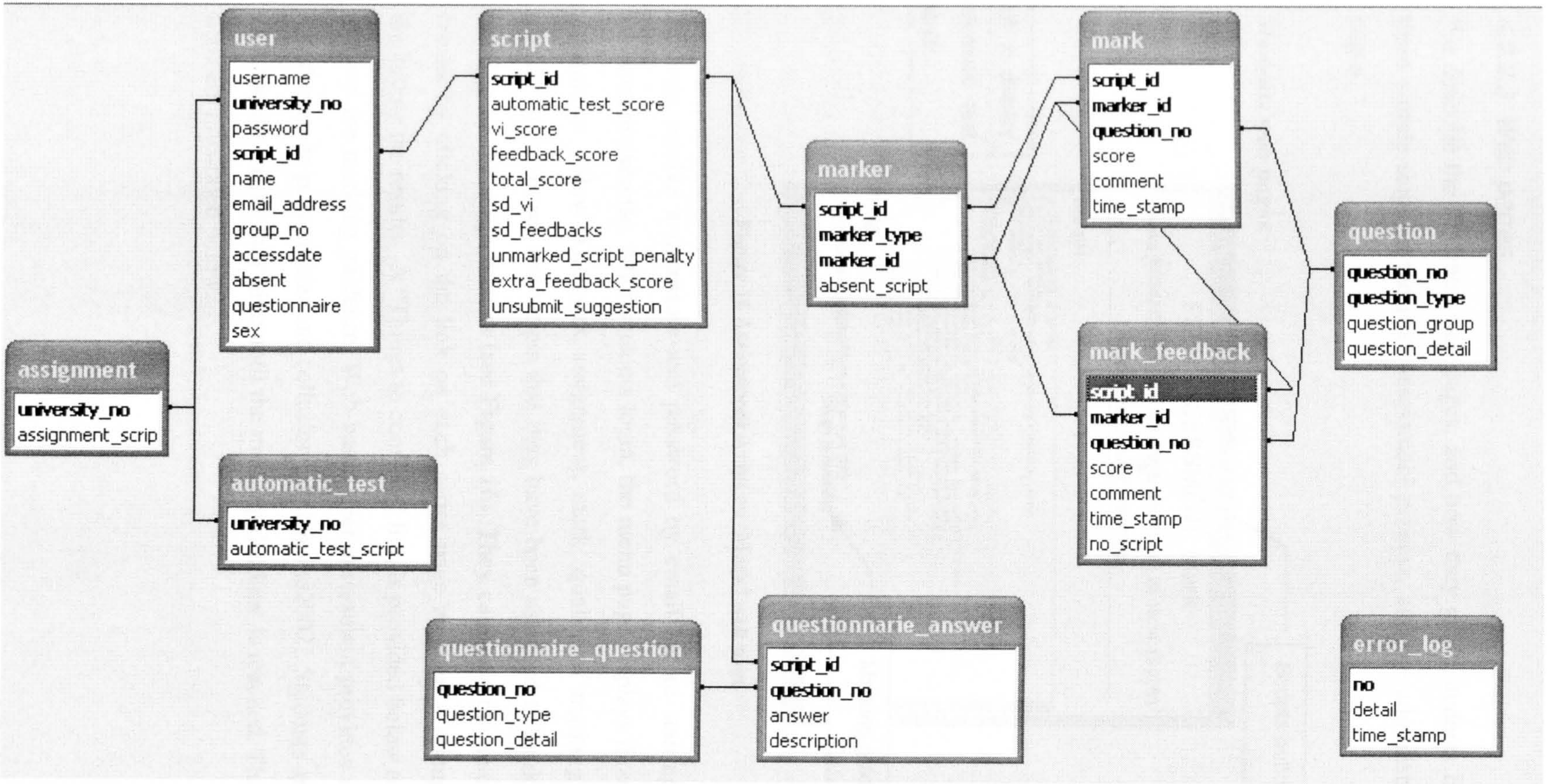


Figure 15 Peer assessment database relationship

4.2.2.3 Web pages

We describe the students' web pages, and how they guide students through the three simple steps in the peer assessment process, and the administrators web pages.

Students web pages

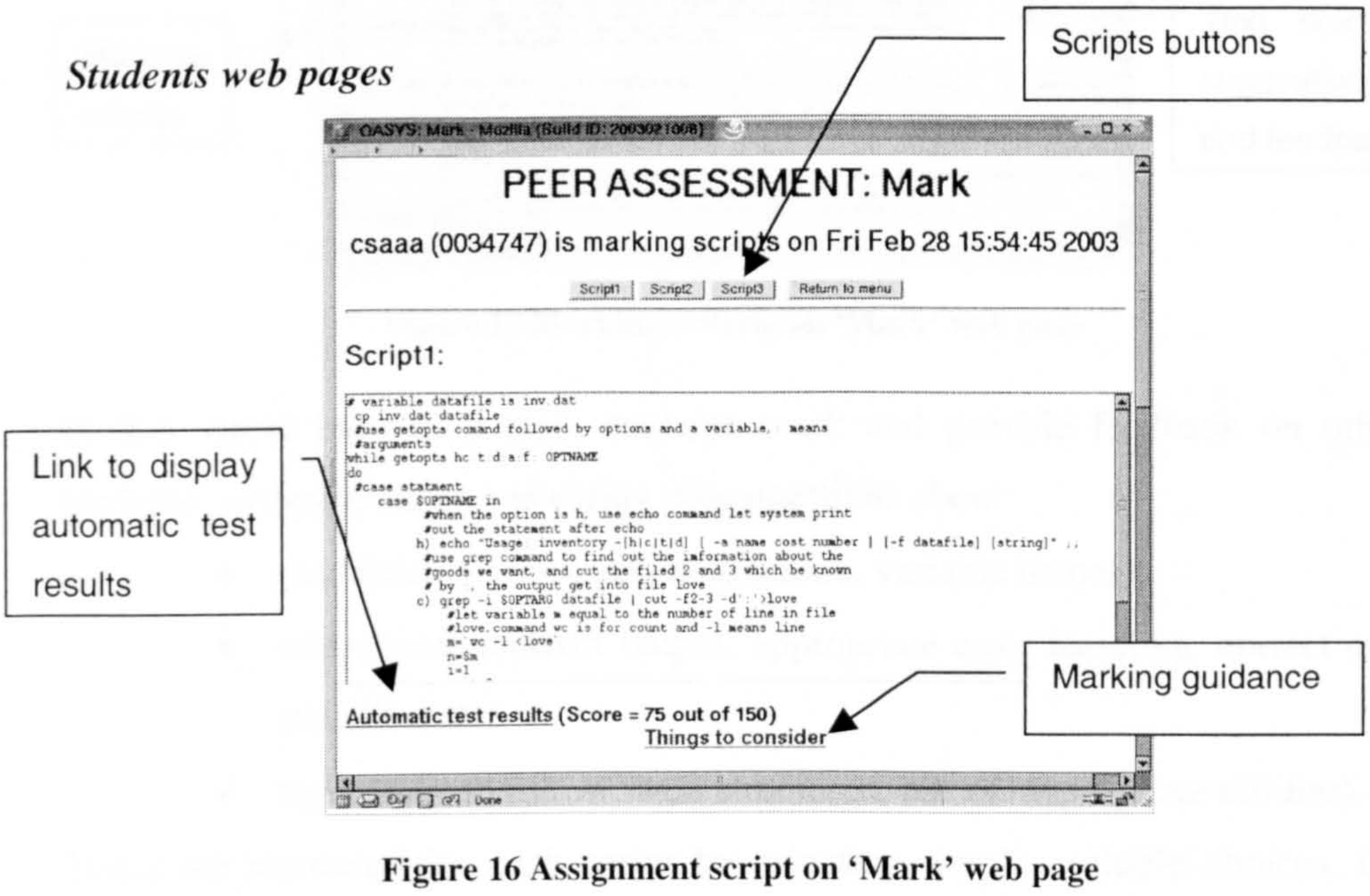


Figure 16 Assignment script on 'Mark' web page

Students receive a username and password by email before starting the peer assessment exercise. After students login, the menu page displays three steps for students to follow (i.e. mark assignment, mark 'quality of marking', and see mark). They can see the scripts that they have been assigned to mark easily by clicking on the script buttons (see Figure 16). They can view the automatic test results by clicking on the link on each script page to open a popup window displaying the results. A "Things to consider" link is provided below each script, to show the marking guidance. Web-based peer assessment provides anonymity for all users to prevent bias and collusion [Ballantyne2002]. Students are allowed to revise the marks they give until the marking deadline is reached. Three simple steps are illustrated below.

Step I: Mark assignment

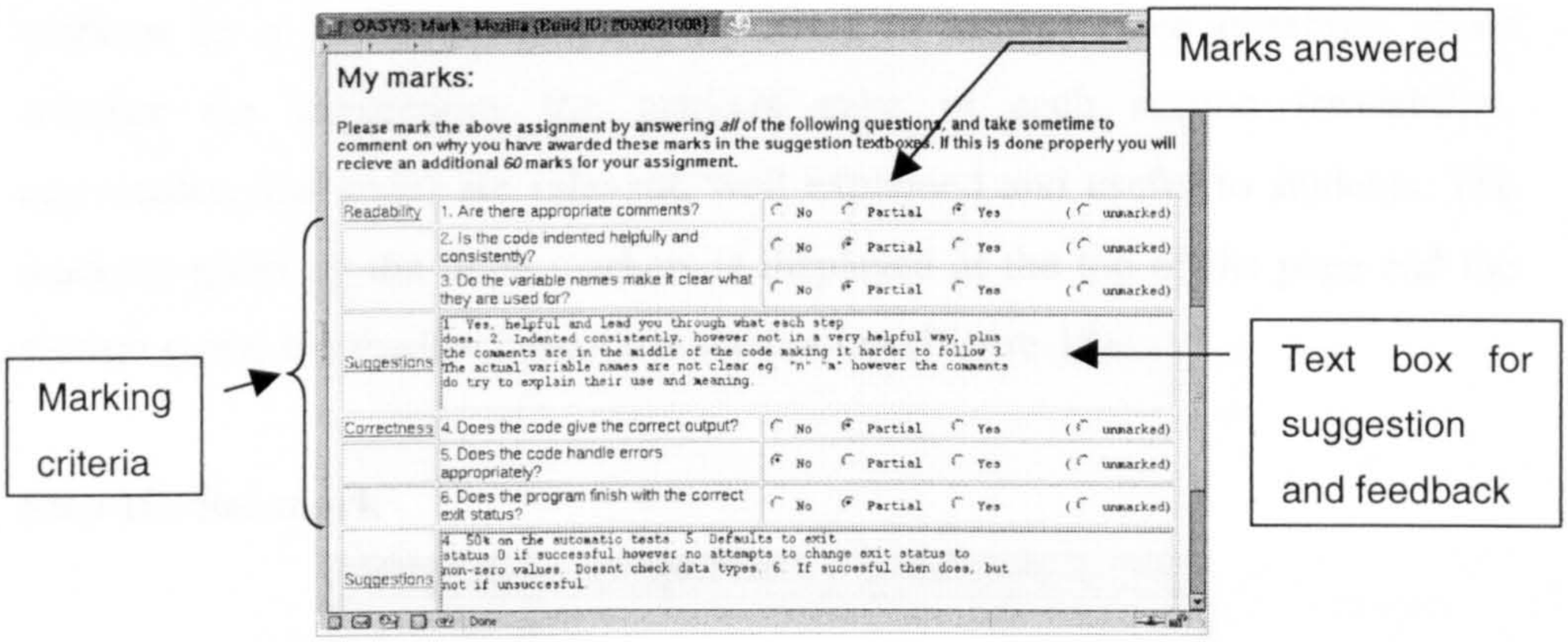


Figure 17 Marking criteria on 'Mark' web page

In this visual inspection step, students mark and provide feedback on other students' assignments by answering nine questions about:

- readability (comments, indentation, variable names);
- correctness (correct output, appropriate error handling, correct exit status); and
- style (easy to follow, well structured, use of appropriate utilities).

These are answered for each script by selecting simple multiple choices, i.e. 'No', 'Partial', and 'Yes'. The default answer is set as 'unmarked' (Figure 17). Students give a comment for each group of three questions. An explanation of the marking criteria is provided for each group of questions by clicking on the links on the left.

Step II: Mark 'quality of marking'

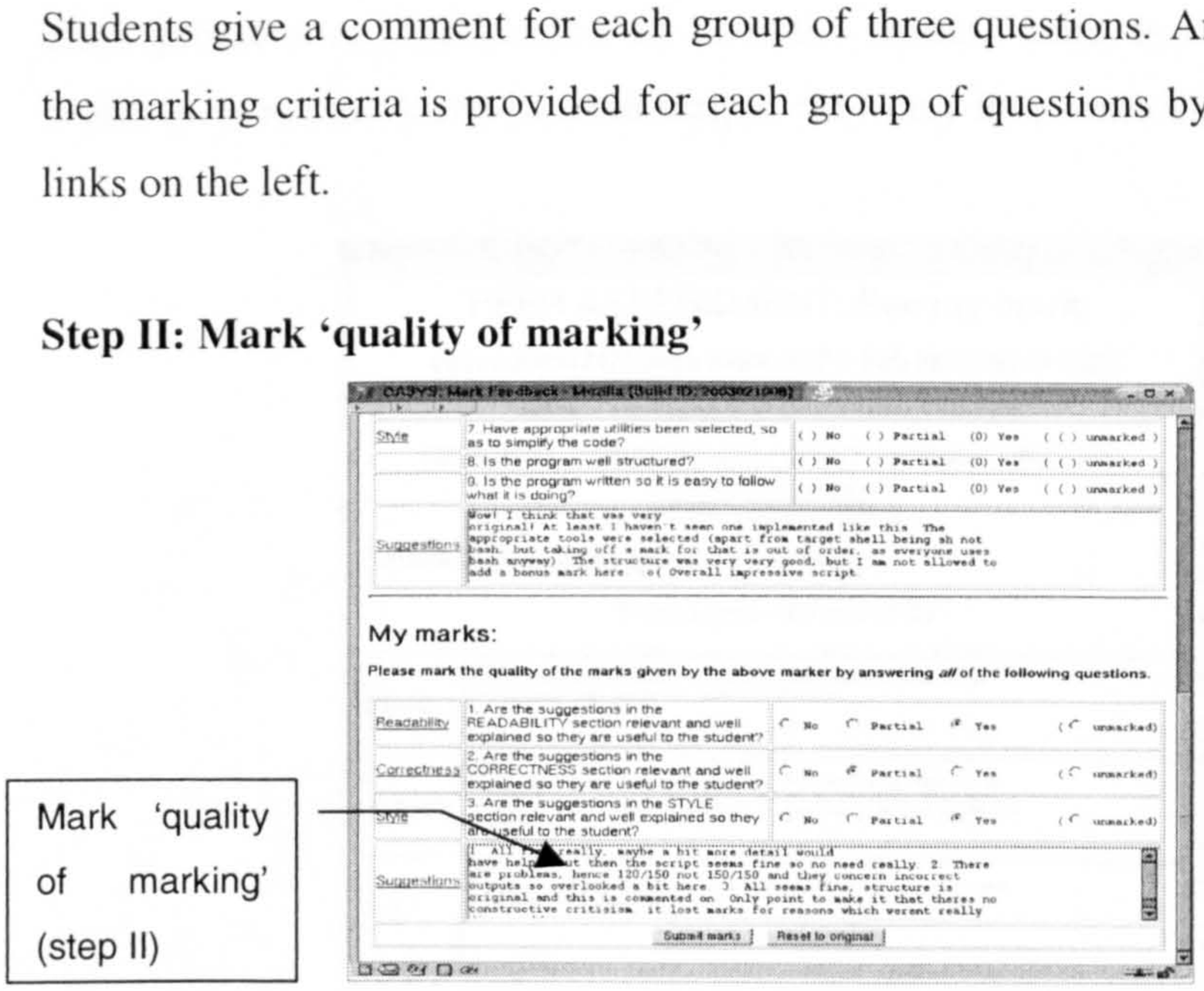


Figure 18 Mark 'quality of marking' web page

In this step, students mark the 'quality of marking' given by each of the three markers on a particular script. They need to answer three questions about whether the suggestions the markers gave in each section (readability, correctness, and style) are relevant, well explained and useful to students. The marking given by the three markers is displayed at the top of the page and the student enters the feedback marks at the bottom (Figure 18).

Step III: See mark

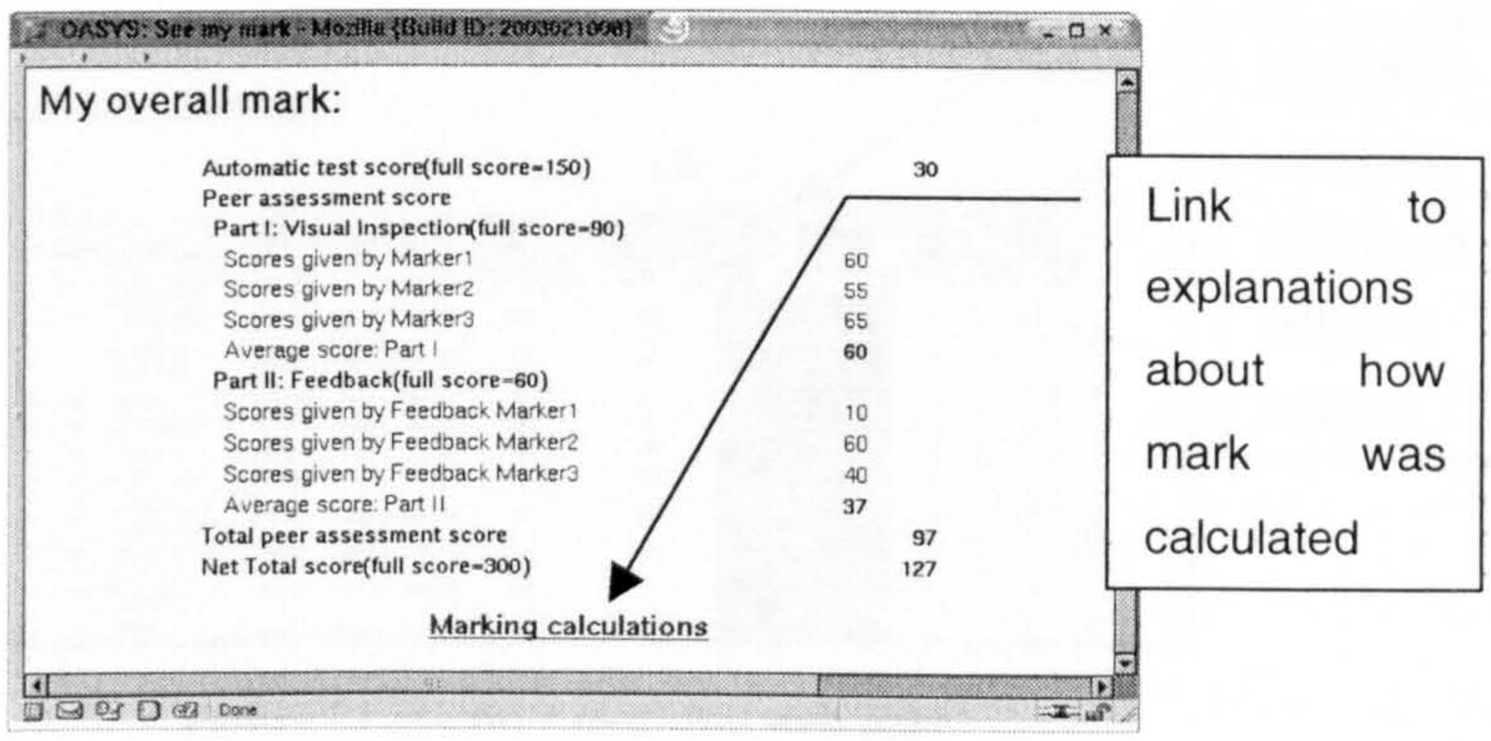


Figure 19 See mark web page

In this final step, students can see their mark from both the automatic test and the peer assessment (Figure 19). A 'Marking calculations' link at the bottom of the page provides an explanation of how the overall mark is calculated. If the students do not mark any of three scripts, they may lose some marks.

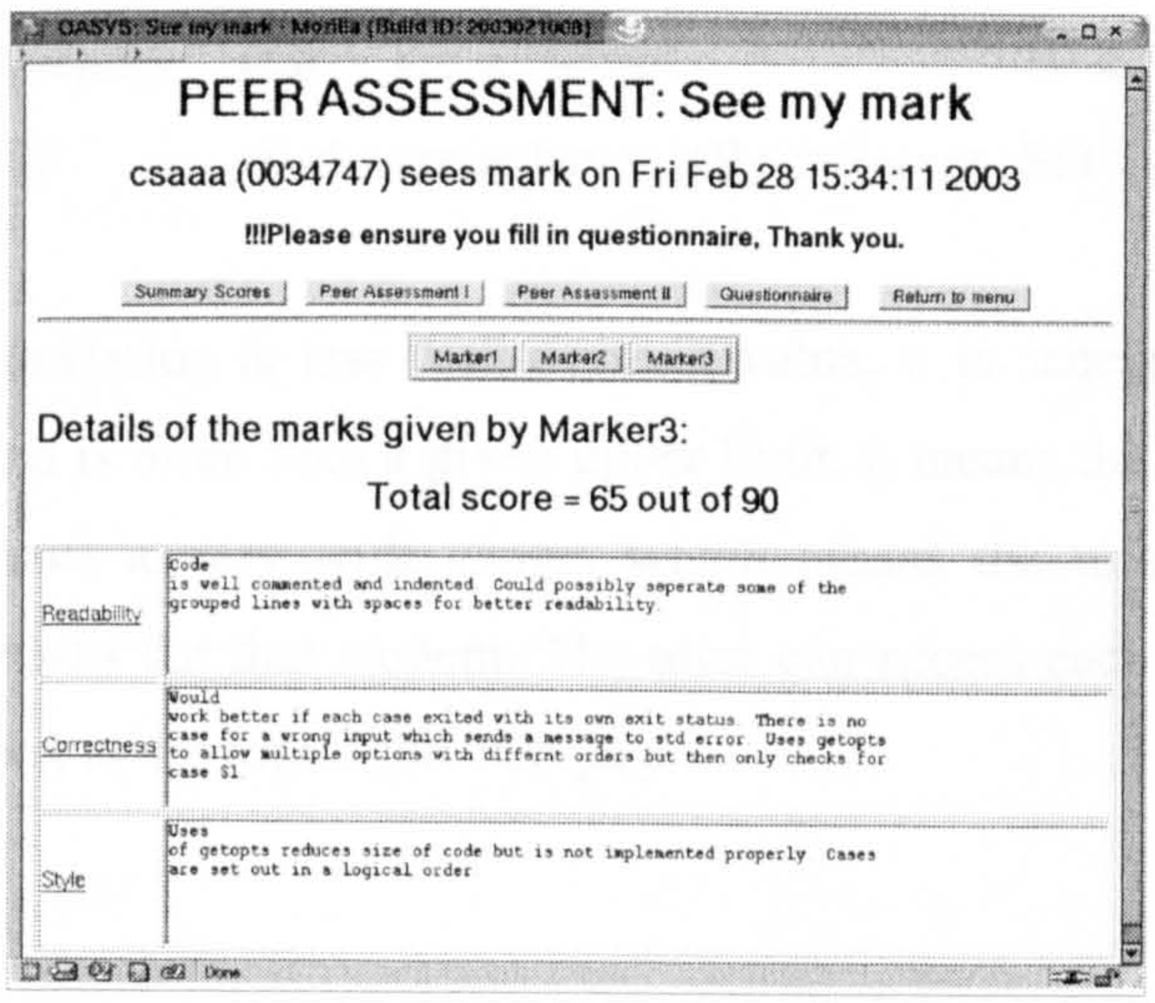


Figure 20 Feedback from peer on 'See mark' web page

The full mark and comments that the three peer markers gave the student's assignment are also available (Figure 20). This also includes the full mark that they were given based on the quality of their own marking.

Administrator web page

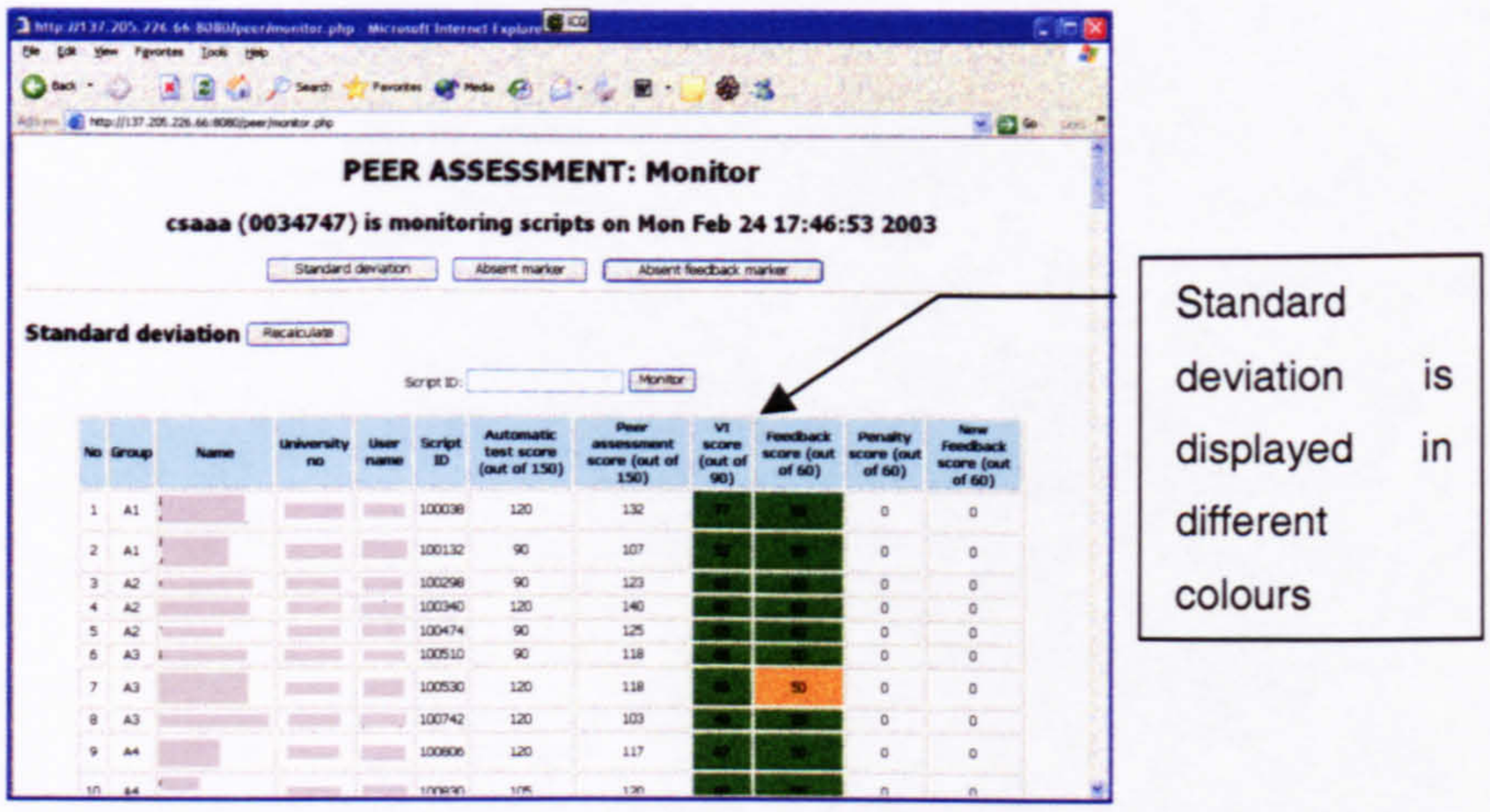


Figure 21 Monitor marking web page

Figure 21 illustrates the 'Monitor marking' web page, which reports the students' marks and any absent markers, and is only available for tutors and administrators. The highlighted columns show the standard deviation (in three colours: green, orange, red) of the three markers for both Step I and Step II in order to know how spread out the marks are.

Low SD (green)	marks are not much different	SD <= 10
Medium SD (orange)	marks are different	10 < SD < 20
High SD (red)	marks are substantially different	SD >= 20

If the standard deviation is less than a preset value, it is acceptable, but if the standard deviation is more than a given upper limit, it means the marks from the three markers have a very wide range, which means the tutor may have to reconsider the marks for that student. The tutor can access each script by using the 'Script ID' box at the top of the web page.

4.2.3 Results and discussions

4.2.3.1 A pre-test and a post-test

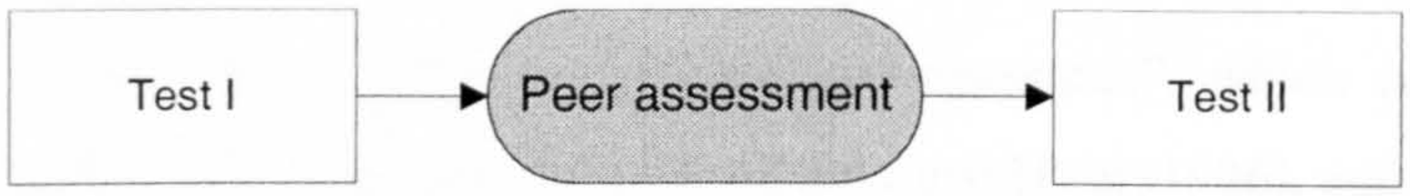


Figure 22 Pre-test and post-test in pilot study of peer assessment

A pre-test and a post-test (see appendix: Test I and test II) were provided in order to evaluate student progress (Figure 22). Students analysed and evaluated short example shell programs (without any guidance) in test I and test II, which are essentially identical in content but cosmetically different. The two tests were taken one week apart, and the peer assessment exercise took place during that seven days period, each student being allocated a one hour lab session in which the exercise was performed. The *numbers of times* that students commented on various aspects of the code were counted. Table 16 illustrates the comparison between the numbers of times that students commented on the programming aspects in test I and test II, as an effect of the peer assessment exercise.

N=180

Programming aspects	Test I	Test II	Difference
Readability			
- appropriate comments	174	174	0
- code indentation	52	75	23
- variable names	106	118	12
Correctness			
- correct output	50	57	7
- handle errors	14	34	20
- exit status	14	32	18
Style			
- appropriate utilities	14	36	22
- program structure	55	93	38
- easy to follow	21	36	15

Table 16 Numbers of times that students commented on the programming aspects

Results from Table 16 suggest that students were able to characterise aspects of a program more finely after they had been through the peer assessment process, especially 'program structure'. The number of times that students commented on 'appropriate comments' is high. This may be because the example shell programs

in both test I and test II had been sparsely commented. We analyse the pre-test and post-test results to find out the significant change in students performance, as an effect of the peer assessment exercise. *Sign test* makes use of "the counts of the direction of any differences between two measures", which is useful for detecting significant change in pre-test and post-test [Peers1996], and the pre-test and post-test analysis according to *sign test* is presented below.

1. Hypothesis H_0: *test I and test II results are not significantly different*
   H_1: *test I and test II results are significantly different*
2. 5% significance level
3. Sign test formula $(s_+ - f)^2 >= 4n$
   s_+ = 8 (the number of designated + signs)
   f = 0 (the count of - signs)
   n = 9 (the number of matched pairs)
4. calculate $(s_+ - f)^2 = (8 - 0)^2 = 64$
5. If $(s_+ - f)^2 >= 4n$ then reject the null hypothesis (H_0)
6. 64, which is > 36, (4 * 9), the null hypothesis is rejected

According to *sign test* results, the performance of students in test I and test II is significantly different. As most of the numbers of times that students commented on each programming aspects in test II are higher than test I, it suggests that after students had been through the peer assessment process, their analysis skills in programming are improved. However we are aware of other factors that may cause this improvement during that one week, such as students have done more programming on their own, and their performance might have improved anyway, and this may form the basis for further investigation. Results from the online questionnaire and interview are reported below.

4.2.3.2 Questionnaire results

At the end of the process, each student was required to fill in a detailed online questionnaire. Table 17 illustrates the number of male vs. female, and English vs. Non-English students think about the peer assessment. The following results suggest that the exercise has been beneficial.

N = 215

	Male (n=189)	**Female (n=26)**	**English (n=153)**	**Non-English (n=62)**
Understand what peer assessment is about	186	26	151	61
Understand the marking criteria in the exercise	179	26	144	61
Realise mistakes in their own answer when marking assignments	131	21	101	51
Discuss with your group when marking assignments	145	19	112	52
Feel comfortable when assigning marks	122	9	89	42
Satisfied with mark from the peer assessment	125	15	106	34
Recommend the peer assessment to friends as a way of learning more	104	18	77	45

Table 17 Questionnaire results in pilot study

Male vs. female students

The majority of students registering on this course are male (87.91%). Almost 100% of students understand both the peer assessment and the marking criteria. 81% of female students realise mistakes in their own answer when marking assignment, but only 69% of male students realise it. Nearly 80% of students discuss with their group when marking assignments. Responses from the questionnaire revealed that 65% of male students feel comfortable when assigning marks, which is more than the female students (35%). More than 50% of students were satisfied with their marks from the peer assessment and would like to recommend the peer assessment to friends as a way of learning more. Therefore most of the male and of the female students had positive responses to peer assessment.

English vs. non-English students

The majority of students registering this course are native English speakers (71.16%). Almost 100% of students understand both the peer assessment and the marking criteria. More than 80% of non-English students realise mistakes in their own answer when marking assignment, and discuss with their group when marking assignments more than English students. Even though the non-English

students feel comfortable when assigning marks more than English students, 55% of non-English students were satisfied with their marks from the peer assessment, less than English students (69%). However nearly 80% of non-English students would like to recommend the peer assessment to friend as a way of learning more, but only 50% of English students would like to do this. Therefore responses from questionnaire revealed that non-English students have a more positive idea about the peer assessment than English students.

4.2.3.3 Interview results

The results from interviewing 18 students yield the following information with the students' quotations support it.

1. The peer assessment helped students to understand more about the assignment.
   - *"When marking and writing suggestion, helping me understand how it suppose to be.*
   - *We go through quite detail how they done it and make sure you could understand.*
   - *Someone said you could be more compact which I'd picked something up from other people program as well. So that was good, kind of circular. Someone picked the same thing up so good and other person had pointed some things out that the specification hadn't point out very clearly and agreed with me so that was good."*
2. The peer assessment started students thinking about what was wrong and what worked well with their own work.
   - *"After I finish marking, I want to see where I went wrong for next time to improve my work.*
   - *When I compare my answer with others, I realise how I do wrong in my assignment.*
   - *Peer assessment help me to remind what I forget and what I should have in my work."*
3. Seeing different ways of solving programming problems helps students write better programs.
   - *"In assignment 3, I change the style, structure and make code clearer.*

- *I learn from everything I read. I pick up different techniques even shorter way of getting around the problem than my and code layout clearer to read than my.*
- *Lots of different ways doing the same thing, using different utilities, different part of computer language, so when marking, when doing peer assessment, some people use different things to me. I don't know how to use it, but when I see it from screen. I can learn from this quite well."*

4. Seeing a lot of programming mistakes helps students test programs.
   - *"It helps me test programs and make sure I don't do it.*
   - *It helps me to review my program that might have problems.*
   - *It helps me to test program and remind me don't do that mistake again."*
5. The peer assessment helps students to evaluate the quality of their own programs.
   - *"I have a chance to see how other people approach to do it, how efficient they manage to, how efficient of their argument as well, which is more effectively than myself.*
   - *I can compare my work with other and see the better one and worse one, which help me to evaluate my own work.*
   - *I'd like to know how people doing in the course. I always think I doing badly. In peer assessment, I can see how other people work and I can compare it."*
6. Peer assessment helps students learn programming more than traditional assessment.
   - *"Traditional assessment, once you hand in, you just forget about it. You stop considering what it was. Two weeks time, you probably don't remember what assignment is about. But in the peer assessment, you get the chance not only review, you have chance to think about it more.*
   - *In traditional assessment, you can hear only what is wrong, but don't know how to make it right. In peer assessment, I see how other achieve it, know how I can do it better myself.*
   - *You know that how many you got right and wrong from automatic test, but in peer assessment, you have chance to review your work and compare it, get comment and suggestion. It helps me think about this next time when I write the program."*

4.2.3.4 Discussions

The students' responses generally appear to support our view that students can learn from each other through this process. Most students seemed satisfied with their marks (with careful tutor monitoring). Although this pilot study produced positive results, problems were reported, such as:

- *face to face discussion* may not be appropriate, because of the time limit (half an hour in lab session); students were feeling intimidated about asking questions; also the absence of some members of some groups;
- *marking scale* is too small (yes, no, partial), it caused students to have difficulty in making decisions;
- *automatic tests scores* should not be displayed, because many students marked other students' programs based on the automatic test scores only;
- *line numbers of the code* should be displayed to help students to associate comments with particular lines of code;
- *a tutor comments* feature is required to allow tutors to add comments for scripts for which peers' marks are significantly different;
- *practising marking* should be provided, because students are not familiar with the peer assessment exercise, and to help them to increase the confidence in marking.

4.3 Peer assessment exercise (main study)

Many academics refer to peer assessment as "peer grading", "peer marking", "peer rating", or "peer review" [Lin2001]. In this experiment, we changed the name from peer assessment to *peer review* in order to encourage students to help each other in learning by discussing and commenting on other students' programs, instead of feeling like they are assessing other students' work. This investigation was performed on 213 first year undergraduate students, enrolled on a first year UNIX programming module in 2003. Students mark and provide feedback on *three consecutive assignments*. The purposes of performing this experiment were to investigate whether peer assessment promotes higher cognitive skills, and to measure the accuracy of this tool. The changes we made in both the methodology and the design of the web-based peer assessment are described below.

4.3.1 Methodology

Peer assessment was changed to an entirely anonymous process. Three assignments in the UNIX programming module were marked using peer assessment. As practising marking should be allowed [Ballantyne2002], the first assignment (weighting only 5%) allowed students to practice marking and to be familiar with using the web-based peer assessment system. The last assignment was independently double-marked by two module tutors, to provide an expert reference against which the marks awarded through the peer assessment process can be compared. The marking scale was also adjusted to offer appropriate choices for marking.

4.3.1.1 Process

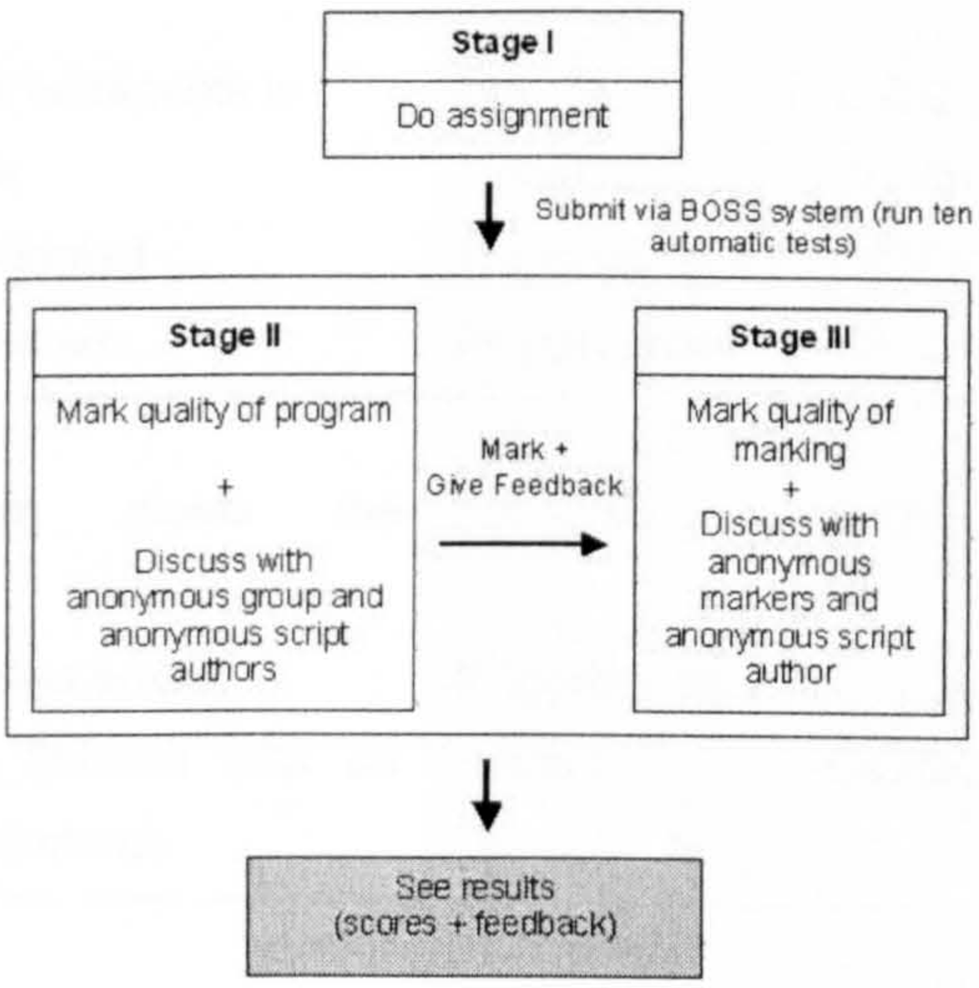


Figure 23 Peer assessment process (main study)

Discussion is an important factor in the peer assessment process and should be allowed in the marking process [Lin2001, Andriessen2003]. In the former peer assessment experiment, students marked and discussed with their group within a one- hour lab session. Some students hesitated to talk to each other for a variety of reasons, including confidence, shyness, and other cultural factors. Some arrived late for the session, some of them wanted to leave early, and some of them felt the seating arrangement was not conducive to discussion. In order to overcome the problems, we have developed a new anonymous online discussion tool to enable students to communicate anonymously when not collocated.

Students could discuss anonymously in both stages II and III in a variety of roles (i.e. script author, marker, and feedback marker) (see Figure 23).

- *Stage II - mark quality of program*: in this step, students can discuss their marking with the other students in their anonymous group who marked the same scripts. In addition, the markers can ask questions about those scripts with the anonymised script authors.
- *Stage III - mark quality of marking*: in this step, students can discuss with three anonymous markers the reasons for the feedback they gave (in the previous stage), and can ask the anonymous script author questions about the script.

4.3.1.2 Marking criteria

Readability			
- The number of comments is	Low	○○○○○	high
- Comments are	Unhelpful	○○○○○	helpful
- The code is indented	Inappropriately	○○○○○	appropriately
- Identifier names are	Inappropriate	○○○○○	appropriate
Correctness			
- The program meets the specification	not at all	○○○○○	completely
- The code handles errors	Inappropriate	○○○○○	appropriate
- The program finishes with an appropriate exit status	Never	○○○○○	always
Style			
- The utilities have been selected	Inappropriate	○○○○○	appropriate
- The program is structured	Poorly	○○○○○	well
- Overall, following what the program is doing is	Hard	○○○○○	easy

Figure 24 Marking criteria in peer assessment for programming

We initially chose to simplify as much as possible the marking criteria and the range of choices available to the students, supported by information from our pilot study. There were only three choices in the pilot study for each marking category, i.e. 'No', 'Partial', and 'Yes' (see appendix: Marking Criteria for peer assessment for programming). However, we found that students felt uncomfortable when assigning marks because of the small number of choices.

Therefore the marking scale was adjusted using a 5-point Likert scale (Figure 24), to help students in marking an accurate and fair judgment [Miller2003].

4.3.2 Design of web-based peer assessment

We have extended the system by developing an Anonymous Communication Device (ACD) to encourage interaction with others, which is a key element in fostering deep learning [Biggs2001]. This new web-based peer assessment which includes the ACD has advantages over ordinary peer assessment because students can be more critical (due to the anonymity the system provides) [Wen2003], they can discuss online and/or leave offline messages, the processing of the marks is automated, and the lecturer can easily monitor the marking and conversation.

4.3.2.1 Architecture

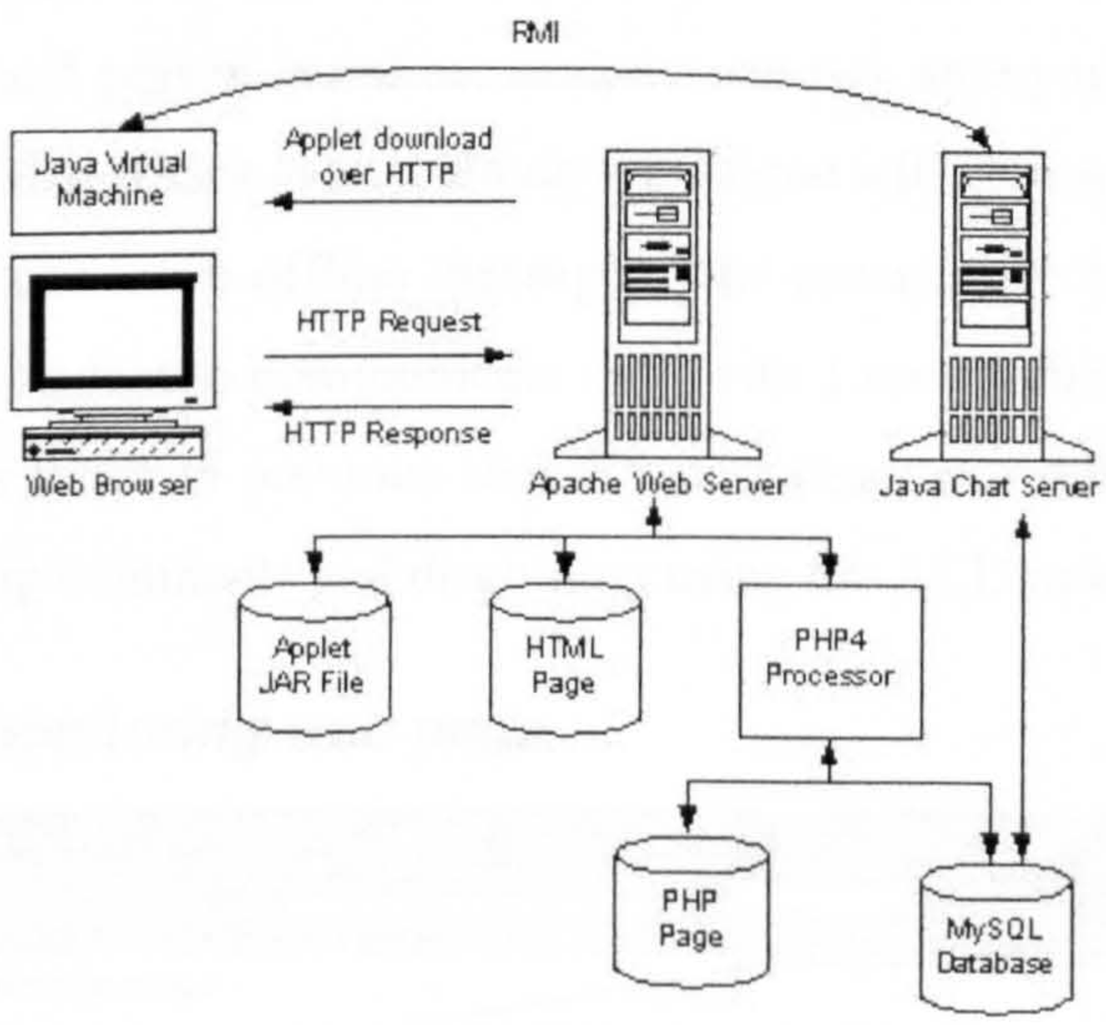


Figure 25 Architecture of the web-based peer assessment system with ACD

Web-based peer assessment is implemented using an Apache server enabled with PHP4 and accessed using a client Java applet for the ACD (see Figure 25). Students' responses to the questions forming the marking criteria were recorded in a MySQL database. The ACD program itself is downloaded via http. Once the program has been downloaded, the applet creates its own connection to another server program, the chat server.

