Implications of an integrated curriculum in a polytechnic or competence based environment

Bahajjaj, Ayusni

Implications of an Integrated Curriculum in a Polytechnic or Competence based Environment

A thesis submitted in partial fulfillment for the degree of Doctor of Education

School of Education
Durham University

by
Ayusni Bahajjaj
2007

Abstract

This is a comparative case study of an innovative approach to teaching computer programming to novices. The focus of this study is to evaluate the integrated curriculum which blends face-to-face interaction with computing practice and online learning to first-year polytechnic students in an engineering informatics diploma course. To examine the efficacy of the blended learning approach, the integrated curriculum is compared to its predecessor which has applied the traditional structured curriculum.

This thesis gains relevance from its study of different dimensions of the curriculum comprising the curriculum aims and objectives, the teaching-learning activities and the different forms of assessment. The research design is mainly qualitative employing analytic induction methods to arrive at its inferences and findings. Content analysis and observation have been performed to evaluate the curriculum of each of the cases. A quantitative analysis is performed on students' performance in the computer programming module to add validity to the qualitative findings. Data were collected for students taking the Principles of Computing module in the first semester of the first year in 2005 and 2006 respectively; a total of 232 students came from the 2005 cohort and 247 students came from the 2006 cohort. The dependent variables are the module score and its sub-components, the project score and the individual test score. The independent variable significant to this study is the student's entry level GCE 'O' levels aggregate; gender is not a significant variable unlike in other studies involving mathematics or science.

The findings highlight the differences that exist between a traditional structured learning environment to the blended learning environment and how students perform under the different learning environments. A major contribution of this study is the constructive alignment framework incorporating the integrated curriculum characteristics to support the blended learning approach. By reviewing the curriculum, the teaching methods, the assessment procedures and the learning environment with regard to the integrated curriculum characteristics, this study has made significant discoveries on the strengths and limitations of the blended learning approach.

The results of this study show how the roles of curriculum, pedagogy and assessment are inter-related and have to be integrated into the curriculum to foster better learning for students. Finally, the findings reveal the importance of the influence of the tutor in the blended learning delivery and the students' preference for tutor interaction. Through these findings, the study is able to recommend future improvements to the Principles of Computing module.

Keywords: integrated curriculum, computer programming, blended learning, online learning, constructive alignment, assessment, teaching-learning activities.

Declaration

The contents of this thesis are the result of my own work except where due reference is made, and have not been previously submitted for a degree in Durham University or any other university.

Statement of Copyright

Acknowledgements

Learning starts with life:
the learning journey has no end:
when one cycle ends, a new cycle begins.

It is this fascination with learning that has spurred me to take up the EdD course with Durham. My brief yet meaningful exchanges with the EdD staff i.e. Mrs Anji Rae, Prof Bill Williamson, Prof Carl Bagley, Dr Julie Rattray, Prof Barry Cooper, Dr Anwei Feng, Prof Mike Byram and Prof David Galloway have made this journey worthwhile. I hope I may emulate your commitment to the quality of learning and research.

I am glad to have made the difficult and expensive step to write up my thesis full-time in Durham despite the wet and cold weather. It was heartening to meet other research students and staff, and to visit the excellent libraries in Durham. Not least, I wish to thank Mses. Jane Watkinson and Anita Shepherd for their excellent support and service throughout the length of my course and throughout my stay in Durham.

It is with deep gratitude that I express my heartfelt appreciation to the following persons without whom this research would not have culminated in its successful completion.

Prof. Steve Higgins as my supervisor and mentor has been invaluable to the development of this research. His encouragements and guidance has given me the vision to see through all the challenges, trials and tribulations of this research.

Daniel Liu, deputy director and department head for the Engineering Informatics diploma in Nanyang Polytechnic, Singapore, whose unmitigated support has been crucial to the basis of this research. Not forgetting dear colleagues and comrades: Jessica, Elaine, Veronica and Rajani whose kind assistance has given me the courage to do this project in the first place.

Friends i.e. Elly whose cooking has brought the taste of home to Durham and kept my sanity; Liz, Wan, Soo Yin, Leen, Diana, Thanu and Nilu – your constant encouragements have given me the will to succeed.

Anwar, my all-seasons partner, for allowing me to take sabbatical leave from family service; Ahmed, Zahrah, Hamud and Kadir for taking good care of yourselves. Your presence and prayers have given me the faith to believe throughout our time together – past, present and future.