4.3.2.2 Anonymous Communication Device

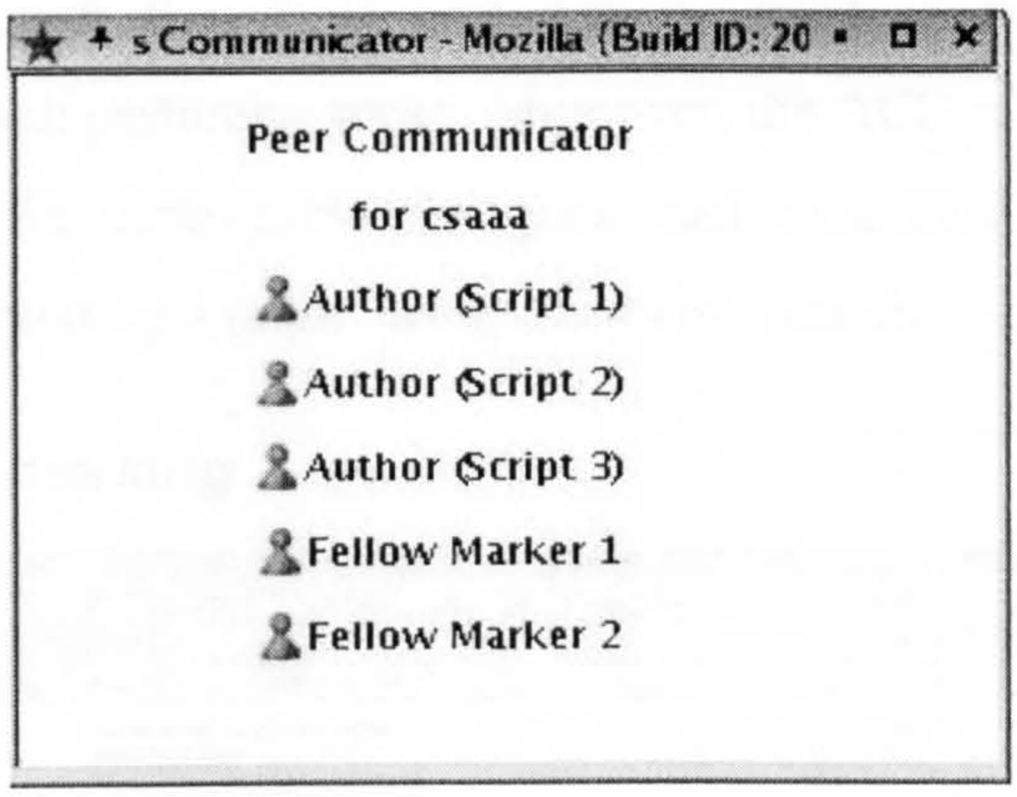


Figure 26 Anonymous communication applet

After students login and access the first step – the ACD window pops up automatically (Figure 26). It displays the 3 script authors of the assignments that students will mark, and 2 other fellow markers who mark the same assignments. If anyone of these 5 person is online, students can talk anonymously by clicking on the picture of that person, and a window dialogue will appear. If that person is offline, students can leave offline messages. The second step is similar, but the ACD allows the student to communicate only with 1 script author and 3 markers who marked this script in previous step. Thus students can reflect on their own ideas by providing comments and discussing using the ACD in a variety of roles.

4.3.2.3 ACD monitoring web page

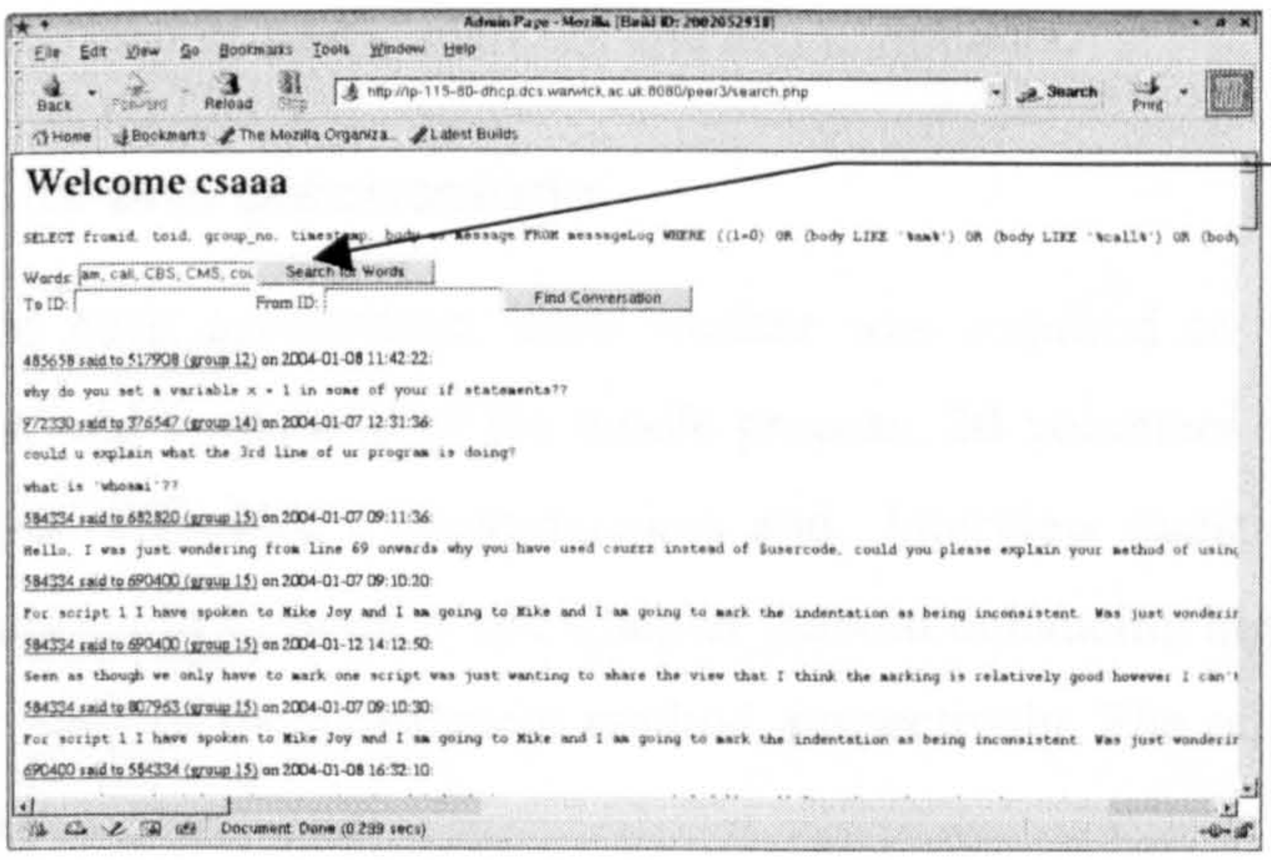


Figure 27 ACD monitoring web page

This monitoring web page (Figure 27) allows an administrator to monitor student conversation, both for the purpose of evaluating the effectiveness of the tool, and

intercepting inappropriate messages and conversations. It reports all the students' conversations through the ACD and can be used to search through the conversations of each particular group. Moreover, the ACD monitoring web page provides a search for undesirable dialogues, such as asking the identity of the participant, or negotiating a mark, using a keyword search.

4.3.2.4 Monitor marking

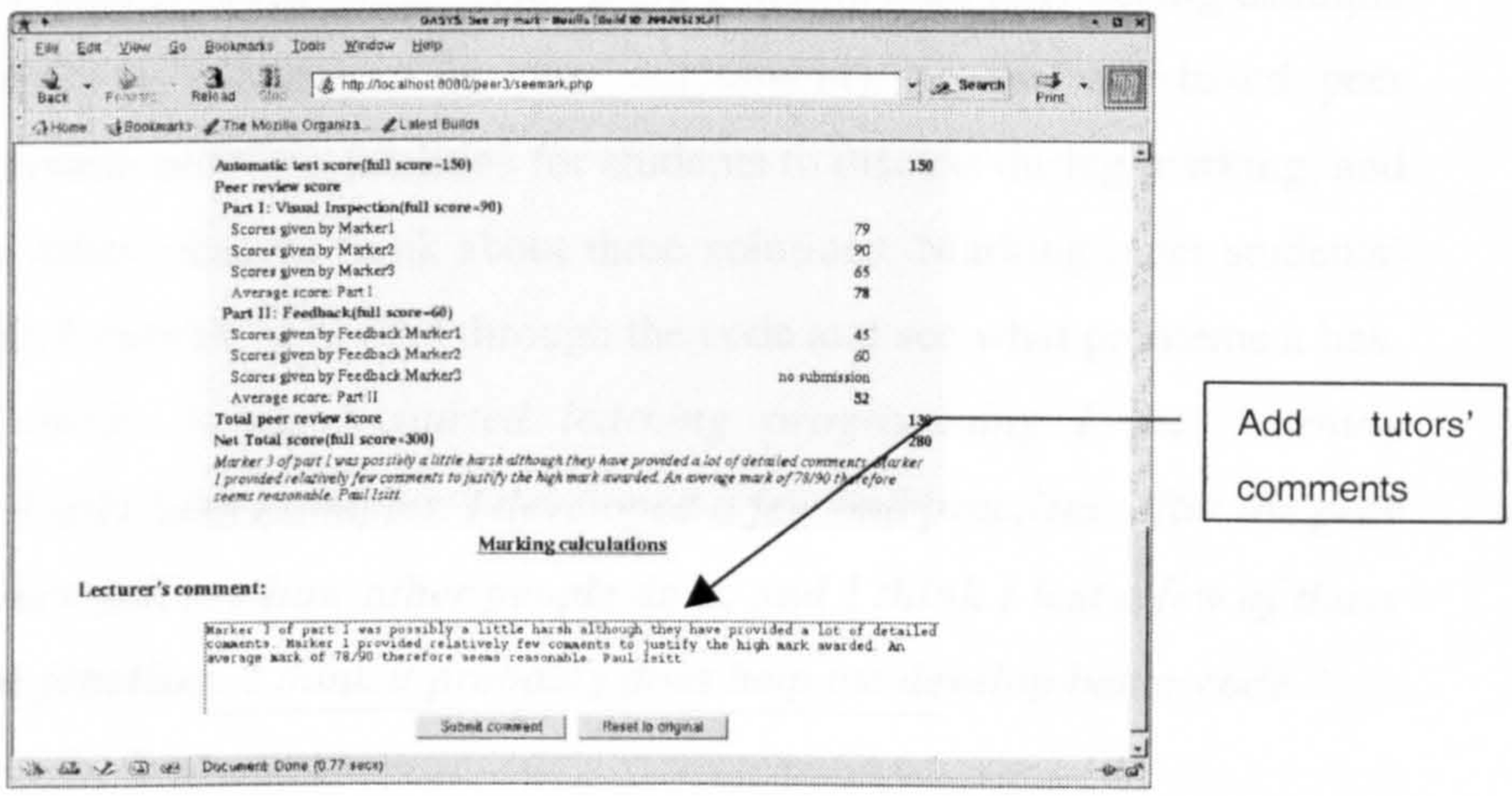


Figure 28 Tutor's comment feature

We modified the web-based peer assessment system to allow tutors to add comments on each script easily. They could monitor the peers' marking by access the 'Monitor marking' web page, which provides a report of students' marking. Then tutors can access each script by clicking on the "name of students" link to inspect peers' marking and add comments.

4.3.3 Results and discussions

At the end of each assignment, each student was required to fill in online questionnaires, and at the end of the whole process, 20 volunteer students were interviewed (see appendix: Questionnaires and Interview questions), and the results are analysed in Chapter 5 and Chapter 6 about enhancing higher cognitive skills and accuracy of the assessment method, respectively. The other results are discussed below.

80 out of 114 students (from the online questionnaire results) reported that peer assessment helps them to learn programming more than traditional assessment.

Peer assessment is a good way of helping them in learning programming, as the student's quote illustrates below.

> *"I would learn more from other scripts that I mark, and I believe if I mark more, my programming skills will probably improve."*

Many students learn programming by examples. Results from interviews indicated that peer assessment provides more benefits than only seeing example programs (without taking part in peer assessment), as the web-based peer assessment system provides facilities for students to discuss during marking, and see what the other students think about these solutions. Marking other students' programs also forces them to read through the code and see what problems it has.

> *"I think... when I started learning programming I was learning basically from examples. I developed a few bad practises, **after the peer review you see how other people do it, and I think I lost a few of those bad practises.** I think it probably does help me develop better code."*

> *"I think the peer review is better. **It gives you a lot more variety,** **rather than just seeing ideal solutions,** you see flawed or imperfect solutions as well, which probably gives you more variety of ideas, and **encourages you to come up with your own way of thinking about it and doing it.**"*

However some students were concerned that the other students may not take this marking seriously, and a few students who are competent in programming preferred comments on their work from an expert. Other issues about the marking are discussed in Chapter 6.

4.3.3.1 Anonymous discussion

The students have different roles as script author, marker and feedback marker in the two stages: marking quality of the program and marking the quality of feedback. They discuss the program to help in better understanding and marking during each stage via ACD, as the following illustrate:

- *unknown utilities* - "Towards the end of your script you use the cut utility and pipe it within the final case statement. I'm pretty sure how it works but to clarify, could you tell me how cut –c works?"
- *assignment specification* - "Hi fellow marker, correct me if I am wrong, but isn't the program suppose to exit with status 0 when checking the file existence and permissions, and status 1 when checking usercode? In script 1 it always exits with 1...!!!"
- *analyse program* – "Your code passes the tests, but I'm not certain that it's fully functional. I think there are cases where your code does not operate fully to the specification. Maybe this is because you haven't used pattern matching in your set of error diagnostics."
- *program understanding* - "Hello, I was just wondering from line 69 onwards why you have used csuzzz instead of $usercode, could you please explain your method of using this."
- *sharing opinion* - "Your marking is pretty good. However, you said that the utility for testing that the owner has write permission for the file will not work correctly in all situations, i.e. –w test is wrong. I thing what you've said is true but what about the other conditions...isn't –x wrongly used as well in the script?"
- *marking discussion* - "Hi, it should be [a-z], shouldn't it? Are you going to mark them down?"

These results suggest that the web-based peer assessment with ACD has successfully helped students to engage more critically when learning computer programming. However it should be noted that although the ACD worked effectively, the technologies employed (a combination of PHP-scripted web pages, Java Applets, and RMI), highlighted interesting technical issues. The University firewall, for example, is configured in such a way that connections to the ACD server were initially forbidden, and the default JVM provided in certain browsers is not fully functional. These are problems that although solvable, require intervention on the part of the participating students if they access the ACD from home, and may have reduced the use of the ACD by some students.

4.3.3.2 Discussions

Our study of peer assessment in programming reveals some issues that contrast with the other studies as follows.

- *Setting marking criteria*: according to Smith [Smith2002], students should be involved in drawing up the marking criteria, to increase confidence and understanding. Results from questionnaire in our study revealed that almost 90% (142 out of 158 students) of students did not want to be involved in drawing up the marking criteria. Falchikov and Orsmond [Falchikov1995 and Orsmond2002] also reported that students seem unenthusiastic about creating marking criteria themselves. Therefore to reduce the time setting up the peer assessment process, marking criteria can be set by teacher.
- *Resubmit work*: according to Lin [Lin2001], students should be allowed to resubmit work (essays) after they took part in the peer assessment exercise. However this programming assignment may not be suitable for doing that, since students may copy the others students' programs after observing them in the peer assessment exercise. Our study revealed that peer assessment makes students realise their own mistakes, picking up many good points from marking other students' programs, which benefits them for the next assignment.
- *Marking quality of marking*: in many researches in peer assessment, tutors award marks for 'quality of marking' [Davies2000, Lin2001, Ballantyne2002]. In our peer assessment process, students mark the quality of marking. Results from interviews indicated that this step not only encourages students to take their assessor roles more seriously, but also develops their critical judgement skills.
- *Reliable marks*: in many studies of peer assessment, the final marks are a combination of peers' and tutors' marks in order to award fair marks to students [Brindley1998, Magin2001]. Our study revealed that there is a strong positive relationship between peers' and tutors' marks. Therefore peers' marks are reliable, and peer assessment is an accurate assessment method in a programming course (see Chapter 6).

- *Confidence in marking*: many studies in peer assessment report that students are not confident in marking other students' work [Orsmond1996, Topping2000, Ballantyne2002]. Our study of peer assessment in three consecutive assignments revealed that the more marking students do, the more confident in marking they become.
- *Anonymous marking*: bias and collusion in marking are problems that may happen in peer assessment [Falchikov1995, Orsmond1996, Topping2000, Ballantyne2002]. In our study, there is no report of these problems. 106 out of 114 students prefer anonymous marking, because they feel more comfortable analysing other students' work with fair marking, instead of possibly favouring their friends.

4.4 Summary of the chapter

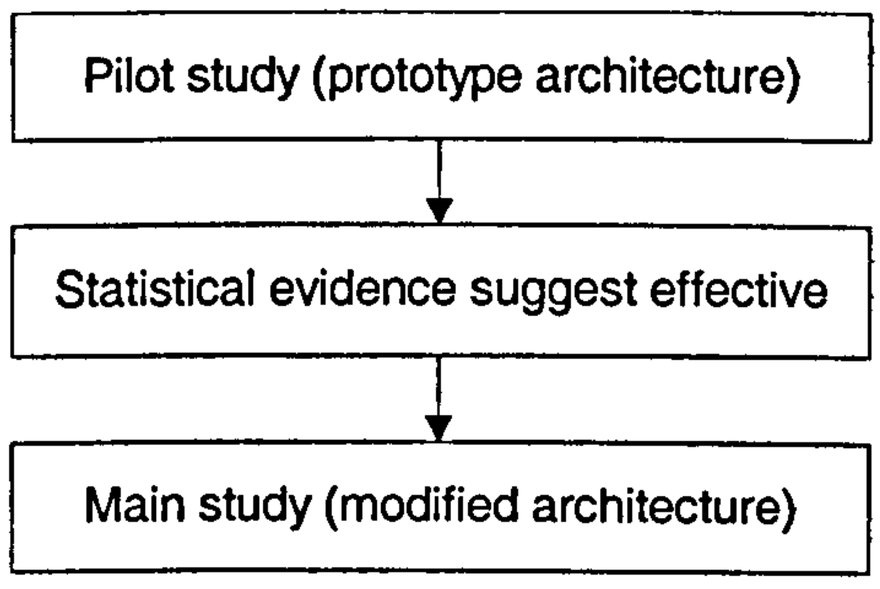


Figure 29 Summary of 'peer assessment for programming' chapter

We have described a peer assessment process, together with supporting web-based software, which we have used to test the effectiveness of peer assessment in learning programming languages (Figure 29). We designed this web-based peer assessment system to provide anonymity for the whole process, in order to ensure the process is fair, to encourage students to discuss without embarrassment, and to allow the process to be closely monitored and supervised. Students have reflected on their own ideas by discussing using the ACD in a variety of roles (script authors, marker, and feedback marker). The process we have used is novel, since students are engaged not only in marking each other's work, but also in evaluating the quality of marking of their peers. This study indicated that it has contributed positively to the students' learning experience.

Chapter 5 Peer assessment and higher cognitive skills

Research question: Does peer assessment enhance higher cognitive skills in a programming course?

5.1 Overview of the chapter

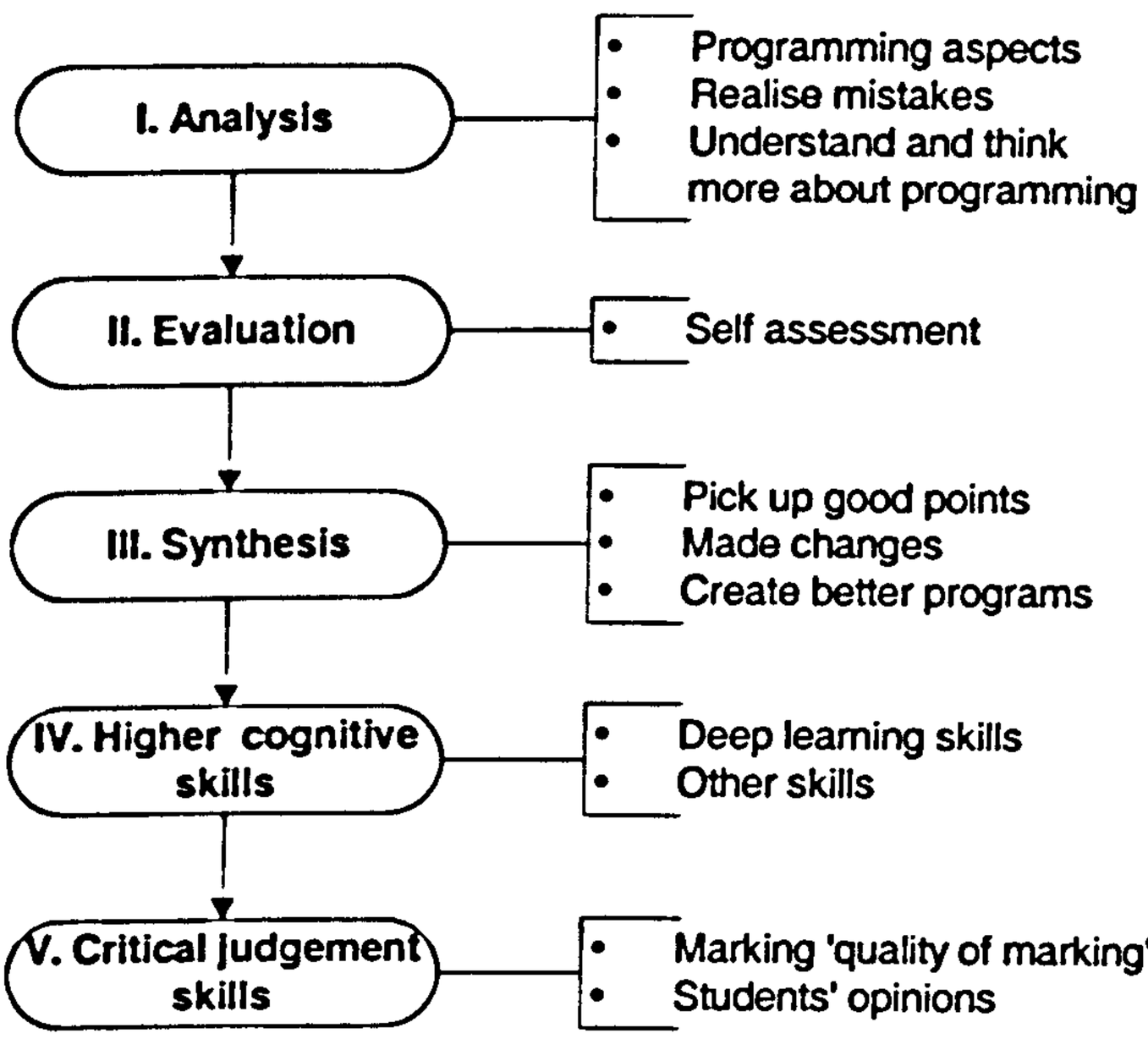


Figure 30 Outline of 'peer assessment and higher cognitive skills' chapter

In this chapter, we discuss how peer assessment encourages students to develop deep learning skills, based on our new learning theory in the context of programming which has been refined from Bloom's taxonomy. We start with *analysis* of both the other students' programs (programming aspects) and their own programs (realising mistakes). Then we consider how students' responses about peer assessment helped students to understand and think more about programming, and how marking in the peer assessment process not only encouraged students to make judgements about other students' work (which will be discussed in the *higher cognitive skills* section), but also encouraged students to evaluate their own work. The link between peer and self assessment is explored in the *evaluation* section. In the *synthesis* section, we explore whether

the students' performances in programming (create better programs) improve after performing peer assessment.

The skills that students could develop from peer assessment are discussed in the *higher cognitive skills* section. Finally in the *critical judgement skills* section, the evaluation of 'quality of marking' that helped students to develop their critical judgement skills is investigated, and students' opinions on this step are reported. The following sub questions are the keys that help in answering whether 'peer assessment enhances higher cognitive skills in a programming course'.

1. Analysis: Does peer assessment help students to understand and think more about programming?
2. Evaluation: What is the link between peer and self assessment?
3. Synthesis: Does the performance of the students improve in subsequent assignments after performing the peer assessment?
4. Higher cognitive skills: Does peer assessment encourage students to develop higher cognitive skills?
5. Critical judgement skills: Do students display critical judgement skills on evaluation of the initial marking?

Analysis of data from online questionnaires and interviews reveals similar students' answers for some questions, but focuses on the different specific issues (e.g. realising mistakes, evaluating the student's own program, and thinking about good and bad points of one's own program). However we do not just summarise the students' responses, as all the relevant evidence should be reported.

5.2 Analysis

Research question: Does peer assessment help students to understand and think more about programming?

> ***"Some better understanding of different ways to develop a specific program, which is kind of useful as far as programming is concerned. Besides that, some, I guess of marking, which might be useful as well because in order to mark, you need to have the knowledge to mark and also to be able to see a program and incorporate your ideas into it. The point is not to learn just how to write something, but to see something and understand it, so that you're handling that as well."***

The above student's quote indicates that peer assessment encourages students to start thinking more deeply about programming than just being able to write the program. As students are involved in marking each others' work in peer assessment, they have a chance to analyse and evaluate the other students' programs, which results in their starting to think how to improve their own work (synthesis). These cognitive skills (i.e. analysis, evaluation and synthesis) are a part of Bloom's taxonomy at the higher level (deep learning) as discussed in Chapter 2 (section 2.2.1 Bloom's taxonomy). In this section, we discuss the way students analysed other students' programs and their own work. We start by exploring the aspects in programming that students specified in comments, followed by students' mistakes that they realised when they had marked other students' programs. Finally we discuss students' responses about whether peer assessment encouraged them to understand and think more about programming. The particularly supportive students' phrases are highlighted in bold.

5.2.1 Programming aspects

In the peer assessment process, students marked each others' work based on specified marking criteria (i.e. program readability, correctness of program and programming style). This step encouraged students to analyse and evaluate other students' programs. Table 18 displays the programming aspects that students specified in commenting on shell programming (assignment 3), the number of

comments which are related to each aspect, and identifies which programming aspects are explicitly mentioned in the marking criteria. Apart from program readability that may be easy to assess, many students' comments focused on the programming errors/mistakes, appropriate utilities and modularity.

Programming aspects	No. of comments from 100 students	Explicit marking criteria
Program readability:		
Number of comments	78	√
Comment quality	80	√
Code indentation	82	√
Appropriate identifier	73	√
Handle errors	45	√
Programming errors/mistakes	79	-
Exit status	51	√
Missing statements/methods	33	-
Appropriate utilities	63	√
Appropriate method	28	-
Program structure	55	√
Sub routines/modularity	82	-
Temp file	2	-
Syntax of programming	12	-
Efficiency of program	19	-
Error message	1	-

Table 18 Programming aspects that students specified in their comments

It comes as no surprise that aspects such as 'appropriate utilities', which are part of the marking criteria, receive a high number of comments, but criteria such as 'programming errors/mistakes' (which are not marking criteria) also receive a large number of comments. Students who commented on those aspects appear to have analysed the programs they were marking. The examples of students' quotes are illustrated below.

- *Programming errors/mistakes*: the programs that did not pass ten automatic tests were investigated to find out what was wrong with the program and how to correct it.

"If multiple -i or -o options are entered, only the first is considered, not the last, which is against the specification which stated only the last would be considered. A suggested fix that would be to look at the man pages for the getopts command and consider using it in a while loop... for example....."

- *Missing statements/methods*: some programs did not fully meet the specification because of missing code.
  "As far as error checking goes the pattern matching on the id codes it very well done, ... However, from what I can see there is no error checking on the dates (which is specifically listed on the spec), no error checking on mup file data ..... I would suggest implementing more error checking routines ....."

- *Appropriate method*: other different/possible solutions are suggested in order to improve the programs.
  "Lines 157 - 160 contain an interesting way of implementing -i, very good! I would suggest another way using a small awk script on a temporary file at the end of the script, this way there is only one line to maintain that applies the awk script to all the files in the file, so 3 copies are not needed as in your current implementation, an example of such an awk script would be....."

- *Sub routines/modularity*: many students suggested that sub routines should be considered in order to decrease repeated code and increase the program readability.
  "Using a lot of subroutines makes following what the program is doing easier. Perhaps you can combine the readPitFile and readCupFile subroutines together in order to avoid unnecessary repeated codes."

- *Efficiency of program*: students' comments on the speed of the program displayed concern for program quality.

> *"The author has decided to store the arguments passed to the script in a temporary file. This is not necessary for the use of Getopts or other argument parsing routines. It would be much more sensible to simply store them in variables and check these rather than outputting them to a file (**which will slow the script down - especially with larger files**)."*

These results suggest that students could analyse other students' work via the marking process in peer assessment.

5.2.2 Realising mistakes

> *"From marking these scripts I feel that I have gained experience because in this coursework **I have learnt how to use getopts and use it efficiently. I made a mistake with one of my functions and I know now that** -f for the test statement is much more recommended for the purpose of this particular specification. **I also realised that** my corruption checking wasn't up to the standard."*

Peer assessment not only encouraged students to analyse other students' work, but also encouraged them to analyse their own work. Through the online questionnaire, 82 out of 156 students realised mistakes that they made in their own solutions when marking other students' programs, and gave details of particular errors or mistakes that they identified. However a few students answered 'No' because they did not make any mistakes, could not follow what the programs did, or discovered new ways of writing the programs instead of realising their own mistakes. The summary of mistakes that students realised when marking other students' work, with examples of students' quotes, and the number of instances (N) are displayed in Table 19.

Mistakes that students realised when marking other students' programs	**N**
1. ***Appropriate utilities*** - Just in style, I missed some useful utilities that would have cut down on the amount of code. - I could of used getopts, I misunderstood how it worked and believed it was more complex than it really was, so didn't use it.	**17**

Mistakes that students realised when marking other students' programs	N
- I realised various mistakes, usually regarding the selection of utilities within my code, also problems with the structure arose on occasions. - Using DATE utility instead of calculating them manually. - Over-complicated use of bc, could have used getopts... - Yes, I noticed they used some utilities such as 'getopts' and 'date' that would have improved my script. - Finding an easier way to consider the number of arguments using getopts.	
2. ***Error handlers*** - Realised I had not accounted for certain errors, particularly to do with dates and times. - I didn't do the date line format checking properly. - That I'd missed some potential errors. - Leap years error in checking whether it was a leap year. - A distinct lack of error checking! - I didn't check for valid dates. - Realised that **my checking for the format of the input was not comprehensive**, and that I should have pattern matched more. Also realised that I could have implemented certain parts of the program in a different way e.g. Is test. - My error checking wasn't quite as thorough as some of the scripts I marked. So I learned from them.	**14**
3. ***Better programming approaches/more efficient ways*** - I saw a lot of better ways of doing things I had done differently in my code, helpful. - It was not as much errors that are noticed but the style and ways in which to do things are very useful from the peer to peer review. - Not mistakes exactly, but instances where I could have better improved my script by using that method or utility. - Not particularly mistakes, but marking other code provided a view of more efficient methods of solving the problem. - I realised alternative ways I could have programmed my script. - Found new solutions to problems I couldn't tackle. - I just noticed different ways of implementing things - perhaps better ways to use in the future.	**11**
4. ***Meet specification*** - Minor errors where my script did not conform to the specification or the way in which my interpretation of the specification differed from the script I was marking.	**11**

Mistakes that students realised when marking other students' programs	N
- Haven't considered more than 4 arguments occurring, which may be valid. - I realised I didn't include a condition for if no option/argument was submitted. - Missed out functions like taking the last argument. - I didn't accommodate for multiple -i/-o options. - Mainly argument checking and that I could have checked for a negative bill instead of checking for every possible error.	
5. Program structure - From seeing how other students structure their code, I realised that my structure (while I believe it to be quite readable), **could be improved and made better** from an "assessed work" standpoint. - I picked up some tips on how to layout code, or some of the utilities that could have been used. - Just other ways in which the solution could be solved. I.e. I didn't think about using functions but all the scripts I marked had done.	5
6. Loop statement - I tried to implement a loop, however it wouldn't work and so I had to delete it totally and submit the previous code as this produced the most automatic test working. **From other people's work I could identify what I had done wrong.**	1
7. Multiple arguments - Mistakes with multiple arguments.	1
8. Code repetition - Could have used functions to cut down code repetition.	1

Table 19 Summary of mistakes that students realised when marking other students' work

A few students who realised their own mistakes when marking other students' programs cannot specify the errors or cannot remember what they are.

"Just seeing bits which had been missed out here and there."

Results in Table 19 suggest that marking other students' programs made students compare their own work with others, which resulted in their realising mistakes in their own work.

5.2.3 Understand and think more about programming

In the online questionnaire, students were asked to either agree or disagree with a number of statements about peer assessment. We first of all consider their responses, and then compare the responses with statements made during the subsequent interviews.

5.2.3.1 Questionnaire: statements about peer assessment

N=129

Which of the following statements about peer review and technical skills do you agree with?	**YES**	**NO**
Peer review forces me to think about what constitutes a good or poor piece of work.	**117**	12
Peer review helps me to think about the specification of a program more deeply.	60	69
Peer review helps me to improve my programming style.	91	38
Peer review is deepening my understanding of what is required in good programming.	71	58
Peer review is providing me with a better understanding of what is required to achieve a particular standard and what tutors are looking for when conducting assessment.	55	74
Peer review encourages me to consider the objectives and purpose of the assessment task as well as the course itself.	56	73
Peer review highlights the importance of presenting work in a clear and logical format.	**108**	21

Table 20 Statements about peer assessment and technical skills that students agree with

Table 20 displays 129 students' opinions about peer assessment and technical skills from the online questionnaire, which suggest that peer assessment encouraged students to think more about issues such as program quality, readability, and style. The percentage of students who agreed with the benefits of peer assessment are shown below in order:

- 91% - forced to think about what constitutes a good or poor piece of work,
- 84% - highlighted the importance of presenting work in a clear and logical format,
- 71% - helped to improve programming style, and

- 55% - deepened understanding of what is required in good programming.

However only 47% of students agreed that peer assessment helped them to think about the specification of a program more deeply, and 43% of students thought peer assessment:

- provided a better understanding of what is required to achieve a particular standard and what tutors were looking for when conducting assessment, and
- encouraged considering the objectives and purpose of the assessment task as well as the course itself.

These results suggest that peer assessment helped students to learn more about programming than to think about the assignment itself (such as program specification, assessment, and objectives). However in addition to these quantitative data from online questionnaires, the qualitative data from interviews are reported in the next section.

5.2.3.2 Interview: peer assessment helped students to understand more about the assignment.

*"Definitely. **I think even more than helping me understand more about the assignment, it helped me learn a lot of things about coding**, especially with CS120, because I found it very strange to start with. I had a lot of problems with my assignment 1, and then we were told we had to do our coursework, and there would be people marking it and I was really scared. **But after going through assignment one, and having people giving comments, and seeing how people were doing their things, it made me feel more comfortable**, because I was more confident with what I was doing, **and it made me see how people actually think, you know there's actually another way of doing what I'm doing, and I think it really helped me, to be honest. And to be able to express myself**, and having other people judging my work, rather than a tutor who knows everything. I think it really helps."*

The above student quote suggests that peer assessment helped him in learning programming and to be able to express himself. Results from interviews with 20 students found that *all* of them agreed that peer assessment helped in understanding more about assignment, especially in interpreting the assignment specification, which was useful for the next assignments.

> *"I think it does help, and* ***it certainly helps progressively, having done the first one, that helps for the second one****. You know a lot more what you're expecting."*

Table 21 displays examples of students' responses, which indicated that peer assessment helped them to understand more about the assignment, with the number of responses.

Peer assessment helped me to understand more about the assignment	**N**
1. Interpretation of assignment specification - **I suppose it would help you in the future to interpret other specifications** because a lot of things I think, some of the things they did left it open to interpretation. I suspect that was probably done on purpose but it might not have been, but I suppose yeah, its all about the way you interpret it, but **seeing the way other people interpret it, I guess that could be useful**. - Reading the peer review for the second one **helped to clarify in the third specification** which was more vague and had to have certain things quantified and classified. Especially the bit with the multiple arguments, multiple inf in the second one, and uni in the third one. You could have multiple options if you used getopt or the PERL equivalent. So you could have the statement and then -i -o -i -o and it would carry out the last one. **The comment in the second one helped doing that in the third one.** - I normally tried to get quite a good idea of what the specification wanted. I would sometimes **see different interpretations of the specification, which I suppose would broaden my understanding**. - Yeah. Definitely. Like on the 1st one because I didn't know how to go about it, I did follow the specification, but I didn't really really look at it in depth. **Because of this peer review you understood that when you were marking it you had to look at the specification in depth**, so for the next one, you'd look at the specification in depth first, then go on to do the things, then re-look at the specification, then change things, the look	5

Peer assessment helped me to understand more about the assignment	N
at the specification again, and it was like a constant... a big circle of checking and criteria, and lots of things like that.	
2. Different programming approaches/advanced solutions - Yes, sure. I mean, as I said it's always useful to see other examples, and also in this type of program there are many many ways of doing things. So, **you see a better way of what you have done and you learn it**. Is nice. - I think it was definitely very useful because if I compare it to other assignments, for example the Java assignment, **it would definitely help me if I looked at how other people approached the problem. Especially with the advanced solutions,** because not everyone can finish the program. **It would help for the future to look at how other people did it.** It's definitely quite useful. - The thing I was looking for was the most efficient way of doing it. Some people wrote very short programs, some people wrote very long ones. And some people who wrote the short programs didn't do it properly, so it was about finding a mixture. But **it helped me approach my next coursework in a better way**.	3
3. Good programming practise - I think it helped me to understand more about **good programming practises**, I think the assignment is just a case of working it out for myself. **It helped me for the next assignment** yeah, and **it helped my general understanding of the subject**.	1
4. Revise programs - Yes, I'd say so I think that especially because I not done any Linux based programming before **it was definitely with my first script I think was particularly weak in comparison with the rest of them** and that was because as over the four weeks that you do it you do the program and then **you're constantly looking back at the material, its all a bit active revision in a way your sort of revising everything you've done** then you'll move on and do the next bit and look back at what you've done and what other people have done. So yes, I think it helped me improve that side of my programming.	1
5. Programming style - It helped me with my general programming style.	1
6. Commenting style - Yeah, in terms of commenting style and using other bits of the language, like functions and stuff, yeah. It did help.	1

Table 21 Peer assessment helped students to understand more about the assignment

The results from interviews support the results from online questionnaire, since most students agreed that peer assessment helped them to learn more about programming. In particular, 5 out of 20 students thought it helped them to interpret the assignment specification, which was useful for the next assignments.

Summary

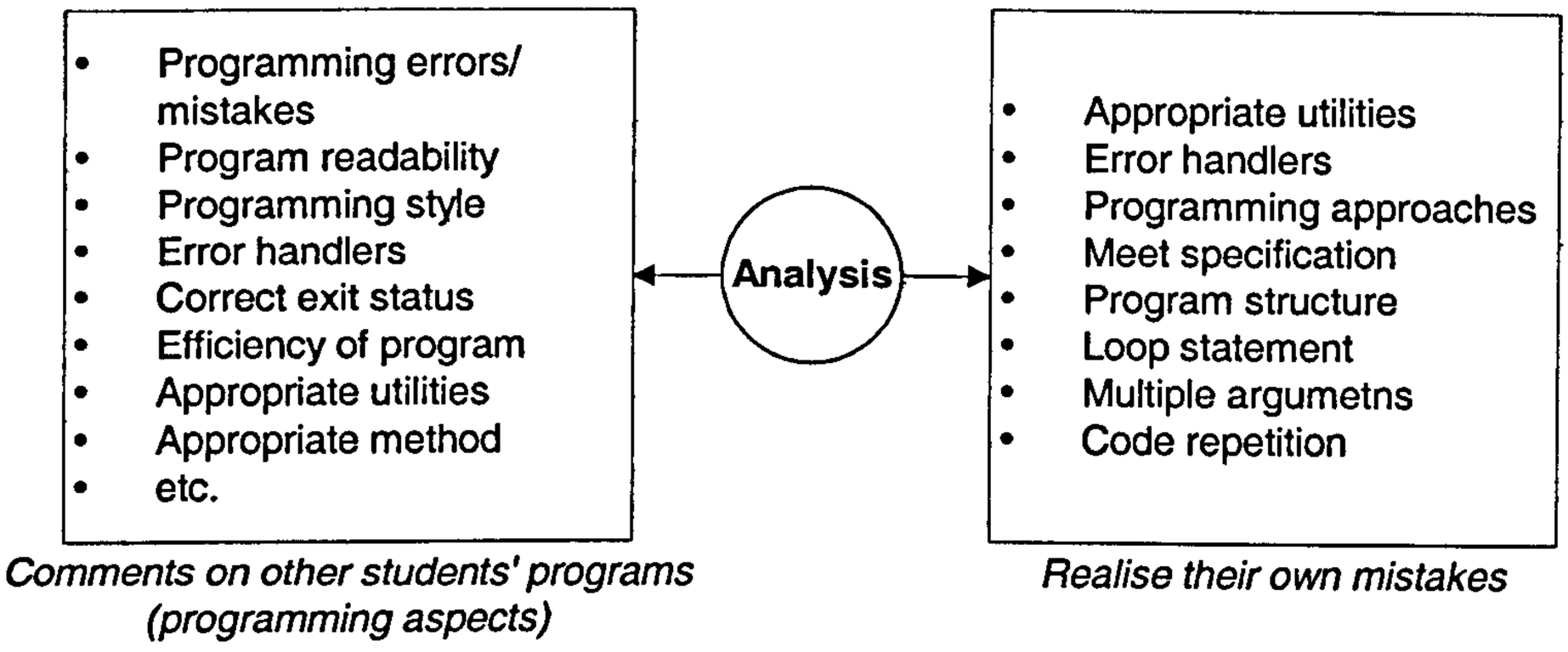


Figure 31 Students' analysis of other students' programs and their own programs

Figure 31 shows the programming aspects that they specified in the comments, against what the students realised about their own mistakes. Peer assessment helped students to understand and think more about programming. They can analyse the other students' programs and their own work via the peer assessment process.

- Students' comments displayed analysis abilities. Besides the marking criteria that students had to follow, students could identify appropriate sub routines, programming problems, etc.
- They also could realise their own programming mistakes (analyse their own programs) when marking other students' work, for example appropriate utility, appropriate error handler, programming approach, meeting the specification, etc.

Results from questionnaires and interviews also indicated that peer assessment encouraged student to learn more about programming, especially thinking about

what constitutes a good or poor piece of work. Analysis of other students' work and their own work resulted in improving of their own programming abilities.

5.3 Evaluation

Research question: What is the link between peer and self assessment?

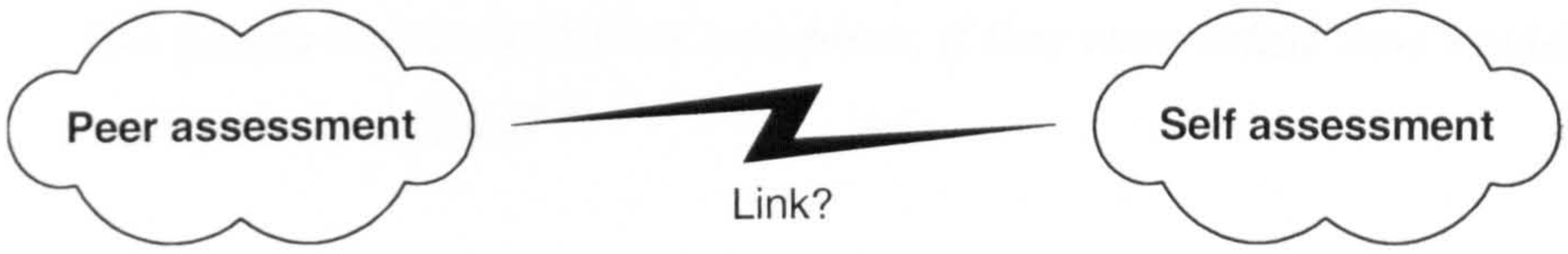


Figure 32 Peer assessment and self assessment

Marking in the peer assessment process not only encouraged students to make judgement of other students' work, but also encouraged students to evaluate their own work. The link between peer and self assessments is explored in this section. *Self assessment* refers to "the involvement of learners in making judgements about their own learning, particularly about their achievements and the outcomes of their learning" [Boud1989], whereas peer assessment engages students in making judgements about other students' work [Somervell1993]. Self assessment is one of the outcomes from peer assessment. Figure 33 displays 100 students' opinions about whether marking helped them to evaluate their own work, using 5 Likert scales (from disagree to strongly agree) from the online questionnaire. From the results, most students agreed that marking helped them to evaluate their own work.

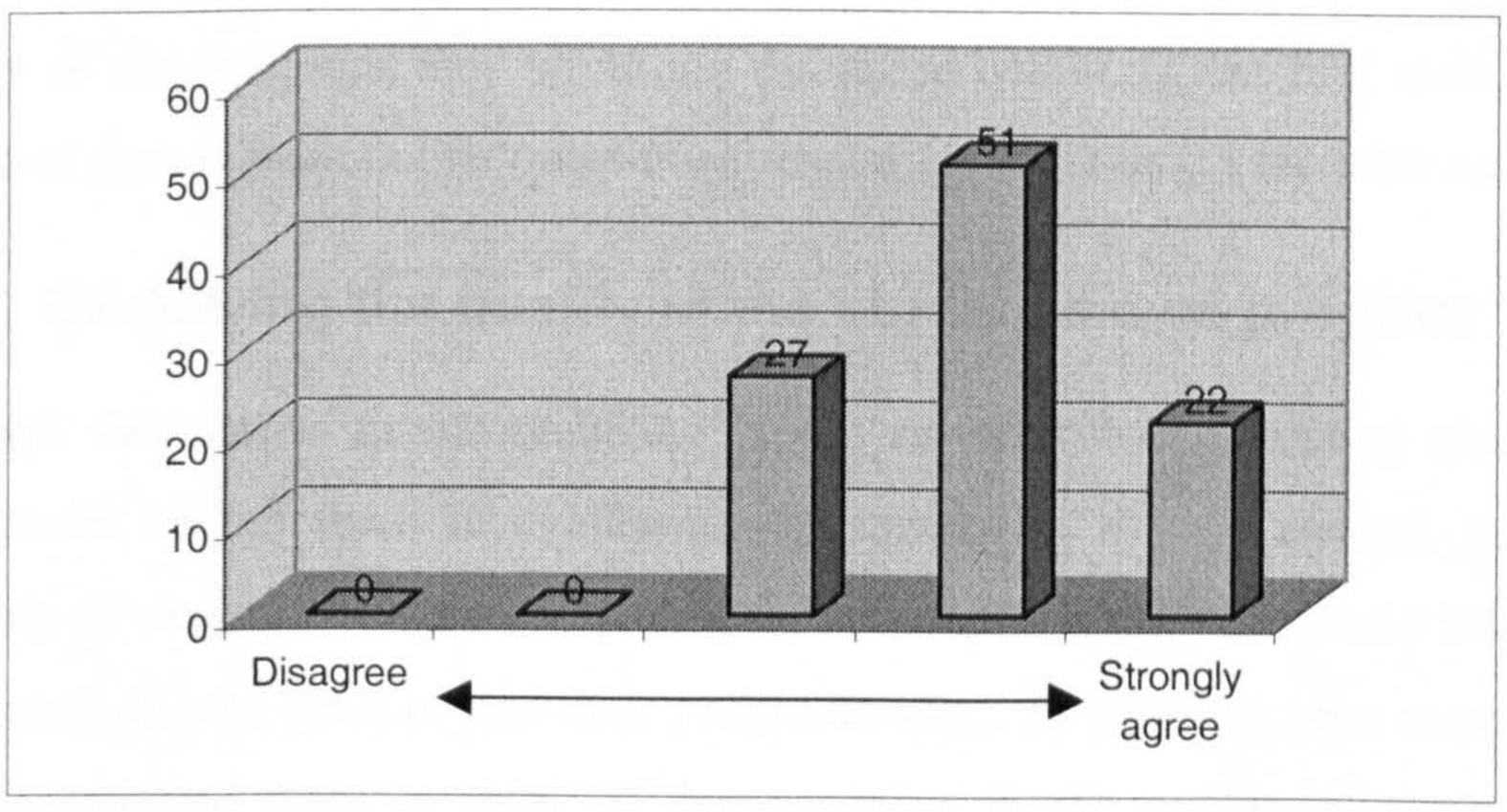


Figure 33 Marking help me to evaluate my own work

> *"When I went through the criteria for the first time I thought "how are they going to mark me?" and I was thinking they might drop me a mark on this area, for example it's not as good as one of the people I'm marking, they might drop a mark there. Or for example, on the first one .....So when I looked through the criteria, I'd decide whether or not I'd get a good mark for something... then I'd know why I hadn't. That did help, if you think about... I think it would be quite interesting to see how people would mark their own piece, if they were asked 'how would you mark your own piece?'"*

The above student quote indicated that peer assessment encouraged him to evaluate his own work via the marking criteria. Self assessment or self evaluation is a good exercise in developing one's own ability [Brown1994]. Students could evaluate the quality of their own work when they marked other students' work. They could recognise the good and bad points of their own programs by comparing with other students' programs. Therefore peer assessment encouraged students to think more about their own work, and the students' responses from the online questionnaires support this:

- 112 out of 129 students agreed that analysing the work of peers leads to an improved *awareness of the quality* of their own work;
- 105 out of 129 students agreed that peer assessment helped them to *criticise* their own work.

Students' opinions on whether peer assessment helped them to evaluate the quality of the programs they wrote are discussed below. In the first section, the source of data is from online questionnaires, and interviews for the later section.

5.3.1 Evaluating the quality of the student's own program

Through the online questionnaires, 52 out of 63 students agreed that peer assessment helped them in evaluating the quality of their programs, and the majority of students thought comparing their own programs with others helped in awareness of how good or bad their programs were. In addition, peer assessment also encouraged them to think of improvement to the quality of their work.

Examples of students' quotes with number of their responses are shown in Table 22.

Yes, peer assessment helped me in evaluating the quality of my programs	N
1. Able to compare with the other students' programs	**12**
- Yeah, able to compare with other people's codes. You roughly have an idea on how good or bad your codes are compared to the others. - Yes because it lets you see where you did things a good way and where others did it a better way. - Yes, because you get to see how other people have gone about the problem and this **allows you to see that maybe there were better ways than your own**. - Yes, I can then see how good my program is compared to other peoples. - It enabled me to evaluate my code against other solutions. - Yes, I feel you can only really tell the quality of the program by comparing it with others.	
2. Realise my own mistakes	**7**
- Yes, it lets me see where I went wrong. - Yes, seeing other peoples code helps to identify problems with your own code. - Yes, because flaws bugs and mistakes were pointed out.	
3. Encourage me to improve my program	**6**
- Helpful in seeing how you can improve your own work by drawing on others. - By looking at others and thinking about the specification and assignment in a different light, **makes you think about how you could change your program**. - It made me see the ways in which it could have been improved and also **made me put more effort into making it logical and clear,** as someone else would read it.	
4. Aware of quality of my program	**5**
- Yes, the marking gave a clear indication of the quality level of my coding. - Yes, as the program would not only have to pass tests but have to be understandable by others which made me think more about the program. - It did, as I had to consider the marks I would have given myself.	
5. Satisfy the marking criteria	**4**
- When writing the program, **I was working with the intent produce code that would fully satisfy the marking criteria for the peer review**, as this would hopefully result in a higher mark. I focused on the criteria given	

Yes, peer assessment helped me in evaluating the quality of my programs	**N**
for marking in the peer review, namely readability, correctness and structure, so I was closely evaluating those aspects of my work. - Yes because I knew what I would look for when marking it. - Yes because it shows you what is expected of you.	
6. Encourage me to improve my programming ability - By looking at better programs it made me strive to achieve a similar level of style and technique. - Yes. I now know what I have to do to ensure my program follows a good style. - It made me think about ways I could have done it more efficiently.	**3**
7. Able to see the alternative ways to implement problems - Yes. Because I can look at other style of codes for the same task. - Yes, to see how other people approach the same problem	**3**
8. Aware of good and bad programming approaches - I became aware of better and worse ways of solving the same problem. - Yes, because you can tell whether other programs are excellent or of poor quality and they help you to think about them in relation to your own work.	**2**
9. Meet specification - Having achieved the specification, the only other parameters for success were style etc, which are extremely subjective.	**2**
10. See better programming methods - Yes. I discovered some better programming methods, etc. - Yes, see better methods of performing functions.	**2**
11. Remind me of what I missed - Yeah it did because I got to see aspects of the program that I neglected and that in reality do matter. - Yes, highlighted points I would have otherwise overlooked.	**2**
12. Useful comments - Yes - the feedback I received was quite useful and gave me some tips on areas that I should work on, such as commenting.	**2**
13. More objective and critical - It taught me how to be objective and critical of people's work, and I am able to apply that to my own.	**1**
14. Review my program - Yes, it did help evaluate the quality of my program because it made me think of the methods others had used and **helped me to think about the ways I had done this in the past**.	**1**

Yes, peer assessment helped me in evaluating the quality of my programs	N
15. Think more about how other people will comment my program - Because as I was writing it, I considered what may be said about it.	1

Table 22 Students who agreed with peer assessment helped them in evaluate their own programs

Results from Table 22 suggest that comparison is the most important link between peer assessment and self assessment, as it encouraged students to evaluate their own programs against others. However 11 out of 63 students who did not think peer assessment helped them to evaluate the quality of their programs are experienced programmers or very confident in their programming abilities (see Table 23). They believed they knew what they were doing, since they considered that they wrote good programs, therefore they did not consider that the comments from weaker students were useful in helping them to evaluate the quality of their programs. One student found it was difficult to compare the quality of his program with other programs that had different programming styles.

No, I don't think so	N
1. I do my best (my program is good) - No, because **the work handed in is normally done to the best of my ability**. Peer review comments are only useful if there is part of the program that is wrong, and the marker suggests how to solve it. - No, since **I went through a design process before writing any code, layout and algorithms were already pre-planned** also the comments were written as the code was written. In that sense the peer review was more about other peoples preferences as opposed to mine, since my program already worked 100%. - Not particularly, all three markers said my program was pretty good and the errors were only little things like identifier names. - To an extent, but I am a fairly experienced programmer and know that my code is to a fairly good standard.	4
2. Not useful comment - Not really, the comments were pretty useless. - Slightly - **I just don't know if I can believe any of the comments being written** - for all I know they could be telling me do something that is not good programming practice and I'm already doing it the right way, but	3

No, I don't think so	N
they believe their way is the right way. - No, not at all...seems the markers are all fixated on readability and comments, which should be made such a great issue when looking at programs.	
3. I learned by myself - I can't say I learned much in terms of the actual programming. Since **mostly I looked up by myself, from the Internet or lecture notes or even asking other people, but I only learned a few points from others' programs**. This is partly due to I learned by myself and partly due to (sad -but happy deep under- to admit), marking criteria didn't require going into the programs' nuts and bolts very much. Of course the decision is completely yours at this point but my will stays on the lazy side.	1
4. I know what I am doing - Not really. I knew what I was doing anyway.	1
5. I lost marks from incompetent student - I have had experience with a variety of Linux/UNIX type operating systems (including Solaris) for several years now. Whilst I am in no way an expert, **I do consider myself to have a good level of proficiency**. As a result, it was often disheartening (and sometimes even offensive) to be criticised *incorrectly* by students with just a few hours' teaching being their only knowledge. As an example, I was told in the first assignment that my use of cs[uvw][a-z][a-z][a-z] was not as good as cs[uvw]???, and probably lost marks as a result. Of course, whilst I could "evaluate the quality of the program" **as being superior to the marker's suggestions, that was little consolation when marks were lost due to other students' incompetence.**	1
6. Can not compare with the different programming style - No. Since they are many different styles of programming, it was hard for me to compare the quality of my program to others.	1

Table 23 Students who disagreed with peer assessment helped them in evaluate their own programs

5.3.2 Thinking about good and bad points of one's own program

Peer assessment started students thinking about what was wrong and what worked well with their own programs, and brought different things to their attention, when they marked other students' work. For example, three students who thought their programs were good, still found what they should improve when they compared their programs with others.

> *"I mean I tried to do it the best way I could do in mine, but there are always small minor things to do, so I always learn from the peer review really."*

Through the interviews, all students agreed that peer assessment encouraged them to think about good and bad points of their own programs. Examples of students' quotes from interviews are displayed in Table 24.

Peer assessment started me thinking about what was wrong and what worked well with my program	Pick up errors	Things I missed out	Efficient program	Meet specification	Commenting style	Appropriate utilities	Appropriate variable names	Program structure/style	Different approaches
I did spot errors in mine or you know things I'd missed out, like there was something about the leap years and I actually picked it up in someone else's program that actually they'd done it the same way as me but actually that wouldn't work in all cases and I knew that I'd done it the same way so mine was wrong to. But yeah you do pick up errors.	√	√							
I got full marks, well above 90 for all three of them, so **I though my script was good but there were a lot of things that when I looked at other people's I though that I could do that to mine to make it more efficient** and just looking at ways to change things around**. I know that there were some of mine that didn't conform fully to the specifications** but looking at other people and seeing how they'd done it made me think oh yeah I could have done it like that. **It did make me look at mine and look at theirs and then think how bits of theirs would be good in mine and bits of mine would be good in theirs.**			√	√					√
Well, yes actually. If there's one thing I would say about my first assignment, though not many people picked up on it ironically, was that **my use of variable names was lacking and as soon as I marked other solutions I though, ah yes I really should have used clearer variable names**. And I didn't make that mistake again.							√	√	√

Peer assessment started me thinking about what was wrong and what worked well with my program	Pick up errors	Things I missed out	Efficient program	Meet specification	Commenting style	Appropriate utilities	Appropriate variable names	Program structure/style	Different approaches
Obviously there were some different ideas. People go about it some very, very long winded ways. Most time I found my program better than others because I coded it well. **My comments probably weren't better but the organisation of the program, how efficient the program was at doing its job**. There was one script that I marked where **I very much liked the commenting style**, and there were some scripts I marked where **they used some utilities I hadn't considered**, that I suppose was helpful.			√		√	√		√	√
Yeah, because **I saw ways things were done more efficiently, and ways of doing things that my script could not do.** There are always things my scripts cannot do. Especially in the last 2 the getopt had some weaknesses, and other people used other stuff.			√			√			
It did make me pick up on a few errors that I'd made. I don't know how serious they were. Once I'd done my assignment, I didn't look at it again, I didn't want to stress over it, because I couldn't change it at that point. **But it did make me have second thoughts that maybe I should have done it a different way**. The way I approached the last one, I think my way of doing it was rather long. It was very easy to read, but it could have been condensed. **Looking at other people's scripts made me realised how I could have shortened it.**	√							√	√
I can't really say because **when I was writing I tried to be up to the specification completely** so I didn't have any problems in the end. **The only thing that was helpful was how to make the program more efficient or shorter**, in a sense. That was better. I did the specification for the			√	√					√

Peer assessment started me thinking about what was wrong and what worked well with my program	Pick up errors	Things I missed out	Efficient program	Meet specification	Commenting style	Appropriate utilities	Appropriate variable names	Program structure/style	Different approaches
automatic and for the others that didn't have automatic tests. And for coursework 3 this data validation and verification that was a lot of people. I mean checking for the dates, some people had 1 to 9 method with square brackets, other people had the number of characters. **That was interesting because you see different ways of doing the same thing**. That was helpful.									
Oh definitely, it did yeah. Because how it helped me was**, once I got my script back with my marks, I would tend to look at it again and see what people had to say about it**, and it really helped because **in the first one my comments weren't that effective** so the person putting your comments could be more effective, or you could put it in a shorter way and **it really helped me because in the second assignment I did that and it gave me higher marks**, so I think it really did help me. The peer review was definitely very helpful to me.					√				

Table 24 Students' responses on peer assessment started them thinking about what was wrong and what worked well with their own programs

Summary

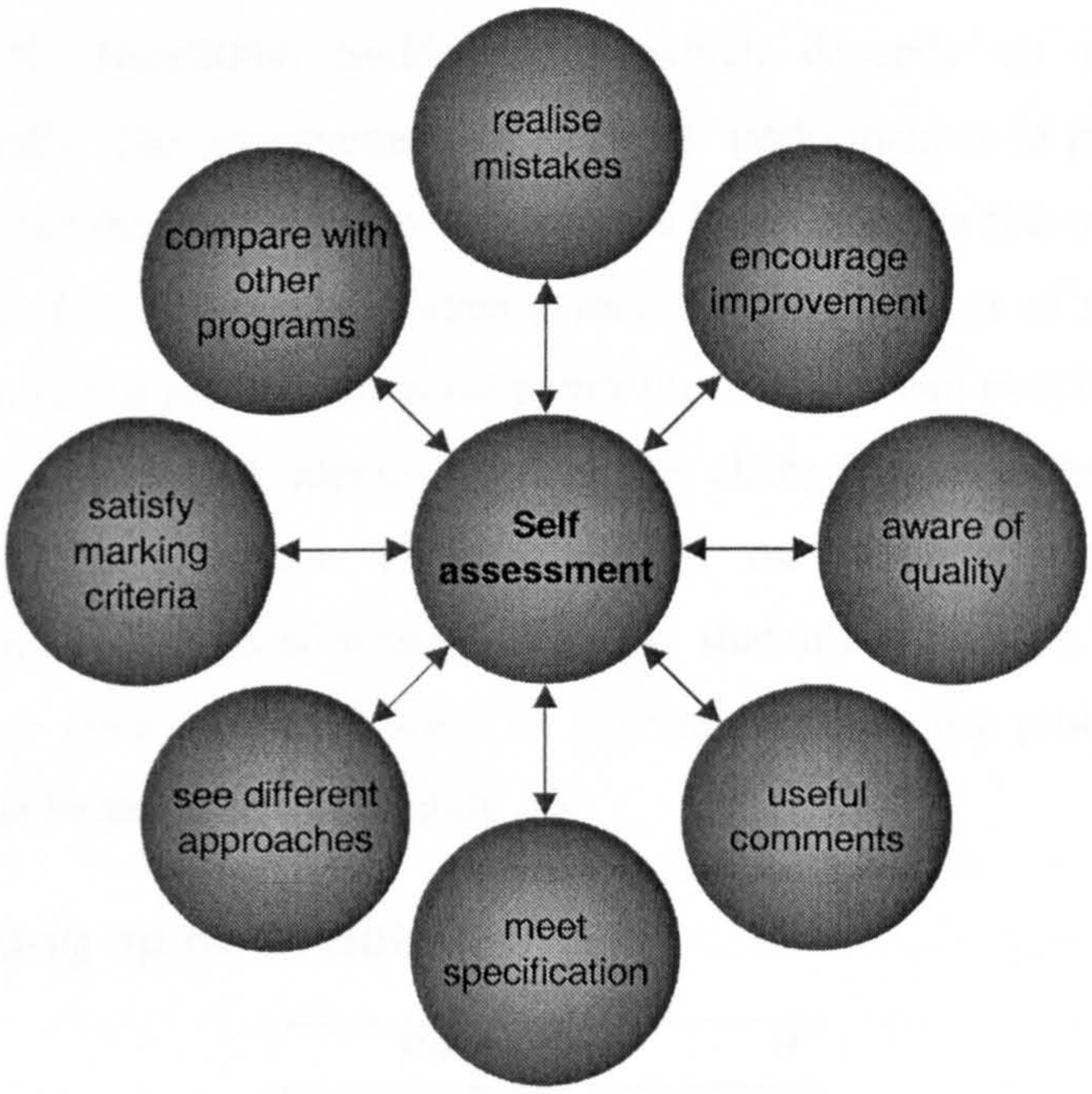


Figure 34 Major students' responses about peer assessment helped them in self assessment

The results from both online questionnaires and interviews indicated that peer assessment encouraged students to evaluate their own work, except students who are experienced programmers. Figure 34 illustrates the students' main responses about how peer assessment helped them to evaluate the quality of their programs. Comparing each others' work is an important link between peer assessment and self assessment that encouraged students to judge the quality of their own works against peers, and particularly to think of improvements to program efficiency.

5.4 Synthesis

Research question: Does the performance of the students improve in subsequent assignments after performing the peer assessment?

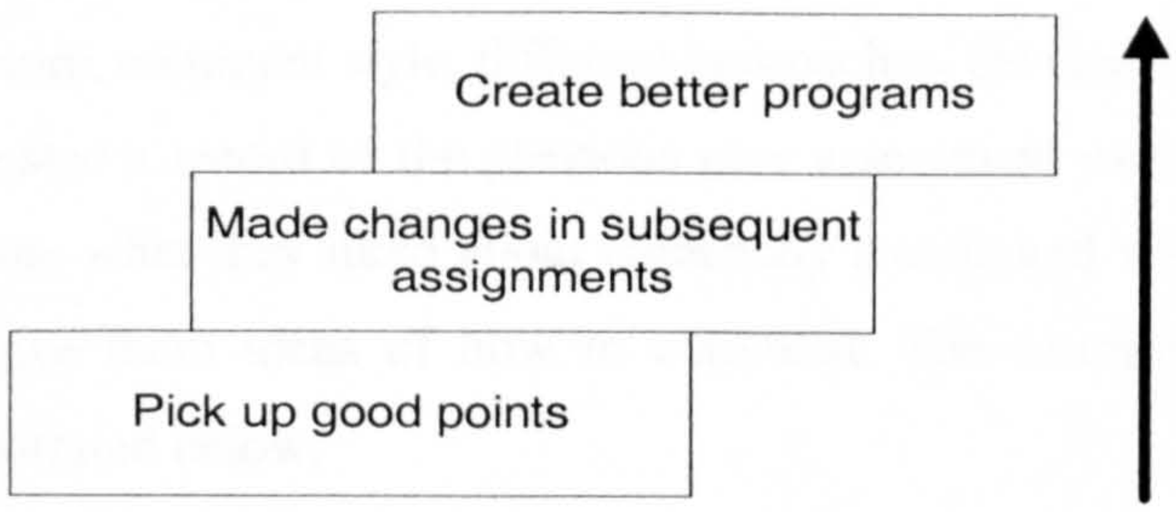


Figure 35 Component sequence of synthesis

Synthesis is the most difficult and complex category [Anderson2001, Corrosion1999, Huitt2004, Seddon1978], which depends on analysis and evaluation skills. The improvement of students' performances in programming, after performing the peer assessment exercises is discussed in this section. Since the difficulty of subsequent assignments increases, the number of marks cannot be compared. In this section, the good points that students can pick from marking other students' work are identified, then the changes that students made in subsequent assignments are analysed, based on students' opinions from interviews and online questionnaires. Finally, students' responses are reported about whether seeing different ways of solving programming problems helped them to create better programs (Figure 35).

5.4.1 Picking up good points

Good points	N
different utilities	12
comment style	11
different approaches	9
how to earn mark	6
program structure	5
new knowledge	5
increase readability	2
receive useful feedback	2
criticise constructive	1

Table 25 Questionnaire results: the good points that students can identify from marking

Analysing each other's work helps students to improve their programming skills. They learnt more about better ways to write more efficient programs with readability, correctness and good programming style. 62 % of students (71 out of 113 students) from the online questionnaire found out many good points that they can pick up from analysing other students' programs, such as different utilities, program structure, comment style, different approaches, etc (see Table 25). Many students requested a record of the previous peer assessment exercise in order to go back and see what they liked about somebody's code and what they did not and also to give them ideas of how to comment. The examples of students' quotes are illustrated below.

- *I picked up on some layout tips from other pieces of code.*

- *I have a better insight into commenting code for clear understanding and structuring code for readability.*
- *Using different utilities to increase efficiency and simplicity, learning new syntax and/or algorithms I may not know.*
- *Structure the code to make it more concise.*
- *You can get some good ideas on how to tidy up section of your code.*
- *I have learned how to criticise constructively."*

Moreover the examples of students' responses from interviews about the good points that they can identify from others students' programs are illustrated in Table 26. The results from interviews are similar to the results from questionnaires. Students saw the interesting different approaches, utilities and programming techniques. They can identify good commenting style and the use of functions to make their programs more readable and easier to follow.

Students' responses from interviews	Different utilities	Comment style	Different approaches	Use of function	Checking exit status	Programming techniques
In my head, yeah. Not like to look at one script and then look at the other, but I knew what I'd written and I knew how I could have changed mine **so it might have been tidier or, you know, more functional**. So I did in my head but not to just look at. Also you get other things, like ..... So yeah, it was useful because a lot of them, I think the people that I marked, most of them did theirs in a different way to mine. **So it was useful to see other ways that it could be approached**. So, yeah it was very useful in that way.			√	√		
Possibly, there was one script that I marked **where I very much liked the commenting style**, and there were some scripts I marked **where they used some utilities I hadn't considered**, that I suppose was helpful.	√	√				
Well it's always good to see the way other people do it because they don't always immediately come to. **Particularly the use of getopts, or lack of use of getopts was interesting** because I considered not using getopts but when I realised exactly what it did I realised it was perfect for the job and used it. So yes **it was interesting to see people how people had implemented it without getopts**. I think it did pretty much everything getopts did. So that was good and that was interesting to read. And there were a few other little bits and pieces like **checking the exit status of previous commands**. Just a few other useful things like that and I though ah yes that's quite a good way of doing things, in fact that may have prompted me to, as I said earlier, check the exit status' of the date command. So, yeah, little bits and pieces here and there, and it's always good to see how other people did things.			√		√	
Em... I just sort of read it through, and then **if there were any ideas it made me think about, then that's very good**. Then I'll remember that and use it the next time I have a similar problem. I found that element very useful. There was... let me think... someone did a very clever loop with the getopt command to iterate through every single option that			√			

Students' responses from interviews	Different utilities	Comment style	Different approaches	Use of function	Checking exit status	Programming techniques
someone had on a line, and then had a boolean variable so it could read them all in and the last one that it read in would be true. **I thought that was very clever. So I expanded on that and put my entire program in a getopt loop, so it worked out pretty well**.						
The **commenting style was a very artistic style, but it was nice to see them**. Other than that I found that **some people knew what they did and had smaller compact ways of doing things**, so I learnt them too, probably I used them in my other scripts. Once you see an example you don't forget it, because it's a mistake of yours and you thought about it before, and there's a solving way. So you learn it and use it in the other examples.		√	√			
I saw different ways of doing it. I hadn't really thought of writing... **some people had written their scripts like in C with functions, which I thought was quite a neat idea**. So they had all the functions and then at the very end they'd have a last command to run the function. I thought that was quite a good idea. **And a few other ways of using different statements, or neater ways of doing things. Like 'getopts',** I never really understood how to use that until after I'd seen someone else's script.	√			√		
Some ideas. If I see a better script than the one I did then **I spend time seeing how this person developed it in a better way, and some parts that were developed in a better way.** I mean, it's not different if he made 4 switch statements and I made 2 and then some other ifs, that's no difference. But if he uses if, and I use something else, then that might be useful to know how this differs.			√			
Mainly to do with commenting, style and layout, and where you put the comments. Yeah, and **there are some simpler and easier techniques for doing certain things in code that I hadn't really thought about before.**		√				√

Students' responses from interviews	Different utilities	Comment style	Different approaches	Use of function	Checking exit status	Programming techniques
Yeah. In the second assignment, there was a use of getopt, that UNIX utility, and I used it in the third assignment, or tried to use it, and it helped in my understanding of it. That was definitely useful. **Comments, and the way people laid out their comments was useful. Like where to put whitespace between a comment and the code. That was useful. The use of functions, as well**, that people used. I only started using functions in the 3rd piece.	√	√		√		
Emm... from the first, I think it was the first or the second assignment, we had to do something for multiple options, and some people just didn't cover that. I think I didn't cover it for the first one, but **it was definitely useful to look how people used different sort of things to look at the problem**, you know getopt, or... just that sort of thing.			√			
Yeah. Even the ones that didn't work, some people had it in a better order, ... They'd put a bit here that linked to something down here, and they'd have it in sections, whereas someone else would have a little bit here and then this bit would go in here. So it was the way people linked the work through, and you could take things like that on board, and then **they'd use a different method to do something**, and OK theirs didn't work, but you could still think "Oh well, obviously they think that method is good, so I'll have a look into why they've used that method, or whatever". It was really good actually; I enjoyed looking through other people's work.			√			
It's just nice to see how other people approached the problem and then they way that they used the tools differently. **It's good to see a variety of solutions, especially if it was something you were having a problems with**, to see how other people tackled that. It's quite useful.			√			
If I went through and I saw, for example, **one person had used some functions, and I hadn't used functions in my code**. I thought afterwards that in some of the situations this was the first piece that we did, so we'd only just come			√	√		

Students' responses from interviews	Different utilities	Comment style	Different approaches	Use of function	Checking exit status	Programming techniques
across them, and I thought that yes that they were quite appropriate for a couple of things, and perhaps they should be used. Whereas he'd used a function for something else and I thought that no I didn't like that**. I could see a problem that he would get, so yeah. It does give you ideas.**						
Ah... Well. I think it was the first assignment, because everyone was new to CS120 and ....., and at that time we found it pretty hard, because ....., and I realised that I had done things in a really long way and then, the script I was marking, then person did it only using ..... I could see that his code was short and precise, compared to mine that was doing the same thing, which was really long. And you could see that he had not put too much effort into it, and yet it was so simple to understand. So it really helped me because **I saw a shorter way of doing exactly what I had done**. **It just made me understand the statements even more, because I could see the way he had applied them, and it made it much easier to understand.**			√			
Yeah. **I would look through their code for ideas as to how I could do mine better, and also how they commented things, was one of the things I looked at because I got some criticism for my commenting**, and I criticised others. I think commenting is one of those things that everybody does differently, whereas the code itself is generally pretty similar, so when you look at the commenting I had a look to see what they'd expect me to put in my code. But some of the coding the way they did it, I think you being to learn what **the best way of doing it is from how they've approached it.**		√	√			√

Table 26 Students' responses from interviews about the good points that they can pick from others students' programs

5.4.2 Making changes in subsequent assignments

> *"**How much I've changed? A lot. Because in the 1st assignment, I'd never done any UNIX, I didn't know to use it properly, I didn't know to gather information and stuff**. So in the 1st one, I basically used my lecture notes, ... By the time we got to the 3rd one, I was using my lecture notes, people, everything that I could, because obviously it was a lot more difficult, **and I'd seen through peer review**, you get to know how much commenting, indenting, where you get your marks, and some things from the 2nd assignment to the 3rd assignment were similar. **So someone might have commented on it in the peer review in the 2nd one, so you could always take that on board, and not do that in the 3rd one, I've got to make sure I do this. It was good like that**."*

The results of changes that students made in subsequent assignment from interview (19 students) are similar to the results of good points that students can pick from marking other students' programs from the online questionnaire (53 students), for example comment style, program structure, different utilities, and different approaches. The above student's quote also confirms that peer assessment influenced him to write better program. The summary of changes that nineteen students made in subsequent assignments after performing the peer assessment is displayed in Table 27.

Changes in subsequent assignments	N
1. Commenting style	**7**
- Emm... I think the 1st assignment, was I think the hardest, because I wasn't sure how we were supposed to do things, **but going to the final assignment I knew how I should comment my code, I knew how to lay it out, how to do it so that other people could see it clearly**, yeah definitely an improvement.	
- It helps me with the commenting style because I can find that this is quite difficult if you have a long script.	
2. Better layout/structure	**7**
- The way people laid out their comments was useful. Like where to put white space between a comment and the code. The use of functions, as	

Changes in subsequent assignments	N

well, that people used. I only started using functions in the 3rd piece.

- First assignment, I didn't lay it out particularly well, I don't think. I don't think it was commented very well either. By the time the 3rd one came around, **I think the comments were quite clearly laid out and it was structured better, and laid out in a more clear manner**.
- Yeah, I'd say so, yeah. Definitely. Just seeing different approaches. **Breaking down things into small tasks**, was the main thing I had to learn to do rather than just try to attack it all.

3. Use more advanced/different utilities **6**

- Emm... from the first, I think it was the first or the second assignment, we had to do something for multiple options, and some people just didn't cover that. I think I didn't cover it for the first one, **but it was definitely useful to look how people used different sort of things to look at the problem**, you know getopt, or... just that sort of thing.
- Yeah, definitely. Because, like, on the 2nd assignment, somebody used getopt and I didn't understand it so then for the 3rd assignment**, I thought that getopt is obviously quite useful, so I tried to understand it.**
- Emm... I think the getopt utility I didn't use before, and I saw some other people using it, and then I decided to use it in the last coursework, I can't quite remember. **But similar things, like utilities, or approaches, or program listing style, and how it looked. I picked it up from other people and used it eventually**.
- I looked at, for example I think it was how someone had done a for loop in the previous exercise and I remembered that and then I just used that myself in the next one. So yes it did help me to positively improve it.

4. Different method/approach **4**

- Yeah, especially in something like PERL because I kind of had to use assembly language commands: chop one letter off at a time, then replace all the colons, and then chop out the As because I didn't have to replace a colon. Things like that. **In their code people had used far more elegant methods of doing it than I had, which was helpful**.
- It did make me pick up on a few errors that I'd made. I don't know how serious they were. Once I'd done my assignment, I didn't look at it again, I didn't want to stress over it, because I couldn't change it at that point. But **it did make me have second thoughts that maybe I should have done it a different way**. The way I approached the last one, I think my way of doing it was rather long. It was very easy to read, but it could have been condensed. Looking at other people's scripts made me realised how I could have shortened it.

Changes in subsequent assignments	N
5. Make the program shorter (increase efficiency)	**2**
- In the beginning I had long statements as other people, but in the last example my code was probably done in less than 100 lines, when people had 500 lines. - **The only thing that was helpful was how to make the program more efficient or shorter**, in a sense. That was better. I did the specification for the automatic and for the others that didn't have automatic tests. And for coursework 3 this data validation and verification that was a lot of people. I mean checking for the dates, some people had 1 to 9 method with square brackets, other people had the number of characters. **That was interesting because you see different ways of doing the same thing. That was helpful.**	

Table 27 Summary of changes that students made in subsequent assignment

Commenting style and program structure are the major changes that students made in the subsequent assignments after they took part in peer assessment exercises. Students were concerned about program readability in order to make their programs easy to follow for the other student markers:

> *"From what I told you... some people did bother to mark seriously, and so that was helpful.* ***In the first coursework, my program was a mess****. I started writing, ..., so indentation was really awful I would say. It was messed with comments, and the spacing was awful. Even my friends who tried to see how it is, they couldn't read ..... And I received, obviously, bad comments as far as indentation and spacing and style was concerned.* ***The second was improved as far as this was concerned, and the third was more improved****."*

5.4.3 Creating better programs

> *"Yeah. Definitely, and* ***I think that's one of the good points, and that's what helped me learn.*** *Because I saw the way that other people solved the problem, and picked up things from there, and come the next coursework I'd include some of their ideas in....."*

From interview, nineteen out of twenty students indicated that analysing other students' programs helped them to improve their own work in the subsequent

assignments. They agreed that peer assessment helped them to learn more about programming, write more efficient programs, and give them other ideas about how to solve the given problems. However one student disagreed because he had been programming for several years, and is more fixed in his own ways. The students' responses on seeing different ways of solving programming problems helped them to write better programs with a variety of reasons as summarised in Table 28 (example of students' responses with the number of responses for each category).

Students' responses	N
1. ***Useful programming techniques/tips***	**4**
- I mean I tried to do it the best way I could do in mine, but there are always small minor things to do, **so I always learn from the peer review really**.	
- Yes, I think it does because, as I now keep saying, seeing other solutions is good and **you ultimately learn different ways of doing things and you can pick the best from your toolkit** and say yep that's the one for the job.	
- Yeah, because **I could see different solutions other people had thought of, and obviously if they thought of a better one, or a smarter way of doing it**, then I'd remember that for the next time that I did it. So it does help in that respect.	
2. ***Encourage thinking about a better solution***	**4**
- Yes**, it made me think about other ways of doing things and making mine better**. **Also made me think about what other people are looking for when they read through a script**, reading through a bit of code, which I've never really thought about because no one's ever read my code before.	
- Yeah. Because you see some things, and you go "Oh, my program wouldn't be able to pass that test if he does it," and "he's done a more general thing and it could pass extra tests," so **it makes you think about yours way**.	
3. ***Compare with their own work and think about improvement***	**3**
- Because you see other people's work, and you could compare it to yours, and see what you could improve. Things like that.	
- I got full marks, well above 90 for all three of them, so **I though my script was good but there were a lot of things that when I looked at other people's I though that I could do that to mine to make it more efficient** and just looking at ways to change things around. I know that there were some of mine that didn't conform fully to the specifications but **looking at other people's and seeing how they'd done it made me think oh yeah I could have done it like that**. It did make me look and mine and look at	

Students' responses	N
theirs and then think how bits of theirs would be good in mine and bits of mine would be good in theirs.	
4. *Give ideas how to solve the specific problem*	**1**
- Yeah. I had problems in specific places, so seeing how other people did those gave me ideas of how to do it.	
5. *Helpful more than seeing different examples from Internet*	**1**
- Yeah, sure. That's always the case. It's how you learn really. And also **they are doing the same thing you do, that's also a good point. If you look for an example on the web, there'll always be something else, so this is concrete** what you should do. **If I set up a course of computing at some point in my life, I will do a version of this, it's really helpful**.	
6. *Encourage thinking of their own mistakes*	**1**
- "Yeah. Well it did make me think so much about what was wrong, and bits that I could do more efficiently.	
7. *Useful for next assignment*	**1**
- Yes. It's definitely more useful for the next assignment. No good for that one, unless you wanted to correct it in some way. Most of the time you just think "OK, **I can use that idea for the next assignment**," and that saves a lot of time. It's definitely helpful for the next assignments.	
8. *Think more about the program specification*	**1**
- Just seeing how people tackle a problem differently can help you look at a problem more laterally and help you improve your work.	

Table 28 Students' responses on benefits of seeing different ways of solving programming problems

In addition, two students said that not only seeing good programs helped them in learning more about programming, but also seeing some bad programs made them realise their own mistakes and helped them to be more analytic of their own work. In addition another student realised that this also helped with other programming languages as the student quote below.

> *"Yes, it has. And **not just in this one either in the other programming ones as well** because if you take out the general points like the structure, the style, its useful for other programming languages as well, because you think well I could do it this way but if I did it this way and split it up into these parts then that would be more reusable and more sort of structured that would look better. So yeah, it is very useful."*

Summary

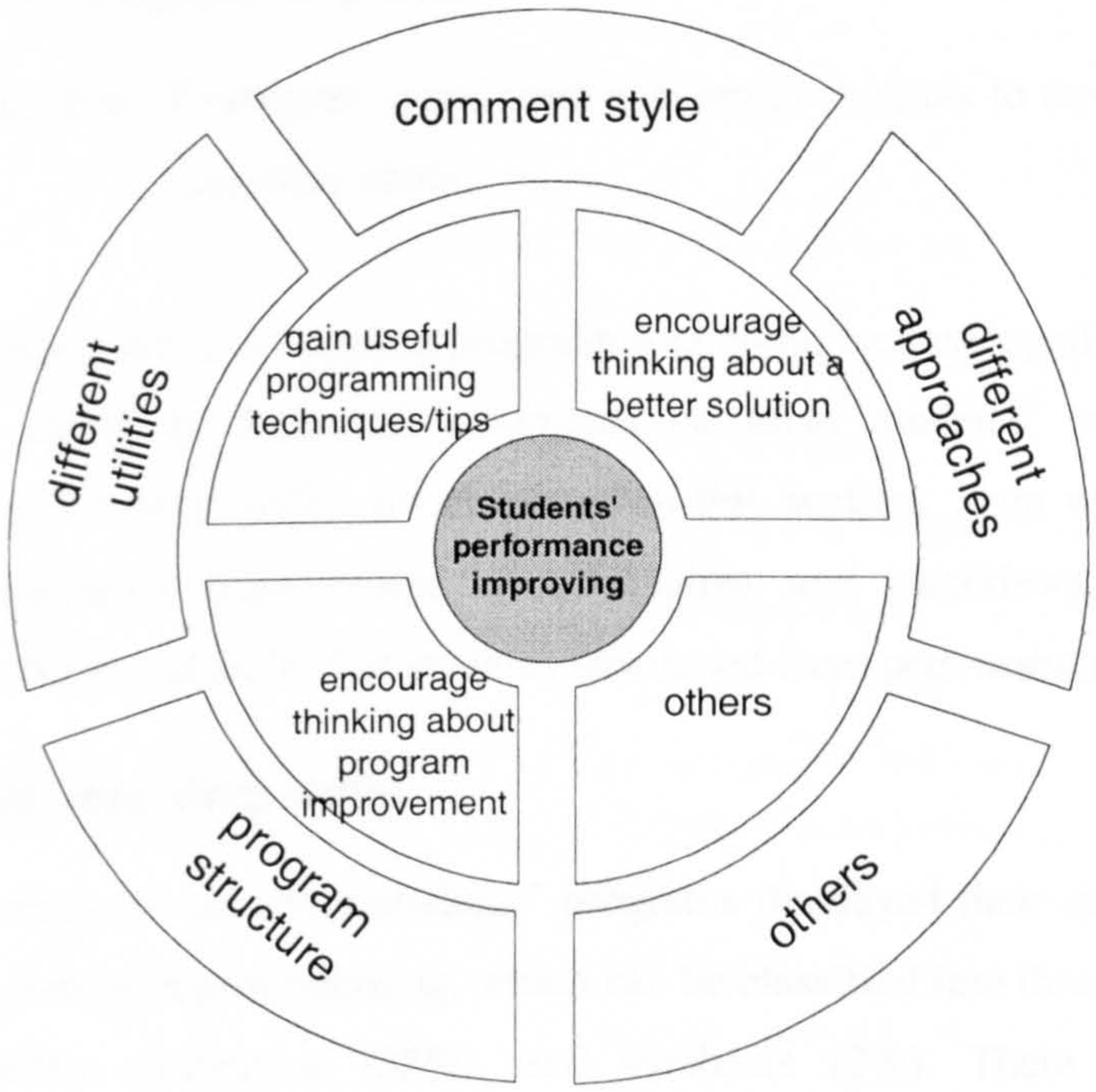


Figure 36 Peer assessment helped students to improve their programming performances

Most of students from interviews (19 out of 20 students) and online questionnaires (71 out of 113 students) agreed that their programming abilities improved after performing the peer assessment exercise. They accepted that their programming styles and strategies were changed a lot from the first to third assignments. Figure 36 presents:

- the major good points that students can pick up from marking other students' programs, which results in changes that they made in subsequent assignments (e.g. different utilities, comment style, different approaches, program structure, etc.);
- the main reasons that students created better programs from seeing different ways of solving programming problems (e.g. gain useful programming techniques/tips, encourage thinking about a better solution, encourage thinking about program improvement, etc).

Therefore peer assessment encourages students to think more about how to create the better programs with readability and efficiency (synthesis).

5.5 Higher cognitive skills

Research question: Does peer assessment encourage students to develop higher cognitive skills?

Higher cognitive skills in learning programming could be developed via the peer assessment approach. Students' comments on other students' work, which indicate deep learning skills, are discussed in this section. Then we report the students' responses from online questionnaires and interviews, about the transferable skills and skills that students developed from peer assessment.

5.5.1 Deep learning skills

Students' comments on other students' programs displayed their deep thinking and understanding in programming, which can be classified into three categories: analysis (66%), evaluation (32%), and synthesis (2%). There is a small percentage of synthesis because this creative thinking ability depends on the analysis and evaluation. Therefore it is the most complex skill, which may not appear in the comments, but may be displayed in writing programs for the next assignment. The detail is shown in Table 29.

	(%)
Analysis	
- Identify causes/fault (program errors/mistakes)	17.41
- Identify the missing statements/methods	7.85
- Identify appropriate utility	13.31
- Find evidence to support conclusions/ideas	4.78
- Analyse the program structure	**19.11**
- Analyse the program command/method	3.41
- Determine the point of view (of method/program)	0.34
Evaluation	
- Judge the validity of program (meet specification)	3.75
- Make judgement based on marking criteria	3.75
- Determine how well the program works (e.g. efficiency, stability, consistency) or judge the quality of program	2.05
- Determine the best solution	1.37
- Judge the style of program (easy to follow)	10.24
- Judge the program readability	10.92

Synthesis	
- Revise program to improve outcome	0.34
- Suggest possible/different solutions	1.02
- Creative thinking	0.34

Table 29 Higher cognitive skills in programming

The level of deep thinking depends on the reason, suggestion or evidence (see Table 30). Each comment may indicate multiple higher cognitive skills. The examples of matching students' comments with Bloom's taxonomy in order to classify them to three categories of deep learning are shown in Table 31.

"With testing your files for readability you need to use 'test -r file' as that specifically tests the readability of the file."	This suggestion is in level 3 – application (surface learning) because the student knew which command should be used in this situation. It is not in the synthesis level because student did not use creative thinking.
"Instead of checking that the file existed (using -e) it may have been better to use open (FILEHANDLE, $filename) or exit 2. Which exits if $filename can't be opened for whatever reason."	The comment displays the analysis skill because student identified the appropriate command, based on understanding (as the analysis level is an extension of the understanding level [Anderson2001]). Therefore this comment shows the deeper thinking about the command than just understanding it.