THANK YOU.

May you all find peace and happiness in your future endeavours.

Contents

Figures

Tables

Chapter 1 Introduction

As long as teaching instruction is dependent on curriculum, an equitable curriculum represents an opportunistic plan as to how to approach the education of students. This initial chapter gives an overview of this study beginning with *section 1.1 Nature of the Study* and how it all started in *section 1.2 Background of the Study*; various areas of research fields related to this study are covered in *section 1.3 Research Interest*; and the *significance of this study* in section 1.4. Subsequently, the *research questions* are put forward to support the aims and objectives of this study (1.5); the *terms* used in this report (1.6); its *assumptions and limitations* (1.7); and finally an *outline* of the remaining chapters (1.8).

1.1 Nature of the study

The motivation for this study arises out of the ever increasing focus on key competences that are inter-disciplinary in nature and affecting many facets of learning and teaching. The growing body of literature on teaching and learning competence indicates a need for integrated measures that are able to fulfill multi-faceted needs. The emphases on learning technologies in general and education technology as a whole have become the new requirements for success in teaching and learning. New trends and developments are finding ways to incorporate their methods and applications in the classroom faster than educators and learners are able to utilise them.

Thinking Schools, Learning Nation

In 1997, Singapore launched its Master Plan for Information Technology in Education (MPITE) as a blueprint for the integration of information technology (IT) in its education system to meet the economic needs of the new millennium (MOE, 2007a). The main objective was to use IT to equip students from young with learning skills, creative thinking skills and communication skills. This was a key strategy for producing a workforce of excellence for the future. By the year 2002, all schools under the purview of the Ministry of Education (MOE) are IT-enabled and 30% of the curriculum time is spent in IT-related activities such as electronic learning (e-learning), surfing the internet for information or desktop publishing. The second master plan, running from 2003 to 2008, is focused on the interactions of curriculum, assessment, instruction, teacher development, student learning and school culture to form a systemic and holistic environment. The aim of the second master plan is to leverage on information and communication technology (ICT) to propel students and staff towards the overall vision of Thinking Schools, Learning Nation.

Students who have completed the Singapore-Cambridge GCE 'O' level certificate, have several options to further their studies (MOE, 2007b): junior colleges which offer the Singapore-Cambridge GCE 'A' level certificate require a maximum of 18 aggregate points of 5 'O' level credits; polytechnics which offer a professional diploma require a maximum of 26 aggregate points of 5 'O' level credits and the institute of technical education (ITE) which offer a professional certificate require a minimum of 3 'O' level credits. Graduates from the ITE with sufficient grade-point average may apply to the polytechnics to earn a diploma.

Polytechnics in Singapore are post-secondary tertiary institutions providing skills training to support the technological and economic development of the nation (MOE, 2007c). Polytechnic graduates serve the middle-level professionals in the workplace to boost Singapore's competitive edge in a knowledge-based economy. Deemed as statutory boards reporting to the Higher Education division of the MOE, polytechnics also serve the needs of continuing education and post-employment professional development. In the polytechnics, IT education is brought another step further where students learn to apply IT for enhanced learning and problem solving. For students who take up the diploma in IT, they have to take up core modules in computer programming.

IT is the enabling factor for all government agencies in Singapore (IDA, 2007a) and industries and companies operating in Singapore require IT connectivity. Based on the 2005 IT manpower survey in Singapore (IDA, 2007b), computer programming and software design remains one of the top three job categories in Singapore, where software development remains the highest skill with the greatest shortage. Education institutions especially polytechnics, have the added responsibility to train IT graduates with solid programming skills to fill the gap within the IT industry. However, as the saying goes one can only bring the horse to the water, the total number of IT graduates in software development remains at less than 40%. This phenomenon seems to occur in other IT education institutions as observed by McGill (2003) where IT graduates are turned off by the perceived complexities of computer programming.

There is apparently a problematic link between the culture of academic life and student culture (Cunningham et al., 2003). The creation of a learning opportunity does not mean that it will be grasped; the setting of course objectives does not mean that they will be achieved. The capacity of students collectively to undermine the best intentions of national and institutional policies is insufficiently recognized. Governments may want more engineers and accordingly create more student places but students nonetheless may fail to enrol, or to seek jobs with different employers and in different labour

markets or sectors. The linkage between macro policies and micro practices breaks down with the autonomy and unpredictability of student behaviour.