Table 30 Examples of analysis from students' comments

Students' comments	Bloom's taxonomy		
	Analysis	**Evaluation**	**Synthesis**
The program barely meets the specification. It does not correctly handle the -i or -o options (in that it does not do what those options are meant to do!). ***It does not process input files,*** apart from an attempt to process a .cup file if the -o option is specified, though it does not complete this. ***The script never prints out any file data.*** The script is no where near finished. The script handles some errors such as non-existent files and incorrect options, and it exits with the correct exit status in this event. Only .cup files are checked for the correct format.	Identify missing methods	Judge the valid of program (meet specification)	
Test 7 fails as the scrip is presented with files that need to be converted then sorted. The script that actually deals with the -i option assumes the files have already been processed, rather than processing the files then sorting the results. ***Test 5 fails because the script processes the temporary data file.*** The script handles error messages correctly, in a concise manner, and always exits with the appropriate exit status.	Identify causes of failed tests		
The program would error if a file named john-joe.cup (for example) was provided, as this matches the regexp /-/. Use /^-/ to match lines beginning with -. ***Additionally, when checking file extensions***	Identify the program errors and providing the correct solutions		

Students' comments	Bloom's taxonomy		
	Analysis	Evaluation	Synthesis
/.cup|.mup|.pit/ would match file.cup.rubbish, and as . is interpreted as any non-null/newline character, a file called buttercup would be matched. ***Use something like*** /\\.(cup|mup|pit)$/.			
Despite passing no tests, I believe this is due to windows to Unix text file formatting problem. As a result BOSS is handling the errors and the code is not being run. ***There also appear to be problems with*** running the script in the joshua environment. I suggest you contact Mike Joy regarding the automatic testing issues. The marks awarded are therefore based on simply reading the script. ***I believe that the program met the specification entirely*** when run in it's environment. the code also appears to handle errors correctly.	Identify causes and make conclusions with supporting statements	Judge the valid of program	
Instead of using substr() to get the various info ***you could have used*** preg_match() with () in the pattern match to return the various parts. Program is structured well, it could be made easier to follow by having a bit more white space.	Identify the appropriate utility		
This program is very well structured in that it breaks down the problem into sections. Particularly the error checking and conversion functions ***simplify matters greatly***. The use of functions to check and convert the pit and mup files ***is useful in making the program easy to follow.***		Judge the program style (easy to follow)	

Students' comments	Bloom's taxonomy		
	Analysis	Evaluation	Synthesis
This program will carry out some of the functions specified for the mup utility. When confronted with multiple cup or pit files to process however, ***the code cannot handle and outputs differently to that specified. This mainly seems to be down to the fact that*** once a particular mup or pit file is processed, it is echoed to the temp file then outputted. As a result, the first file is saved to temp and outputted, then the second processed file is saved to temp alongside the first and outputted with it. ***This results in*** the same data appearing more than once.	Identify the problem		
Although the program passes all the automatic tests, it will not pick up all cases of corrupt files. For example, there is no checking that the date of birth is valid. Also some format errors such as too many fields or numerical data in name fields will not be caught. ***There is also no section to validate and process*** mup files, which was mentioned in the specification, probably to allow cup/pit files to be merged with and existing mup file. ***The argument checking seems to be not quite correct***, if "mup -i -o" was called it would try to filter using "-o" as the filter rather than giving an error. ***There is also no handling of*** both -i and -o being specified, i.e. outputting a filtered selection to a file.	- Finding evidence to support conclusion/idea - Identify missing method - Identify the program mistakes		

Students' comments	Bloom's taxonomy		
	Analysis	**Evaluation**	**Synthesis**
The utilities you choose is really makes the program look more complex. There are two examples here. ***You should use the method*** getopts instead of the way you use. It can give a more clearly idea when someone read. ***Another example, instead using*** some if and elif method, you can use case. The case method gives a more structured situation to read and it can show your clear logic.	Identify appropriate utilities		
The use of classes to store the data is a terribly inefficient way of handling the information that is presented in this exercise - ***a simple array of output strings would be much more effective***. Then you could have called the array sort function, which is built into PHP from version 3 and above I believe, which would have finished the job. No classes would be required and only a couple of regular expressions to format the input properly. ***The program is structured in an orderly fashion*** and it is easy to follow what is going on for the most part thanks to the comments. ***One other thing is that there is absolutely no need to use 2 files.*** The php file could have been named mup and had the line "#!/usr/bin/php -q" at the top - there is no need to have 2 files ***and again it is an inefficient approach***.		- Determine the best solutions - Judge the quality of program	
There is also a lot of repeated code throughout the script, a lot of the code could be integrated so that it is only present once - making a	Analyse the program structure		

Students' comments	Bloom's taxonomy		
	Analysis	Evaluation	Synthesis
much more efficient code. ***For example***, the script handles lines that contain module marks and lines with awaited module marks separately. There is no reason why this cannot be done by the same part of the program.			
The program does not handle the last option if multiple -i / -o are used, ***for example the following command:*** /mup -o me -o mine johndoe.cup will just output /Bad data file -i/ and the exit status is 2 and the same thing would happen in case of -o option, ***this could be done by*** including some logic inside the part of the case statement handling -i / -o options like checking if the next option is the same as the current by using the value of the variable $OPTIND in some way.	Identify missing statements /methods with example and provide solution		
It seems you only allow a certain number of data lines in a file of each type. The files could contain any number of lines. ***Try getting the*** number of lines in the file, and using that for iteration, ***rather than*** a fixed number. ***Instead of duplicating most of the script*** for the -i option, ***you could have simply filtered the file before*** output, e.g. loop thought all of your output array, and only keep lines where the code matches the pattern.			- Revise program to improve outcome - suggest different solutions with creative thinking
You are only allowing muppet.mup, rather than *.mup . That is a fair enough conclusion from the background info sheet, but the	Analyse the program command/method		

Students' comments	Bloom's taxonomy		
	Analysis	Evaluation	Synthesis
manual page implies it can take any .mup formatted file. ***It's nice to see proper validation of dates being done***. Leap years are actually those that are divisible by 4 and not divisible by 400, but as 2000 was excluded, that hardly matters.			
Author used lots of utilities here, however, he did not fully understand the functional programming. Because using functions means you can simplfy the main part, and makes your structured clear, you did not, especially in main part. ***Repeatly using case to solve the multiple options is not very reasonable***, may try to write a loop in arguments or use getoptions can make this process much easier. Why don't you write file(line 303) loop in a function? ***which helps improving your programme structure.***	Analyse the program structure		
All tests are passed, ***however there are some mistakes in error handling***. In the case statements that include the pattern matching '*.[pc][iu][tp]', if a file 'johndoe.pip' ***for example*** is read in, the code would not detect an error until control is passed to the next functions. In the specification it states that error checking should be done before any processing is done.	Identify programming mistakes		
Getopts is well known for not properly handling non-options, amongst other failings. Lines are cut and stored in explicitly named	Analyse command and suggest appropriate utility		

Students' comments	Bloom's taxonomy		
	Analysis	Evaluation	Synthesis
variables. ***It would have been far better***, far easier, and far more easy to get to work to use bash arrays. These have been in Bash since version two, and the scripter could have easily used them to loop through all the lines in the file, not just the first thirteen.			
Generally a good program let down by poor commenting and bad layout. The use of subscripts is a good idea, and implemented well. If commenting was improved and the checking of input data was improved a little this would be an excellent program.		Judge the program readability	
The way that the code is segregated into different parts is very helpful in understanding what each section does. ***The switch statement at the beginning is a really clear way of dealing with the options.*** In general the layout and style of programming that has been used, taking into account the use of utilities such as sort within subroutines, ***is very good and made following what the program was doing easier than normal.*** Some of the subroutines were quite lengthy and it may be worth considering using a larger number of more simply defined subroutines to help the program structure further.	Analyse the program structure	Judge the style of program	

Table 31 Matching of students' comments in Bloom's taxonomy (deep learning)

5.5.2 Other skills

The higher cognitive skills and other skills that students developed from peer assessment exercises are discussed in this section. Both results from online questionnaires and interviews are reported. Through the online questionnaires, 129 students' opinions about transferable skills, that peer assessment helped them to improve, are displayed in Table 32.

N=129

Which of following transferable skills did peer review help you to improve?	**YES**	**NO**
Peer review enables me to view and critique a range of programming styles, techniques, ideas and abilities, thus encouraging me to learn from both the mistakes and exemplary performances of peers.	**107**	22
Analysis and evaluation of other students' work helps me to improve my ability in learning programming.	82	47
Peer review helps me to develop the ability to judge the performance of peers.	99	30
Peer review helps me to improve my self-confidence in learning computer programming.	43	86
Reading code skill and understanding of other students' work can be developed in peer review.	95	34
Peer review helps me to develop team working skills by discussing ideas and concepts with peers.	22	**107**
Peer review enhances independent study.	43	86
I have more confidence in my ability to assign marks when participating in the assessment next time.	62	67

Table 32 Transferable skills that peer assessment helped students to improve

Results from Table 32 indicate that peer assessment helped students to improve many transferable skills, including:

- 83% of students agreed that 'peer review enables me to view and critique a range of programming styles, techniques, ideas and abilities, thus encouraging me to learn from both the mistakes and exemplary performances of peers'.
- 77% of students agreed that 'peer review helps me to develop the ability to judge the performance of peers'.

- 74% of students agreed that 'reading code skill and understanding of other students' work can be developed in peer review'.
- 64% of students agreed that 'analysis and evaluation of other students' work helps me to improve my ability in learning programming'.

However 48% of students thought that they had more confidence in their abilities to assign marks when participating in the assessment next time. Only 33% of students agreed that:

- 'peer review helps me to improve my self-confidence in learning computer programming', and
- 'peer review enhances independent study'.

Therefore only 17% of students thought 'peer review helps me to develop team working skills by discussing ideas and concepts with peers'. This may be because of the technical problems with the anonymous communication tool. Students could use this tool only from machines on campus. Therefore there were not many students using this tool to discuss marking.

Table 33 shows the examples of students' responses from interviews about skills that students developed from peer assessment, with the number of responses.

Skills that students developed from peer assessment	N
1. ***Think more critical/logically***	**5**
- **Think more logically and break it down into the simplest steps** and then comment them and write then in a nice style. It helps me come back again later and use the code again later.	
- Being able to criticise and make constructive criticism.	
- **Being able to criticise my own work was one of the most important things**. Being open minded was another thing. And I think the ability to compromise, because when I first did my work, I had my views on one thing, but then after reading other people's things **you tend to think there is another way of doing things and the way you assess them it changes a lot, it varies a lot, so I think it really helped me that way**. But I think the most important thing was criticising my own work, it really really helped me in that way.	

Skills that students developed from peer assessment	N
- I'm much at designing the problem and programs.	
2. Able to think of better approach resolving the given problem	**5**
- The fact that there are different ways of doing things. At first I didn't really think of that, ... Then you think "somebody else used that, so I'll look what that does, and see if I can use that in this situation," and **it taught you that there were a lot of different ways to do one problem, to solve it**. And it taught you that ..... Things like that show you the different ways of programming, and showed you how it can be done.	
- I think it's just looking at how everybody's doing the thing. **There's always when you try to develop your own solution**, is everybody doing it this way, or am I off track with this thing, and especially if it doesn't work, and you definitely need to see how the problem is solved.	
- When I looked at the next piece of coursework I remember to think about how someone else is going to approach marking my piece of work, looking at. Whereas before I would have done it how I thought I should have done it without taking into account how anyone else would... well I suppose a little bit of thought would go into it, but not as much. **Whereas now, I'm aware of how my commenting might need to be structured, and the best way of doing some things, and laying things out as well. So it's changed my approach to the programming assignments.**	
3. More confidence in analysing/marking the program	**4**
- More confident in analysing the program in Unix and Perl.	
- **It's helped me be more analytical of my code and other's**. I've seen really bad scripts and really good script and it helps you to gauge whereabouts your level of coding is, and hopefully show you how to make it better. **It's all about analysing your own code, and help yourself improve your code**.	
- The peer review was very easy to use... ..... And then it was the fact... the 1st one you thought "Oh, I'm giving someone marks," so you weren't so sure, but as you got on, **you got to learn how programs worked, and what was better and what was good, what was expected**. So **looking at someone's work was a lot easier the more you did it**. So you'd look at the first of the 3 codes, and you'd think....., but then the 3rd one would be a mixture of yours and the 2nd one, things like that. **And by the time you got to the marking of the marking, you could go through the script quite easily** and think "Oh that's quite nice, that's good, that's a good point,	

Skills that students developed from peer assessment	N
that's a bad point," and things like that. - if you give me a script right now, I think **I'd do it more confidently, and I know exactly what you're looking for when you're marking my script.** I know exactly what things I have to highlight and really focus on. I know there are key things like my comments, my indentation, my quality, you know. I think that really matters. And I know that whatever I put down, even though....., you'd still appreciate the way that I did as long as my program made sense. I'd still be able to portray it to you and know it wouldn't be wrong, because there is no wrong answer in programming.	
4. Able to read and follow other people's code easier - The ability to read a script as well, and try and follow it in your head, rather than having to run it or compile it or whatever. - I developed a lot of skills and **a kind of way of interpreting other people's code**, that mightn't be as clear as code that's given by a lecturer, which is going to be good. **You have to work a little bit harder sometimes to figure out what it's doing... so that's probably a good skill** to have in following it in that way. - **My analysis of code has gotten better from peer review, from reading other people's code. I find it easier to follow other people's code now.** Because following your own code is far simpler because..... When looking at someone else's code you have to..... It helps with analysing their code and other people's code because you can remember these traits for the future when you're analysing another person's code, **which could speed it up in a workplace environment, or a project in the second or third year**.	3

Table 33 Skills that students developed from peer assessment

Summary

Peer assessment encouraged students to develop higher cognitive skills. Comments on other students' work indicated deep learning skills, especially analysis skills. Most students focused on analysing the program structure and identifying program errors. Students' responses from the online questionnaire suggest that peer assessment encouraged them to develop transferable skills, such as critical thinking, judgement, etc. Students' opinions from the interviews also indicated that peer assessment helped students to develop critical thinking and synthesis skills.

5.6 Critical judgement skills

Research question: Do students display critical judgement skills on evaluation of the initial marking?

In the second step, students analysed and evaluated the 'quality of marking' from the previous step (mark the quality of program). This assessor role is called *feedback marker*. The students who are the feedback markers can see not only the other students' programs, but also the other markers' ideas on those programs. The expectation of this stage was that it should help students to develop their critical judgement skills and encourage them to take the assessor role in the previous stage seriously. This section consists of two parts. Firstly, the students' comments on the 'quality of marking' are discussed about whether they exhibited the critical judgement skills. Secondly, students' opinions on the benefits and problems of this extra step are reported.

5.6.1 Marking 'quality of marking'

Critical judgement involves "being able to go beyond stereotypes, prejudices, preconceptions, and intuitive assumptions to do a rigorous analysis" [Comp2004]. The Teaching and Educational Development Institute [TEDI2004] defines critical judgement as:

- "The ability to define and analyse problems
- The ability to apply critical reasoning to issues through *independent thought* and informed judgement
- The ability to evaluate opinions, make decisions and to *reflect critically* on the justifications for decisions."

However in the context of programming in this particular exercise, critical judgement is *the ability to evaluate other students' marking (initial marking), and make judgements about whether the marking is valid, based on understanding the program being marked.* Table 34 displays the analysis of students' comments on the initial marking and whether they indicated critical judgment skills.

Students' comments for the quality of marking	Critical judgement on initial marking					
	Marking correctness	Feedbacks agreement	Marks agreement	Feedbacks usefulness	Missing points in an initial marking	Analyse program problems
This marking is generally very good. **Despite the high quality of the code, the Marker has still found helpful comments and suggestions, yet have generally marked fairly**. Good. I personally would have deducted a mark or two for ..... There remains one element of confusion, however**... the Marker appears to have advised the Author uses getopts- which they have done.** I don't quite understand what has happened, there, unless the marker simply missed it, which seems unlikely.	√		√	√		
The only thing you missed is the fact that the script assumes that all student are registered in 2000 or later, that means students registered in 1999 (or before) would have id code 2099******.					√	√
I generally agree with what this marker has put. However, I think that the number of comments in the code is not really a lot. Although the comments are rather helpful. The code is also indented properly so should have got full marks. **I think the program is on whole correct. However, it does not seem to** recognise false dates. I think that the program has not selected all the utilities as the code has a lot of repetition.			√			√
I think that everyone has their own style of writing their codes and comments. The marker seems to have a problem with the student writing comments for every part. I personally ..... **Thus I don't think such suggestions are useful for the student**. And the marker said that ..... **It doesn't seem to be true.**	√			√		
I disagree slightly with this markers opinion of the comments used within the program, I feel that more direction could have been given by the marker, in explaining that more effective comments could		√	√	√		√

Students' comments for the quality of marking	Critical judgement on initial marking					
	Marking correctness	Feedbacks agreement	Marks agreement	Feedbacks usefulness	Missing points in an initial marking	Analyse program problems
I also feel that it is slightly harsh to mark the student down fully regarding the correctness of the code, although the code passes none of the automatic tests **it seems as though this may be due to a problem with** the environment that the code is running in, and not down to actual errors with the code. I agree with all the comments made regarding the style and utilities used within the script and they are relevant and helpful to the student who has written the code.						
So much more could have been suggested on the error checking in this script! Incomplete error checking on the id number for each line, no error checking on ..... **Generally error checking on this script is patchy at best! Inappropriate in some parts, for example** ..... **The marker really should have picked up on this and made suggests to the student**, the "code handle errors" section should **most certainly not be worth all those marks, and of course this error checking would have impacts on the marks for the other two sections**, I have only listed a few of the flaws I found while reading this script, I would suggest that ..... **However, well spotted on the weak date checking**		√	√		√	√
Sorting after the 12th byte would not do the same - it should ....., this persons output will be sorted by name. I think this comment should ..... **Also, you have not noticed that the code cannot handle** mup files. The writer had split the code up into cup and pit functions as you suggest that you have but (s)he has not.	√	√			√	√
There were quite a few different utilities that could have been used to make the script better **I do not believe the use of utilities was great, certainly not worth 5 marks**. There were no suggestions in this section when so			√		√	√

Students' comments for the quality of marking	Critical judgement on initial marking					
	Marking correctness	Feedbacks agreement	Marks agreement	Feedbacks usefulness	Missing points in an initial marking	Analyse program problems
much could have been said. **Student could have used getopts in a while loop to make reading the arguments easier.** Separating fields in a file could be far more easily done with a read loop and the IFS set to ":". **Program spots NO errors in a mup file! This was not mentioned nor suggested.** Those are just two that I found, a wider awareness of the utilities available in bash would be advised. The program is indeed structured very well, but again, no suggestions... Can't be blamed for that.						
For the correctness section I think the marks were given too generously. **It doesn't fully meet the specification for MUP, so marks should have been deducted here despite it passing all the tests.** In addition the exit status' were always 0 which is incorrect. **Although the correct output was achieved for errors there wasn't** a non-zero exit status and this wasn't picked up on, so marks should be deducted.			√			√
Good review technique for the readability and style sections, not only pointing out the errors in the code, but also giving tips on how to actually fix the problems mentioned. Valid, insightful comments on the correctness section, but without tips on how to actually fix the problems. Impressive broad thinking in the style section (portability comment).		√		√	√	
Not many suggestions for the student who created the script to improve on readability - you have not specified where indenting is not as it should be - something that the scriptwriter does not currently see. For correctness, it is mostly correct, as you have said. **However, you have failed to pick up on the fact that** ..... For instance..... and therefore ..... **Good suggestion to use**		√		√	√	

Students' comments for the quality of marking	Critical judgement on initial marking					
	Marking correctness	Feedbacks agreement	Marks agreement	Feedbacks usefulness	Missing points in an initial marking	Analyse program problems
For the correctness section I think the marking was a little harsh. The comments made were reasonable but they failed to recognise that all the error statuses were 0. **This should have been highlighted, yet because the code dealt with errors adequately, i.e.** ..... **Further improvements could have been suggested by the marker, e.g.** checking dates aren't in the future.			√		√	√
Valid comments in readability sections, but highlighting specific examples increase its usefulness. **Good suggestions for correctness section, giving some specific ways to improve the code. Nothing useful in the style section, could have commented on use of** temp file and shell sorting **instead of just using** the sort function in Perl etc.		√		√	√	
I agree with the marker's analysis of the script. The suggestions regarding the indentation and commenting are important to the programmer in improving his/her script. **I can fully understand the marker's concern with the use of complex pattern matching rather than REs.** Grep could be used in this situation to ..... The comment regarding the ">" symbol being used to append to the temporary **is another key issue that has been overlooked by some other markers.** The use of sed, grep and awk are features that could be used to improve the script further.		√		√	√	
The marker has not specified the areas in which the author had missed error checking - e.g. especially the date error checking being a single pattern match which would accept 39/19/2999 which is a completely invalid date, and date error checking was specifically mentioned in the specification. **The suggestions seem to be**				√	√	√

Students' comments for the quality of marking	Critical judgement on initial marking					
	Marking correctness	Feedbacks agreement	Marks agreement	Feedbacks usefulness	Missing points in an initial marking	Analyse program problems
quite useful. Though I think the program was not too easy to follow as the case statements were not too easy too follow and the author of the script should have put in useful comments to explain it.						
I do not entirely agree with the marks given for Correctness. Although the marker has identified an inefficiency in sorting lines lexicographically, I believe the program will sort data appropriately in most situations - nevertheless they are still correct that ..... **The marker makes some remarks on how option handling has missing features.** However, from my knowledge 'Getopt::Long' will return the appropriate error when arguments are missing, and the author of the script has actually commented this functionality in line 17. **Delivering inaccurate remarks such as this would not be helpful to the student.**	√		√	√		√
Agree with the marker's readability not a lot of suggestions to give to the marker might be a bit informal commenting about self in the commenting, but ..... **Agreed with the correctness pointed out the MUP running is missing. Agreed with the marker's suggestions, definitely point some thought into the marking.** Could have explained how getopts could have worked. **Furthermore could have pointed out the author has not used** $array[0] to access elements but used @array[0]. Will not throw up an error in compiling but if put a parameter -w it will then throw up an error.	√	√		√	√	√

Table 34 Students' comments for the 'quality of marking' and indication of critical judgement skills

Results from Table 34 indicated that marking the initial marking helped students to develop critical judgement skills, based on the analysis of the program problems. Then students can evaluate other students' marking, such as evaluate the valid feedbacks and marks, usefulness of feedbacks, and identify missing points in the initial marking.

5.6.2 Students' opinions

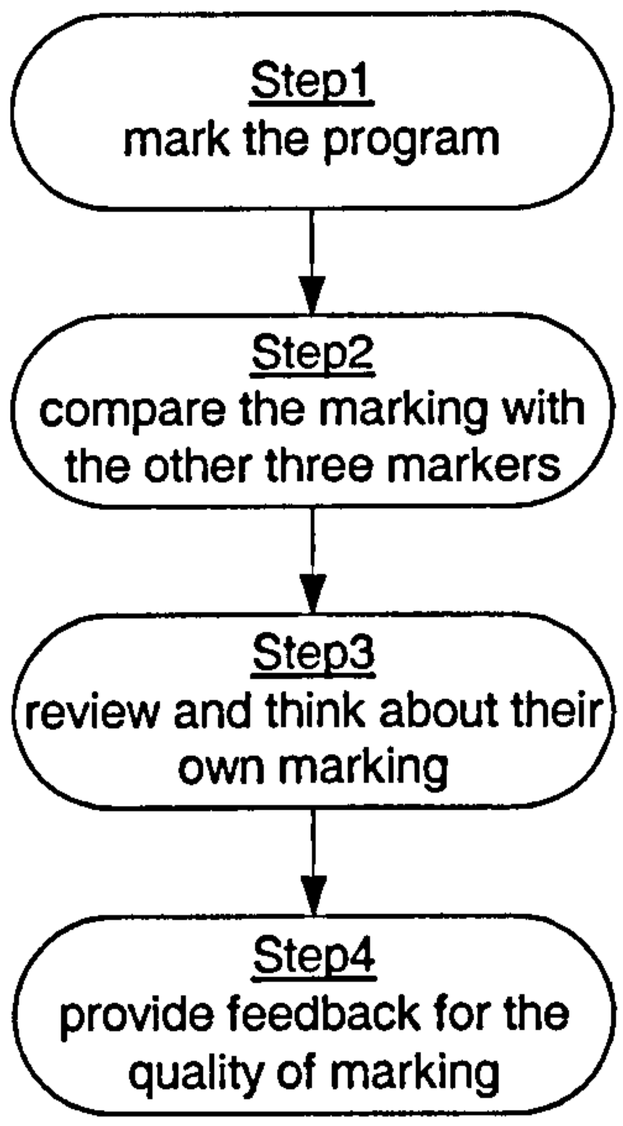


Figure 37 Marking the 'quality of marking' process

Twenty students were interviewed about the benefits and problems of this marking the 'quality of marking' step. Nineteen students agreed that this step was very helpful, and some of them also brought up the problems, which they should be aware of in this step. However one student did not see any benefit of this step. Twelve students mentioned that before making judgement on the other students' marking, they read through the programs and marked by themselves first and then compared their own marking with the other marking (see Figure 37). Students' opinions on the benefits and problems of this step are summarised below.

5.6.2.1 Benefits

> *"I think it's the most beneficial out of all of the steps. Rather than marking a script without having anything to go on, **I think step 2 is more useful, because you already have someone with an opinion, and then***

> ***you have yours, and you think about it**, so I think, to be honest, it's the most beneficial of the steps."*

In the first step, students can compare the three programs, but in this step they can compare the three other students' marking of the same script. Many students also compared their own marking with the other students' marking. They could see how the marking was different and relate it to their own marking. They knew what other students thought about the program. They could see the different issues that the other students picked up, which they might have missed. They saw where they would agree and disagree with comments from the other markers. It gave them more ideas about what other students thought about the code and marking. The above reasons suggest that peer assessment forced students to think critically. Students' opinions on the benefits of step 2 and the number of students who gave these opinions (one student may give more than one opinion) are summarised in Table 35.

	Students' opinion	N
1.	***Marking control***	11
	- A controlling mechanism helps to make the markers do their job carefully in the first step.	
	- The marking of the marking should be used to adjust the mark awarded. I think it could be helpful for ensuring the marks are fairer.	
	- It is a sort of moderation of the marks; you can be punished for not taking it seriously (students get marks on how much of effort they put in).	
	- The marking the markers is useful in weeding out the comments that aren't justified.	
2.	***Thinking more about marking***	9
	- It gives you a chance to see what other students think of when they mark, to make you think of different things when you are marking.	
	- It is interesting to see how other people mark things and pick up different issues.	
	- It helps you see what other people have thought of programs and you can look back on that or agree or disagree with that correspondingly.	
	- If they're marking similar to you, you know you're on the right track. Then if the marks differ slightly it doesn't really matter, because you miss something that they've spotted.	
	- It helps to see what other people are looking for when they look through the code, mostly likely not the same things that everybody's going to look for.	

	Students' opinion	N
3. Learning more about commenting		**8**
	- It helps to expand my comments when I see what other people say about them.	
	- Learning more about commenting because I know what not to do, and what is good.	
	- Help to correct the comment, realise the mistakes that the other person has picked up on.	
	- Sometimes a student might put in something, and I might have a different opinion, and being able to know that, and since I had a different opinion, being able to see what he's trying to say, and putting in my work I think helps more.	
	- Everybody's going to have their own style, but they can always learn from other people and adapt to new things, like your commenting, how you're going to mark in the future, that kind of thing.	
4. Better understanding of program		**4**
	- You know lots of different ways of coding; it gave you a better understanding of program.	
	- You can see how you've gone wrong and what way you could have done it better.	
5. Helping in subsequent assignments		**3**
	- Like some people would say something about it and it would bring that to your attention, when you were doing your work you'd make sure that you did that.	
	- It highlights some problems in a way that you might have missed, that maybe you won't miss them again.	
	- I do read the comments and remember them, I'm trying to apply them to subsequent projects which is helpful.	
6. Increasing confidence in marking		**2**
	- It was really good to see what other people were saying about other peoples' work, because the fact that there were other markers as well, meant you weren't as nervous giving out your mark.	
	- It's the progression of marking. You'll learn to understand what's good and what's bad, and that's the only thing that makes the marking aspect less scary.	
7. Others		**4**
	- You learn what the average level of your fellow students is.	
	- It helps me to develop the ability to acquire knowledge through this process, which means that to develop other skills that are transferable, not just as far as programming is concerned.	

Students' opinion	N
- Benefits for the other students, help the other students in improving their programming abilities and providing feedback skills by suggestion of a different way than they had. - Students who were the markers in the previous step received the feedback on the quality of their marking from the feedback markers in this step	

Table 35 Student's opinion on the benefits of step 2 – mark the 'quality of marking'

Marking the 'quality of marking' not only encouraged students to take the assessor roles in the previous step seriously, but also helped them to think more about programming and marking, which results in improving the quality of commenting, better understanding of programs, helping them in doing the next assignment, increasing confidence in marking, etc. Students' quotes are illustrated below.

- *"**It does encourage people to spend more time and do a proper job of marking in the first place, which is very useful.** It's good to see other people's comments and what they've said about a particular script, and compare that to what you would have said about it, and **see if there's faults that you'd pick up and they haven't, or you wouldn't have spotted and they had**. So it's quite useful from that point of view as well."*
- *"Yeah, because **sometimes I may have not thought of it, and seeing another person saying 'oh, they could have improved it like that,' makes me think even more "why didn't I think of such a point?" and he actually makes sense.** It's very useful because I get to see what other students are thinking as well, so it really does help actually. Step two really helps to be honest."*

However many students were concerned about their marks. They thought this step gave them a chance to improve the commenting style in order to get a good mark on the next assignment. One student said that he can improve his comment style, and see what people do like and do not like to hear in order to get a good mark for the quality of marking.

*"**You can see how other people are thinking and you get the chance to think**, well actually this what's important to people and if I want a good*

mark on the next one then ***this is what I have to, sort of improve****. I mean, if you're saying something about some technical aspects then I think it can be quite useful because obviously they're probably going to* ***give a different way that they could have done it****, because you have to suggest an alternative, they're probably going to give a different way to do it than you are so you get more ways of doing it."*

5.6.2.2 Problems

A few students who disagreed with the benefits of this second step, focused on the low or unfair marks that they received from the harsh feedback markers. They thought the feedback markers did not pay attention when marking their work because there is no mark reward for this step. The student who did not see any benefits of this step was not satisfied with his marks because two feedback markers had not marked his work (quality of marking) but one feedback marker simply tried to find faults with his marking and gave him a low mark. Therefore he suggested that marks from this step were unfair. Students suggested three main problems of this step:

- it is not accurate marking;
- nothing to mark because of lack of feedbacks;
- students did not mark properly because there was no mark award for this step.

The examples of students' quotes with the numbers of opinion are displayed in Table 36.

Students' opinion	N
1. It is not accurate marking	**4**
- No matter how well you mark it, there's always going to be someone who marks you down for what you've written, you could drop marks because someone doesn't like what you've written whereas someone else could think it's brilliant.	
- Marks from this step was unfair, it depends how well they've actually looked into it.	
- That's not so good for accurate marking; it depends on a strict or lapse marker.	
- The questions are very subjective; they're based on people's opinions	

Students' opinion	N
rather than easy things that you can say.	
2. Nothing to mark because of lack of feedback	**3**
- The problems occur if people don't do it, or don't take it seriously, but I don't know how you avoid that.	
- People just don't put enough effort into giving the feedback in the first place. So that's the main problem with marking the feedback: there's not much to mark.	
- Some people just click on the bullets and give marks without writing suggestions.	
3. Students did not mark properly because there was no mark award for this step	**2**
- I suppose the main problem with that is that people won't listen, that's about it. It's very easy to be very blasé, and that's the end of it, you've already got the marks so you can put that behind you.	
- I just think that in the first step you look through something and you do it in detail because your going to receive a mark for your feedback whereas in the second stage you don't, it doesn't have any bearing at all.	
4. Others	**3**
- The problem is that if everyone did it right, there's not a lot you can pick up on.	
- The problems arise in the actual marking of the work itself, and people making wrong or right comments there.	
- The comments they're very much, a lot of them are opinions, I think this looks messy, a lot of it you're mostly just commenting on other people's opinions and saying whether you agree or disagree.	

Table 36 Student's opinion on the problems of step 2 – mark the 'quality of marking'

One student did not think this step helped him much in improving his programming analysis ability because he had in his mind what he should do, therefore the other students' marking did not help him much. Another student thought it was good enough that he had been looking at other students' code in the first step and he can discover for himself how to improve his code. In addition, one student thought this step forced him to be more lenient in the previous step in order to get good marks from this step.

> *"Although personally I took it to heart, and if somebody said I was being a bit harsh, I made a concerted effort to be a bit more lenient or be less picky."*

However one student disagreed with this step because it was not always necessary to penalise someone for their opinion because everyone has got different opinions about programs. Another student suggested discussion on this step instead of marking. Two students recommended that the feedback from a tutor is more beneficial than the feedback from their classmates, and students will accept it (especially the negative feedback).

> *"Em... Marking the quality of the program, yeah. Marking the quality of the marking, it's a good way of vetting what people have said, but whether people pay any attention to it, is another matter. Because I know personally when I was given some negative feedback for my marking, I just sat there and said* ***"Well, I don't agree with that, you're wrong, I think my marking was alright,"*** *so that perhaps not.* ***Comments, maybe from tutors, on marking would be more beneficial, because we would be more inclined to listen to it, because it's not just some random stranger who could know nothing about the subject."***

Summary

Evaluation of initial marking encouraged students to develop critical judgement skills as they could make a judgment of whether the marking is valid, such as identifying the marking correctness, comment usefulness, missing points in the comments, etc. 95% of students (19 out of 20 students) thought marking the 'quality of marking' not only encouraged them to take the assessor roles in the previous step seriously, but also encouraged them to think more about marking (critical judgement). However marks should be awarded for this step in order to encourage proper marking.

5.7 Summary of the chapter

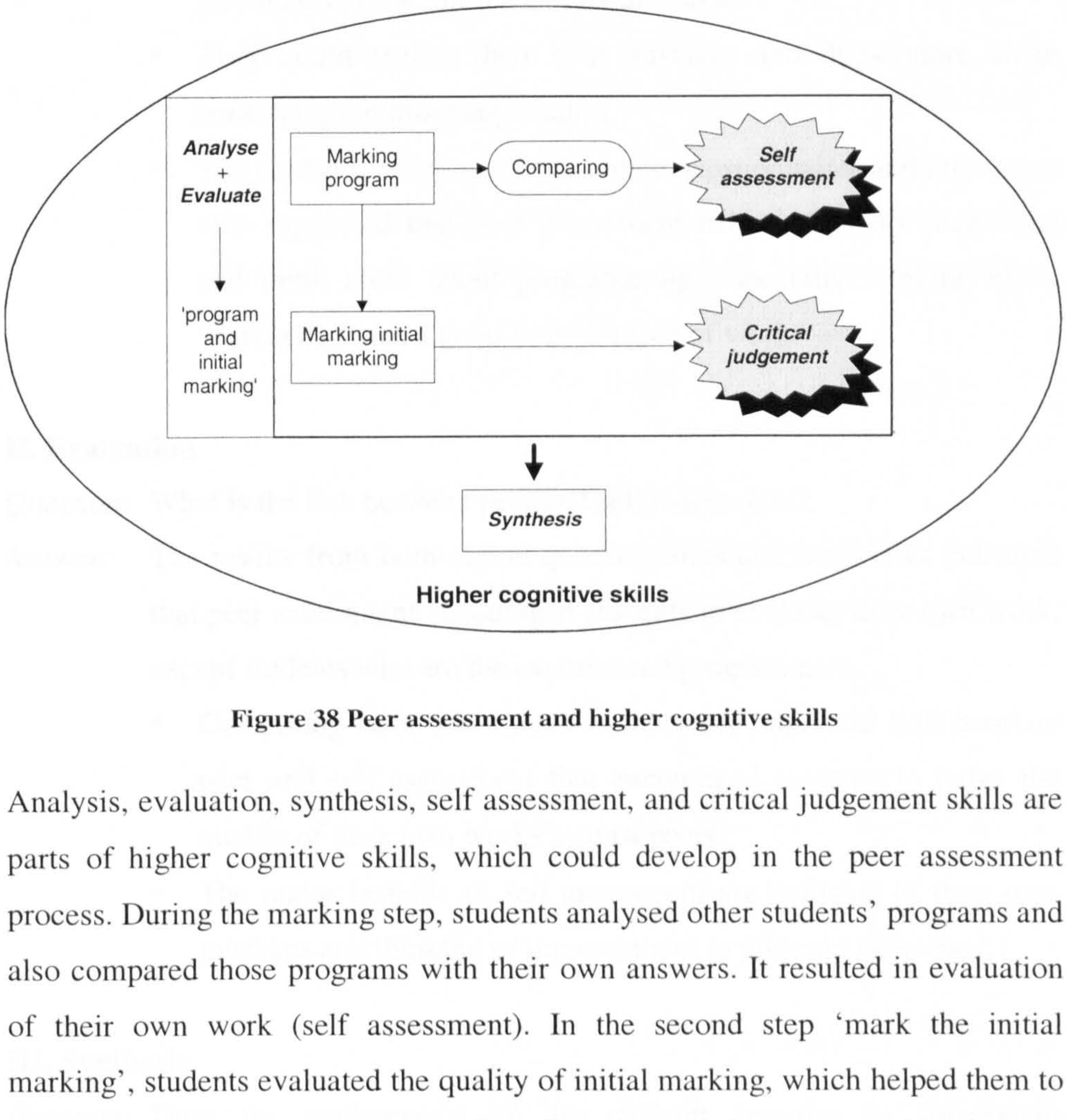


Figure 38 Peer assessment and higher cognitive skills

Analysis, evaluation, synthesis, self assessment, and critical judgement skills are parts of higher cognitive skills, which could develop in the peer assessment process. During the marking step, students analysed other students' programs and also compared those programs with their own answers. It resulted in evaluation of their own work (self assessment). In the second step 'mark the initial marking', students evaluated the quality of initial marking, which helped them to develop critical judgement skills. Then all of these skills encouraged students to think of better solutions and create the efficient programs (synthesis). These results are supported by the students' responses from online questionnaires and interviews, and the analysis of students' comments. The answers of individual sub research question are summarised below.

I. Analysis

Question: Does peer assessment help students to understand and think more about programming?

Answer: Yes, results from analysing other students' programs and their own works indicated that peer assessment helped them to understand and think more about programming.

- They could identify the program problems, appropriate utilities, and analyse program readability and style.
- They could realise their own mistakes and think more about better programming approaches.
- The students' responses from online questionnaires and interviews also suggested that peer assessment helped them to understand and think more about programming, especially thinking about what constitutes a good or poor piece of work.

II. Evaluation

Question: What is the link between peer and self assessment?

Answer: The results from both online questionnaires and interviews indicated that peer assessment encouraged students to evaluate their own work, except students who are the experienced programmers.

- Comparing each other work is the most important link between peer and self assessment that encouraged students to judge the quality of their own works against peers.
- The major benefits of self assessment are realising of their own mistakes and thinking of improvement to program efficiency.

III. Synthesis

Question: Does the performance of the students improve in subsequent assignments after performing the peer assessment?

Answer: Yes, most students (95% from interviews and 63% from online questionnaires) agreed that their programming abilities improve after performing the peer assessment exercises.

- They could pick up the good points from marking and make changes of their subsequent programs, such as appropriate utilities, comment styles, better approaches, program structures, etc.
- Peer assessment helped students to create better programs because they gained useful programming techniques, which encouraged thinking about better solutions and improving the quality of their own programs.

IV. Higher cognitive skills

Question: Does peer assessment encourage students to develop higher cognitive skills?

Answer: Yes, peer assessment encouraged students to develop higher cognitive skills.

- Comments on other students' work indicated deep learning skills, especially analysis skills. Most students focused on analysing program structure and identifying program errors.
- Students' responses from the online questionnaire suggested that peer assessment encouraged them to develop transferable skills, such as critical thinking, judgement, etc.
- Students' opinions from the interviews also indicated that peer assessment helped students to develop critical thinking and synthesis skills.

V. Critical judgement skills

Question: Do students indicate their critical judgement skills on evaluation of the initial marking?

Answer: Yes, evaluation of initial marking encouraged students to develop critical judgement skills.

- They could make a judgment of whether the marking is valid, such as identifying marking correctness, comment usefulness, missing points in the comments, etc.
- 95% of students (19 out of 20 students) thought marking the 'quality of marking' not only encouraged them to take the assessor roles in the previous step seriously, but also encouraged them to think more about marking (critical judgement).

Chapter 6 Accuracy of peer assessment

Research question: Can peer assessment be an accurate assessment method in a programming course?

6.1 Overview of the chapter

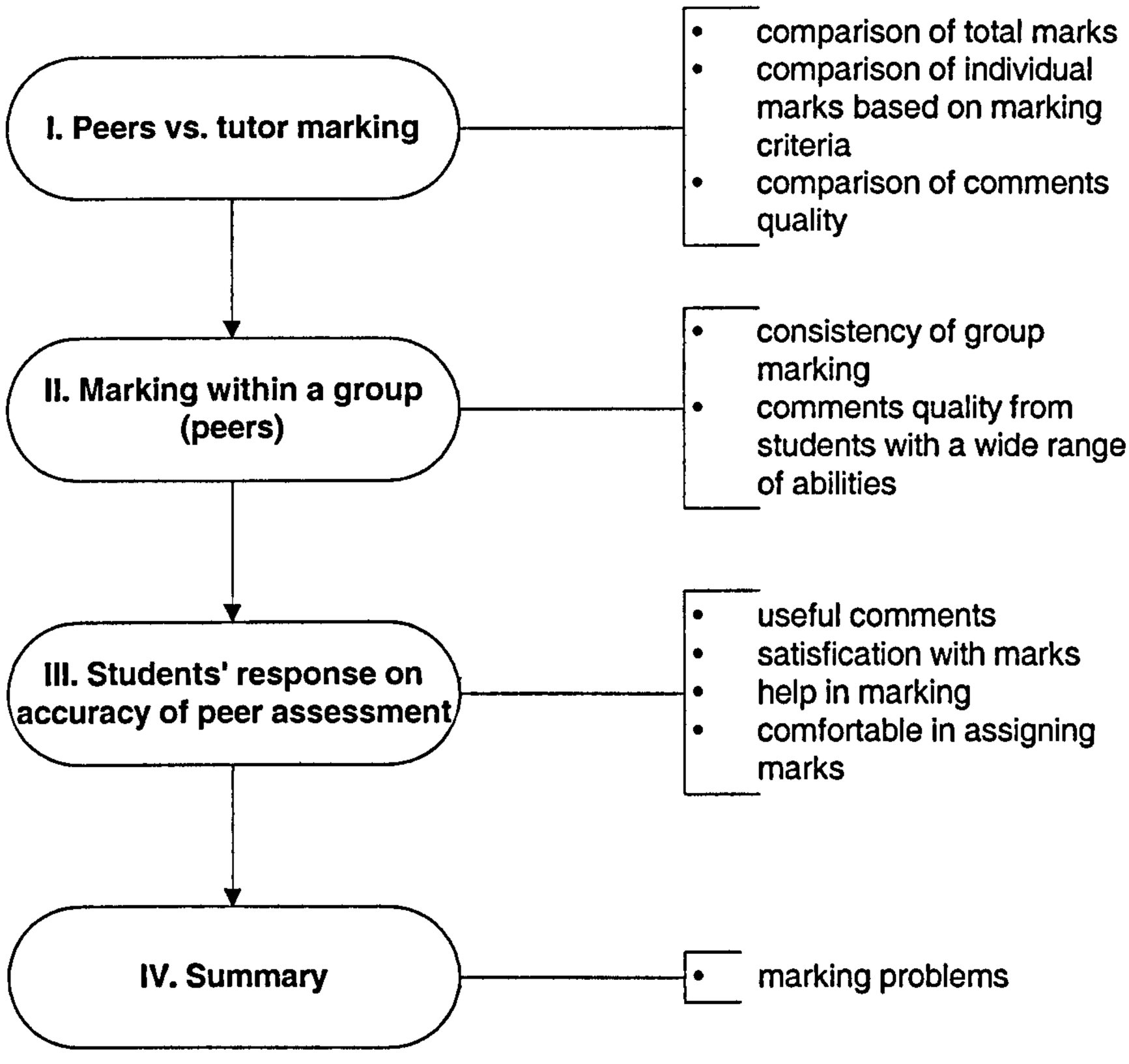


Figure 39 Outline of 'accuracy of peer assessment' chapter

Figure 39 displays the outline of this chapter. The comparison between peer and tutor marking is examined by starting from the analysis of the total marks to the details of each mark and comments according to the marking criteria. Then marks within individual groups also are analysed to find out the consistency of group marking, and the marking from students with different abilities is analysed. Besides the analysis of marking performances of both peers and tutors, students' opinions about satisfaction with their marks, useful feedback, and marking (from questionnaire and interview) are discussed. The following sub

research questions and the questionnaire and interview analysis questions are considered.

Sub research questions

1. Is there a relationship between the marks given by the tutor and peer markers?
2. Is there a significant difference between the marks given by the tutor and peer markers?
3. For which marking criteria are peers' marking as accurate as tutors' marking?
4. Do peers comment on similar or different issues compared to tutors?
5. Are marks within a group of students consistent?
6. Are the comments from students with a wide range of abilities different?

Questionnaire and interview analysis questions

1. Are the comments from peers useful?
2. Are students satisfied with marks from peer assessment?
3. What are the facilities that help students in marking the program?
4. Do students feel comfortable when assigning marks?

N = 166

		Range	Minimum	Maximum	Mean	Std. Deviation
Assignment1:	Peer marks	82.00	17.00	99	77.86	16.79
	Tutor marks	50.50	23.50	74	60.97	10.80
Assignment2:	Peer marks	81.00	18.00	99	84.27	12.46
	Tutor marks	79.00	9.00	88	77.77	10.16
Assignment3:	Peer marks	48.00	52.00	100	85.95	8.83
	Tutor marks	46.50	39.5	86	77.90	5.92

Table 37 Descriptive statistics of tutors and peers' marks in 3 assignments

An "*expert*" is represented by a tutor for this module who has knowledge and experience in marking of shell programming. Table 37 shows the summary statistics of tutors' and peers' marks on the quality of program, based on 166 students from 3 assignments. Each mark is out of 100. Each of the marks in the 3 assignments is calculated from the average marks, given by 2 tutors (tutor marks) and the average marks given by 3 students (mixed range of student ability – peer marks). The average peer marks in all of 3 assignments are higher than tutors' marks because of a variety of reasons. For example, some students were overly generous with the mark they awarded in the hope that this would be reflected in the marks they received, and students tend to give more marks when they did not fully understand the scripts.

> *"I suppose I did in a way to assign marks, but I didn't really know how to mark mine in the first place, so it was hard to. I usually tended to give slightly more marks if I was erring, I'd usually err on the side of give more marks rather than less marks for a script."*

In this experiment the average of peers' marks is higher than the average of tutors' marks: 17%, 7%, and 8% in assignment 1, 2 and 3 respectively. The different of average marks decreased from assignment 1 to 3 because students had experience in marking from the first assignment. They knew exactly what the markers are looking for and they learnt more about how to mark properly.

> *"The 1st peer review, the marking is like "Oh my god, I'm marking someone else's work," whereas by the 3rd one you're like "They don't deserve that many marks, and I can't give it to them."*

Assessing normal distribution

	Skewness		Kolmogorov-Smirnov		
	Statistic	Std. Error	Statistic	df	Sig.
Assignment1 Peers' marks	-1.354	.188	.137	166	.000
Assignment1 Tutors' marks	-1.567	.188	.212	166	.000
Assignment2 Peers' marks	-2.582	.188	.182	1661	.000
Assignment2 Tutors' marks	-4.564	.188	.273	166	.000
Assignment3 Peers' marks	-.987	.188	.099	166	.000
Assignment3 Tutors' marks	-3.303	.188	.143	166	.000

Table 38 Tests of normality of total marks from 3 assignments

It is important to study the shape of the distribution of data. Ideally for most statistical techniques, a distribution should be symmetrical and normally distributed (bell-shaped) [Howitt2003]. Assessing the normality of the distribution of marks is displayed in Table 38. The result of the Kolmogorov-Smirnov test indicates that the distribution of marks is non-normal because the Sig. value is less than .05. The Sig. value is less than .0005 (Sig. value .000 is rounded to 3 digit [Pallant2001, Bryman2001]) for each group, suggesting violation of the assumption of normality. This is quite common in large sample sizes [Pallant2001, Hair1998]. Moreover the value of Skewness of each group is not equal to zero, which means a set of marks is asymmetrically distributed [Cramer1998]. The distribution of marks is negative skewed because most of the marks cluster to the right and there is a long tail to the left (see Figure 40). However the bell-shaped curve of each group in Figure 40 looks closely symmetrical and the sample size is large enough (more than 100). Therefore it can be assumed approximately normal distribution [Hair1998] and it is reasonable to use statistics that assume a normal distribution [Chatfield1983].

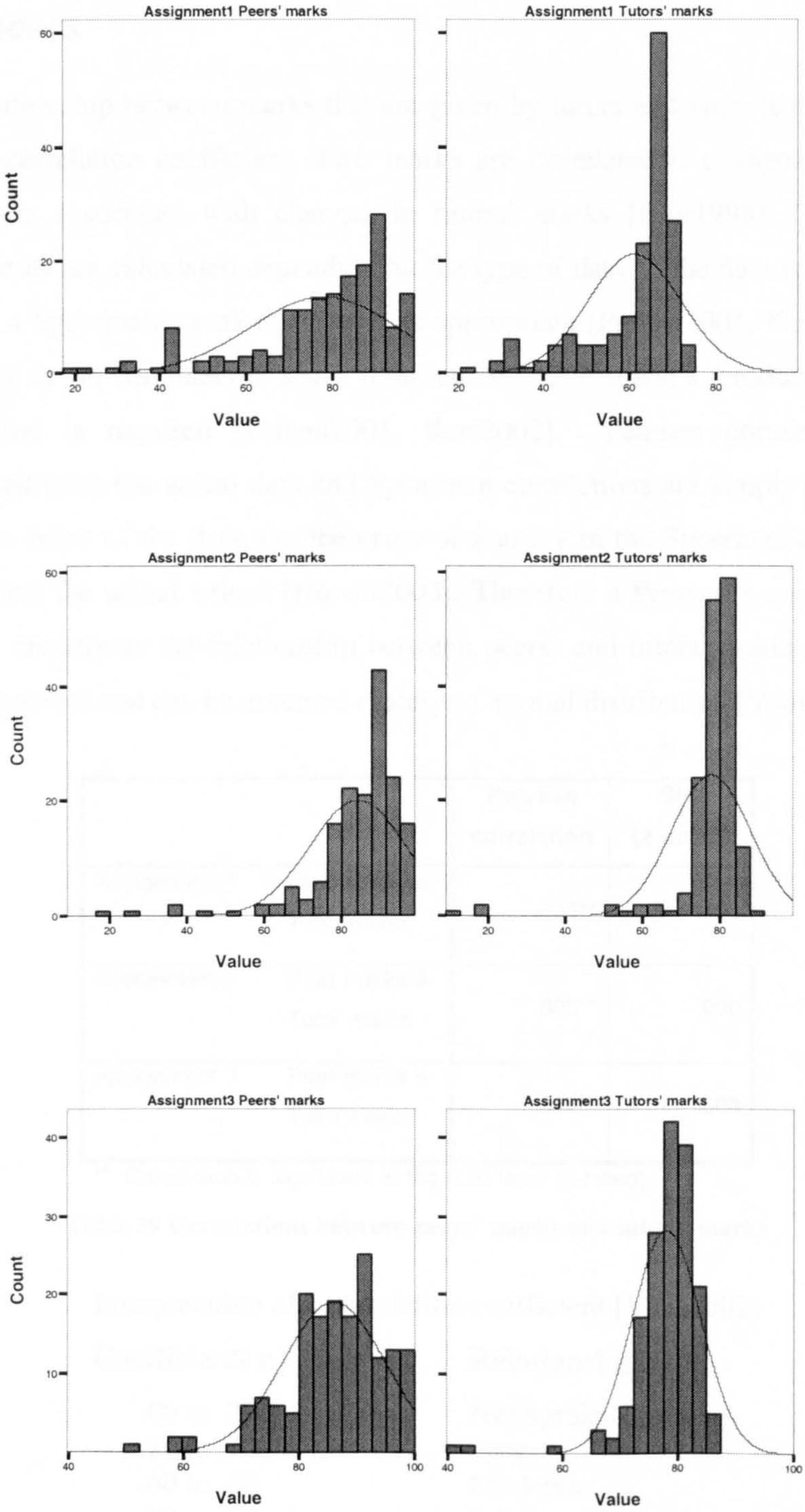


Figure 40 Histograms of peers' and tutors' marks in assignment 1, 2, and 3

6.2 Relationship between marks given by the tutor and by peers

The relationship between marks that are given by tutors and peer, is represented by the correlation coefficient. Two marks are correlated if changes in peers' marks are associated with changes in tutors' marks [Hair1998]. Correlation coefficients are calculated depending on the type of data. If the data is ordinal or ranked, a Spearman's rank correlation is appropriate [Pallant2001, Kerr2002]. If the data is on an interval scale (continuous), a Pearson's product moment correlation is required [Pallant2001, Kerr2002]. Pearson correlations are calculated from the actual data and Spearman correlations are simply calculated from the order of the data, i.e. the order of x and y in the Spearman coefficient rather than the actual values [Howitt2003]. Therefore a Pearson's correlation is used to investigate the relationship between peers' and tutors' marks, since the data is interval and can be assumed closely to normal distribution [Vaus2002].

N = 166

		Pearson correlation	Sig. (2-tailed)
Assignment 1	Peer marks & Tutor marks	.848**	.000
Assignment 2	Peer marks & Tutor marks	.809**	.000
Assignment 3	Peer marks & Tutor marks	.617**	.000

** Correlation is significant at the 0.01 level (2-tailed).

Table 39 Correlations between peers' marks and tutors' marks

Interpretation of a correlation coefficient [Vaus2002]

Coefficient(r)	Relationship
.00 to .20	Negligible
.20 to .40	Low
.40 to .60	Moderate
.60 to .80	Substantial
.80 to 1.00	High to very high

Pearson correlation coefficients in 3 assignments (Table 39) are positive and substantial ($r > .60$). The positive correlations achieve a high level of statistical

significance because $p < .01$ [Miller2002] (a significance level of .000 means $p < .0005$) [Best1989, Pallant2001]). There was the strongest positive relationship between peers and tutors marks in assignment 1 [$r=.85$, $n=166$, $p < .0005$]. The marks that are given by tutors are high; the corresponding marks that are given by peers are high as well. The scattergrams in Figure 41 give strong evidence of a high positive correlation between tutors' marks and peers' marks in assignment 1, 2, and 3, respectively, because all the points neatly arranged in a narrow shape. An upward trend indicates a positive relationship, high marks on X (tutors' marks) associated with high marks on Y (peers' marks).

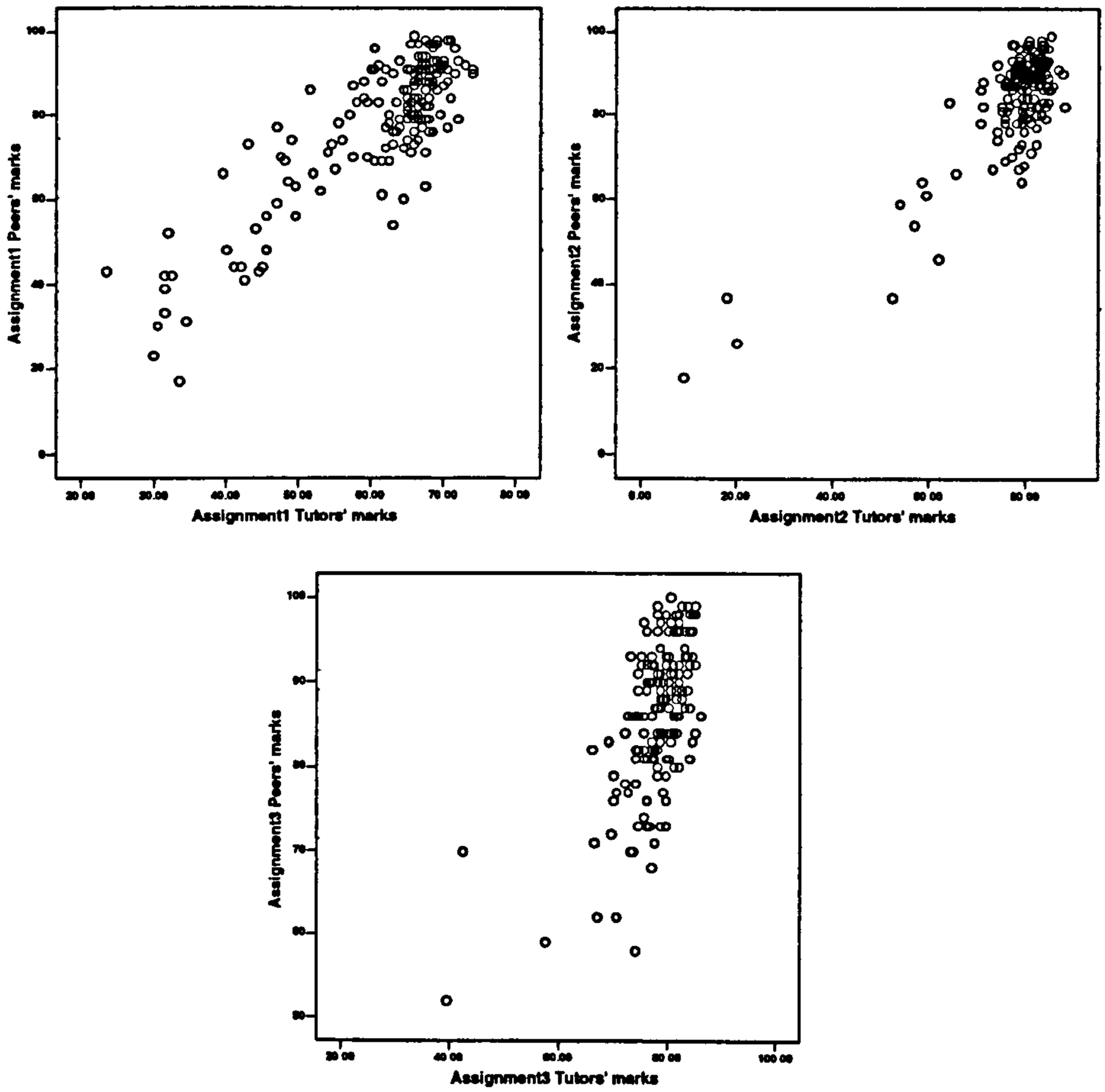


Figure 41 Scattergram of peers' and tutors' marks in assignment 1, 2 and 3

However the correlation coefficient in assignment 3 [$r=.62$, $n=166$, $p < .0005$] is lower than the correlation coefficients in assignment 1 and 2. This may be because assignment 3 is the most difficult assignment. The following are suggested reasons for the decreased correlation coefficient in assignment 3.

1. Students have many assignments to finish in the same time at the end of term, therefore they don't really spend enough time marking carefully.

   "Last term was very work hectic, we had about 5 due in for the last, coming to the end of term. So it was very rushed. After getting the assignments done, you had got it done and move on to the next one. Did you see the timetable we had?"

2. Two tutors gave different marks (high/low marks) on some scripts (see more detail in section 6.4.3), which affected the marks. For example one tutor gave a high mark but another gave a low mark, therefore the average mark from tutors was different from the average mark from peers, which was high.
3. Students gave high marks for short and incomplete scripts (see more detail in section 6.6.2).

   "The program is easy to follow since it is short and incomplete, and the utilities used so far are simple." (mark 3 out of 4)

4. Students gave an over generous mark, which was inconsistent with their comments.

   "Exceptional, extensive comments and great indentation. The only thing could be that the comments are TOO extensive, but I am not reducing marks for that. Sorry" (mark 4 out of 4)

5. A few students misunderstood how to answer the question about the program correctness (program meets the specification, appropriate code handle errors, program finishes with an appropriate exit status) in marking criteria. They gave a zero mark for the script, which does not pass the ten automatic tests (marking base on the automatic test mark), but the other students found out what was wrong with the script and tried to give marks.

   "I'm afraid as you failed every test I cannot give you any marks for this section. Your theory seems to be very good and right for the most part, but is it doesn't work at all then I can't see how I can give you any marks, sorry."

> *"The program failed all 10 automatic tests, which is a shame. The reason for that is..... I will mark based on ....., since I am trying to exam the program, not picking mistake."*

Summary

From the results above suggested that there is a strong positive relationship between tutors' marks and peers' marks at a high level of statistical significance in all 3 assignments. Students and tutors marked in the same way, when the marks that are given by tutors are high, the marks that are given by peers are high as well. On the other hand when the marks that are given by tutors are low, the marks that are given by peers are low as well. This result is supported by Segers [Segers2001] studies of peer assessment. However the average of marks given by peers and tutors are different, and there are variations in individual cases (see more detail in 6.4.1). In the next section will examine a significant difference between the average marks from these two groups of markers.

6.3 Difference between marks given by the tutor and by peers

The results in the previous section suggested that there is a strong correlation between the marks awarded by a tutor and those awarded by peers, but the averages of the marks given by peers are higher than the average of marks given by tutors. In this section, the differences between the marks given by the tutors and peers are analysed. The purpose of *paired-samples T-test* and *Wilcoxon Signed Rank test* is to test for significant differences between two group means from data that has been gathered by means of related measures design [Best1989, Pallant2001]. An assumption for a paired-samples T-test is that the observed data are from the same subject or from a matched subject and are drawn from a population approximately normally distributed (this assumption is less critical the larger the value of n) [Rees1989]. On the other hand the assumption of the Wilcoxon Signed Rank test is that the observed data are drawn from a population with a non-normal distribution. Therefore both of these statistical tests are considered to accept or reject the hypothesis H_0 below.

Hypothesis

H_0: the mean of peers' and tutors' marks is not significantly different

H_1: the mean of peers' and tutors' marks is significantly different

6.3.1 Paired-samples T-test

	Paired Differences					t	df	Sig. (2-tailed)
	Mean	Std. Dev.	Std. Error Mean	95% Confidence Interval of the Difference				
				Lower	Upper			
Pair 1: assignment 1 Peers' marks & tutors' marks	16.89	9.54	.74	15.43	18.35	22.811	165	.000
Pair 2: assignment 2 Peers' marks & tutors' marks	6.51	7.33	.57	5.38	7.63	11.435	165	.000
Pair 3: assignment 3 Peers' marks & tutors' marks	8.05	6.96	.54	6.98	9.12	14.891	165	.000

Table 40 Paired-samples t-test comparing peers' and tutors' marks

A 95% confidence interval for the difference in means is commonly used [Bryman2001]. If the value in the Sig. (2-tailed) column is less than 0.05 then there is a significant difference in the mean marks of tutors and peers. The t-value of the difference between the sample means, its degrees of freedom and its two-tailed significance level are also shown in Table 40. The difference of mean marks of the 3 pairs are 16.89, 6.51, and 8.05, respectively. Since the values in the Sig. (2-tailed) column of all 3 pairs (0.0005) are less than 0.05, then there are the significant differences in the mean marks of tutors and peers (reject H_0).

6.3.2 Wilcoxon Signed Rank test

		Z	Asymp. Sig. (2-tailed)
Pair 1	Assignment1 Peers' marks - Assignment1 Tutors' marks	-10.892[a]	.000
Pair 2	Assignment2 Peers' marks - Assignment2 Tutors' marks	-8.539[a]	.000
Pair 3	Assignment3 Peers' marks - Assignment3 Tutors' marks	-9.884[a]	.000

a Based on positive ranks.

Table 41 Wilcoxon Signed Ranks test comparing peers' and tutors' marks

The Z value and the associated significance levels presented as Asymp. Sig. (2-tailed) are displayed in Table 41. If the significance level is equal to or less than .05 then it can be concluded that the difference between the two marks is statistically significant. In Table 41 the Sig. value is .000 (which means less than .0005). Therefore it can be concluded that the two sets of marks are significantly different (reject H_0).

The difference between peers' and tutors' marks may result from the different marking perspectives. Students tend to give full marks (see the marking details in the next research question) for the program that they think is good, but tutors tend to give only 70% for the good program, as the following quote from a tutor illustrates. It is interesting that students tend not to worry about high marks, whereas tutors do.

> *"I don't think I would give full marks for an assignment as a whole as it's very difficult to say that something is 100% correct and it also gives the student the impression there is no room for improvement. In my view 70% is a good assignment mark. I consider 80% as a high mark and would very rarely give students a mark greater than this."*

Summary

The results from paired-samples T-test and Wilcoxon Signed Rank test are the same – reject H_0. It can be summarised that the mean marks of peers and tutors is significantly different ($p < .0005$). Peers' marks are higher than tutors' marks. More details about the different marking between tutors and peers are investigated in the next section.

6.4 Marking criteria

The correlation between peers' and tutors' marks is analysed in more detail (each marking criterion) in this section, to find out for which marking criterion peers' marking is as accurate as tutors' marking. The causes of low correlation between peers' and tutors' marks are discussed for each assignment. The correlation between two tutors' marks are reported comparing with the correlation between peers' and tutors' marks in order to find out the marking criteria that most people (students and tutors) have different opinions about.

Tutors and peers marked the assignment by answering the same questions in the marking criteria, which relate to program quality (i.e. readability, correctness and style). There are 8 questions in the marking criteria for assignment 1 and 10 questions in the marking criteria for assignments 2 and 3 (see appendix: Marking Criteria for peer assessment for programming). The number of questions is different because of the size and difficulty of the assignment. The Pearson correlation is used to identify the relationship between tutors' marks and peers' marks on 143 individual assignments for each marking criteria (see Table 42), since the data is in the interval level.

6.4.1 Peers' and tutors' mark correlation in each assignment

N = 165

Marking Criteria	**Assignment 1**		**Assignment 2**		**Assignment 3**	
	Pearson Correlation	Sig. (2-tailed)	Pearson Correlation	Sig. (2-tailed)	Pearson Correlation	Sig. (2-tailed)
Readability:						
1. The number of comments	.853**	.000	.613**	.000	.630**	.000
2. Helpfulness of comments	**.862****	.000	.548**	.000	.470**	.000
3. Appropriate indented code	.620**	.000	.450**	.000	.281**	.000
4. Appropriate variable/ function names	.522**	.000	**.694****	.000	.324**	.000
Correctness:						
5. The program meets the specification	-		.668**	.000	.618**	.000
6. Appropriate code handles errors	-		.638**	.000	**.789****	.000
7. The program finishes with an appropriate exit status	.672**	.000	.661**	.000	.492**	.000
Style:						
8. Appropriate utilities have been selected	.380**	.000	.655**	.000	.628**	.000
9. Good program structure	.388**	.000	.501**	.000	.351**	.000
10. Easy to follow what the program does	.478**	.000	.472**	.000	.190*	.015

**Correlation is significant at the 0.01 level (2-tailed)
*Correlation is significant at the 0.05 level (2-tailed)

Table 42 Correlation between tutors' and peers' marks in each marking criterion

***Assignment 1**:* There was the strongest positive relationship between the tutors' marks and peers' marks in marking criterion 2 [r=.86, n=165, p < .0005] at a high level of statistical significance, and a less positive relationship in marking criterion 8 [r=.38, n=165, p < .0005] at a high level of statistical significance. Therefore the marking criterion 2 – 'helpfulness of comments' is the easiest for students to follow, but marking criterion 8 – 'appropriate utilities have been selected' is the most inaccurate marking, because the assignment 1 is small (there are not many choices of utilities). Most students have no experience in marking and this is the first assignment to mark. Although a program had some mistakes or used inappropriate utilities, they hesitate to assign low marks as the following students comment.

- *"The incorrect selection of the -w test, as previously mentioned, means that the code does not fulfil its purpose and so was* ***a poorly selected utility****. The other tests chosen are appropriate."* (mark 3 out of 4)
- *"Could have cut down half your code by not using so many if then else statements. Could have used a case for say usercode, if it was malformed output the error then exit, therefore rest of program does not need to run. If the usercode was valid then move on."* (mark 4 out of 4)

This is illustrated by the histogram in Figure 42 where most of peers' marks are in the range of 8-10, but tutors' marks are varied. The bell-shaped curve of tutors' marks is wider than that of the peers' marks.

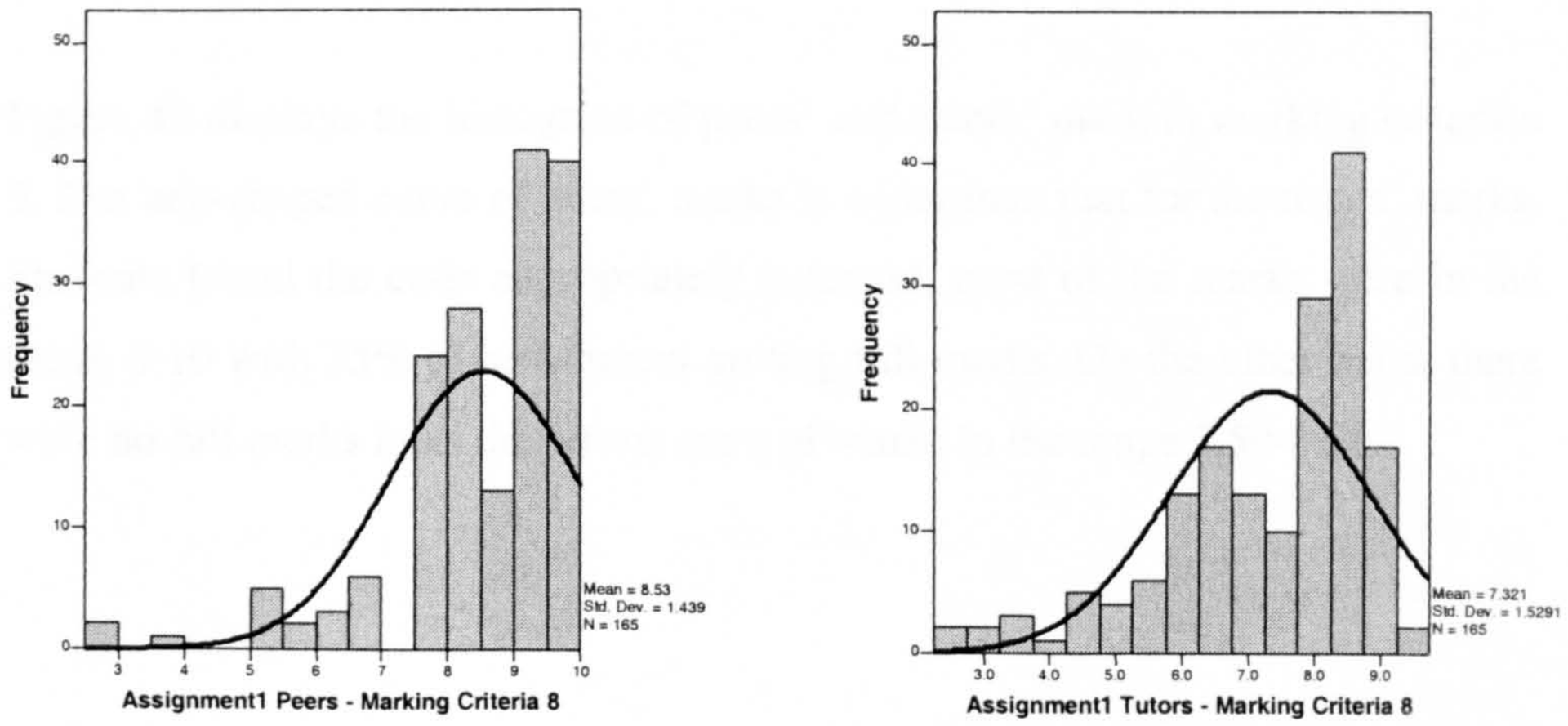


Figure 42 Histogram of peers' and tutors' mark in marking criterion 8, assignment 1

The correlation coefficients of marking criteria 1 to 7 are above .50 at a high level of statistical significance. This result indicates that programming readability (especially number of comments and helpfulness of comments), and program correctness are the most accurate for students' marking. However programming style (correlation coefficient in marking criteria 8 to 10 are below .50 at a high level of statistical significance) is the most difficult for students to follow.

Assignment 2: There was both the strongest and the weakest positive relationship between the tutors' marks and peers' marks (at a high level of statistical significance) in programming readability - marking criterion 4 [r=.69, n=165, $p < .0005$], and marking criterion 3 [r=.45, n=165, $p < .0005$] respectively. Most of the correlations are higher than .50, which means most of peers' marks and tutors' marks have a strong positive relationship at a high level of statistical significance. However students and tutors have a different opinion on marking criteria 3 and 10. For marking criterion 3, students gave different marks on the same script and hesitated to give low marks for inappropriate indentation, which made the marks higher than the tutors' marks, as the following different students' comments on the same script illustrate.

- *"not very clearly indented I felt it moved about to much to be able to follow and had very little commenting."* (mark 2 out of 4)
- *"The indentation is quite good, with appropriate layering."* (mark 3 out of 4)

Figure 43 displays the histogram of peers' and tutors' mark in marking criterion 3. The bell-shaped curve of peers' marks is wider than that for the tutors' marks. Students found the code appropriately indented; most of the marks were in the range 8-10 with 35% of assignment getting full marks. On the other hand, there were no full marks from the tutors; most of marks in the range 7.50-8.50.

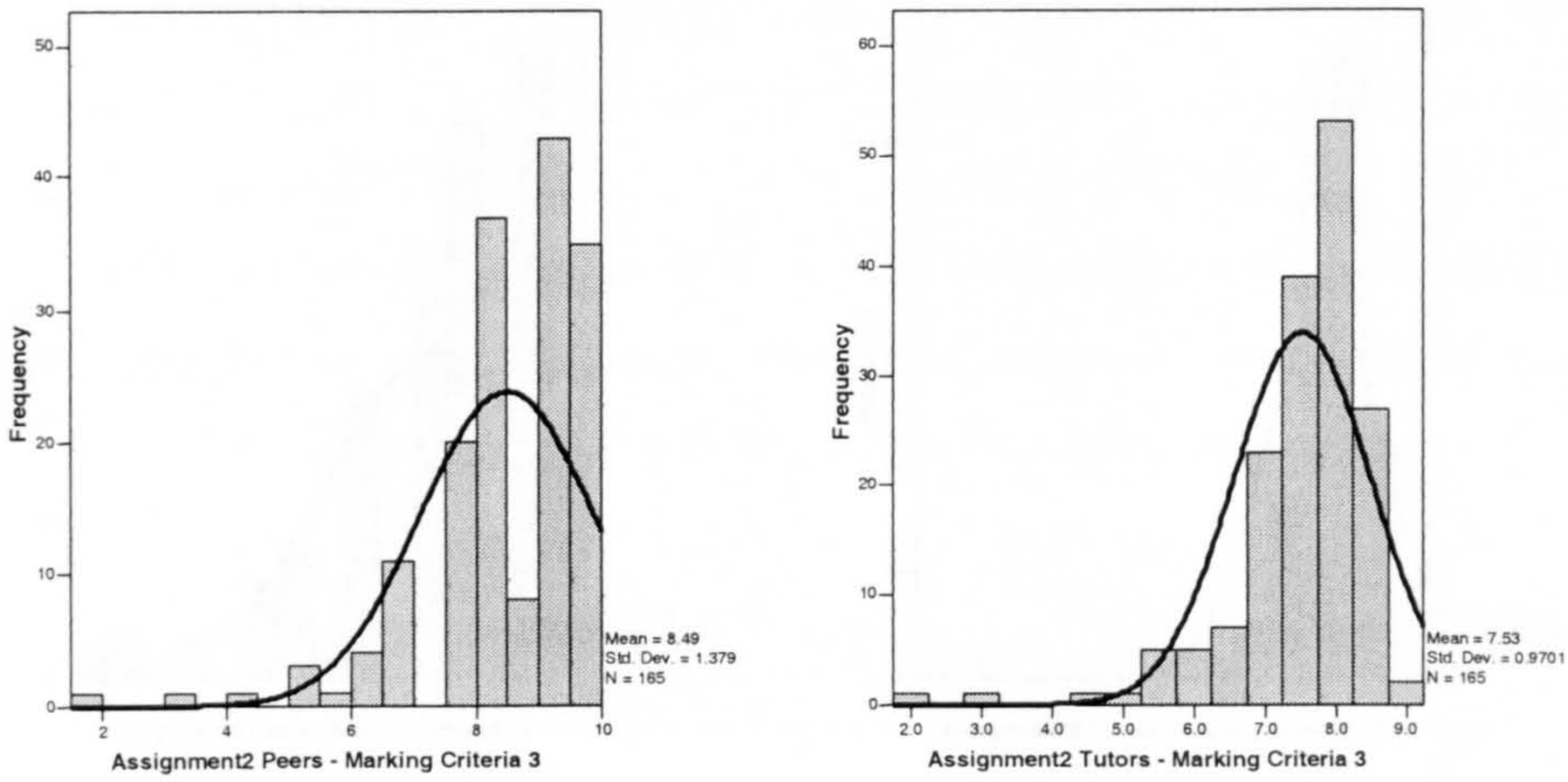


Figure 43 Histogram of peers' and tutors' mark in marking criterion 3, assignment 2

For marking criterion 10, some marks from peers are higher than the marks from tutors because there is no agreement on how to mark the incomplete script; students and tutors have different opinions on marking the script that did not pass any automatic test. The following are the comments from students on the programming style with the "marks higher" than the tutors' marks.

- *"It's easy to follow the code because there's so little of it. Of what it does."*
- *"This script should be marked with 0 grade, but since this script is an actual shell script I am going to mark it as a program as well. The style chosen is correct and the program performs its job."*

The comments below from students on the style of program with the "marks lower" than tutors' marks.

- *"It would have made your code a lot easier to follow if you had split your code up into functions. Also the nesting of case statements."*
- *"After line 49 everything is fine (when HEADLINES is introduced) and if the entire program was styled like that then it'd be great. But that first half is a real chore to understand as it's just one big unindented block."*

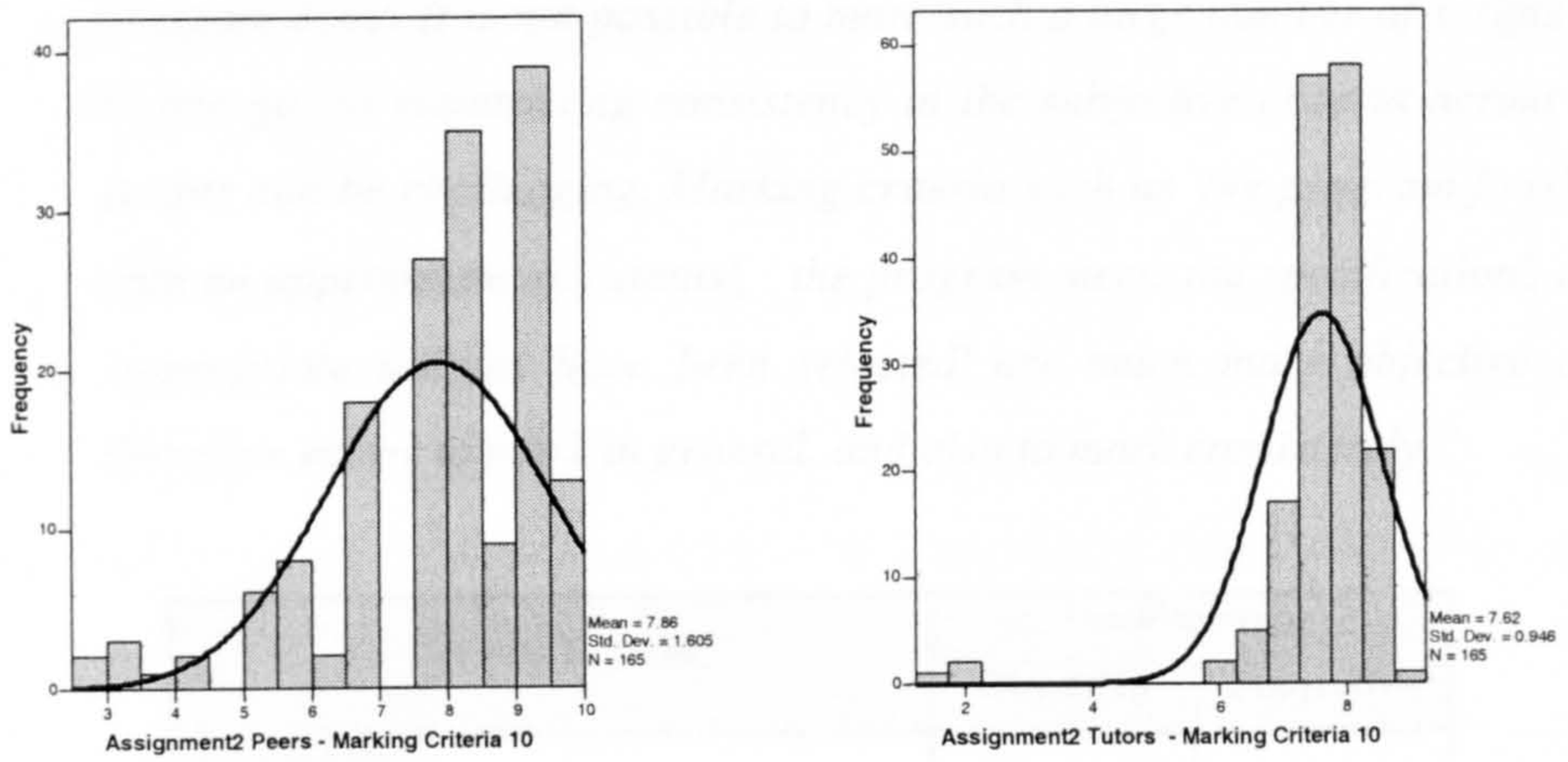


Figure 44 Histogram of peers' and tutors' mark in marking criterion 10, assignment 2

Figure 44 displays the marks from peers having a wider range than the marks from tutors. Tutors who have much greater knowledge and experience of this programming found it was easy to follow what the program doing; most of the marks were in the range 7-8. However some students found it was easy to follow what the program does, if it was coded in a similar style to their own program.

> *"Following what the program is doing is very easy, however this may be to do with the fact that I carried it out in a similar way."*

Assignment 3: There was a strongest positive relationship between the tutors' marks and peers' marks in marking criterion 6 [r=.78, n=165, $p < .0005$], and a low positive relationship in marking criterion 10 [r=.19, n=165, $p < .015$[1]] and marking criterion 3 [r=.28, n=165, $p < .0005$], at a high level of statistical significance. The level of relationship between peers' marks and tutors' marks varies within each category of marking criteria, as there are high and low correlations in the same category. The highest correlation was in the program correctness category. This is also supported by the following comment on marking criteria from the tutor who found that the subjective questions are difficult to mark consistently compared to the objective questions. The subjective and objective questions are summarised in Table 43 below.

> *"The marking criteria I found the most difficult were those which were* ***subjective*** *such as 'helpfulness of comments', and 'easy to follow what the*

[1] The correlation is significance, if $p < .05$.

program does'. It is not possible to mark such a large number of scripts all in one go, so maintaining consistency in the subjective criteria across all scripts can be challenging. Marking criteria such as 'the program finished with an appropriate exit status', ' the program meets the specification', and 'appropriate utilities have been selected' are much more ***objective*** *and therefore easier to mark in general, and also to mark consistently."*

Marking Criteria	**Question type**	
	Objective	**Subjective**
Readability:		
1. The number of comments	√	
2. Helpfulness of comments		√
3. Appropriate indented code		√
4. Appropriate variable/ function names		√
Correctness:		
5. The program meets the specification	√	
6. Appropriate code handles errors	√	
7. The program finishes with an appropriate exit status	√	
Style:		
8. Appropriate utilities have been selected	√	
9. Good program structure		√
10. Easy to follow what the program does		√

Table 43 Objective and subjective question type in marking criteria

Figure 45 displays the distribution of marks from peers and tutors on the marking criterion 10 – 'easy to follow what the program does'. The bell-shaped curve of peers' marks is wider than that of the tutors' marks. Most of marks from tutors were in range 7-8, but most of marks from peers were in range 7.5-10. Some students got high marks from peers because the program was short or incomplete, when it is hard to make an accurate judgement.

- *"The program is easy to follow since it is* ***short and incomplete****, and the utilities used so far are simple."*
- *"The use of getopts and case is good, and implementing functions is good practise. But again* ***the incompleteness of the program precludes a high mark here****. Program structure is excellent however, and following what*

your script does manage is easy. You have excellent programming technique, ..."

However some students got low marks from peers because the program was too long, with too few comments, which made it hard to follow what the program did, but tutors found the programs were well written with clear layout. Many students recommended that subroutines should be used to simplify the programs.

- *"You have **not made use of subroutines**, which would have helped to tidy your script, and save some duplication, especially as you chose to repeat everything for the -i handler."*
- *"A **long long program**... You really need to think about using **subroutines** or simply your code. Not that easy to follow and easy to read. Try to splitting up problems and don't ....."*
- *"The program is structured well and the break down into functions makes it easier to follow what the script is doing. However **the lack of comments** makes it really difficult to follow ....."*

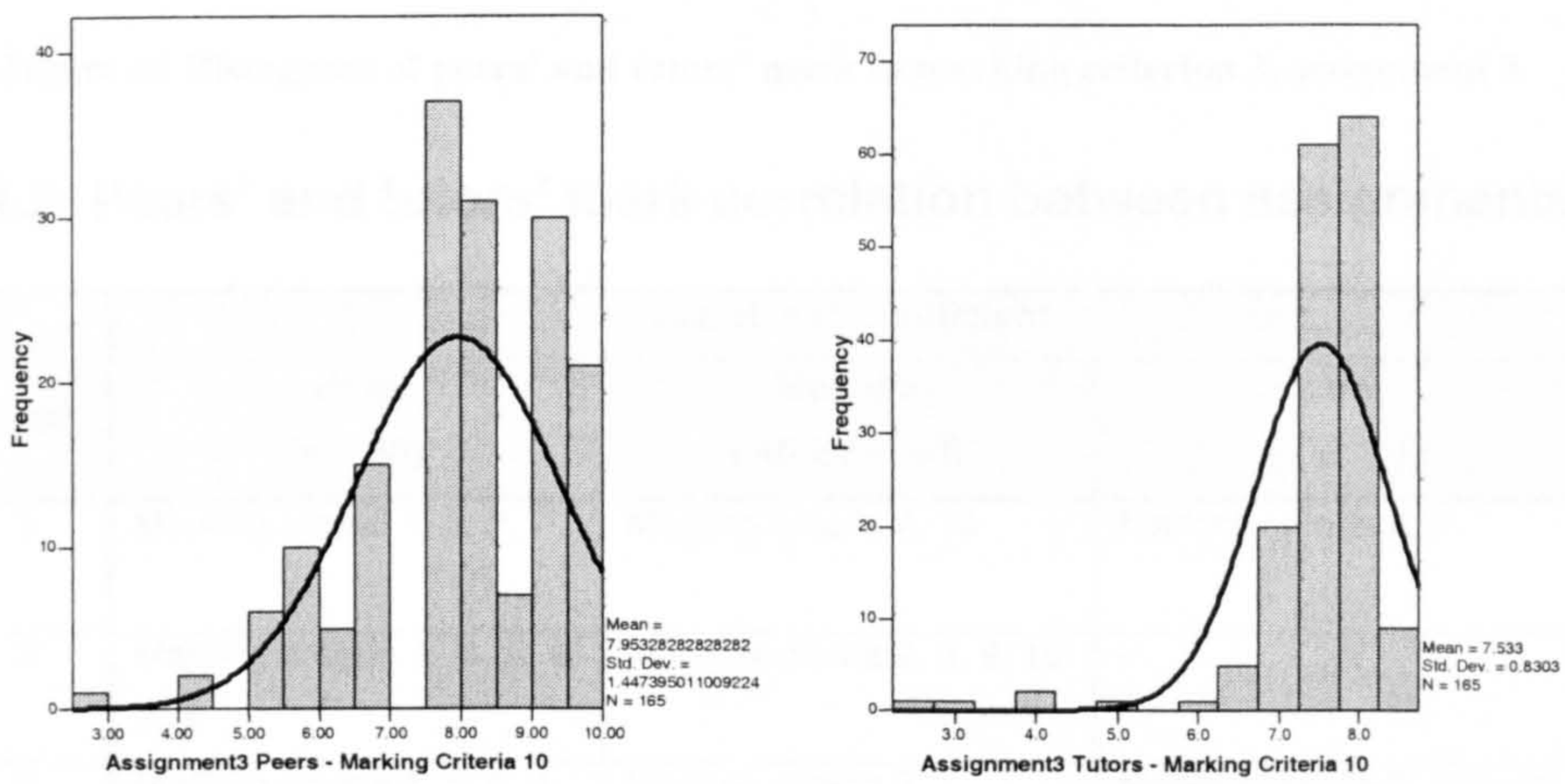


Figure 45 Histogram of peers' and tutors' mark in marking criterion 10, assignment 3

There was a low positive relationship between peers' marks and tutors' marks in the marking criterion 3 – 'appropriate indented code'. Figure 46 displays the distribution of marks from peers and tutors on the marking criterion 3. The bell-shaped curve of peers' marks is wider than the tutors' marks. Most of marks from tutors were in range 7-8, but most of marks from peers were in range 9-10. Students gave high marks for short and incomplete programs. Students were

harsh or too generous marking on some scripts. Some students in individual groups were too harsh or generous with their marking, while the rest gave reasonable marks, which made the marks high or low overall for the group. The comment below was from a student who gave a very low mark, but tutors gave a medium mark.

"The code is indented poorly and lines spread for far too long. It is a good idea to stick to an 80 column screen width."

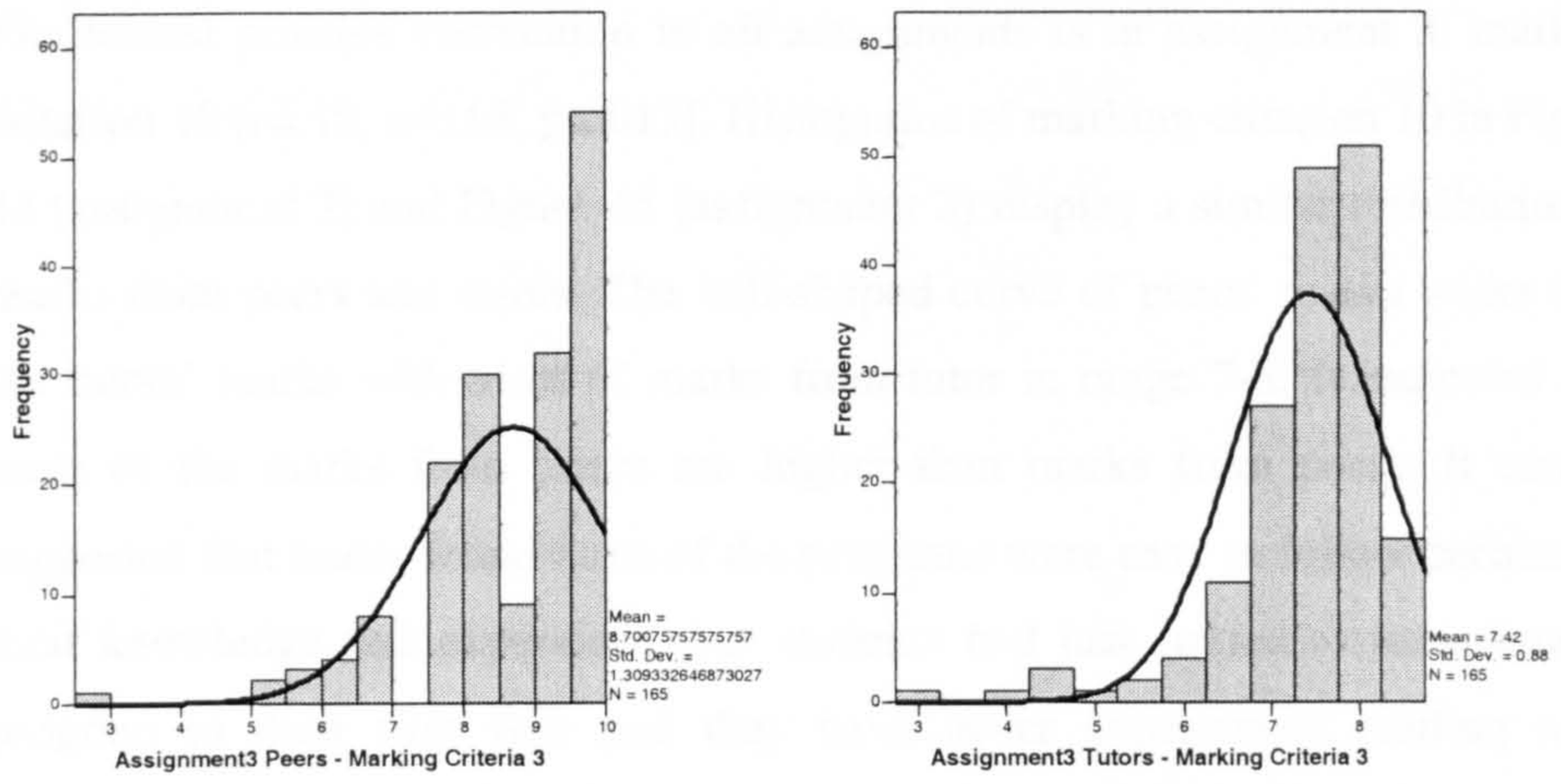


Figure 46 Histogram of peers' and tutors' mark in marking criterion 3, assignment 3

6.4.2 Peers' and tutors' mark correlation between assignments

Assign ment	Correlation coefficient		
	High (r > .60)	**Medium (.40 < r < .60)**	**Low (r < .40)**
1	Marking criteria 1, 2, 3, 7	Marking criteria 4, 10	Marking criteria 8, 9
2	Marking criteria 1, 4, 5, 6, 7, 8	Marking criteria 2, 3, 9, 10	-
3	Marking criteria 1, 5, 6, 8	Marking criteria 2, 7	Marking criteria 3, 4, 9, 10

Table 44 Level of correlation coefficient of 10 marking criteria from 3 assignments

Table 44 shows the grouping of strong, medium and low positive relationship between peers' marks and tutors' marks of 10 marking criteria. The group arrangement is based on the interpretation of a correlation coefficient by Best and Kahn [Best1989]. Most correlations from 3 assignments are high, which mean there is a strong positive relationship between peers' marks and tutors' marks.

Especially in assignment 2, there was no low correlation. As can be seen in Table 44,

- marking criterion 1 – 'the number of comments' is marked accurately by students in every assignment;
- marking criterion 9 – 'good program structure' is in the low correlation group in assignment 2 and 3.

The lowest positive correlation in all assignments is in assignment 3, marking criterion 10 [r=.19, n=165, p<.015]. Histograms of marking criterion 10 in Figure 44 (assignment 2) and Figure 45 (assignment 3) display a similar distribution of marks from peers and tutors. The bell-shaped curve of peers' marks wider than the tutors' marks with most of marks from tutor in range 7-8. It indicated that most of the marks from tutors are higher than marks from peers. It can be suggested that tutors found most of the programs were easy to follow because of their knowledge and experience, but students had just started to learn how to program in their first year and they have never experienced reading other people's code before. However this skill can be developed after taking part in peer assessment exercise.

> *"My analysis of code has gotten better from peer review, from reading other people's code.* ***I find it easier to follow other people's code now****. Because following your own code is far simpler because you know exactly ..... When looking at someone else's code you have to learn ..... It helps with analysing their code."*

Assignment	Correlation coefficient	
	Most increase	**Most decrease**
1to 2	Marking criterion 8 – Appropriate utilities have been selected From [r = .38, n = 165, p < .0005] To [r = .65, n = 165, p < .0005]	Marking criterion 2 – Helpfulness of comments From [r = .86, n = 165, p < .0005] To [r = .54, n = 166, p < .0005]
2 to 3	Marking criterion 6 – Appropriate code handles errors From [r = .63, n = 165, p < .0005] To [r = .78, n = 165, p < .0005]	Marking criterion 4 – Appropriate variable/function names From [r = .69, n = 165, p < .0005] To [r = .32, n = 165, p < .0005]
1 to 3	Marking criterion 8 – Appropriate utilities have been selected	Marking criterion 2 – Helpfulness of comments

Assignment	Correlation coefficient	
	Most increase	**Most decrease**
	From [r = .38, n = 165, p < .0005] To [r = .62, n = 165, p < .0005]	From [r = .86, n = 165, p < .0005] To [r = .47, n = 165, p < .0005]

Table 45 The movement of correlation coefficient between 3 assignments

The movement of correlations between the 3 assignments is shown in Table 45. The correlation coefficient in marking criterion 8 – 'appropriate utilities have been selected' between assignment 1 and 2 is the most increased. Comparing the histograms between Figure 42 and Figure 47, the bell-shaped curve of tutors' marks in assignment 1 is wider than in assignment 2. Tutors found most programs in assignment 2 had the appropriate utilities.