In Singapore, the national policy of excellence has caused a brain drain which in turn brings about the foreign talent policy (Chan, 2002). It is not enough that educators turn out quality students but the government has to attract quality brains (talent) from overseas. Since Singapore lacks a domestic market, the driving message is to establish Singapore as a business and investment hub where technology particularly IT enables Singapore to become a global hub for regional companies and a regional hub for global companies. Although there is a plethora of reasons why IT graduates do not pursue computer programming jobs, improving the learning of computer programming remains an important responsibility of IT schools. This study seeks to evaluate the curriculum for first-time computer programmers or novice programmers so as to examine the effectiveness of developing the learning skills for computer programming.

1.2 Background of the study

Nanyang Polytechnic is inaugurated in 1992 (Chiang, 1998) as the fourth polytechnic in Singapore. However, its history can be traced back to the specialized training institutes set up by the Economic Development Board of Singapore with foreign governments: Japan-Singapore Institute that specialised in precision engineering, German-Singapore Institute that specialised in manufacturing engineering and the French-Singapore Institute that specialised in electronics and communications technologies. The staff from these institutes provide the expertise and groundwork for the faculties or schools in the polytechnic.

The school of IT in Nanyang Polytechnic (NYP, 2007a) runs five different IT diplomas, each with a specialised area of interest. The common aim among the IT diplomas is to develop competent computer programming and analytical skills in the students so that they can be effective IT professionals. In the first year of the 3-year diploma, the fundamentals of computer programming are taught along with other foundation modules. Each year comprises two semesters where students take six core modules and one complementary module in each semester.

The IT diploma for Engineering Informatics (EI) is established in 1997 as a multi-disciplinary diploma incorporating IT, engineering and business modules (NYP, 2007a). The dynamic nature of technological changes in IT and engineering has meant several curriculum changes in the last nine years for the EI diploma. For the first five years, curriculum changes were made on the content and domain specific areas. As the EI diploma matures, thereafter from 2003, the curriculum changes are focused on course

delivery and student learning. This is in line with the MOE's IT plan for education to foster more student-centred learning through the use of education technology.

The curriculum development cycle in EI and the school of IT is compliant with the quality standards set by the polytechnic. As shown in figure 1.1, the course management committee identifies curriculum changes based on ministry, industry and technological updates. Once these changes are approved by the senate or board of directors, the course manager assigns the module convener and the module supervisor to develop the curriculum aims, objectives and module syllabus under the course design and development process. Subsequently, the course management committee approves the module syllabus; thereafter the module convener designs and develops the module's study materials and teaching-learning activities. Within the course delivery process, the module convener sets up the resources required. At the end of the study semester, about four months after the start of the semester, the module convener and the module supervisor perform a module review based on student and tutor feedback and other technical updates. Any new updates to the module are subject to approval of the course management committee. Each module is audited annually by the internal audit team in the school of IT to ensure that student records are correctly maintained and materials are updated in a timely manner. The student assessment process verifies students' performance and progress and the staff loading and appraisal process assigns and validates staffs' performance and progress against the targets set by the course management committee.