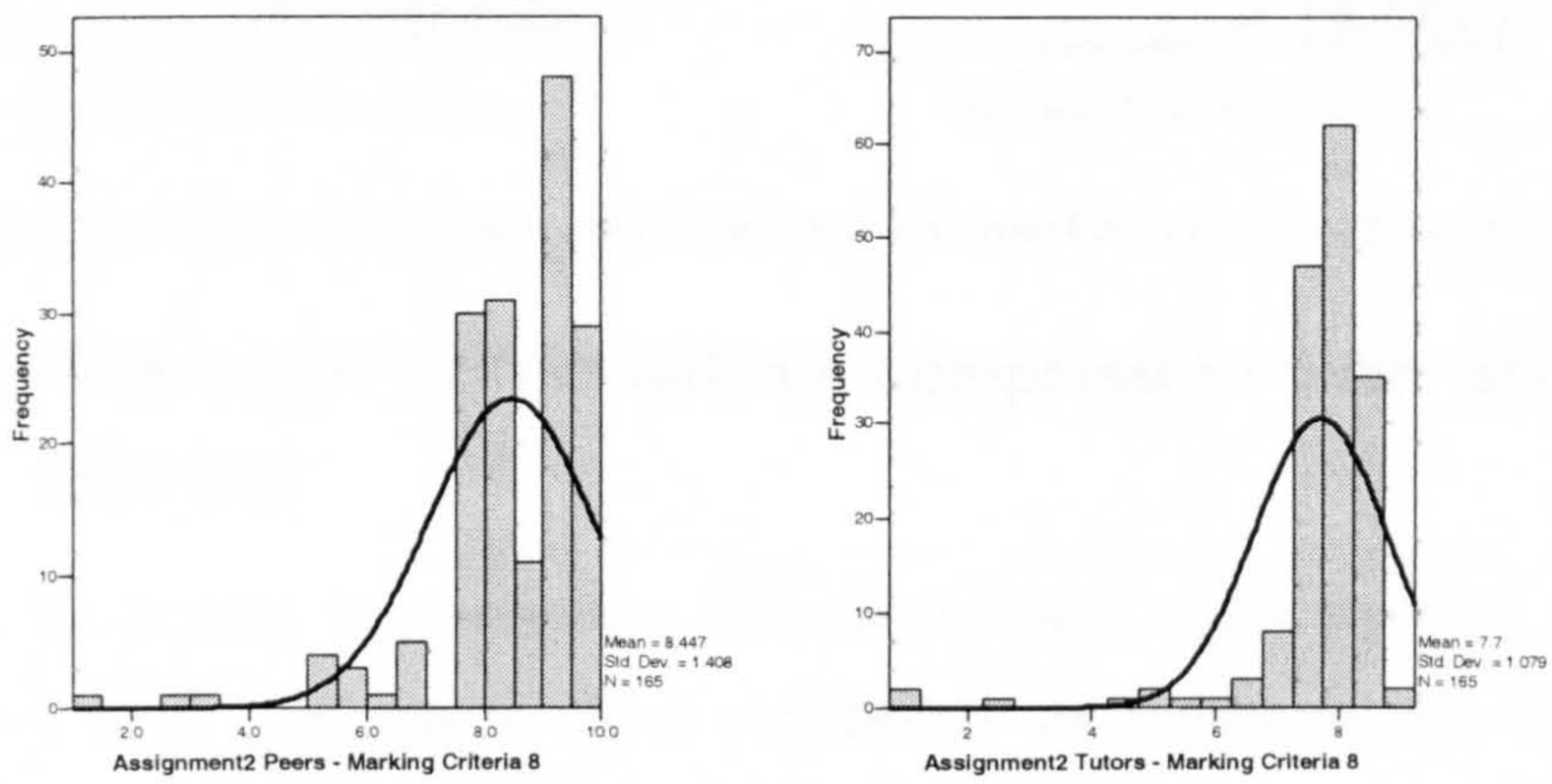


Figure 47 Histogram of peers' and tutors' mark in marking criterion 8, assignment 2

In contrast, marking criterion 2 – 'helpfulness of comments' is the most decreased correlation between peers' marks and tutors' marks in assignments 1 and 3 because the first assignment was small and not complicated. Students can follow what the program does easily without a lot of comment, but clear comments were required for assignment 3, which was the most difficult and complicated assignment. However some students tended to give high marks that contrast with their comments. Figure 48 shows the distribution of marks from peers and tutors for marking criterion 2, assignment 3 with most marks from peers in the range 8- 10.

- *"The comments are helpful, but* ***not terribly explicit*** *in their description of what is going on. Also the comments go from being before the bit of code, to after the bit of code they are talking about which is confusing."* (mark 3 out of 4)

- *"Some of the **comments don't quite make sense**, as I assume the author of the code if given more time will finish and then when everything builds up it will make sense."* (mark 3 out of 4)

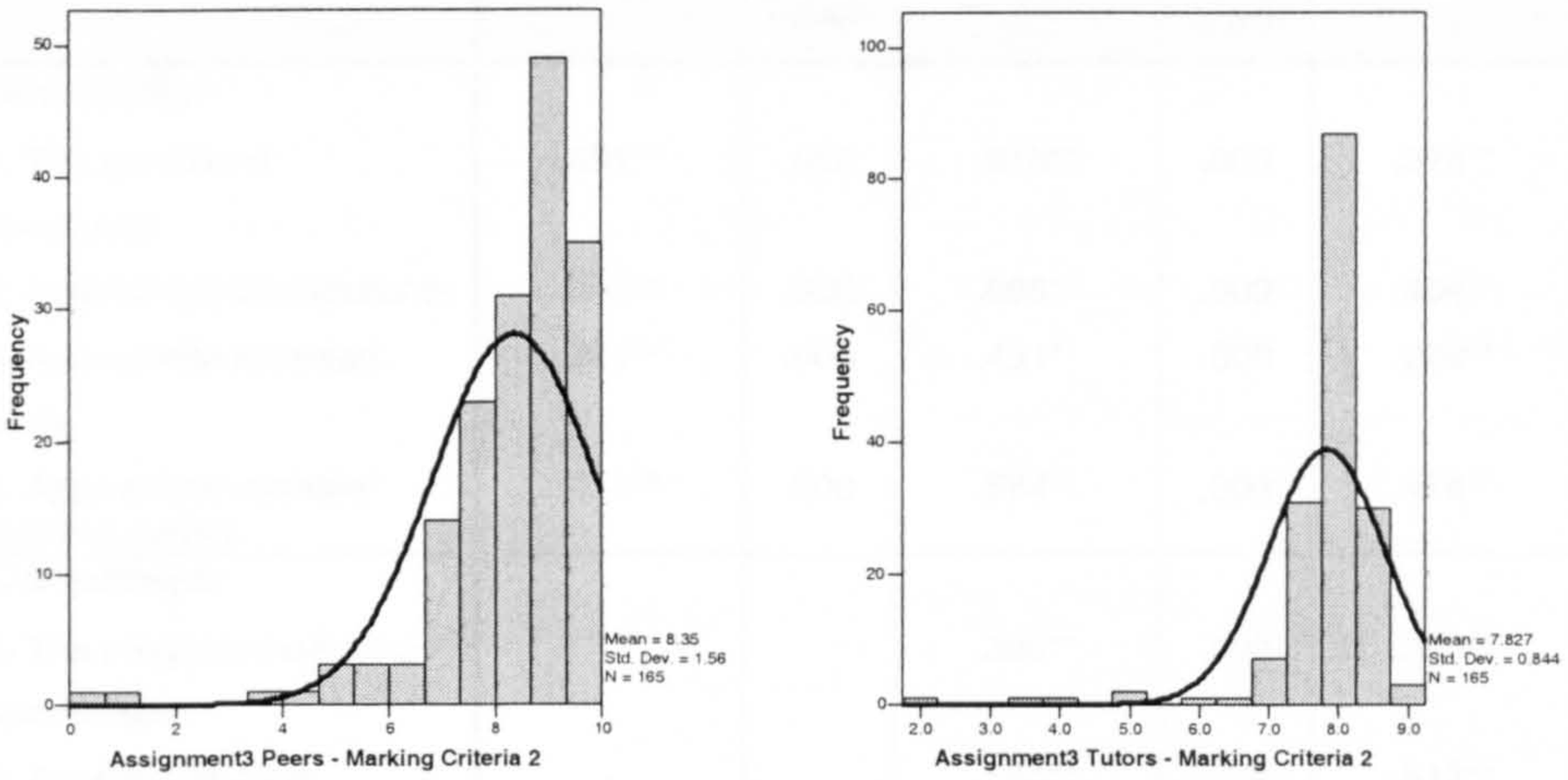


Figure 48 Histogram of peers' and tutors' mark in marking criterion 2, assignment 3

6.4.3 'Peers and tutors marking' compared to 'tutor and tutor marking'

Table 46 displays the correlation between 2 tutors' marks for each marking criterion in 3 assignments. These correlations are similar to the correlation between peers' and tutors' marks in Table 42 as following.

- In assignment 1, both of the correlations of marking criteria 1 and 2 are quite high. The correlations of style of program are low.
- In assignment 2, the correlations for style of program increased and the correlations for marking criteria 1 and 2 decreased. The correlations for marking criteria 4 and 8 increased from assignment 1 to 2 by a similar amount.
- In assignment 3, most of the correlations decreased.

N = 192

Marking Criteria	Assignment 1		Assignment 2		Assignment 3	
	Pearson Correlation	Sig. (2-tailed)	Pearson Correlation	Sig. (2-tailed)	Pearson Correlation	Sig. (2-tailed)
Readability:						
1. The number of comments	**.891****	.000	.618**	.000	.374**	.000
2. Helpfulness of comments	.883**	.000	**.698****	.000	.404**	.000
3. Appropriate indented code	.394**	.000	.431**	.000	.286**	.000
4. Appropriate variable/ function names	.459**	.000	.584**	.000	.478**	.000
Correctness:						
5. The program meets the specification	-		.399**	.000	.283**	.000
6. Appropriate code handles errors	-		.550**	.000	**.613****	.000
7. The program finishes with an appropriate exit status	.390**	.000	.269**	.000	.220**	.000
Style:						
8. Appropriate utilities have been selected	.387**	.000	.678**	.000	.525**	.000
9. Good program structure	.534**	.000	.619**	.000	.451**	.000
10. Easy to follow what the program does	.208**	.004	.549**	.000	.416**	.000

**Correlation is significant at the 0.01 level (2-tailed)

Table 46 Correlation between 2 tutors' marks in each marking criterion

The comparison between the correlations of peers' and tutors' marks, and two tutors' marks revealed that the variance of students' marks is not much different from the variance of tutors' marks. It depended on the type of marking criteria (objective/subjective question) and the difficulty of assignment. Table 47 shows the highest and lowest correlation coefficient in each assignment from both two groups (peers & tutors and tutor & tutor). Both groups have the highest correlation in program readability in assignment 1 and 2, and in program correctness (marking criterion 6) in assignment 3. The lowest correlations from the 2 groups are varied.

Assign ment	Highest correlation coefficient		Lowest correlation coefficient	
	peers & tutors	tutor & tutor	peers & tutors	tutor & tutor
1	Marking criterion 2 – helpfulness of comments [r = .86, n = 165, p < .0005]	Marking criterion 1 – the number of comments [r = .89, n = 192, p < .0005]	Marking criterion 8 – appropriate utilities have been selected [r = .38, n = 165, p < .0005]	Marking criterion 10 – easy to follow what the program does [r = .20, n = 192, p < .0005]
2	Marking criterion 4 – appropriate variable/function name [r = .69, n= 165, p < .0005]	Marking criterion 2 – helpfulness of comments [r = .69, n = 192, p < .0005]	Marking criterion 3 – appropriate indented code [r = .45, n = 165, p < .0005]	Marking criterion 7 – the program finishes with an appropriate exit status [r = .26, n = 192, p < .0005]
3	Marking criterion 6 – appropriate code handles errors [r = .78, n = 165, p < .0005]	Marking criterion 6 – appropriate code handles errors [r = .61, n = 192, p < .0005]	Marking criterion 10 – easy to follow what the program does [r = .19, n = 165, p < .015]	Marking criterion 7 – the program finishes with an appropriate exit status [r = .22, n = 192, p < .0005]

Table 47 Comparing of highest and lowest correlation coefficient in 3 assignments

Summary

It can be concluded that peers' marking is typically as accurate as experts' marking on the objective marking criteria, but they have different opinions on the subjective questions. This depends on their knowledge and experience in programming. In addition, there are several problems which caused the low positive relationship between peers' and tutors' marking which are summarised in Table 51 (in section 6.6).

6.5 Comment issues

Feedbacks	Peer (%)		Tutor (%)	
	+	-	+	-
Program readability				
The number of comments	4.91	3.68	2.27	2.27
Helpfulness of comment	4.91	4.91	5.68	9.09
Style of comment	3.68	0	0	0
Appropriate indented code	6.14	6.14	1.14	2.27
Appropriate variable/function names	3.07	4.30	0	1.14
Program correctness				
The program meets the specification	2.45	3.68	10.23	5.68
Appropriate code handles errors	3.07	4.30	0	1.14

Feedbacks	Peer (%)		Tutor (%)	
	+	-	+	-
The program finishes with an appropriate exit status	1.23	6.75	2.27	1.14
Point out the program mistake	5.52	0.61	12.50	6.82
Style of program				
Appropriate utilities have been selected	7.36	3.07	1.14	3.41
Good program structure	10.43	0.61	6.82	10.23
Easy to follow what the program does	7.36	1.23	2.27	1.14
Other				
Summary of program readability and style	0	0	0	11.36
Point out the good point of program	0.61	0	0	0
Total	60.73	39.27	44.32	55.68

+ with further explanation/ suggest other solutions
- without further explanation/ no suggest other solutions

Table 48 The comparison between peers and tutor' feedbacks

Calculation:

$$\text{Percentage of feedbacks} = \frac{(\text{no. of comments} * 100)}{\text{total no. of comments}^2}$$

$$= \text{x } \%$$

Most research in peer assessment has analysed only the different marks between peers and tutors. In this research, the quality of comments from peers and tutor are analysed as well. The comments from tutors in assignment 3 were shorter than the comments from students, but this analysis focuses on the comment issues and the quality of comments rather than the length of comments. A random selection of 72 students' sets of comments is analysed, comparing with the comments from tutors on the same assignments (see Table 48). These results are valid because of the reasons below

- as the comments from students are random with the large number of sample size (72 comments), these results can reflect the population [CRS2003];
- the peers' results are similar to the results from research question "Are the comments from students with a wide range of abilities different?" (in section 6.7).

[2] total number of comments from each group of marker (i.e. peer, tutor)

6.5.1 Peer and tutor comment behaviours

Feedbacks	Peer			Tutor		
	+	-	Total	+	-	Total
Program readability	22.70%	19.02%	**41.72%**	9.09%	14.77%	23.86%
Program correctness	12.27%	15.34%	27.61%	**25.00%**	14.77%	**39.77%**
Style of program	**25.15%**	4.91%	30.06%	10.23%	14.77%	25.00%
Other	0.61%	0.00%	0.61%	0.00%	11.36%	11.37%
Total	**60.73%**	39.27%	100%	44.32%	**55.68%**	100%

+ with further explanation/ suggest other solutions
- without further explanation/ no suggest other solutions

Table 49 Summary of peer and tutor feedback percentage

Tutor comments – most of comments from tutors focused on the correctness of program (39.77%) with the extra comments (beyond marking criteria) that point out the program mistakes with further explanation (Table 49).

> *"Missing "shebang" line. Would have been better to use a system temp dir (e.g. /tmp, /var/tmp), rather assuming working dir is writeable. Doesn't sort and output data correctly. Problem with mup file checking."*

However most comments from tutors on programming readability and style are concise without further explanation, for example 'use of alternative utilities could have……' – there is no suggestion of what the alternative utilities are. It can be concluded that most tutor comments are focused on program correctness, pointing out the program mistakes, but lacking explanation and suggestions for program readability and style (55.68%).

- *"Well written, clear layout and very well commented. Satisfies specification well.";*
- *"Layout could do more to improve readability. Use of* ***alternative utilities*** *could have made script shorter and improved readability."*
- *"Script is rather long and comments/layout could do more to increase readability".*

Peers' comments – most comments from students focused on the program readability (41.72%) with a high percentage (60.73%) of explanation relating to mark awarding, recommendation for other programming styles and suggestions for alternative solutions, as the examples of comments below illustrate.

- Pointing out program mistakes – *"The program would error if a file named john-joe.cup (for example) was provided, as this matches the regexp /-/. Use /^-/ to match lines beginning with -. Additionally, when checking file extensions /.cup|.mup|.pit/ would match file.cup.rubbish, and as . is interpreted as any non-null/newline character, a file called buttercup would be matched. Use something like /\.(cup|mup|pit"*
- Suggestion for the alternative solutions - *"To reduce code, the line 'open(TEMPFILE, "<$file") or (print STDERR "Bad data file $file\\n" and exit 2);' could be used instead of using -s and -r tests on the file. This would open the file as required, or, if this is not possible, print the error and exit appropriately."*
- Suggestion for comment style – *"I'm a huge fan of comments with brief descriptions before every function, so it's nice to see them here. Additionally, don't nest multiple line comments within the condition part of an if statement, it just hampers readability when you have what looks like a code block when in fact you just have an if statement with 3 conditions. At most you should be placing very brief comments on the same line as each condition just to pick out which is which."*
- Comments on the program structure – *"The structure of the program is, on the whole, very good. Splitting the code into subroutines helps readability, and gives a good structure to the program. The only suggestion I have would be to include headings for blocks of subroutines and the main program e.g.*

  #----------

  # SUBROUTINES

  #----------

  .....subroutines...

  #----------

  # MAIN Program

  #----------

  ...main program..."

Some students' comments showed that they really did spend time analysing the code carefully.

> *"**Despite the code passing all the automatic tests, it still does not meet the specification fully**. For instance, if one of the input files were of the type .mup, your code would output 'bad data file', whereas it should according to the spec be able to handle this type of input. Date checking in your script is also a bit crude. It would seem reasonable to check the date is valid i.e. each month has the correct number of days, leap year checking etc."*

6.5.2 Peers' comments against tutors' comments

The results in Table 49 indicate that peers' comments have more quality than tutor comments because of the high percentage of comments with explanations and suggestions for other solutions. However there is a high percentage of tutor comments on program correctness with explanations, but a low percentage of explanation about program readability and style. It contrasts with to peers' comments, which have a high percentage of explanation and suggestion on program readability and style. The following are examples of feedback from peers compared with tutors on the same script. In general students and tutor have a similar opinion on program readability, but they identify some *different* programming issues. The tutor focused on programming mistakes, while students focused on the error handling functions, appropriate utilities, and variable names. It seems clear that people provide different comments, which depend on what they are interested in, but in general their opinions on the same program are not much different, as the following examples illustrate.

- Tutor: *"Script doesn't check for missing arguments to -o and -i options, otherwise satisfies specification. Clearly written, good use of functions. Comments could be more helpful."*
- Marker1: *"The code is littered with useful comments ..... Overall, the readability could not really be improved. Well done."; "As far as I can work out the program meets the specification fully.* ***The error handling functions look correct as do the*** *preg_match() calls that pick out incorrect files etc..... "; "**The utilities used have been selected***

appropriately, particularly *the use of preg_match() which is a much faster alternative to the more common ereg() function. The program is well structured and follows PHP coding conventions - meaning that this program has a good style.* ***The choice of shell_exec() has interested me*** *- I didn't know it existed - and was using fopen('php://stdout') etc and fwrite() to output my data - this seems like a much better alternative! Following what the program is doing is also easy."*

- Marker2: *"Overall good readability - well spaced out code which makes for easier reading..... maybe some of the functions could have a short description of what they are likely to expect as parameters....."; "Ok, pretty good style, some pretty long if statements - (line 283) with embedded functions..... these might have been better broken up into seperate variables. Just improves readability. Also a few if/elseifs maybe* ***could have been better written with a*** *switch/case....."*
- Marker3: *"The comments are good and descriptive, and the indenting is consistent..... However,* ***the variable names aren't always good; for example****, 'dobYY' and 'iArg' could be better.....";*

Summary

It can be concluded that peers identify similar comment issues to tutors (because of using the same marking criteria) with more quality of comments, because of more explanations and suggestions than tutors who provide concise comments without further explanation and suggestion. However they focus on different issues i.e. tutor focuses on program correctness (especially meeting the program specification, and programming mistakes), while peers focus on the program readability and style. Students also suggest the comment style, alternative utilities, and program structure. They preferred the subroutine functions as they help in easy to follow what the program does. Students are concerned about the

program readability and style more than program correctness when they analyse other students' programs. However most students provided useful comments.

6.6 Consistency of marks

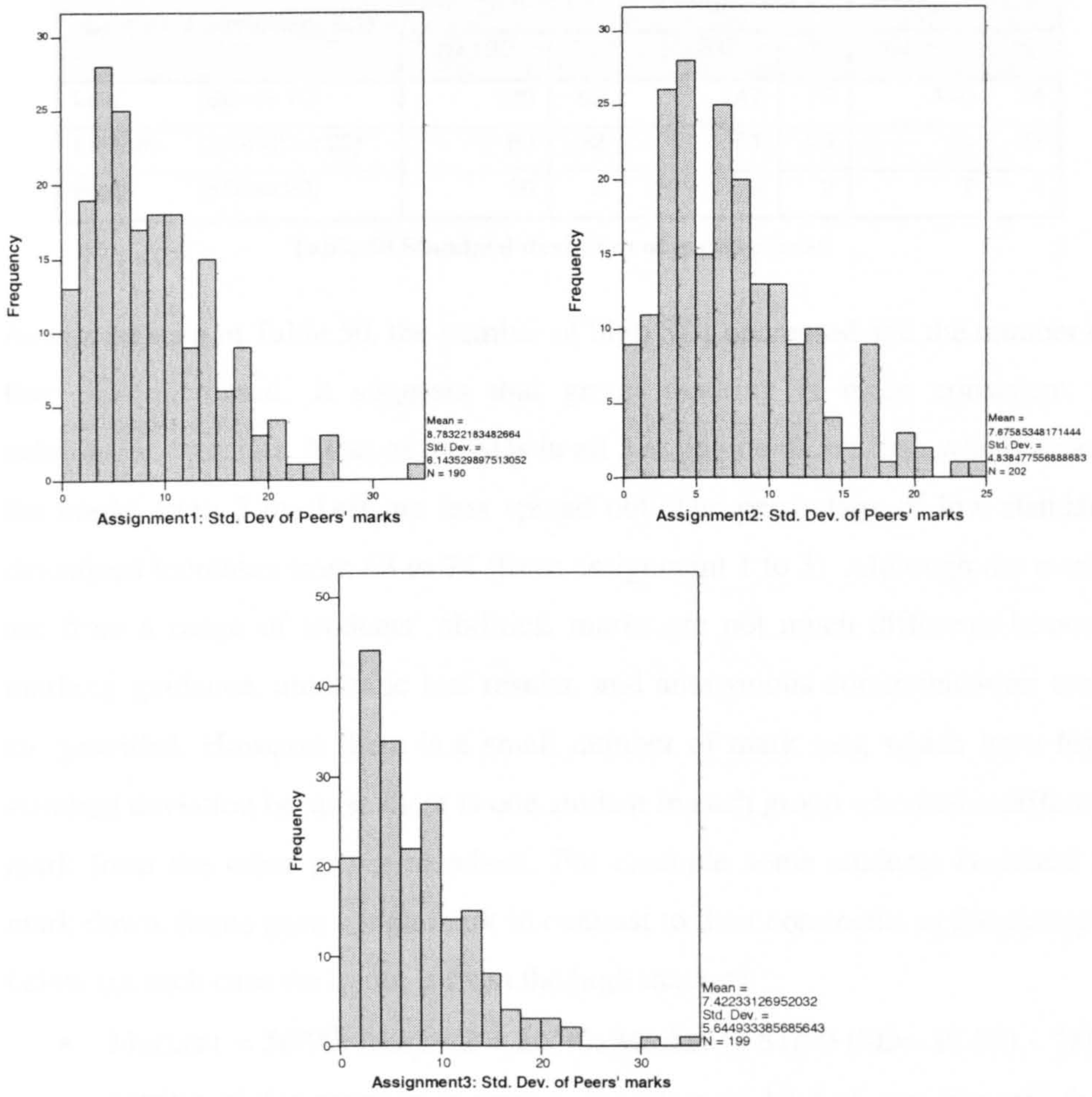


Figure 49 Histogram of standard deviation of peers' marks in assignment 1, 2 and 3

6.6.1 Statistical analysis

Each group consists of 3 students, with a mixed range of ability. The standard deviation (SD) is calculated from 3 marks (out of 90) from 3 students in each group. The spread of the marks awarded by group marking is analysed in this section. The problems of large mark differences within a group are reported. Figure 49 shows the distribution of SD of peers' marks from each assignment. It is clear that the number of SDs more than 20 is decreased in assignment 3. Three categories of SD are considered as follows:

Low - marks are not much different SD <= 10

Medium - marks are different 10 < SD < 20

High - marks are substantially different SD >= 20

Standard deviation (SD)		Assignment 1		Assignment 2		Assignment 3	
		N=190	%	N=202	%	N=199	%
Low	(SD <= 10)	120	63	147	73	148	74
Medium	(10 < SD < 20)	60	32	51	25	44	22
High	(SD >= 20)	10	5	4	2	7	4

Table 50 Standard deviation of group marks

As can be seen in Table 50, the number of high SDs decreased and the number of low SDs increased. It suggests that group marking is more consistent in subsequent exercises. Most of the SDs in all 3 assignments are low, which means the marks from 3 markers are less spread out. The percentage of low standard deviations increases from 63 to 74 (from assignment 1 to 3). Although the marks are from a range of students' abilities, marks are not much different, because marking guidance, automatic test results, and anonymous communication tools are provided. However there is a small number of mark sets, which have high standard deviation because there is one student in each group who had a different mark from the other group members. For example some students hesitated to mark down. Some gave a high mark in contrast to their comments as the example below (in each case the quote is from the high marker).

- Marker1 = 56/90; Marker2 = 20/90; Marker3= 81/90 (SD= 30.67) - *"The sorting of this program is wrong. It sorts each kind of .cup line and then sorts them again in bash. Getting a list off all of the lines and then sorting them. Using Perl would have been a better option as it would have then passed the automatic tests (line 90)."*
- Marker1 = 23/90; Marker2 = 54/90; Marker3= 68/90 (SD= 23.58) - *"The readability of the code is not too good, as the author has not indented the code too well. Also, he has used ambiguous identifier names. The comments are definitely inadequate."*

6.6.2 Marks in group marking

An examination of the groups with a high standard deviation from 3 assignments found that there is one student in each group who had a different opinion from the rest of the group. The factors that cause the high standard deviation in group marking, are summarised in Table 51.

Problems	Assignment 1	Assignment 2	Assignment 3
Harsh marking	2	2	3
Lenient marking	4		1
Do not read the code carefully	2	-	-
Marking based on the automatic test mark	1	1	1
Incomplete script	-	1	-
Fake program	-	-	2
Do not answer some of marking criteria	1	-	-
Total	**10**	**4**	**7**

Table 51 The factors, which cause the high standard deviation in group marking

The results in Table 51 shows harsh and lenient marking are the major factors that effect the different marks from 3 students, as the following students' comments on the same script.

- *Harsh marking*: marker1 pointed out at the confused comment and gave low mark.

  Marker1: *"Confused comments such as..... and the rather perplexing lack of understanding regarding the appropriate use of 'if' and 'case', epitomised."* (mark 1 out of 4)

  Marker2: *"Comments are helpful to understand mostly what is going on, but it is occasionally difficult to understand exactly what is being done, eg ....."* (mark 3 out of 4)

  Marker3: *"The comments were good, although it may be useful to proof read them first. The number of comments was good and every section of code was well."* (mark 4 out of 4)

- *Lenient marking*: marker1 gave full marks for the program readability, although the identifier names are inappropriate and the code is indented inconsistency.

  Marker1: *"It was useful for the identifier names to be consistent throughout the program. The code was practically self-explanatory, although a few more comments would have been preferable."* (mark 4 out of 4)

  Marker2: *"The use of "[1]" in the case statements is a semantically-correct, but a syntactic misuse of ..... The lack of indentation for ....."* (mark 1 out of 4)

  Marker3: *"Z is an extremely unhelpful variable name. Choose whether or not to indent code inside 'if' statements....."* (mark 1 out of 4)

However some too harsh and lenient marking on the same script make the overall mark for that script appropriate, as the following comment from tutor illustrates:

> *"Marker1 of part I has been rather harsh and provided no justification for the low mark awarded. Marker 3 has been rather generous and failed to provide any comments to justify the award of full marks. An overall mark of 77/90 for part I therefore seems reasonable."*

Some students marked the program without reading the code carefully, or marked the program based solely on the mark from automatic test. Students have different opinions on marking the incomplete program or the "fake" program, as the following example below.

- *Did not read the code carefully*: marker 1 gave full marks for the program readability without reading the program carefully. The others gave zero marks.

  Marker1: *"The code is easy to understand, with following helpful comments."* (mark 4 out of 4)

  Marker2: *"The comments and number of comments are useful for this program and make it easier to follow, but the code*

has not been indented at all. *The spacing used does also not appear to be consistent. If spacing and indentation had been used in a correct manner."* (mark 0 out of 4)

Marker3: *"No indentation was present which made the whole script a bit difficult to understand. Good indentation is necessary for the reader to understand the code."* (mark 0 out of 4)

- *Marking based on the automatic test mark*: marker 1 gave marks based on the number of failed automatic tests without answering the questions correctly, but marker 2 analysed the program carefully to response to the question in marking criteria, i.e. the program meets the specification, the code handles error appropriately, and the program finishes with an appropriate exit status.

  Marker1: *"The program fails to pass tests 7,9,5,8. Test 9 fails because ..... A while loop could have been implemented to fix that problem. Test 8 fails because ....."* (mark 2 out of 4)

  Marker2: *"Due to the fact that the program passed 6 tests, it can be implied that ..... Semantically, the code looks correct, and you certainly had the right idea about ....."* (mark 4 out of 4)

- *Incomplete script*: 2 markers have different opinions on marking the incomplete program. Marker2 a gave high mark for the quality of the short program the same as for a full program.

  Marker1: ***"Little amount of code to mark,*** *cannot really make an accurate judgement. I suggest that with the indenting perhaps not so much white space is required."* (mark 1 out of 4)

  Marker2: *"It's a rather unusual method of displaying comments, it owes perhaps .....!* ***However, I don't think it would be fair to deduct marks for a preference thing,***

and it is actually very clear. Not sure if would work with a really long section of code, *but for a short piece of code like this it does the job well. The comments themselves are ....., although ....., I don't think I can actually mark down for that either."* (mark 4 out of 4)

- *Fake program*: All 3 markers have different opinions on assigning marks to a script which was written only to pass the automatic test without the appropriate programming, as the following comments from 3 markers on the program correctness.

  Marker1: *"although all tests are passed this program deserves* ***absolutely no marks as it a cheat. no error handling and*** *none of the specifications met."* (mark 0 out of 4)

  Marker2: *"No doubt about it that this program accurately finished all the tasks."* (mark 3 out of 4)

  Marker3: *"Actually, it is not a good excuse that BOSS went down. I revised your code, only one thing you have done is check if the file exits, what is worse, you did not use this one in your original code, which I can not give you mark for that."* (mark 4 out of 4)

In order to solve all the above problems, it may be appropriate to provide more training for marking. Giving the guidelines on marking, for example, a minimum or maximum mark for a particular case (i.e. incomplete program, fake program, incorrect program) and an example of a mark for each kind of program, etc.

Summary

It can be concluded that the marks within a group are consistent because the number of groups with a low standard deviation of marks within the group is high, and increases in the subsequent assignments (i.e. 63%, 73%, 74% in assignment 1, 2, 3, respectively), which means the marks from 3 markers (within a group) are less spread out. The small percentage of high standard deviations arises for a variety of reasons (i.e. harsh marking, lenient marking, did not read the code carefully, marking based on the automatic test mark, incomplete script,

fake program and did not answer some of the marking criteria). Marking training is suggested to solve these problems.

6.7 Student ability ranges

As results from the previous section indicate marks within a group are not generally spread out. In this section, comments from individual students within each group are analysed. Each group of markers who marked the same scripts comprised students of different abilities. Each student's ability is classified by the number of marks from the automatic test, as the following:

Good	> 80% of marks for automatic test
Average	automatic test marks in range 40-80% inclusive
Poor	< 40% of marks for automatic test

This classification of student abilities is valid because the students who can write a correct program (pass the automatic tests) have better programming ability than the weak students, and it is the only available evidence (number of marks from the automatic test) to classify students' abilities.

Feedbacks	**Students' abilities (%)**					
	Good		Average		Poor	
	+	-	+	-	+	-
Program readability						
The number of comments	7.22	2.06	2.94	5.88	4.94	2.47
Helpfulness of comment	5.15	2.06	5.88	4.90	6.17	6.17
Style of comment	0	0	2.94	0	3.70	0
Appropriate indented code	4.12	4.12	6.86	6.86	6.17	4.94
Appropriate variable/function names	8.25	4.12	4.90	3.92	2.47	8.64
Program correctness						
The program meets the specification	10.31	1.03	7.84	0.98	2.47	0
Appropriate code handles errors	7.22	3.09	1.96	0.98	6.17	1.24
The program finishes with an appropriate exit status	2.06	8.25	2.94	0	1.24	2.47
Point out the program mistake	1.03	0	8.82	0	6.17	0
Style of program						
Appropriate utilities have been selected	9.28	3.09	8.82	1.96	2.47	4.94
Good program structure	7.22	3.09	6.86	0	8.64	6.17
Easy to follow what the program does	5.15	2.06	7.84	3.92	6.17	3.70
Other						

Feedbacks	Students' abilities (%)					
	Good		Average		Poor	
	+	-	+	-	+	-
Suggestion to improve the program but it is not required in the program specification	0	0	1.96	0	0	0
Do not totally understand the program	0	0	0	0	2.47	0
Total	67.01	32.99	70.56	29.40	59.25	40.74

+ with further explanation/ suggest other solutions
- without further explanation/ no suggest other solutions

Table 52 The feedbacks from different ability groups

Calculation:

$$\text{Percentage of feedbacks} = \frac{(\text{no. of comments} * 100)}{\text{total no. of comments}^{3}}$$

$$= x\ \%$$

6.7.1 Providing feedback

Since the students who have poor programming ability are few (most students have a good programming abilities), the comments from 45 students who have different abilities in assignment 3 are compared in Table 52. The results in Table 52 suggest that the good students gave many comments with further explanations on 'the program meets the specification' and 'appropriate utilities have been selected', which are similar to the high percentages on these feedbacks from the average students, but there are low percentages from weak students. However the highest percentage of feedback with further explanations from the weak students is on 'good program structure'. This indicates that students who have poor abilities in programming are not interested (not capable) in providing much feedback on the program correctness, but they provided more feedback with suggestions on program readability and style of program (see Table 53).

---

[3] total number of comments from each group of marker (excellent/average/poor students' abilities)

Feedbacks	Good (%)			Average (%)			Poor (%)		
	+	-	Total	+	-	Total	+	-	Total
Program readability	**24.74**	12.37	**37.11**	**23.53**	21.57	**45.10**	**23.46**	22.22	**45.68**
Program correctness	20.62	12.37	32.99	21.57	1.96	23.53	16.05	3.70	19.75
Style of program	21.65	8.25	29.90	**23.53**	5.88	29.41	17.28	14.82	32.10
Other	0	0	0	1.96	0	1.96	2.47	0	2.47
Total	**67.01**	32.99	100	**70.59**	29.41	100	**59.26**	40.74	100

+ with further explanation/ suggest other solutions
- without further explanation/ no suggest other solutions

Table 53 Summary of feedbacks from the different students' abilities

Table 53 shows the summary of each category of feedback from students of different abilities. The good and average students provided many feedbacks with suggestions in every category, but the weak students provided a low amount of feedback on the program correctness, compared with the other groups of students' abilities. This may be because of the poor abilities in programming, as a few percentages of students who did not fully understand the program.

> *"**I'm not quite sure** where you were going when you first got into the actual code. I think your first error was ..... With the -o option **I'm not sure where the problem is**. **I'm not sure what your trying to attempt** with .....but I would make a presumption that this could be the source. Again, there isn't much commenting here so **I don't know what your aims are**."*

6.7.2 Comparing feedback

> *"Some of them did talk about different things like some of them wouldn't have spotted a thing, so mostly its "yeah, this passed the test, it's OK, a bit more laid out would have been better," and the other one "OK, it's OK laid out, but you've missed out this, this, this," **so most of them were pretty similar, but some of them had extra points**."*

In general the comments from a wide range of students abilities are not much different (Table 53), but they may identify different issues as different people have different ideas (see above student's quote). The following are comments on

the same assignment from students with different abilities. The comments are not much different, especially the comments on program readability. Students identified different problems in the program and suggested other solutions for program correctness and style of program. For example, the first marker who has a poor programming ability, focused on not enough comments making the program difficult to follow, and programming mistakes. The second marker and the third marker, who have an average and good programming abilities, focused on programming readability and suggested alternative appropriate utilities.

- Marker1: (poor programming ability marker)*"Firstly yes the comments are there but ..... It would be helpful if you ..... Other than that, the actually **code is lay out fine** and the **identifiers** are self explanatory. The comments you have put in are not **indented** with the rest of the coding but ..... In future I would strongly recommend ..... It is very difficult for me to check through your code because of the comments being too vague. Such as ..... I am unsure as to why your -i option doesn't work but I believe ..... The output is totally **correct**, all that has happened is that **it's printed out twice**. As I've already said, it's mainly down to the comments ....."*
- Marker2: (average programming ability marker) *"The **code layout** is good, and **variable names** are well chosen, the comments are well defined and..... Some comments are misspelling, but are still fairly clearly readable. **More comments could have helped** ..... The program works fairly well, but ..... **The use of a case statement instead of "getopts" or similar, and no use of the "grep" or "awk" utilities** have made this program ..... More white space between code and comments could make the program a little easier to understand."*
- Marker3: (good programming ability marker) *"**This script is well commented in terms of the number appearing**, and they describe adequately what the script is doing. The comments*

*are helpful to a degree, but ..... Code is **indented** well, and ..... Generally, **identifier** names are ..... This program will carry out some of the functions specified for ..... As a result, the first file is ..... However, pit and cup transformations could have all been handled by ..... **Sed and Awk could have been used to** ..... **Grep is used for** ..... The functional approach used means that the **program structure** is greatly enhanced."*

However different people always have different opinions. The different programming mistakes or issues identified by different markers collectively provide holistic feedback, which help the script author to improve his/her program as illustrate below.

- Marker1: *"The pattern matching only matches 2 digit scores only, so if the user gets <10 or 100 then it will fail on a valid mark. For your loops, you can use foreach(@DATA) instead of for ($i = 0; $i <= $#DATA; $i++)"*
- Maker2: *"Your code the handle the dates has errors in it, u have explained some of them, but they are still there. Why have $mode = "help"? As if -h is detected only the usage msg should be output and no other operations carried out then you could have just put 'print "Usage ... "; exit;' within the argument handler."*
- Marker3: *"Only slight question mark is over lines 90-94 where there appears to be some code which has been turned into a comment so that it is not executed."*

Summary

The results above suggest that feedbacks from students of different abilities are similar. Every group of students focused on providing many comments with explanations on program readability. The percentage of feedbacks on each feedback category from students of average ability are not much different from students of good ability, since they have a potential to understand the program

better after analysing the other students' programs. However the group of students who have poor programming ability provided not many comments on the program correctness, as they did not fully understand the program, compared with the other student groups. Therefore all students have abilities to provide feedback on program readability and style of program but the ability of providing the feedback with good suggestions on program correctness depends on their programming abilities.

6.8 Questionnaire and interview analysis

At the end of the peer assessment exercise, each student was required to fill in an online questionnaire. In addition 20 volunteer students were interviewed. This section discusses the students' opinions in marking the program and how useful comments from peers were, with the following questions.

- Are the comments from peers useful?
- Are students satisfied with marks from peer assessment?
- What are the facilities that help students in marking the program?
- Do students feel comfortable when assigning marks?

6.8.1 Are the comments from peers useful?

Comments	Students' responses from			
	Questionnaire (Yes/No question) N = 104		Interview N = 20	
Useful	63	**61%**	12	**60%**
Most of them are useful	-	0%	5	25%
Not useful	41	39%	2	10%
Not certain	-	0%	1	5%
Total	104	100%	20	100%

Table 54 Responses from questionnaire and interview on useful comments

Results in Table 54 suggested that most students found the comments from peers were useful with the suggestions for improving their programming abilities, and helped them for doing the next assignment, as the following students' quotes.

- Suggest alternative program solutions - *"Some markers give me alternative answer to the same question which is good."*

- Point out program weaknesses - *"It helped me focus on the weaknesses that I had and gave me more confidence in my ability."*
- Suggestions for program structure improvement - *"They pointed out areas I could improve the commenting of my code and how I could improve the structure of my programs."*
- Help for next assignment - *"Since this assignment was the most complex, and worth the largest number of marks of the three, I paid close attention to the comments given by students in the previous peer reviews to ensure that the work I submitted was likely to be correct, readable, and easily understood, as this would probably result in a better mark."*
- Encourage being a good programmer - *"Comments are always helpful. Even if you have a perfect program that you spent a lot of time on, even if people just say,....., this is great encouragement. Or if they spot some mistake in your style or code, it is even better; you get to learn very important things. (being anonymous has a point here, you get less annoyed since you know the other person most surely concentrated on your code, not on yourself). **It is really a great opportunity to get into a programmers' atmosphere and have something to say.**"*
- Increase program readability- *"In my first assignment my structure and indentation was terrible and my lines were not 80 columns, and comments from people, from my peers helped me improve that, so next time it wasn't like that. And that's an example I remember. There were more, obviously. In comments I got back there were suggestions on how to go around things, **so I learnt from the comments.**"*

Students believe and follow the comments from peers for improving their programs, especially in order to gain more marks in the next peer assessment.

- *"I took on board the comments made in the previous peer review exercise to make my script for this one better than the first 2."*
- *"Useful to see what people were looking for, and better ways of implementing certain programming tasks."*

However some students found that some comments were not useful because markers did not read the code carefully in order to provide the good comments.

> *"Most of them. Some of them were just... they didn't really care. But occasionally there were some good comments with ideas of what's wrong with my code. It's very difficult to judge that yourself."*

6.8.2 Are students satisfied with marks from peer assessment?

> *"I felt the guys gave me fair marks, I admit that I did make most of the mistakes that they pointed out. good work guys!"*

There are two sets of marks awarded by students in peer assessment (i.e. quality of program and quality of marking). Results from questionnaires indicate that 74% (116 out of 156 students) of students were satisfied with marks from peer assessment. However some students were not satisfied with the marks awarded by peers because of the following reasons.

- Harsh marks for the quality of marking -*"I thought the marks received from part II were extremely harsh. Sure, I'm not the best marker in the world, but I feel that 10 and 15 out of 60 were unjustified."*
- Penalty marks for incomplete peer assessment process – *"I marked the first script but then managed to forget to mark the remaining 2. I do realize that this was fully my fault, but it does disappoint me."*
- Missing markers – *"my work was not marked by all 3 and I only scored 60/150 in the automatic tests when all of them were successful*
- Marker did not understand the program – *"Why should I be penalised for having other people not understand my code when two other markers understood it perfectly. And why should I lose marks if I answered it correctly in a different style choice than to that of the marker (Even when the markers method would probably take up more lines of code)."*
- Different interpretation of assignment specification– *"One guy marked me down, because he disagreed with my interpretation of the assignment, which I deliberately put as a comment in the code, so the marker could see what I was doing and why. Yet, the marker decided because he has a different interpretation"*

- Unfair mark comparing with the others- *"Not fair. I know some who didn't do much like didn't do much on error checking, multiple option get higher than those who did!!!"*

6.8.3 What are the facilities that help students in marking programs?

Facilities help in marking	N = 20
Automatic test result	12
Marking guidance (Things to consider)	7
Discussion (via anonymous communication tool)	0
All of the above lists	1

Table 55 The facilities that help students in marking the program

Automatic test result

"Yes, the automatic tests results, I did use this. And I was surprised actually at some of the tests that people failed, just very simple errors, one that I remember is ..... and I really couldn't understand how he'd could have not noticed and corrected this."

Results from interviewing 20 students found that the automatic test result is the most helpful (see Table 55) for students in understanding and marking the programs because it helps to highlight major problems in the code. However some students found that the automatic test results cannot cover everything. They liked to read through the script and find out how it related to the specification, although that script passed all the automatic tests. One student recommended that the peer assessment should not provide the automatic test result in order to force students to actually read through the script and try and understand it by themselves before marking.

Marking guidance

"It is really helpful when I marking, because at first I did not understand what they want me to do, the computer have been text the code already, if the result is well the code should have no problems."

The lecturer provided marking guidance in order to help students in marking the program assignment. From the results in questionnaire, 70% of students (110 out

of 158 students) found that the marking guidance or 'Things to consider' helped them in marking program because of the following reasons:

- help to outline exactly what students should be looking for when marking the scripts;
- help in trying to standardise the marking and provide consistent marks; how to be fair when marking;
- give an overview of how to mark, which help students who have no experience in marking before;
- specify in more depth what the assignment criteria wanted;
- guide which other utilities could be used;

Besides the marking guidance helping students in marking, students realised it also helped them in better understanding of the assignment, improving their programming abilities and guiding what they should consider in doing the next assignment.

- *"It helped me realise important aspects of the program that I did not consider."*
- *"Because it helped me draft my program better"*
- *"Gave you direction as to what you should try and improve."*
- *"Not only did it outline how this particular script could be done well but also good practice which could apply to any script"*
- *The things to consider gave a clear guideline as to where marks should be awarded. This is very helpful information for my programming as well.*

Discussion

The peer assessment system provided the anonymous communication tool that allowed students to discuss about the program and marking within their groups and ask questions about that script from the author. Because of technical problems with this tool, many students could not use it on their machine at home (as more than 89% (141 out of 157 students) of students reported in the online questionnaire).

- *"The peer communicator doesn't work properly."*

- *"I couldn't get the software working in my room"*
- *"Couldn't get Peer Communicator to work again."*

A few students reported that there was nobody online when they were marking the script resulting in no discussion. Only one student found that everything was clear without discussion. However 13 students who used this tool reported it helped them to understand the script more easily, especially complex and unclear code.

The results above suggest that the automatic test result, marking guidance, and discussion via the anonymous communication tool are helpful facilities in understanding and marking the programs, although there was a technical problem in using the anonymous communication tool.