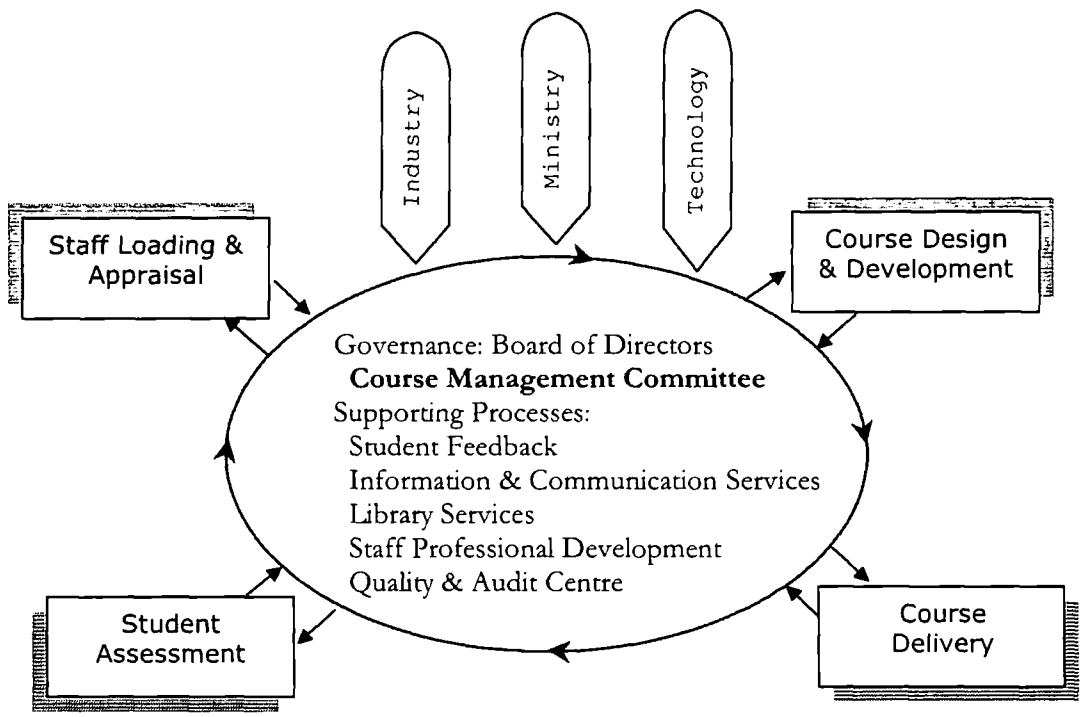


Figure 1.1 The curriculum continuous improvement cycle in NYP

Supporting processes ensure the four main processes are integrated and running smoothly: Student feedback are stored in databases and are processed through software application; network facilities and application systems are maintained by the information and communication systems; reference texts and study or discussion facilities are available through library services; staff training are handled by the staff professional development system; and the quality and audit centre ensures that quality standards and all processes are in place and up-to-date.

There are twelve core modules in the first year and these modules are equally spread across each semester in a year (see Appendix A for course structure and related modules). One of the core modules in the first-year EI diploma is the Principles of Computing (PrC) module which covers computer programming topics for first-time programmers. The PrC module is a single track module in that it is offered to all first year EI students in the first semester. Being IT oriented, electronic learning (e-learning) is highly emphasised in the course delivery of the EI diploma (NYP, 2007b) and various teaching-learning online activities are organised with the aim of improving students' competence. Other programming languages covered in the first year include web based programming and spreadsheet programming. However, the programming aspects of these modules are less than 20% of the curriculum compared to the PrC that is fully programming. As a foundation module, the programming language covered in the PrC is applicable to other related modules up to the third year, and the programming concepts are transferable to all programming modules and projects in the EI diploma. As such, it is important that students have a clear grasp of the topics covered in PrC in order for them to progress to other programming related modules in the EI diploma.

1.3 Research interest

Departmental and student improvement – The IT diploma for Engineering Informatics(EI) has a vested interest in improving computer programming skills as the profiles of its students are the lowest median with regard to entry-level GCE 'O' level aggregate points compared to the other diplomas in the school of IT (NYP, 2005b). For students who failed this module, they are allowed to repeat this module but they will not be able to take other related programming modules until they have cleared this module. For students who scored a grade D in the overall assessment for PrC, their programming skills are barely there and these students struggle every semester to clear other programming related modules. Based on the significance of improving students' understanding of computer programming in their first year, the EI department has implemented several changes in its curriculum, course delivery and assessment. It is of interest to this research to evaluate the effectiveness and efficacy of the changes

that were put in place in order to determine the variables and their influence on students' competence in computer programming. In doing so, this research aims to discuss the problems faced by first-time programmers and thereafter recommends effective methods to teach novice programmers.

Educational theory for software education – According to the Association for Computing Machinery (ACM, established since 1947), computer programming falls under the software engineering sub-discipline of computer science (ACM, 2007). The discourse on software education and primarily on teaching computer programming to first-time or novice programmers can be seen in two perspectives (Robins et al., 2003): 1) software engineering based studies that are concern with the development of programming tools or methods to solve specific problems; 2) educational studies are concerned with computer program comprehension, cognitive or mental models and the knowledge and skills of computer programming. Educational studies of learning computer programming may include psychological discourse on motivation, mastery and behavioural patterns. This study supports the latter perspective revolving around the notion of learning and instructional strategies for building competence in computer programming. The findings from this study will add to the growing literature on identifying effective methods for teaching and learning computer programming.

Learning technology implementation – Since the dawn of the internet and the world-wide-web in the mid-1980s and early 1990s (Bates, 2005), opportunities for using computer technology in enhancing teaching and learning have increased dramatically. Education technology has evolved as the vehicle to global education and online learning (Littlejohn and Pegler, 2007). It is perceived as an enabling tool to the development of independent learning skills, active engagement and self-directed learning. Yet, some reports from current discourse in e-learning confirms otherwise. Mason (2001) reports students feeling overwhelmed, lost in cyberspace, isolated and apart from community. These issues are the potential causes for the consistently high drop-out rate in online learning programs (Bates, 2005). What was intended to be promoting competency has instead brought obstacles to learning. Clearly, what is lacking in these purely online environments is adequate support and infrastructure for the advantages of exploratory learning to be fully realised. Conversely, traditional classroom environments lack visual expression, flexibility and recall.

Driscoll (2002) puts forth the blended learning paradigm which combines the best of both onsite and online learning environments. By including face-to-face interaction, blended learning provides possibilities for open-ended and learner controlled activities; promotes active and engaged learning that is self-directed and regulated in phases along with instructional events that are structured from the expertise of the teacher. In

this respect, blended learning has the potential to address both sides of the coin, resolving the problems in each instructional strategy. This study explores the use of online learning and blended learning and makes comparisons on their impacts on students' learning. In this manner, this study is uniquely positioned to report on both learning technologies.

Competence based environment – also known as competency based education (CBE) involves the improvement of students' ability to deal with non-routine and abstract work processes, to operate in dynamic or ill-defined environments, to understand evolving systems and to work meaningfully in groups (Keen, 1992). The relevance of CBE in a polytechnic institution is that it enables the integration of professional knowledge into the academic curriculum. This implies that graduates must have the ability to coordinate skills, knowledge and attitude to solve problems in complex environments. Competence is more than the sum of knowledge and skills; it integrates knowledge, skills and attitudes holistically to enable adequate and effective action in a given situation (Kirschner, 2005). This makes the teacher less of an instructor and more of a facilitator. Similarly, student assessment has to accommodate a more diagnostic nature as in formative evaluation rather than judgmental as in summative evaluation. In CBE, performance evaluation has to include both components in order to realise the objectives of building competence levels in student learning.

Curriculum analysis and evaluation – The study of curriculum analysis in the wider social science context is concerned with the institutional setting in schools and classrooms (Franklin, 2000) and its impact on the patterns of resource allocation, legitimacy and power relations. There is a large academic community who engage in curriculum analysis, and education ministries over the world are concern with curriculum evaluation (Posner, 2004). In its narrowest concept, curriculum refers to the intellectual material to be transmitted to students. As the concept of curriculum broadens, it includes the reference texts, the teaching-learning materials and activities and even the pedagogical techniques employed to module delivery. Within the educational context, the study of curriculum is mainly prescriptive – its analysis and evaluation is based on the knowledge it is meant to fulfill (Pinar et al., 2005). In terms of curriculum development and implementation, IT curriculum is currently concerned with how technology and industry forces will necessitate curriculum changes (Irons et al., 2004). However, in this study, the research interest is limited to the learning objectives of the curriculum and its impact on the teaching, learning and assessment systems. In this respect, this study is interested in the constructive alignment of the curriculum objectives (Biggs, 2003) and how it has driven the students' learning. Related to this notion of constructive alignment, is the assessment for learning.

1.4 Significance of the study

Just as there are many stakeholders in education, a meaningful research should be able to benefit as many if not all stakeholders in education. This research although limited in scope aims to provide some insight to the stakeholders in education. The prevalence of programmable instruction in any machinery be it a digital watch or calculator, makes the learning of computer programming significant to almost every industry and profession. In fact many courses in higher education provide for students from other disciplines who wish to take up computing (Alexander et al., 2003).

Industry and technology

Advances in technology are strongly tied to software development and computer programming capabilities (Alexander et al., 2003); the maintenance of technology alone in the form of network and communications resources and its supporting software, accounts for more than half of the pool of computing professionals. To meet this high demand, industries turn to educational institutions to churn out IT professionals. As such the issues raised in this research should be of significance to industries especially those who support industry attachments and fund research programmes in IT and education development.