> *"**The 'things to consider', I read through that first of all,** and jotted it down on a piece of paper, important bits, so I didn't forget anything. And then, looking through the automatic tests, you could see "well this bit doesn't work", and you'd spent.....If it doesn't work, you have to understand why it doesn't work, what you can do to change it, **so the automatic tests were a great big help. And the communication obviously is quite important,** because you want to know why they said something or whatever."*

6.8.4 Do students feel comfortable when assigning marks?

> *"Emm... not at first. In the first peer review I was like "hmmm... if I put it one lower then they're going to get a bad mark", But because you do have 3 you learn what's a good one, what's an OK one, and what's a bad one. So you are able to, if you see a good one, you'll be like "Oh, yeah definitely I'll give you a good mark", whereas if you see a bad one you'll be like "Well... I can't give you marks for that, but you've done alright on that bit, so maybe I can give you a few marks". **So as you've done 3, the 3rd one you understood what you did, and weren't really bothered about giving marks.**"*

Results from the online questionnaire indicate that 122 students felt comfortable when assigning mark, but 44 students did not. There are three main reasons (see details in Table 56) that caused students to feel uncomfortable when assigning marks:

- Not enough knowledge, not fully understanding the program
- Not enough confidence in assigning marks
- The marking guidance and marking criteria are not clear with the small answer scale

The major problem seems to be that students believe that they are not qualified to be a marker because they have just started to learn how to program and they do not have enough knowledge to mark their friends' work.

Students' opinions	N = 44
I have not enough knowledge, I am not qualified to be a marker	9
I have no experience in marking	5
I do not want to be a harsh marker	2
It's difficult for me to make a decision of which mark to give	6
I'm not confident to mark the program which better than mine	1
If it is difficult to follow what the program does, I give low mark	2
I'm not confidence in marking	1
The answer scale is very small	4
The marking guidance is not clear	2
The marking criteria is not clear	2
I don't want to be a marker, I don't want to read the other people's programs	1
I do not want to assign marks which count to the final grade	2
I like to give high marks because the others may mark in the same way as me	1
No comment	6

Table 56 The students' opinions on being comfortable when assigning marks

From interview, 11 out of 20 students felt comfortable when assigning marks, one student felt comfortable depending on the difficulty of the script. Most students understand what they are marking and 1-5 scales are appropriate and not difficult for them to make decisions on which marks to give. They seemed more comfortable with assigning marks in the subsequent peer assessment exercise because they learnt more from the other markers and they have more experience in marking as well.

> *"With the first assignment it was difficult to know where to pitch, how strict you were going to be, but after the second and by the time the 3rd came around it was much much easier, because you'd seen what other people had given you, so..."*

"Anonymous marking" is another reason that made students felt comfortable in assigning marks because they can concentrate on the program, not the person (friendship marking). However a few students found that assigning marks is easy, but commenting on a good program is difficult.

Summary

More than 60% of students agreed that the comments from peers were useful with suggestions of how to improve the program, which helped them in increasing their programming abilities. Moreover 74% of students were satisfied with the fair marks from peers. The marking facilities provided (i.e. automatic test result, marking guidance and anonymous communication tool) helped students in understanding and marking the programs. In addition, students felt more comfortable when assigning marks in the subsequent peer assessment exercises. All of these results indicate that students were satisfied with the marking results from their peers and felt comfortable in assigning marks in this peer assessment process.

6.9 Summary of the chapter

From the qualitative and quantitative analysis above it is suggested that peer assessment is an accurate assessment method in a programming course. There is a strong positive relationship between peers' and tutors' marks, and students and tutors identify similar commenting issues. Most students were satisfied with marks from peers and accepted that the comments from peers were useful. However students' marks are significantly higher than tutors' marks because they have a different perspective on mark scales. The marking in this peer assessment process is possible because the system provided automatic test results, marking guidance and an anonymous communication tool. The more marking students do, the more confident in marking they become. The answers for each sub research questions are summarised below.

I. *Question*: Is there a relationship between the marks given by the tutor and peer markers?

Answer: There is a strong positive relationship between tutors' marks and peers' marks at a high level of statistical significance in all 3 assignments. Students and tutors marked in the same way.

II. *Question*: Is there a significant difference between the marks given by the tutor and peer markers?

Answer: Peers' marks are higher than tutors' marks and it is a significant difference.

III. *Question*: For which marking criteria are peers' marking as accurate as tutors' marking?

Answer: Objective questions (i.e. the number of comments, the program meets the specification, appropriate code handles errors, the program finishes with an appropriate exit status, and appropriate utilities have been selected) and one subjective question - helpfulness of comments.

IV. *Question*: Do peers identify similar or different comment issues compared to tutors?

Answer: Peers identify similar comment issues to tutors with more quality of comments because of more explanation and suggestions than tutors, who provide concise comments without further explanation and suggestions. However they focused on the different issues i.e. tutor focused on correctness of program (especially the program meeting the specification, and programming mistakes), while peers focused on program readability and style.

V. *Question*: Are marks within a group consistent?

Answer: Yes, marks within a group are consistent. Few groups exhibit a high standard deviation of marks within the group, and the percentage

exhibiting a low standard deviation increased in the later assignments.

VI. *Question*: Are the comments from a wide range students' abilities different?

Answer: No, the comments from a wide range students' abilities are similar. Every group of students focused on providing a lot of comments with explanations on the program readability. However students who have poor programming ability provided few comments on the program correctness, as they did not fully understand the program. It can be said that all students have abilities to provide the good feedback on program readability and style of program, but good feedback on program correctness depend on their programming abilities.

6.9.1 Marking problems

Although peer assessment is an accurate method in a programming course because most marks and comments from between peers and tutors are similar based on the same marking criteria, there are a small number of marks and comments, which are different. The following is a summary of problems that caused the different marking between peers and tutors.

- The subjective questions (for example helpfulness of comment, easy to follow what the program does) are difficult for consistent marking.
- Tutors and peers have a different perspective on mark scales.
- Tutors and peers have different levels of knowledge and experience in analysing programs.
- Students awarded too generous marks because they hope that this would be reflected in the marks they received.
- Students hesitated to mark down programs that they are not confident with or do not fully understand.
- Students misunderstood marking the program correctness (meet specification, handle errors, appropriate exit status). They marked the program based on the number of marks from the automatic tests.
- Unusual scripts not catered for in the marking scheme.

Chapter 7 Peer Assessment for Essays

7.1 Overview of the chapter

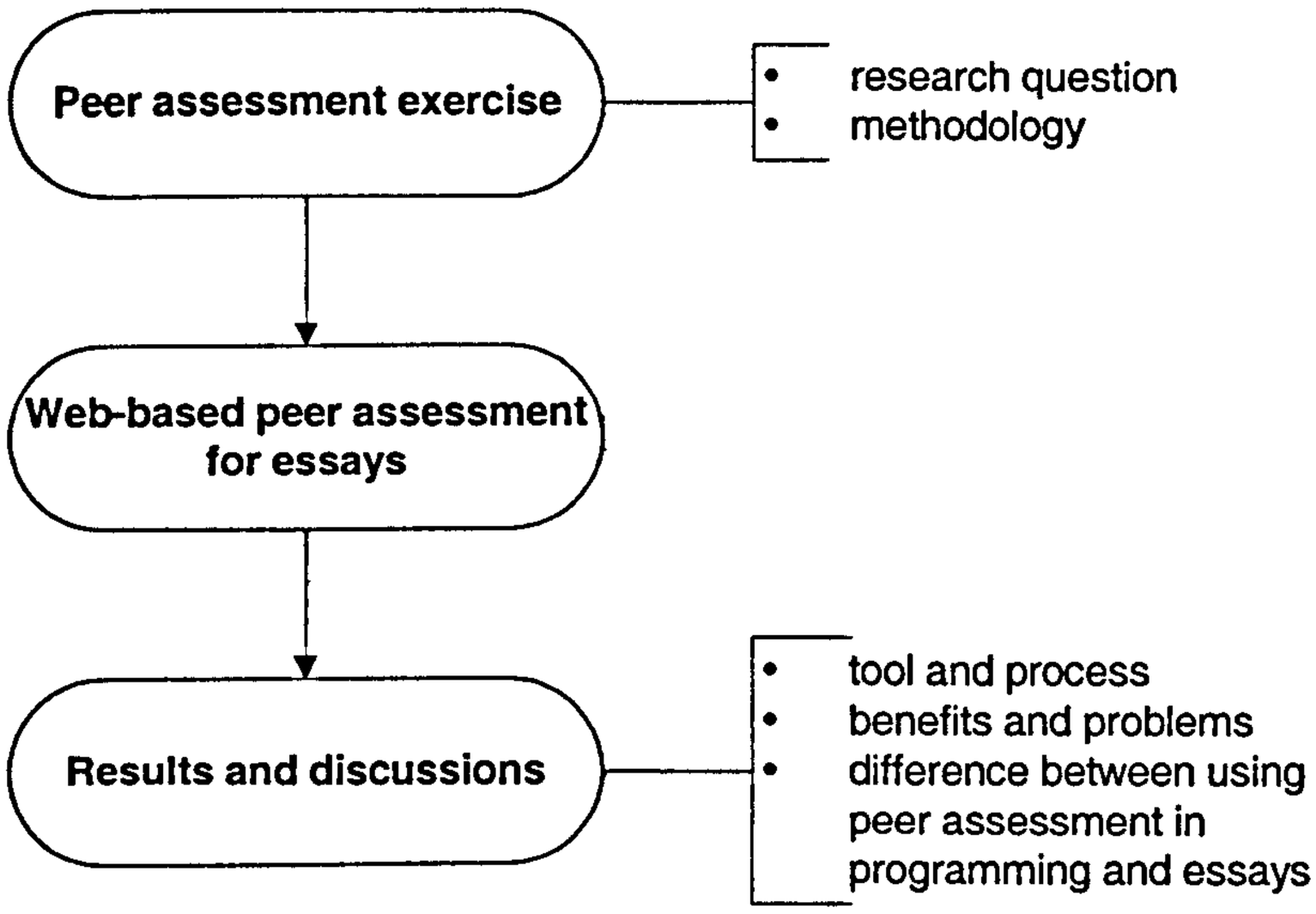


Figure 50 Outline of 'peer assessment for essays' chapter

The peer assessment for essays experiments were performed on first year undergraduate students who have used peer assessment for programming before. A simplified version of the web-based peer assessment tool for programming is used in this experiment. We describe the peer assessment for essay method, and the design of web-based peer assessment for essay system, followed by the results and discussions. The differences between using peer assessment in programming and essays are examined.

7.2 Peer assessment exercise

We use the phrase "peer review" instead of "peer assessment" in this exercise, because it involved students analysing each others' work without assessment (no marks were awarded), and it was used before submission. The method is shorter than peer assessment for programming, as marking the initial marking step is removed. Students were required to write 1500 word essays in 'professional aspects of computing' module (CS122). This module aims at "introducing students to the concept of professional ethics and behaviour, the place of

computers in society and the legal aspects of computing" [Simmonds2003]. Six students volunteered in this experiment. As the assignment deadline was after the vacation, there were not many students enthusiastic to finish their work earlier in order to participate in peer assessment. In addition, many students were concerned that the other students may steal their ideas, and they may get marked down for plagiarism.

7.2.1 Research question

Many peer assessment studies [Lin2001, Fiore2001, Davies2000] report the benefits of peer assessment for essay writing. We are interested in finding out how students in computer science perceived a similar exercise working for an essay assignment, so that we can make a comparison with their experience with programming module. Therefore students who have experienced peer assessment for programming were chosen to participate in this exercise. The purpose of performing the experiment in peer assessment for essays was:

- to investigate the differences between using web-based peer assessment in learning computer programming and writing essays (for students in computer science department).

7.2.2 Methodology

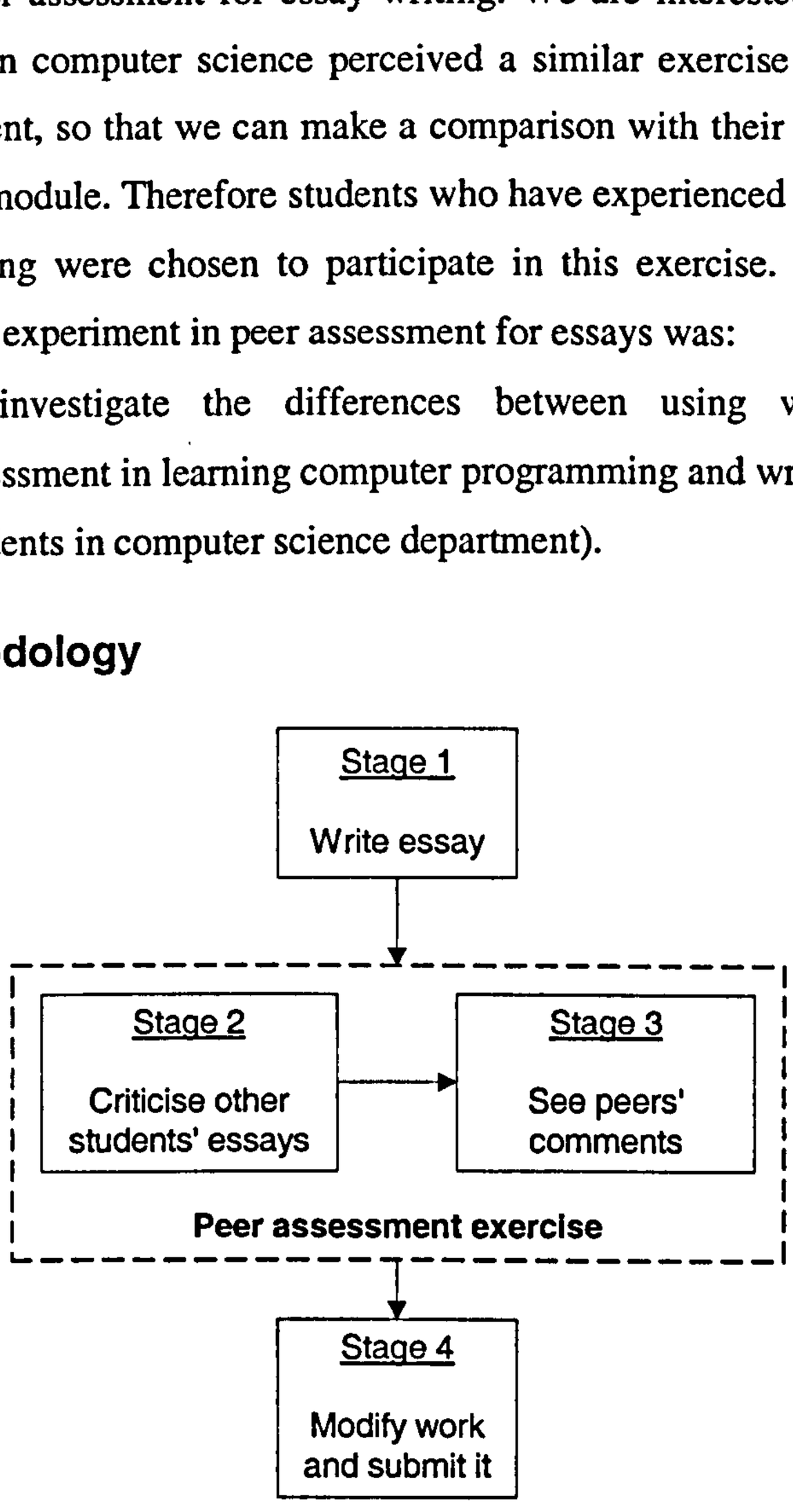


Figure 51 Peer assessment for essays method

This peer assessment process was divided into four separate stages, as shown in Figure 51. Students comment on each other's work via the web-based peer assessment tool for essays system, before submission.

Stage 1: Students write the 1500 word essay in their own time, and then email it to the administrator for participating in peer assessment exercise.

Stage 2: Students criticise each other's work (three essays) by answering questions about "good writing". Then they discuss their criticisms with partners who analysed the same essays in two hours of a lab session.

Stage 3: In the following day, students can view the peers' comments of their essays, both on the particular line (annotation) and other feedback.

Stage 4: Students modify their own work before submitting it to be marked by the tutor.

7.3 Web-based peer assessment for essays

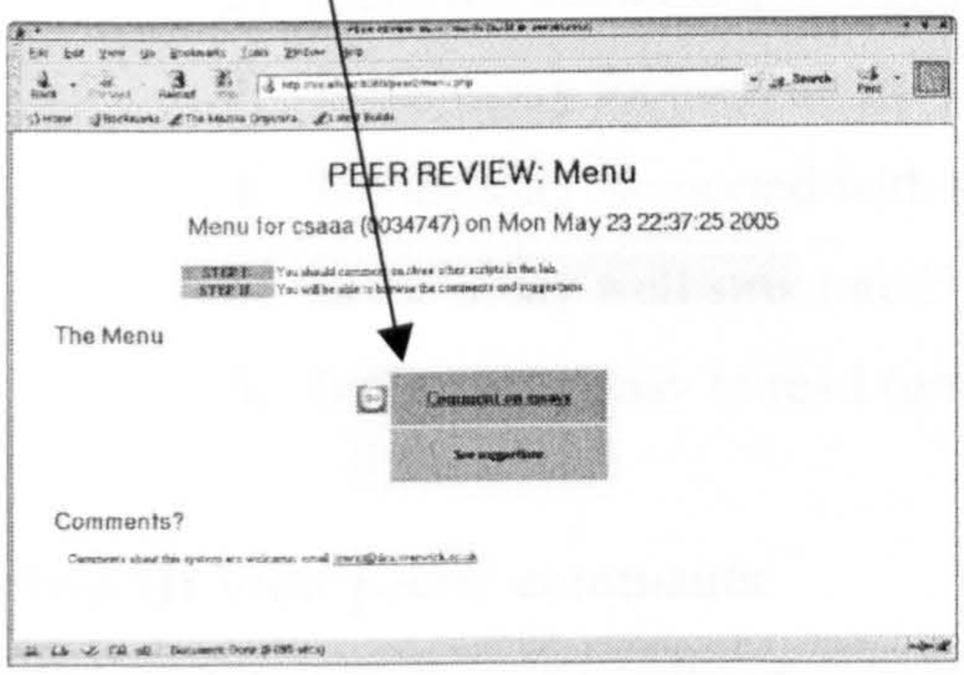


Figure 52 Menu web page

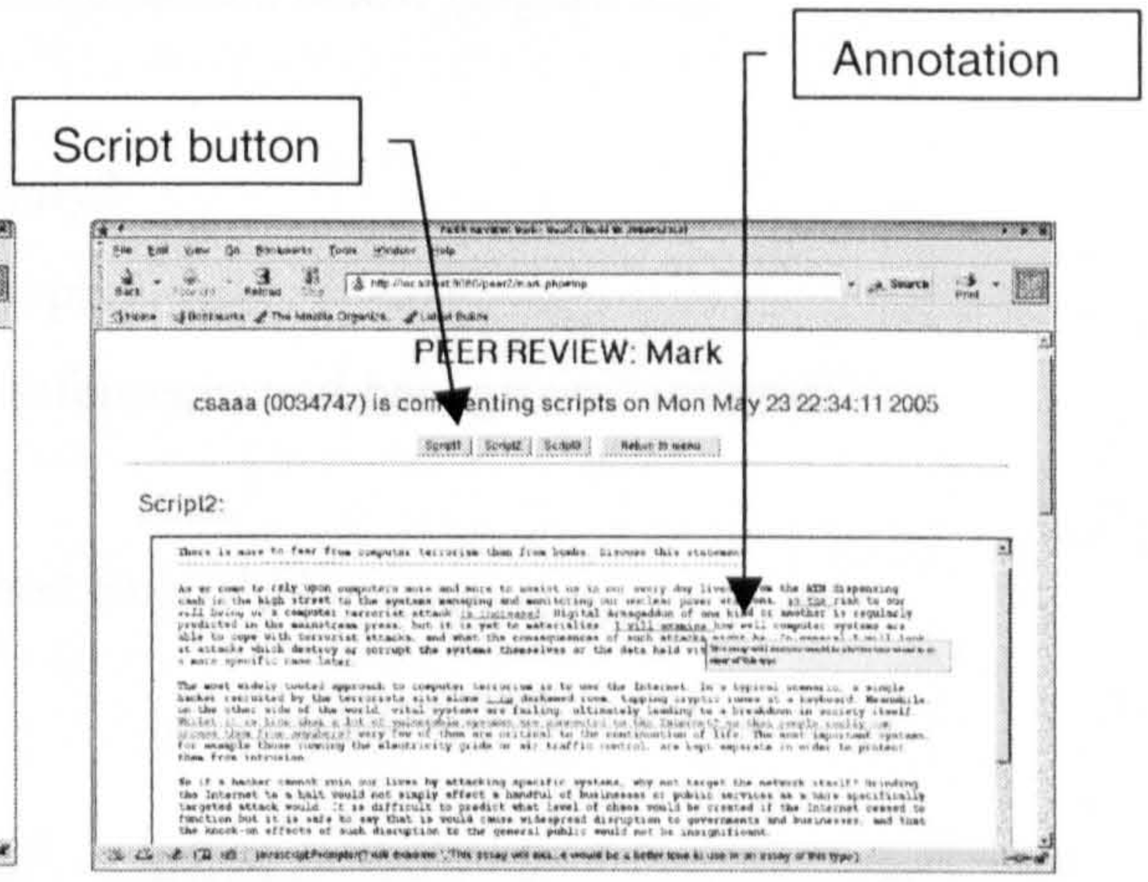


Figure 53 Mark web page

Web-based peer assessment for essays is a version of the web-based peer assessment for programming tool, simplified by removing marking of initial marking step (Figure 52). We have also extended the system by developing an annotation tool, to encourage more effective comments by associating them with specific lines of the essay (Figure 53). Two steps in peer assessment for essays are illustrated below.

Step I: Criticise essays

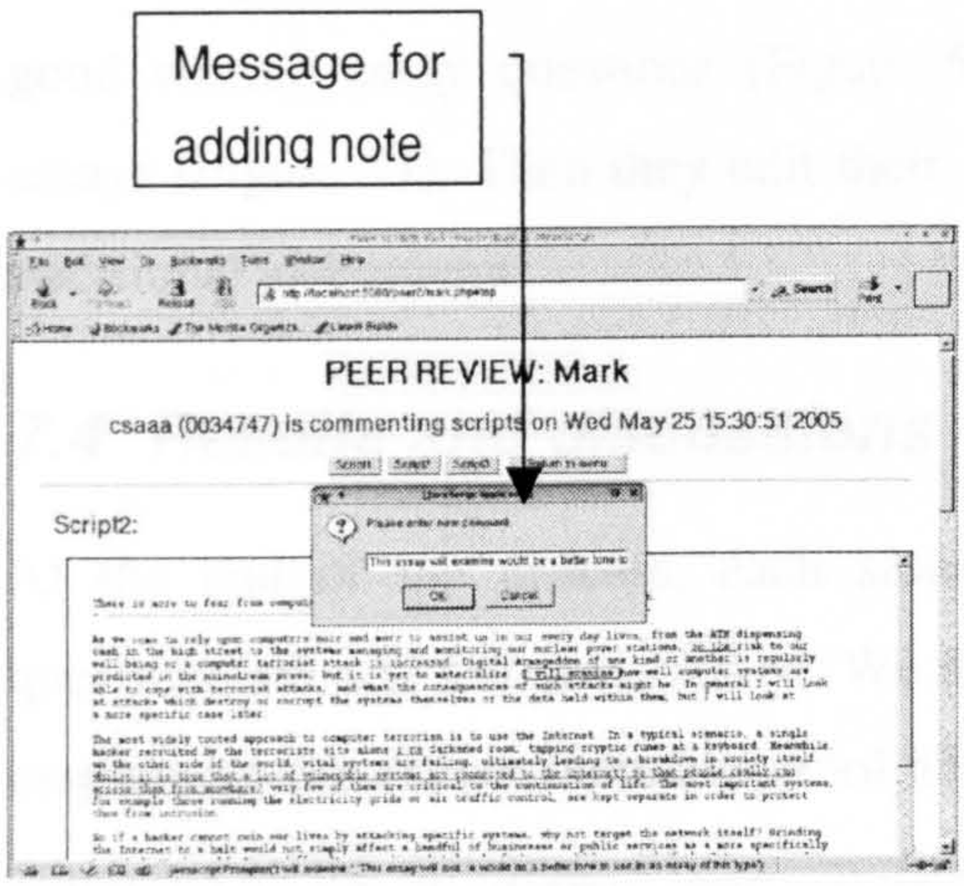


Figure 54 Annotation window

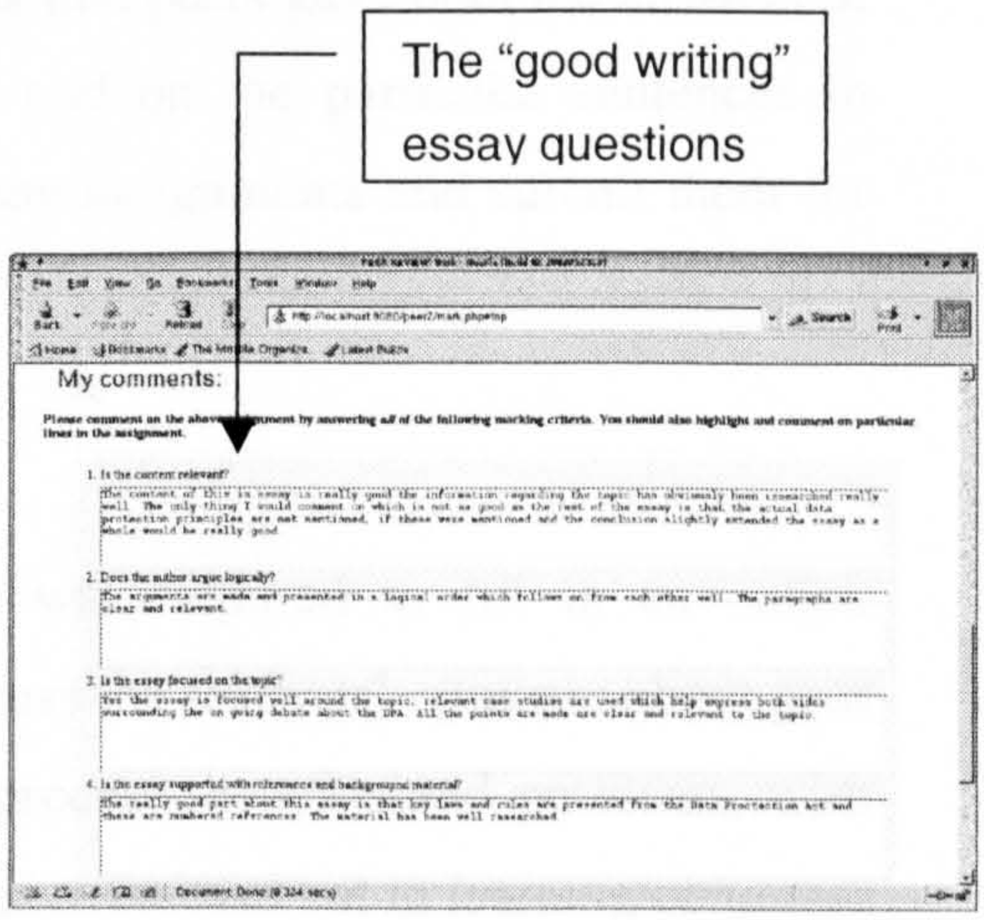


Figure 55 Comment web page

In this step, *firstly* students are required to highlight and comment on the particular lines in the essays. They can annotate easily by highlighting on the particular sentences, then the message box for adding notes will appear (Figure 54). Students can also edit and delete their notes by clicking on those highlighted sentences, then their note appears for editing or deleting. *Secondly* students should analyse the essay by answering the questions below (Figure 55).

1. Is the content relevant?
2. Does the author argue logically?
3. Is the essay focused on the topic?
4. Is the essay supported with references and background material?
5. Is the essay well structured?
6. Is the essay easy to read (good English)?

Step II: View peers' comments

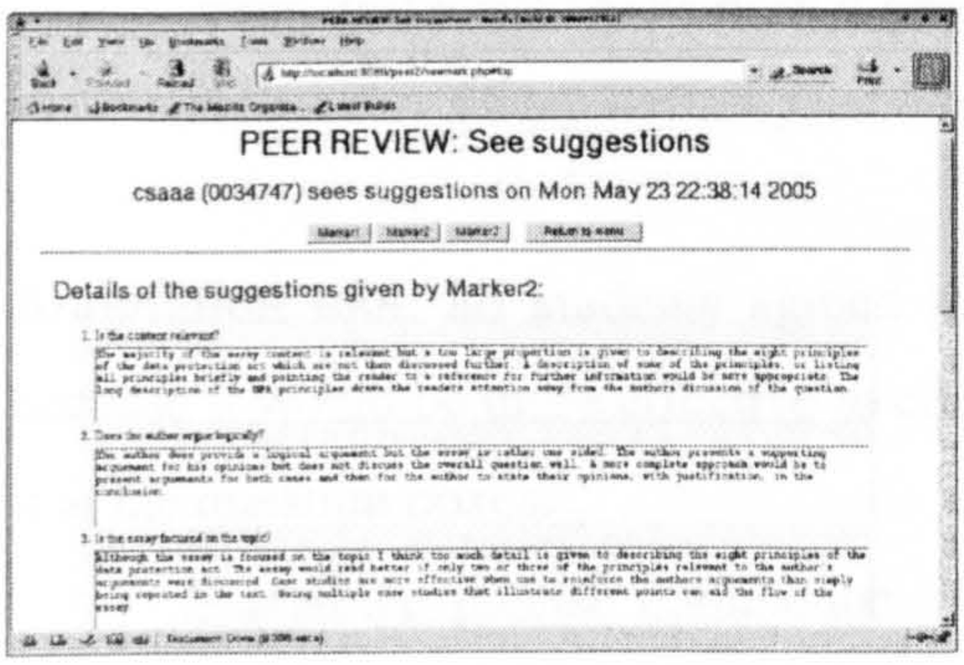


Figure 56 See suggestions web page 1

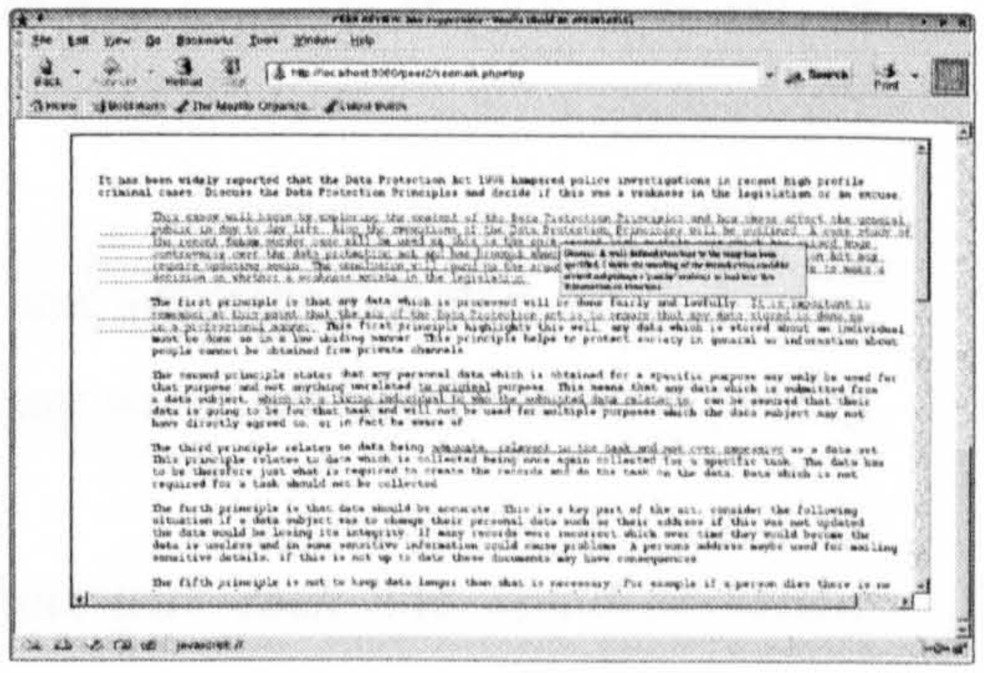


Figure 57 See suggestions web page 2

In this step, students can view the comments that peers gave both the answers of good writing essay questions (Figure 56) and on the particular sentences in essays (Figure 57). Then they edit their essay assignments and submit them for traditional assessment.

7.4 Results and discussions

At the end of the process, each student was required to fill in an online questionnaire and was interviewed. We discuss the students' opinions about peer assessment for essays including, tool and process, benefits and problems, with emphasis on the differences between using peer assessment in learning computer programming and writing essays.

7.4.1 Tool and process

Summary results from the questionnaire and interview yield the following information.

- 2 out of 4 students discussed (face to face) with their partners when analysing essays, about major problems, the structure and the style of the essay. The other two students preferred to discuss anonymously via the online communication tool.
- All students thought that web-based peer review is easy to use with the simple interface.

  "The web based peer to peer software is easy to use and logical."
- Peer assessment for essays is considered as a peer review method, providing feedback to improve the essay writing.

  "The fact that there is no overall score awarded by the marker for an essay, this would have been a guide to how good the essay was as well as the marker giving written feed back."

Annotation tool: all students agreed that the annotation tool is a good idea, because it would be more effective to comment with each line rather than putting it at the question boxes.

- Encourage precise comments: *"I think with each line would be better, yeah. It was definitely a good idea to just put the line there. For example, if it was done with the coding then **I could just look through my code***

and know exactly where the problem was, or what exactly could be improved. Whereas with reading just text box, you just read it generally and don't put that much thought into it, I suppose."

- Increase the quality of comments: *"I think, yeah, I think you can write a better comment because* ***you don't have to explain where in the text that it was, it's already there,*** *you know it's in that bit of text, and then you can explain with short comments, like... I would prefer less colloquial expressions. Without having to explain at the bottom, "in paragraph 2, in sentence 3, you wrote this,"* ***you can just say it, and it means you can write more comments in the time, which means that you can improve the overall quality of your commenting.*** *Whereas if you have to write long explanations it means you'll write less comments, and it will be less helpful, less quality overall. Important skills that are required in writing and comment on essays: good English (spelling and grammatical corrections), concise and precise about what you are saying."*

Peer assessment before submission: peer assessment for essays before submission may not work well. Since students may interpret the question differently, they do not want the other to copy their innovative ideas. Some students may just want to observe other students' answers, without handing in the good essay.

- Hand in different essay: *"Yeah but some people, seriously, they might not be helpful at all because* ***they know that if they hand in this essay and you read it then you might steal all your ideas,*** *so you might miss them out and improve it later and hand in a different essay."*
- Stealing ideas: *"I think it's not fair to take everybody else's ideas and interpretation and use it. I mean even* ***if you put it in your own words, it's still taking their ideas really."***
- Unique idea: *"I think if you write an essay you're looking for something, you might be looking to say something that nobody else has said, so* ***you might want to get an idea into your essay that nobody else would have thought of,*** *which is a lot different to programming.* ***You might have something that you really want to say that you don't want anyone else to see."***

7.4.2 Benefits and problems

Summary results from the questionnaire and interview yield the following information.

- 3 out of 4 students thought that seeing different ways of writing an essay help them to write better essays.

  "It helped me think about how to develop my essay with more structure and relate things back to the essay title"

- All students considered that peer review exercise helped them to evaluate the quality of the essays they wrote.

  " I think it gave me an insight into what other people thought of my work, and also what a marker would feel marking the same assignment. This has helped me think about refining my essay"

- All students agreed that the comments from peers were useful and identified the mistakes that they have.

  "Yes. Feedback was useful. Helps to improve the essay."

Therefore peer assessment for essays provides the benefits to improve essay writing. Students can observe the different styles of essays and receive the useful feedback.

> *"People think about what you have written before it actually is marked.* ***This gives the author a chance to refine their work before handing in.*** *Also if the author has strayed from the point or made several errors the peer can point this out leading to better quality work being produced."*

However several problems of peer assessment for essays are reported, such as the difficulty to concentrate on reading essays on screen, students found reading on paper a lot easier; lack of a communication tool to discuss with the authors and another marker; repeating comments on both annotation and in question boxes.

> *" The best part is being able to highlight sections and comment them, you sometimes find yourself repeating what you said there in the question boxes however."*

7.4.3 Using peer assessment in programming and essays

"The main difference being that when you write a program you have the opportunity to perfect it, where as with a essay you do not have the benefit of being able to test it. Also when writing essays you need to have a skill to be able to communicate with a reader and reflect your ideas ***accurately.*** *This is a completely different skill than what is required to write a successful computer program"*

When writing a computer program, students can test the output, and correct the mistakes, but writing an essay depends on students' interpretation of the title, with no right or wrong answer. Students should be able to express themselves in good English language in writing essays. Results from the experiment revealed that computer science students consider peer assessment is more useful in programming than in essays. Students' opinions about the differences between using peer assessment in learning computer programming and writing essays are:

- **analysis:** analysing essays is more difficult than analysing programs. There are no right or wrong answers for essays. Students may interpret an essay title differently.

  "It's too judgemental. Different markers will give different opinions on an essay. In some cases ***there are no right ways to write a sentence or an essay.*** *Even if you mark someone's essay, or* ***a different day you may interpret it differently after reading someone else.*** *But there are wrong ways of writing program."*

- **skill development:** feedback from peer assessment may provide more benefits to students in learning programming than writing essays. It helps them to improve the programming skills, and prevent the same mistakes from happening again in the future.

  "I suppose with CS122, even after I got the feedback from people I didn't change anything in my essay so... I think especially in writing an essay, you can't change much in a short period of time. ***Whereas in programming it's a challenge to look at other solutions, for future***

improvements. It helps you to develop your problem solving skills, other than improving your own code for the certain assignment."

- **accomplished task:** writing a good essay is much more dependent on their understanding and good research on each topic rather than observing other students' essays.

  "I think programming is definitely a good idea, even if it's after submitting the coursework, it just helps to look at how problems can be solved in different ways. With the essay I'm not to keen about it before or after, to be honest. It's good to look through other people's essays, it's interesting but it doesn't help me personally, if I was to write my essay again. Because at the end of the day ***if you've done a lot of research then you develop your own idea, and you're not that easy to convince by just another essay."***

- **language:** analysing an essay is more difficult than marking programming for the international students who are not native speakers, since they may have difficulty with the English grammar and spelling.

  "Yes, yeah. I think that especially with programming it worked, ***I didn't quite like it with the CS122 because it was reading an essay, and I'm a non-English speaker,*** *and all of them sounded good to me, to be honest. I can't spot the exact problem with structure of an essay, that sort of thing, it's better than from an academic point of view.* ***But with programming we're all learning at the same problem,*** *we're all solving the same problem. It's not a matter of somebody having a different idea to somebody else or going in a different direction, we have targets."*

- **plagiarism:** students can apply other students' programming styles to improve the quality of their programs. However, they cannot apply other students' essay writing styles, and it may cause plagiarism.

  "For an essay I might not generally use someone else's way of writing it as the thing. I'd still keep my own style probably. While for a program I'd say "Oh look, he's managed to do this code in 100 lines whereas I've used 3 times that amount. He's managed to simplify, this is really useful, I can

use that kind of stuff," Whereas for an essay you'd say... you wouldn't be able to use that kind of stuff, it would be plagiarising on his style. But for programming you'd see that if he uses all these different subroutines then it's far better than me just writing it out with no subroutines, or something."

7.5 Summary of the chapter

We have described a peer assessment for essays process, together with supporting web-based software. An annotation tool has been developed to encourage more effective comments. Although peer assessment for essays provides benefits to students in improving the quality of essay writing, the results indicated that peer assessment in essays might not be as useful as in programming for students in a computer science department. Analysing other students' programs is challenging, and helps them to improve their programming skills, whereas writing good essays depends on good research and interpretation of the titles. In addition, fluency in English language is necessary and may be more difficult for the international students in analysing essays than programming.

Chapter 8 Conclusions

8.1 Checklist for designing a peer assessment system

The success of an online exercise such as our web-based peer assessment depends on four steps: design, software development, deployment and monitoring.

(√) Design

- **Process:** in many peer assessment exercises, students only mark and provide feedback on other work. Some students do not perform this task seriously because it does not affect their mark. Therefore we should add another stage – *mark quality of feedback* - in order to encourage students to take the assessment seriously and make this marking process more effective. The average mark for this part of the exercise was 79%, indicating that students provided quality marking and feedback. This stage also helps students to develop their critical judgement skills, they can see how other students mark and provide feedback to compare with their own judgement. Thus the more marking students did, the better their own results became [Bhalerao2001].
- **Anonymity:** peer assessment should be anonymous, and this is supported by interviews with our students, in which 93% of students interviewed agreed, in order to avoid friendship marking and lessen the embarrassment of marking friends' work. This aspect also has been reported by Segers et al. [Segers2001] and Davies [Davies2000], that anonymous peer assessment reduces the opportunity of collusion and biased marking.
- **Group discussion:** group discussion is an important factor in the peer assessment process. Students can share knowledge and learning [Andriessen2003], resulting in greater understanding of the assignment and the development of transferable interpersonal skills. In our research, we divided students into small groups (three students per group), each group consisting of students with a range of abilities. Each

student was assigned three other students' assignments to mark from other groups (each consists of a mixed range of quality of assignments). They can discuss their marking with the other students in their group, who marked the same assignments. In the experiment, students were allocated timetabled laboratory sessions where they were able to talk together in a group, with tutors present to offer assistance if required. With face to face discussion may arise the problem of students being unwilling to talk to each other, and feeling intimidated about asking questions. Furthermore the absence of some members of the group may complicate the process. Thus an anonymous communication system is recommended to solve these problems, where students can discuss online or leave offline messages to their anonymous group.

- **Marking criteria and guidelines:** marking criteria should be clear and marking scales should offer appropriate choices for marking (such as a 5-point Likert scale) in order to help students in making an accurate and fair judgment [Miller2003]. Using specific marking criteria also helps students to understand what is expected of a good program. Orsmond et al. [Orsmond2002] reported that students seemed unenthusiastic about creating marking criteria themselves, therefore in this peer assessment process, marking criteria were set by the teacher. As students have different levels of knowledge and ability in marking, marking guidelines or 'things to consider' should be provided to point out potential aspects of a good answer. We found that 95% of students understood the marking criteria and guidelines, indicating that the clarity was appropriate. In this peer assessment process, we also provided the students' automatic test results as well. Students can see the automatic tests, the expected output and the actual output, which helps them in marking.

(√) **Software development**

In this peer assessment process, students submit assignments via the department's online submission system [Joy1998], and then evaluate each other's work through the web. Web-based peer assessment has advantages over ordinary peer assessment because students can be more critical due to

the anonymity the system can provide. Although each student initially marks in a laboratory environment (in order that support can be provided by tutors) they can finish the process wherever and whenever they choose before the marking deadline, thus allowing anonymity to be preserved. The processing of the marks is automated, and teachers can easily monitor the marking. Provision of online discussion and an offline messaging system helps students in discussing marking.

(√) **Deployment**

One of the major factors in the successful implementation of peer assessment is to be careful in introducing it to staff and students, and to ensure they have a correct understanding of the objectives and benefits in enhancing learning rather than just as a method of grading [Fallows2001]. A few students misunderstand the purpose of peer assessment, thus it is important to provide enough time to discuss the process with students at the beginning of the exercise. Moreover the following issues also should be considered.

- Taking the mystery out of the process, enabling students to appreciate why and how marks are awarded [Brindley1998].
- Training students to rely on their own judgement and their peers, and to develop a belief that a tutor or lecturer is a coach, who supports and adjusts the decision that students make [Sluijsmans2002].
- Using of a tutor moderation system would be of value to the peer assessment process to prevent cheating and enable the process to be seen to be fair [Ballantyne2002].

(√) **Monitoring**

It is inevitable that issues will arise during such an experiment, such as student absences, and students who – for whatever reason – do not fully engage with the process. Monitoring of the exercise is thus an essential part of the process, to ensure that no student receives an unfair mark or inappropriate feedback. Availability of appropriate basic statistical data, such as the standard deviation of marks for each group of markers, allows the teacher to intervene and – if appropriate – remark. In order to evaluate

the accuracy of the marks awarded by the students, tutors also remarked all the students' pieces of work.

8.2 Summary of the thesis

> *"I think for subjects like computer science, peer review for programming which is basically computer science, because you're programming all the time and I think* ***it's very good to have other student's views on what you're doing,*** *because I think it really helps you, it makes you more confident.* ***It makes you actually make an effort to make your program better.*** *So I think peer review for programming is a very good idea and you should definitely continue with it."*

We have described theories of learning and learning computer programming pedagogy, together with tools for learning programming that can be classified as surface and deep learning tools. The design and implementation of a novel web-based peer assessment system are described. The results of evaluating the tools through several experiments involving large programming classes and essay writing module are reported. Our investigation in peer assessment for programming revealed three important issues.

- Peer assessment enhances students' higher cognitive skills. Analysis, evaluation, synthesis, self assessment, and critical judgement skills that are parts of higher cognitive skills, can develop in the peer assessment process.
- Peer assessment is an accurate assessment method in a programming course. There is a strong positive relationship between peers' and tutors' marks, and students and tutors identify similar comment issues.
- Peer assessment is a useful technique for computer science students in learning programming more than in essay writing. Analysing other students' programs is challenging, and helps them to improve their programming skills, whereas writing good essays depends on good research and their interpretation of the essay title.

In this thesis, we also proposed:

- a new variation of Bloom's taxonomy, which is appropriate to describe the skills required for tasks such as programming, and
- a checklist for designing a peer assessment system.

The process we used is novel, since students are engaged not only in marking each other's work, but also in evaluating the quality of marking of their peers. This stage helps students to develop their critical judgment skills and encourages them to take the assessor role in the previous stage seriously. Moreover this system is designed to provide anonymity for the whole process, in order to ensure the process is fair, and to encourage students to discuss without embarrassment by using an anonymous communication device (ACD). Encouraging interaction with others is a key element in fostering deep learning, and students have reflected on their own ideas by discussing using the ACD in a variety of roles (script authors, marker, and feedback marker). Thus this novel web-based peer assessment which includes the ACD has advantages over ordinary peer assessment because students can be more critical (due to anonymity the system provided), they can discuss online and/or leave offline messages, the processing of the marks is automated, and the lecturer can easily monitor the marking and conversation.

Although this study indicates that peer assessment has contributed positively, it is a time consuming process. Therefore some students prefer the traditional marking by module tutors, rather than spend more time on this exercise. In addition it should be noted that peers might not have adequate knowledge and experience to evaluate others' work, even when guidance and well-explained marking criteria are provided. The nature of the programming assignment did not lend itself to there being only one model answer, and a variety of styles of solution were possible. The tutors should therefore give students adequate guidance during the marking process to assist students.

8.3 Future work

Although evaluation of the web-based peer assessment system indicates that it has contributed positively to the students' learning experience, further research in this area is suggested.

- The effectiveness of different marking criteria within the context of a peer assessment tool for computer programming.
- The application of other communication technologies (e.g. forums) within the tool to allow student dialogue outside of their allocated groups.
- Provision of sample "good" and "bad" programs to ensure that all students see programs from the full range of quality.
- Inclusion of a self assessment component to allow students to reflect on their skills before they take on board their peers' views.
- Inclusion of technologies (such as email, SMS) within such a tool to assist students in their time management when engaging with the tool.
- Case studies with different students cohorts, outside of the Warwick context.

Chapter 9 References

AAHE1993 AAHE Bulletin. (1993). Deep Learning, Surface Learning. *American Association for Higher Education (AAHE) Bulletin*, **45**(8), pp. 10-13.

Aggelia2004 Aggelia Internet Publishing. (2004). *Bibles studies: Kohlberg's stages of moral development.* [Online]. (URL http://www.aggelia.com/htdocs/kohlberg.shtml).

Altamura1995 Altamura, O. and Roselli, T. (1995). Hyperex: An Intelligent Tutoring Hypertext System for Learning Programming. *International Conference on Multimedia Computing and Systems*, Washington, DC, May 15-18, pp. 341-344.

Anderson2001 Anderson, L., Krathwohl, D., Airasian, P., Cruikshank, K., Mayer, R., Pintrich P., Raths, J., and Wittrock, M. (2001). *A Taxonomy for Learning, Teaching, and Assessing: a revision of Bloom's taxonomy of educational objectives.* Complete edition. United States: Addison Wesley Longman, Inc.

Andriessen2003 Andriessen, J. (2003). *Working with Groupware: Understanding and Evaluating Collaboration Technology.* UK: Springer-Verlag London Limited.

Armstrong1998 Armstrong, T. and Idaho Virtual Campus. (1998). *Bloom's Taxonomy: Sample Questions*. [Online]. (URL http://www.officeport.com/edu911/bloomq.html).

Azalov2003 Azalov, P. and Zlatarova, F. (2003). Teaching Programming through Successive Problem Transformations. *JCSC*, **18**(4), pp. 175-182.

Baldwin2001 Baldwin, L. and Kuljis, J. (2001). Learning Programming Using Program Visualization Techniques. *The 34th Hawaii International Conference on System Sciences*, USA: Hawaii.

Ballantyne2002 Ballantyne, R., Hughes, K. and Mylonas, A. (2002). Developing Procedures for Implementing Peer Assessment in Large Classes Using an Action Research Process. *Assessment & Evaluation in Higher Education*, **27**(5), pp. 427-441.

Barak2005 Barak, M. (2005). From order to disorder: the role of computer-based electronics projects on fostering of higher-order cognitive skills. *Computers & Education,* **45**, pp. 231-243.

Barg2000 Barg, M., Fekete, A., Greening, T., Hollands, O., Kay, J., Kingston, H. and Crawford, K. (2000). Problem-Based Learning for Foundation Computer Science Courses. *Computer Science Education*, **10**(2), pp. 109-128.

Beattie1997 Beattie, V., Collins, B., and McInnes, B. (1997). Deep and surface learning: a simple or simplistic dichotomy? *Accounting Education,* **6**(1), pp. 1-12.

Bell1995 Bell, J. and Scott, F. (1995). The Investigation and Application of Virtual Reality as an Educational Tool. *American Society for Engineering Education 1995 Annual Conference*, June 1995.

Best1989 Best, J., and Kahn, J. (1989). *Research in Education.* 6th ed. New Jersey: Prentice Hall.

Bhalerao2001 Bhalerao, A. and Ward, A. (2001). Towards electronically assisted peer assessment: a case study. *Association for Learning Technology Journal (ALT-J)*, 9(1), pp. 26-37.

Biggs2001 Biggs, J. (2001). Assessing for Quality in Learning, Assessment To Promote Deep Learning. *American Association for Higher Education (AAHE)*, Washington, DC, pp. 70-73.

Bloom1956 Bloom, B., and David R. (1956). *Taxonomy of Educational Objectives: The Classification of Educational Goals – Handbook I: Cognitive Domain.* London: Longman Group Ltd.

Blumenstein2004 Blumenstein, M., Green, S., Nguyen, A. and Muthukkumarasamy, V. (2004). An Experimental Analysis of GAME: A Generic Automated Marking Environment. *Proceedings of the 9th Annual SIGCSE Conference on Innovation and Technology in Computer Science Education: ITiCSE 2004*, UK: Leeds, June 28-30, 2004.

Boud1989 Boud, D. and Falchikov, N. (1989). Quantitative studies of self-assessment in higher education: a critical analysis of findings. *Higher Education*, **18**, pp. 529-549.

Boud1997 Boud, D. and Feletti, G. (1997). *The challenge of problem based learning*. 2nd ed. London: Kogan Page.

Bowling2002 Bowling, A. (2002). *Research methods in health: investigating health and health services*. 2nd ed. Buckingham: Open University Press.

Boyle2003 Boyle, T., Bradley, C., Chalk, P., Jones, R. and Pickard, P. (2003). Using Blended Learning to Improve Student Success Rates in Learning to Program. *Journal of Educational Media*, **28**, pp. 165-178.

Brindley1998 Brindley, C. and Scoffield, S. (1998). Peer Assessment in Undergraduate Programmes. *Teaching in Higher Education*, **3**(1), pp 79-89.

Brown1994 Brown, S., and Knight, P. (1994). *Assessing Learners in Higher Education.* London: Kogan Page.

Brown1997 Brown, G., Bull, J., and Pendlebury, M. (1997). *Assessing student learning in higher education.* London: Routledge, pp. 21-39.

Bryman2001 Bryman, A., and Cramer, D. (2001). *Quantitative Data Analysis with SPSS Release 10 for Windows: A guide for Social Scientists.* London: Routledge.

Buck2001 Buck, D. and Stucki, D. (2001). JkarelRobot: A Case Study in Supporting Levels of Cognitive Development in the Computer Science Curriculum. *SIGCSE*, **2**(1), pp. 16-20.

Buckland2001 Buckland, W. (2001). Promoting Deep Learning through the Use of Effective Textbooks. *Cinema Journal*, **41**(1), pp.121-127.

Byrne2001 Byrne, P. and Lyons, G. (2001). The Effect of Student Attributes on Success in Programming. *ACM*, **33**(3), pp. 49-52.

Campbell2002 Campbell, W. and Bolker, E. (2002). Teaching programming by immersion, reading, and writing. *32nd ASEE/IEEE Frontiers in Education Conference*, MA: Boston, November 6-9, 2002.

Carbonell2004 Carbonell, L. (2004). *Learning Theory.* [Online]. (URL http://www.my-ecoach.com/idtimeline/learningtheory.html).

Carlisle2005 Carlisle, M., Wilson, T., Humphrier, J. and Hadfield, S. (2005). RAPTOR: A Visual Programming Environment for Teaching Algorithmic Problem Solving. *ACM SIGCSE bulletin*, Missouri: St. Louis, February 23-27, 2005, pp. 176-180.

CELT2001 Centre for the Enhancement of Learning and Teaching (CELT). How Student Learn. [Online]. (URL http://www2.rgu.ac.uk/subj/eds/pgcert/how/how1.htm).

Chatfield1983 Chatfield, C. (1983). *Statistics for technology*. 3rd ed. London: Chapman and Hall.

Chin2000 Chin, C., and Brown, D. (2000). Learning in Science: A Comparison of Deep and Surface Approaches. *Journal of Research in Science Teaching*, **37**(2), pp. 109-138.

Churcher1998 Churcher, N., Cockburn, A., McMaster, B. and Horlor, J. (1998). CUTE – The Design and Evolution of a First Year Programming Environment. *Proceedings of the 1998 International Conference on Software Engineering: Education & Practice*, Washington DC, January 26-29, 1998, pp. 142-148.

Clark1999 Clark, D. (1999, June 5). Learning Domains or Bloom's Taxonomy. [Online]. (URL http://www.nwlink.com/~donclark/hrd/bloom/html).

Clear1997 Clear, T. (1997). The nature of cognition and action. *ACM SIGCSE Bulletin*, **29**(4), pp. 25-29.

Cohen2000 Cohen, L., Manion, L. and Morrison, K. (2000). Research Methods in Education. 5th ed. London: RoutledgeFalmer.

Comp2004 Competencies Module, William Howard School. *Critical Judgement.* [Online]. (URL

http://www.williamhoward.cumbria.sch.uk/leading_edge/competencies/critical_judgement.htm)

Corrosion1999 Corrosion Doctors. (1999, August 8). *Bloom's Taxonomy for Corrosion Training*. [Online] (URL http://www.corrosion-doctors.org/Training/Bloom.htm).

Coull2003 Coull, N., Duncan, I., Archibald, J. and Lund, G. (2003). Helping Novice Programmers Interpret Compiler Error Messages. *In:* O'Reilly, U. *4th Annual Conference of the LTSN Centre for Information and Computer Sciences*, Ireland: NUI Galway, 26-28 August 2003, pp. 257-258

Cox1998 Cox, K., and Clark, D. (1998). The Use of Formative Quizzes for Deep Learning. *Computers and Education*, **30**(3 / 4), pp. 157-167.

Craighill1994 Craighill, E., Fong, M., Skinner, K., Lang, R. and Gruenefeldt, K. (1994). SCOOT: An Object-Oriented Toolkit for Multimedia Collaboration. *Proceedings of the second ACM international conference on Multimedia*, California: San Francisco, October 15-20, 1994.

Crain1985 Crain, W. (1985). *Theories of development: Kohlberg's stages of moral development.* [Online]. (URL http://faculty.plts.edu/gpence/html/kohlberg.htm).

Cramer1998 Cramer, D. (1998). *Fundamental Statistics for Social Research: Step-by-step calculations and computer techniques using SPSS for Windows.* London: Routledge.

CRS2003 Creative Research Systems. (2003). *The survey system.* [Online] (URL http://www.surveysystem.com/sscalc.htm#terminology).

Daly1999 Daly, C. (1999). RoboProf and an Introductory Computer Programming Course. *Proceedings of the Annual*

Conference on Innovation and Technology in Computer Science Education: ITiCSE 1999, Poland: Cracow.

Davies2000 Davies, P. (2000). Computerized Peer Assessment. *Innovations in Education and Training International (IETI)*, **37**(4), 346-355.

Deek1998 Deek, F. and McHugh, J. (1998). A Survey and Critical Analysis of Tools for Learning Programming. *Computer Science Education*, 8(2), pp.130-178.

Deek1999 Deek, F. (1999). The Software Process: A Parallel Approach through Problem Solving and Program Development. *Computer Science Education*, **9**(1), pp. 43-70.

Diwan2004 Diwan, A., Waite, W. and Jackson, M. (2004). PL-Detective: A System for Teaching Programming Language Concepts. *ACM SIGCSE bulletin*, Virginia: Norfolk, March 3-7, 2004, pp. 80-84.

Dochy1997 Dochy, F. and McDowell, L. (1997). Assessments as a tool for learning. *Studies in Educational Evaluation*, **23**(4), pp. 279-298.

Elliott1996 Elliott, S., Kratochwill, T., Littlefield, J. and Travers, J. (1996). *Educational Psychology: effective teaching, effective learning*. 2nd ed. London: Brown & Benchmark publishers.

Ellis1998 Ellis, A., Carswell, L., Bernat, A., Deveaux, D., Frison, P., Meisalo, V., Meyer, J., Nulden, U., Rugelj, J. and Tarhio, J. (1998). Resources, Tools, and Techniques for Problem Based Learning in Computing. *Proceeding of ITiCSE'98: Working Group Reports*, pp. 41-56.

Entwistle2001 Entwistle, N. (2001). *Promoting Deep Learning Through Teaching and Assessment: Assessment To Promote Deep Learning*. Washington: American Association for Higher Education (AAHE).

Falchikov2001 Falchikov, N. (2001). *Learning together: peer tutoring in higher education*, London: RoutledgeFalmer.

Falchikov1995 Falchikov, N. (1995). Peer feedback marking – developing peer assessment. *Innovations in Education and Training International*, **32**, pp. 175-187.

Fallows2001 Fallows, S. and Chandramohan, B. (2001). Multiple Approaches to Assessment: reflections on use of tutor, peer and self-assessment. *Teaching in Higher Education*, **6**(2), pp. 229-246.

Fercheri1994 Fercheri, P. and Molfino, M. (1994). Software tools for the learning of programming: A proposal. *Computers Education*, **23**(4), pp. 269-276.

Fincher1999 Fincher, S. (1999). What are We Doing When We Teach Programming?. *Proceeding of Frontiers in Education*, November 1999, IEEE press, pp. 12a4-1 to 12a4-5.

Fiore2001 Fiore, M. and Chapman, O. (2001). *Calibrated Peer Review (CPR): web-based writing and peer review*. [Online]. (URL http://cpr.molsci.ucla.edu/).

Fowler1996 Fowler, B. (1996). *Critical Thinking Across the Curriculum Project: Bloom's Taxonomy and Critical Thinking*. [Online] (URL http://www.kcmetro.cc.mo.us/longview/ctac/blooms.htm).

Fox2003 Fox, S. and MacKeogh, K. (2003). Can eLearning Promote Higher-order Learning Without Tutor Overload? *Open Learning*, 18(2), pp. 121-134.

Funders2001 Funderstanding. (2001). *About Learning*. [Online] (URL http://www.funderstanding.com/about_learning.cfm).

Gagne1965 Gagne, R. (1965). *The Conditions of Learning*. London: Holt, Rinehart & Winston.

Gagne1979 Gagne, R. and Briggs, L. (1979). *Principles of Instructional Design.* 2nd ed. London: Holt, Rinehart & Winston.

Gibbs2002 Gibbs, G. (2002). *Qualitative Data Analysis: Explorations with NVivo*. Buckingham: Open University Press.

Good1990 Good, T. and Brophy, J. (1990). *Educational psychology: a realistic approach.* 4th ed. London: Longman.

Grauerholz2001 Grauerholz, L. (2001). Teaching Holistically to Achieve Deep Learning. *College Teaching*, **49**(2), pp. 44-50.

Hair1998 Hair, J., Anderson, R., Tatham, R., and Black, W. (1998). *Multivariate Data Analysis.* 5th ed. New Jersey: Prentice Hall.

Hartley1996 Hartley, J. (1996). Managing Models of Collaborative Learning. *Computer Education*, **26**(1-3), pp. 163-170

Heaney2004 Heaney, D. and Daly, C. (2004). Mass Production of Individual Feedback. *Proceedings of the 9th Annual SIGCSE Conference on Innovation and Technology in Computer Science Education: ITiCSE 2004*, UK: Leeds, June 28-30, 2004.

Henry1990 Henry, R., Whaley, K. and Forstall, B. (1990). The University of Washington illustrating compiler. *Proceeding of ACM SIGPLAN on programming language design and implementation*, New York: ACM Press, pp. 223-233.

Higgins2003 Higgins, C., Hegazy, T., Symeonidis, P. and Tsintsifas, A. (2003). The CourseMarker CBA System: Improvements over Ceilidh. *Journal of Education and Information Technologies*, 8(3), pp. 287-304.

Howitt2003 Howitt, D., and Cramer, D. (2003). *A Guide to Computing Statistics with SPSS 11 for Windows*. Revised edition. UK: Prentice Hall.

Huitt2004 Huitt, W. (2004) *Bloom et al.'s Taxonomy of the Cognitive Domain, Educational Psychology Interactive*. GA: Valdosta State University. [Online]. (URL http://chiron.valdosta.edu/whuitt/col/cogsys/bloom.html).

Imrie1995 Imrie, B. (1995). Assessment for Learning: quality and taxonomies. *Assessment & Evaluation in Higher Education*, 20(2), pp. 175-189.

Jarvis2003 Jarvis, P., Holford, J., and Griffin, C. (2003). *The theory and practice of learning*. 2nd ed. UK: Kogan Page Limited.

Jarvis2004 Jarvis, S. (2004). *CS118 Programming for Computer Scientists*. UK: Warwick University. [Online]. (URL http://www.dcs.warwick.ac.uk/undergraduate/modules/cs118.html).

Jenkins2001 Jenkins, T. (2001). The Motivation of Students of Programming. *SIGCSE Bulletin*, 33(3).

Jenkins2002 Jenkins, T. (2002). *On the Difficulty of Learning to Program*. [Online]. (URL http://www.ics.ltsn.ac.uk//pub/conf2002/jenkins.html).

Johnson1997 Johnson, S. D. (1997). Learning technological concepts and developing intellectual skills. *International Journal of Technology and Design Education*, 7, pp. 161-180.

Johnston1985 Johnston, H. (1985). *Learning to Program.* UK: Prentice-Hall International.

Jonassen2000 Jonassen, D., Peck, K., and Wilson, B. (2000). *Learning with technology: A constructivist approach.* NJ: Prentice Hall.

Joy1998 Joy, M. and Luck, M. (1998). The BOSS System for On-line Submission and Assessment. *Monitor: Journal of the CTI Centre for Computing,* **10**, pp. 27-29.

Kannusmaki2004 Kannusmaki, O., Moreno, A., Myller, N. and Sutinen, E. (2004). What a Novice Wants: Students Using Program Visualization in Distance Programming Course, *Proceedings of the Third Program Visualization Workshop (PVW'04)*, UK: Warwick, July 2004.

Kerr2002 Kerr, A., Hall, H., and Kozub, S. (2002). *Doing Statistics with SPSS.* London: SAGE Publications.

Kirkwood2000 Kirkwood, M. (2000). Infusing higher-order thinking and learning to learn into content instruction: a case study of secondary computing studies in Scotland. *Journal Curriculum Studies*, **32**(4), pp. 509-535.

Kiper1996 Kiper, J. Cross, V., Delisio, D. Sobel, A. and Troy, D. (1996). Perspectives on Assessment through Teaching Portfolios in Computer Science. *SIGCSE Bulletin*, **28**(1), pp. 200-203.

Knoy2001 Knoy, T., Lin, S., Liu, Z. and Yuan, S. (2001). Networked Peer Assessment in Writing: Copyediting Skills Instruction in an ESL Technical Writing Course. *International Conference on Computers in Education: ICCE 2001*, Korea: Seoul, November 11-14, 2001.

Kolling2001 Kolling, M. and Rosenberg, J. (2001). Guidelines for Teaching Object Orientation with Java. *SIGCSE Bulletin*, **33**(3), pp. 33-36.

Kreijns2003 Kreijns, K., Kirschner, P. and Jochems, W. (2003). Identifying the pitfalls for social interaction in computer-supported collaborative learning environments: a review of the research. *Computers in Human Behavior*, **19**, pp. 335-353.

Kruse2004 Kruse, K. (2004). *Gagne's Nine Events of Instruction: An Introduction.* [Online]. (URL http://www.e-learningguru.com/articles/art3_3.htm).

Lejk2002 Lejk, M. and Wyvill, M. (2002). Peer Assessment of Contributions to a Group Project: student attitudes to holistic and category-based approaches. *Assessment & Evaluation in Higher Education*, **27**(6), pp. 570-571.

Lin2001 Lin, S, Liu, E. and Yuan, S. (2001). Web-based peer assessment: feedback for students with various thinking-styles. *Journal of Computer Assisted Learning*, **17**, pp. 420-432.

Lin2002 Lin, S., Liu, E. and Yuan, S. (2002). A Pilot Study of Students' Attitudes Toward and Desired System Requirements of Networked Peer Assessment System. *International Conference on Computers in Education: ICCE 2002*, New Zealand: Auckland, 3-6 December 2002.

Lister2000 Lister, R. (2000). On Blooming First Year Programming, and its Blooming Assessment. *SIGCSE Bulletin*, pp. 158-162.

Magin2001 Magin, D. and Helmore, P. (2001). Peer and Teacher Assessments of Oral Presentation Skills: how reliable are they? *Studies in Higher Education*, **26**(3), pp. 287-298.

McGill1997 McGill, T. and Volet, S. (1997). A conceptual framework for analysing students' knowledge of programming. *Journal of Research on Computing in Education*, **29**(3), pp. 276-297.

Mergel1998 Mergel, B. (1998). Instructional Design & Learning Theory. [Online]. (URL http://www.usask.ca/education/coursework/802papers/mergel/brenda.htm).

Miller2002 Miller, R., Acton, C., Fullerton, D, and Maltby, J. (2002). *SPSS for Social Scientists*. New York: Palgrave Macmillan.

Miller2003 Miller, P. (2003). The Effect of Scoring Criteria Specificity on Peer and Self-assessment. *Assessment & Evaluation in Higher Education*, **28**(4), pp. 383-394.

Mukherjea1994 Mukherjea, S. and Stasko, J. (1994). Integrating algorithm animation capabilities within a source level debugger. *ACM Transactions on Computer-Human Interaction*, **1**(3), pp. 215-244.

Newcomb1987 Newcomb and Trefz. (1987). *Bloom's Levels of Cognition.* [Online]. (URL http://www.oersted.dtu.dk/personal/kah/31652/Learning/main.html).

NHMRC2001 NHMRC. (2001). Human Research Ethics Handbook: Students, Research Involving. [Online]. (URL http://www.nhmrc.gov.au/hrecbook/02_ethics/43.htm).

Notting1992 Nottingham University. (1992). *A Review of CAL Packages: Ceilidh.* [Online]. (URL http://www.doc.ic.ac.uk/~nd/surprise_95/journal/vol2/hst/article2.html).

Oliver2004 Oliver, D., Dobele, T., Greber, M. and Roberts, T. (2004). This Course Has A Bloom Rating Of 3.9. *In*: Lister, R. and Young, A. *Sixth Australasian Computing Education Conference (ACE2004)*, Dunedin, NZ, January 2004. Australian Computer Society, Inc., pp. 227-231.

OLTC1996 Open Learning Technology Corporation Limited (OLTC). (1996). *Learning with software: pedagogies and practice – Learning Theories.* [Online]. (URL http://www.educationau.edu.au/archives/CP/04.htm).

Orsmond1996 Orsmond, P. and Merry, S. (1996). The importance of marking criteria in the use of peer assessment. *Assessment & Evaluation in Higher Education*, **21**(3), pp. 239-250.

Orsmond2002 Orsmond, P., Merry, S. and Reiling, K. (2002). The use of Exemplars and Formative Feedback when Using Student Derived Marking Criteria in Peer and Self-assessment. *Assessment & Evaluation in Higher Education*, **27**(4), pp. 309-323.

Pallant2001 Pallant, J. (2001). *SPSS Survival Manual: a step by step guide to data analysis using SPSS for Windows (Version 10)*. USA: Open University Press.

PBS2005 Public Broadcasting Service: PBS. (2005). *A science odyssey: people and discoveries.* [Online]. (URL http://www.pbs.org/wgbh/aso/databank/entries/bhpavl.html).

Pea1983 Pea, R. and Kurland, D. (1983). *On the Cognitive Prerequisites of Learning Computer Programming*. New York: Bank Street College of Education.

Pea1984 Pea, R. and Kurland, D. (1984). *On the Cognitive Effects of Learning Computer Programming: A Critical Look*. New York: Bank Street College of Education.

Peden1994 Peden, B. (1994). Assessing Student's Critical Thinking Skills and Attitudes Toward Computer Science. *SIGCSE Bulletin*, pp. 263-267.

Peers1996 Peers, I. (1996). *Statistical Analysis for Education and Psychology Researchers*. London: Falmer Press.

Perry2002 Perry, D. (2002). *P540: Learning and Cognition in Education – Unit 6: Gagne's instructional design theory.* [Online]. (URL http://education.indiana.edu/~p540/webcourse/gagne.html)

Reek1996 Reek, K. (1996). A Software Infrastructure to Support Introductory Computer Science Courses. *SIGCSE Bulletin*, **2**, pp. 125-129.

Rees1989 Rees, D.G. (1989). *Essential statistics*. 2nd ed. UK: Chapman & Hall.

Robins2003 Robins, A., Rountree, J. and Rountree, N. (2003). Learning and Teaching Programming: A Review and Discussiong. *Computer Science Education*, **13**(2), pp. 137-172.

Robling2004 Robling, G., Trompler, C., Muhlhauser, M., Kobler, S. and Wolf, S. (2004). Enhancing Classroom Lectures with Digital Sliding Blackboards. *Proceedings of the 9th Annual SIGCSE Conference on Innovation and Technology in*

Computer Science Education: ITiCSE 2004, UK: Leeds, June 28-30, 2004.

Rosie2000 Rosie, A. (2000). Deep Learning: A dialectical approach drawing on tutor-led. *Learning in Higher Education*, **1**(1), pp. 45-59.

Rowe1999 Rowe, G. and Gregor, P. (1999). A computer based learning system for teaching computing: implementation and evaluation. *Computers & Education*, **33**, pp. 65-76.

Sadiq2004 Sadiq, S., Orlowska, M., Sadiq, W. and Lin, J. (2004). SQLator – An Online SQL Learning Workbench. *Proceedings of the 9th Annual SIGCSE Conference on Innovation and Technology in Computer Science Education: ITiCSE 2004*, UK: Leeds, June 28-30, 2004.

Sandberg1997 Sandberg, J. and Barnard, Y. (1997). Deep learning is difficult. *Instructional Science*, **25**(1), pp. 15-36.

Sapsford1996 Sapsford, R. and Jupp, V. (1996). *Data Collection and Analysis*. London: SAGE Publications.

Schorsch1995 Schorsch, T. (1995). CAP: an Automated Peer Assessment tool to check Pascal Programs for Syntax, Logic and Style errors. *Twenty-sixth SIGCSE technical symposium on Computer science education*, Tennessee: Nashville, March 1995.

Scott2003 Scott, T. (2003). Bloom's Taxonomy Applied to Testing in Computer Science Classes. *JCSC*, **19**(1), pp. 267-274.

Seddon1978 Seddon, G. (1978). The properties of Bloom's taxonomy of educational objectives for the cognitive domain. *Review of Educational Research*, **48**(2), pp. 303-323.

Segers2001 Segers, M. and Dochy, F. (2001). New Assessment Forms in Problem-based Learning: the value-added of the students' perspective. *Studies in Higher Education*, **26**(3), pp.327-343.

Simmonds2003 Simmonds, R. (2003). CS122 Professional Aspects of Computing. [Online]. (URL http://www.dcs.warwick.ac.uk/undergraduate/modules/cs122.html).

Sivan2000 Sivan, A. (2000). The Implementation of Peer Assessment: an action research approach. *Assessment in Education*, **7**(2), pp.193-213.

Sluijsmans1999 Sluijsmans, D., Dochy, F., and Moerkerke, G. (1999). Creating a learning environment by using self- peer- and co-assessment. *Learning Environments Research*, **1**, pp. 293-319.

Sluijsmans2002 Sluijsmans, D., Brand-Gruwel, S. and Merrienboer, J. (2002). Peer Assessment Training in Teacher Education: effects on performance and perceptions. *Assessment & Evaluation in Higher Education*, **27**(5), pp. 443-454.

Somervell1993 Somervell, H. (1993) Issues in assessment, enterprise and higher education: the case for self-, peer and collaborative assessment. *Assessment and Evaluation in Higher Education*, **18**, pp. 221-233

Smith2002 Smith, H., Cooper, A. and Lancaster, L. (2002). Improving the Quality of Undergraduate Peer Assessment: A Case for Student and Staff Development. *Innovations in Education and Teaching International*, **39**(1), pp. 71-81.

Smith2003 Smith, M. (2003). *Learning Theory*. [Online]. (URL http://www.infed.org/biblio/b-learn.htm).

Somervell1993 Somervell, H. (1993). Issues in assessment, enterprise and higher education: the case for self-, peer and collaborative assessment. *Assessment and Evaluation in Higher Education*, **18**, pp. 221-233.

TEDI2004 Teaching and Educational Development Institute (TEDI), The University of Queensland, Australia. *Course mapping – Learning activities grids (grid 4 – Critical judgement).* [Online]. (URL http://www.tedi.uq.edu.au/teaching/GradAttributes/grids/pm-grid4.html).

TEP2001 Teaching Effectiveness Program (TEP) – Academic Learning Services, University of Oregon, Georgeanne Cooper, USA. *Writing Multiple-Choice Questions that Demand Critical Thinking.* [Online]. (URL http://darkwing.uoregon.edu/%7Etep/assessment/mc4critthink.html).

TIP2004 Kearsley, G. (2004). *Theory Into Practise (TIP) database.* [Online]. (URL http://tip.psychology.org/).

Topping1998 Topping, K. (1998). Peer assessment between students in colleges and universities. *Review of Educational Research,* **68**, pp. 249-276.

Topping2000 Topping, K, Smith, E., Swanson, I. And Elliot, A. (2000). Formative Peer Assessment of Academic Writing Between Postgraduate Students. *Assessment & Evaluation in Higher Education*, **25**(2), pp. 149-169.

Tsai2002 Tsai, C., Lin, S., and Yuan, S. (2002). Developing science activities through a net worked peer assessment system. *Computers & Education*, **38**, pp. 241-252.

Vaus2002 Vaus, D. (2002). *Analyzing Social Science Data: 50 Key problems in Data Analysis*. London: SAGE Publications.

Venables2003 Venables, A. and Haywood, L. (2003). Programming students NEED instant feedback! *5th Australasian Computing Education Conference (ACE2003)*, Australia: Adelaide, Australian Computer Society Inc.

Verdejo1993 Verdejo, M., Fernandez, I. And Urretavizcaya, M. (1993). Methodology and design issues in Capra, an environment for learning program construction. In G. Dettori, B. du Boulay and E. Lemut, eds. *Cognitive Models and Intelligent Environments for Learning Programming,* Berlin: Springer-Verlag, pp. 156-171.

Vygotsky1962 Vygotsky, L. (1962), *Thought and language*. Cambridge, MA: MIT press.

Vygotsky1978 Vygotsky, L. (1978). *Mind in Society: The development of higher psychological processes*. London: Harvard University Press.

Wen2003 Wen, M., Tsai, C., & Chang, C. (2003). Attitudes toward Peer Assessment: Perspectives from College Students and Inservice Teachers. *In:* Chee, Y., Law, N., Lee, K. and Suthers, D., eds. *The "Second Wave" of ICT in Education: from Facilitating Teaching and Learning to Engendering Education Reform*, Proceedings of the International Conference on Computers in Education 2003: ICCE2003, Hong Kong, pp. 181-184.

Yue2004 Yue, K. and Ding, W. (2004). Design and Evolution of An Undergraduate Course on Web Application Development. *Proceedings of the 9th Annual SIGCSE Conference on Innovation and Technology in Computer Science Education: ITiCSE 2004*, UK: Leeds, June 28-30, 2004.

Zilles2005 Zilles, C. (2005). SPIMbot: An Engaging, Problem-based Approach to Teaching Assembly Language Programming. *ACM SIGCSE bulletin*, Missouri: St. Louis, February 23-27, 2005, pp. 106-110.

Chapter 10 Appendix

10.1 Peer assessment for programming

10.1.1 Test I and test II

Test I

CS120 Programming Laboratory University No _____

```
        # list the vital statistics (in the manner of ls -l) of each directory
        # between the root and the current directory
1       A=$( pwd )
2       until [ $( pwd ) = "/" ]
3       do
4               R=$( pwd ); cd ..
5               ls –d –l $R >>$A/del
6       done
7       cd /
8       ls –d –l / >>$A/del
9
10      sort –k 9,9 $A/del
11      rm $A/del
```

Imagine you are a marker providing some feedback to the student who wrote the code above. Write a short paragraph in the space below that will help the student improve their programming. Continue overleaf if necessary.

For example, if the program was badly indented so it was hard to read, you might comment that "additional, consistent indentation would make this program easier to read".

Test II

CS120 Programming Laboratory University No _____

```
# list the vital statistics (in the manner of ls -l) of each directory
# between the root and the current directory
1       ls -d -l / >>$HOME/trashf
2       while [ "$( pwd )" != / ]
3        do
4               X=$( pwd ); cd ..
5               ls -d -l $X >>$HOME/trashf
6        done
7       cd /
8
9       sort -k 9,9 $HOME/trashf
10      rm –f $HOME/trashf
```

Imagine you are a marker providing some feedback to the student who wrote the code above. Write a short paragraph in the space below that will help the student improve their programming. Continue overleaf if necessary.

For example, if the program was badly indented so it was hard to read, you might comment that "additional, consistent indentation would make this program easier to read".

10.1.2 Marking Criteria for peer assessment for programming

Pilot study

Readability			
- Are there appropriate comments?	ONo	OPartial	OYes
- Is the code indented helpfully and consistently?	ONo	OPartial	OYes
- Do the variable names make it clear what they are used for?	ONo	OPartial	OYes
Correctness			
- Does the code give the correct output?	ONo	OPartial	OYes
- Does the code handle errors appropriately?	ONo	OPartial	OYes
- Does the program finish with the correct exit status?	ONo	OPartial	OYes
Style			
- Have appropriate utilities been selected, so as to simplify the code?	ONo	OPartial	OYes
- Is the program well structured?	ONo	OPartial	OYes
- Is the program written so it is easy to follow what it is doing?	ONo	OPartial	OYes

Main study

Readability			
- The number of comments is	low	OOOOO	high
- Comments are	unhelpful	OOOOO	helpful
- The code are indented	inconsistently	OOOOO	consistently
- Identifier names are	inappropriate	OOOOO	appropriate
Correctness			
- The program meets the specification	not at all	OOOOO	completely
- The code handles errors	inappropriately	OOOOO	appropriately
- The program finishes with an appropriate exit status	never	OOOOO	always
Style			
- The utilities have been selected	inappropriately	OOOOO	appropriately
- The program is structured	poorly	OOOOO	well
- Overall, following what the program is doing is	hard	OOOOO	easy

10.1.3 Difference between OASYS and our web-based peer assessment

OASYS was deployed in the first year Design of Information Structures module in the Computer Science Department in an attempt to give students effective and timely feedback on their progress in laboratory sessions. Students took 30 minutes of on-line testing, run under exam conditions. The on-line tests assessed the students' understanding of their earlier work during the lab session, and were comprised of multiple choice (MCQ) and free response questions. In the latter type of question, students were typically asked to write a few lines of Java code or a few English sentences. The answers to the free response questions were then peer assessed.

In OASYS, most peer assessment was not examined in detail by the tutors, and there was a high focus on the exact marks given and not the discursive feedback. In our web-based peer assessment, we have shifted from an emphasis on *marks* in OASYS to an emphasis on *useful feedback*. The arrangement of assessors into groups is a significant change. Assessors make *comparisons* between submissions and are encouraged to reflect on their decisions through discussion with their peers, instead of making judgements in isolation (in OASYS). The changes we made are summarised below.

OASYS	**Our web-based peer assessment system**
Aims at giving students effective and timely feedback on their progress in laboratory sessions	Aims at enhancing students' higher cognitive skills in programming
OASYS gave short tests MCQ and Java exercises (small tests)	Tutor set a sizeable shell programming assignment
Students created *short scripts* in lab session	*Online submission* of programming assignment.
No group discussion, individual marking	Group discussion and group marking
No control for ranges of ability	All students observe a range of ability
Automatic test results not shown to assessors	Automatic test results shown to assessors

OASYS	Our web-based peer assessment system
Emphasis on *marks*	Emphasis on *useful feedback*
Assessment judgements made in isolation	Groups make comparisons between submissions
Tutors monitored the quality of marking	Peers mark the quality of marking

10.1.4 Questionnaires

You have recently completed an assignment involving peer review. We would welcome your feedback on this experience and encourage you to share any comments you may have with us.

"Data Protection Act 1998"

This questionnaire will not be used for assessment of your assignment. It will be used for monitoring and evaluation of the peer assessment exercise only.

Please answer the following questions.

Questionnaire after assignment 1

1. What is your gender?

   ○ Male ○ Female

2. Is English your first language?

   ○ Yes ○ No

3. Have you had any experience of programming before studying it at Warwick university? If the answer is *Yes* please indicate how much experience you have and which languages you know.

   ○ Yes _____

   ○ No

4. How long did you take to complete assignment1?

   ○ More than 10 hours ○ 5-10 hours ○ Less than 5 hours

5. Did you fully understand what was required in assignment1?

   Unclear ○ ○ ○ ○ Fully understood

6. Here are a number of statements concerning your approach to study in general. Please select the appropriate response.

	Never---Always
a. I find it easy to organize my study time effectively.	O O O O
b. It is important for me to do very well in every module.	O O O O
c. I would like to learn only the subjects which really interests me.	O O O O
d. I often find myself questioning things that I hear in lectures or read in books.	O O O O
e. When I don't understand the lecture or what I have read, I always find out more to ensure that I understand it.	O O O O
f. I find that I have to concentrate on memorizing what I have learned.	O O O O
g. I enjoy solving complex programming problems.	O O O O
h. I enjoy discussing interesting topics with other people.	O O O O
i. I tend to spend little effort beyond what's required for completing assignments.	O O O O
j. I spend my spare time in finding out more about interesting topics which have been discussed in class.	O O O O
k. I regularly spend my spare time reviewing what I have studied.	O O O O

l. In classes I have taken

O I have usually got to know many of the students.

O I have rarely got to know many of the students.

m. In a study group working on difficult material, I am more likely to

O jump in and contribute ideas O sit back and listen

n. When I start an assignment problem, I am more likely to

O start working on the solution immediately.

O try to fully understand the problem first.

o. When I am learning a new subject, I prefer to

O stay focused on that subject, learning as much about it as I can.

O try to make connections between that subject and related subjects.

7. Is Web based peer review easy to use? Please explain your answer briefly.

O Yes O No

8. Did seeing different ways of solving programming problems help you write better programs (Design, Coding, Testing, Maintenance)? Please explain your answer briefly.

O Yes O No

9. Any other comments.

Questionnaire after assignment 2

1. Did you fully understand what was required in the assignment?

   Unclear O O O O Fully understood

2. How long did you take to complete assignment2?

   O More than 10 hours O 5-10 hours O Less than 5 hours

3. Did you fully understand the marking criteria in the exercise?

   Unclear O O O O Fully understood

4. Would you have preferred to have been involved in drawing up the marking criteria? If the answer is *Yes* please indicate what changes you would make.

   OYes

   O No

5. Did the 'Things to consider' help you in marking and make you understand more about the assignment? If the answer is *No* please indicate why.

   OYes

   ONo

6. Did you discuss with your group when marking assignments? If the answer is *Yes* please summarized briefly what you discussed about. If the answer is *No* please indicate why you did not discuss with your group.

O Yes O No

7. Did you discuss with the script authors when marking assignments? If the answer is *Yes* please summarized briefly what you discussed about. If the answer is *No* please indicate why you did not discuss with the script authors.

O Yes O No

8. Did you find having a discussion with the script author and group about the assignment helpful? Why?

O Yes O No

9. How long did you spend marking other students' work (Step I) for 3 scripts?

O More than 1 hour O 30-60 minutes O Less than 30 minutes

10. Would you like to mark more than 3 scripts? Why?

O Yes O No

11. Did you feel comfortable when assigning marks? If the answer is *No* please indicate what you did not feel comfortable with.

OYes

ONo_____

12. Did you realise mistakes that you made in your own answer when marking other students' work? If yes, what particular errors were they?

    OYes_____

    O No

13. Are you satisfied with your mark from the peer review? If the answer is *No* please indicate why you are not satisfied with it.

    OYes

    ONo_____

14. Were the comments from your peers useful? Why?

    O Yes O No

    _____

    _____

    _____

15. Which of the following statements about peer review and technical skills do you agree with? (tick as many as apply)

    ☐ Peer review forces me to think about what constitutes a good or poor piece of work.

    ☐ Analysing the work of peers leads to an improved awareness of the quality of my own work.

    ☐ Peer review helps me to think about the specification of a program more deeply.

    ☐ Peer review helps me to improve my programming style.

    ☐ Peer review is deepening my understanding of what is required in good programming.

    ☐ Peer review is providing me with a better understanding of what is required to achieve a particular standard and what tutors are looking for when conducting assessment.

    ☐ Peer review encourages me to consider the objectives and purpose of the assessment task as well as the course itself.

    ☐ Peer review highlights the importance of presenting work in a clear and logical format.

16. Which of following transferable skills peer review help you to improve? (tick as many as apply)

- ☐ Peer review enables me to view and critique a range of programming styles, techniques, ideas and abilities, thus encouraging me to learn from both the mistakes and exemplary performances of peers.
- ☐ Peer review helps me to criticise my own work.
- ☐ Analysis and evaluation of other students' work helps me to improve my ability in learning programming.
- ☐ Peer review helps me to develop the ability to judge the performance of peers.
- ☐ Reading code skill and understanding of other students' work can be developed in peer review.
- ☐ Peer review helps me to develop team working skills by discussing ideas and concepts with peers.
- ☐ Peer review enhances independent study.
- ☐ Peer review helps me to be enthusiastic in learning computer programming.
- ☐ I have more confidence in my ability to assign marks when participating in the assessment next time.

17. Any other comments.

_____

_____

_____

Questionnaire after Assignment 3

1. How many lectures out of 10 have you attended?

   O More than 8 lectures O 4-8 lectures O Less than 4 lectures

2. Did you fully understand what was required in the assignment?

   Unclear O O O O Fully understood

3. Has this been your first experience of peer review? If the answer is *No* please indicate your previous experience of peer review.

   O Yes

   O No _____

4. Do you fully understand what peer review is about?

   Unclear O O O O Fully understood

5. Did you take the peer review exercise seriously? Why?

O Yes O No

6. Do you think it is a good idea that the peer review exercise is anonymous? Why?

O Yes O No

7. Did you compare your work with your peers? What are the good points that you can use to improve your programming style?

O Yes

O No

8. Were the comments from your peers of previous assignment useful for doing this assignment? Why? (please indicate what are the particular points which is useful)

O Yes O No

9. In your opinion, what are the benefits of peer review in learning programming?

10. In your opinion, what are the problems with peer review process?

11. Here are a number of statements concerning peer review. Please select the appropriate response.

	Disagree	Strongly agree
a) Seeing good and bad programs helps me in learning programming.	O	O O O O O
b) Marking helps me to think more deeply about my own work.	O	O O O O O
c) Seeing different ways of solving programming problems helps me create better programs.	O	O O O O O
d) Using specific marking criteria helps me to understand what is expected of a good program.	O	O O O O O
e) Marking helps me to evaluate my own work.	O	O O O O O
f) Giving feedback helps me to reflect on my own ideas.	O	O O O O O
g) It is important that I should take part in peer review as part of my studies.	O	O O O O O
h) The peer feedback I received was relevant and useful.	O	O O O O O
i) Marking in groups stimulates discussion about the assignment.	O	O O O O O
j) With adequate guidance, the marks from peers can be as reliable as the mark from a tutor.	O	O O O O O

12. Do you think that the peer review exercise helped you to evaluate the quality of the program you wrote? Why?

13. Do you think that peer review helps you learn programming more than traditional assessment? Why?

14. Can you think of other ways of using peer review in a computer programming course?

15. Do you think you the peer review web pages could be improved? What changes would you make?

O Yes _____

O No

16. Would you recommend peer review to your friends as a way of learning more?

    O Yes O No

17. Please give us any other suggestions you may have to improve the peer review process.

_____

_____

18. We would value the opportunity to speak to you about the peer review exercise. This would take about 30 minutes, and would be at a mutually convenient time. May we contact you to arrange an interview?

    O Yes O No

10.1.5 Interview questions

Learning programming

1. Could you please tell me your strategy in learning programming?

_____

_____

2. What is your goal from learning programming?

   Or What do you want to get out of learning programming?

_____

_____

3. Skills for learning programming

   a. pre: In your opinion, what are the necessary skills for learning programming?

_____

_____

   b. post: what are the skills that you develop from learning programming?

_____

_____

4. What help or facilities do you require to support learning programming (i.e. surgery, tutor, exercise)?

Peer review exercise

Tool

5. Do you think it is a good idea that the system provides the anonymous communication device? Why? *YES* *NO*

6. Do you prefer anonymous or non-anonymous discussion? Why?

7. Is anonymous communication device easy to use? *YES* *NO*
   Did you use it? Why not? *YES* *NO*

8. Please give me any suggestions you may have to improve the anonymous communication device in order to encourage discussion.

Process & Results

9. Which of these help you in understanding and marking the programs?
   ___ Discussion by using anonymous communication device
   ___ Automatic test result
   ___ Things to consider
   ___ Others (please specify)

10. Did you feel comfortable when assigning marks starting from assignment1 to assignment3?

11. How carefully did you read the script before marking?
    a. What are the good points that you can pick from those scripts to improve your programming style?

12. Did the peer review exercise **start you thinking** about what was wrong and what worked well with (the **good and bad points** of) your own program? Why? (evaluate your own work) *YES* *NO*

13. Did seeing different ways of solving programming problems help **you write better programs?** *YES* *NO*

    a. What change have you made in each assignments?

14. Do you agree with the comments that you receive from peers? *YES* *NO*
    Were those comments useful? Please give me an example.

15. Correctness & quality of program (i.e. readability, style, efficiency)

    a. How do you weight of correctness and quality of program?

    b. How does peer review relate to correctness and to quality of program?

16. STEP2 – mark quality of feedback

    a. In your opinion, what are the benefits of step2 – mark quality of feedback?

    b. In your opinion, what are the problems of step2 – mark quality of feedback?

17. Do you think that the peer review exercise helped you to understand more about the assignment? Why? *YES* *NO*

Other

18. Is it a good idea using peer review to help in learning programming?

19. Have you heard of other assessment tools or other systems, which help in learning programming?

20. Please give me any suggestions you may have to improve the peer review system.

21. Any other comments?

10.2 Peer assessment for essays

10.2.1 Questionnaire

1. Is English your first language?

   O Yes O No

2. Is your English fluent?

   O Yes O No

3. Did you fully understand an essay topic?

   Unclear O O O O O Fully understood

4. Did you fully understand the marking criteria in the exercise?

   Unclear O O O O O Fully understood

5. Did you discuss with your group when analysing essays? If the answer is *Yes* please summarize briefly what you discussed. If the answer is *No* please indicate why you did not discuss with your group.

   O Yes O No

   _____

   _____

6. Did you find having a discussion with group about the essay helpful? Why?

   O Yes O No

   _____

   _____

7. Do you prefer anonymous or face to face discussion? Why?

   O anonymous discussion O face to face discussion

   _____

   _____

8. Did seeing different ways of writing an essay help you write better essays? Please explain your answer briefly.

   O Yes O No

   _____

   _____

9. Do you think that the peer review exercise helped you to evaluate the quality of the essay you wrote? Why?

10. Were the comments from your peers useful? Why?

O Yes O No

11. In your opinion, what are the benefits of peer review for essays?

12. In your opinion, what are the problems with this peer review process?

13. Is Web based peer review easy to use? Please explain your answer briefly.

O Yes O No

14. In your opinion, what are the different skills that you require to write a good program and essay in computer science context?

15. Did your experience in peer review in CS120 help you to analyse essays and give comment easier?

16. What do you like and dislike about this process compared with peer review in CS120?

17. What are the differences between using Web-based peer review in learning computer programming and writing essays?

18. Please give us any other suggestions you may have to improve the peer review process.

_____

_____

19. We would value the opportunity to speak to you about the peer review for essay exercise. This would take about 30 minutes, and would be at a mutually convenient time. May we contact you to arrange an interview?

O Yes O No

10.2.2 Interview questions

1. What are the important skills that are required in writing essays?

_____

_____

_____

2. Did analysing other students' work help you to improve your essay? Why?

_____

_____

_____

3. Do you think you give more effective comments by associating them with specific lines of the essay?

_____

_____

4. When you analyse the essay, which parts do you find are .....? How do they compare with your work?

   a. very interesting, good ideas ?

_____

_____

_____

b. weak point, disadvantages?

5. What are your expectations from this peer review? How much did you got out of it?

6. What are your general opinions about these peer review exercises (both in programming and essays)?