Government and ministry

In the pursuit for a global and knowledge based economy, governments are pushing for greater IT growth and use in the educational curriculum to meet the demands of industry and technology as stated above. Understanding what drives the learning of computer programming is a first step towards developing a benign culture for developing budding computer programmers. By considering the findings of this study, the government or its ministries in education and in information technology may examine its own policies into the teaching-learning of computer programming. Irons (et al., 2004) claims that the chaotic nature of higher education resulting from the lack of appropriate policies and funding is exacerbated within the computing discipline; yet with a ready resource through student internships and industry attachments, the computing department is well placed to source for non-government funds. Thus continuing research into the teaching-learning of computer programming, which is the interest of this research, can only serve to promote government's aims to promote skilled IT professional for an ever expanding industry.

Management and administrators in education

Being the policy makers especially in curriculum issues, education senate and administrators need to be aware of technology advances and its impact on education. With the greater emphasis on education technology, there is a need to make educators IT literate and to assist in their proficiency in education technology. The next natural

step is to allow educators to take up computer programming courses which is suggested in this study (refer to chapter 7). Presently there are PGCE (post graduate certificate in education) courses where in-service teachers may take up introductory computer programming courses (HEA, 2007). When teachers are effective IT users, students benefit and inadvertently or otherwise are motivated to take up computer programming.

Students and educators

This research examines if the performance of students are aligned to the objectives of the curriculum. The results of this research will provide valuable information to course managers in designing curriculum and delivery of computer programming modules for novice programmers. In addition, course managers will obtain a better understanding of how assessments affect students' learning and students' performance in computer programming modules.

This study shows how the roles of curriculum, pedagogy and assessment are inter-related and have to be integrated into the curriculum to foster better learning for students. For educators, this knowledge may assist them in preparing appropriate teaching-learning activities to cultivate students' learning and performance in computer programming. For researchers of computer programming or technology training, the findings will add another outlook to the teaching and learning of computer programming by novice programmers.

Current discourse in learning IT skills is advocating for integrating computer programming skills into the content areas (Bach et al., 2007). Teaching computer programming as a separate task does not help students to apply computer skills in meaningful ways. There is a need to use technology as a tool for organising information, communicating and exchanging ideas and finding new solutions (Yelland, 2007). This research builds on the integrated curriculum framework to enable novice programmers to relate computer programming to its meaningful content areas. For students who are keen to pick up computer programming, the findings will give them an awareness of what and how to focus when picking up computer programming skills. For students and teachers already in IT, this study shares with them the best practices of teaching novice programmers and the problems that novice programmers face.

1.5 Research questions

It is to the benefit of any new initiative to be compared to its predecessor. Accordingly in this research, the analysis of the blended learning framework implemented in the integrated curriculum of computer programming is compared to the traditional structured framework that it replaces. The main research questions explored in this study are:

Q1. In what ways have the change in curriculum from a traditional structured approach to an integrated, blended learning framework influence the students' learning in computer programming?

Q2. To what extent are the assessments affected by the traditional structured curriculum and the integrated, blended learning curriculum?

Q3. How have students performed in the computer programming module in the traditional structured environment compared to the integrated, blended learning environment?

The above questions fulfill the research aims to seek an appropriate learning framework and environment and to serve the integrated curriculum in a competence based environment that the polytechnic education embraces. Students today must be able to sort and validate information through critical thinking and applying integrated skills to enable them to find meaning in their learning career (Fink, 2003). Each research question corresponds to the following research objectives:

i. To examine the ways in which integrated, blended learning framework influence the students' learning in computer programming;
ii. To investigate the extent assessments are affected by the integrated, blended learning environment; and
iii. To analyse how students are performing in the integrated, blended learning environment.

The theme emphasised throughout this report is students' learning and understanding of computer programming especially for first-time programmers. The curriculum analysis explored in this study is an evaluation of the impacts and influences that curriculum aims, objectives and syllabus have on students' learning of the introductory programming module. Other components of the curriculum that affect students' directly in their learning are the course delivery and assessments. In this respect, the combination of curriculum, pedagogy and assessment are examined to evaluate students' learning. Several frameworks that involve these components are discussed in subsequent chapters of this study to give a rich holistic meaning to the integrated curriculum.

1.6 Terminology used in this study

Assessment: method or procedure to evaluate students' understanding or knowledge of the topic being tested. (www.dictionary.com)

Formative assessment: a non-standardised assessment that provides feedback for improvement of the students' learning. (Biggs, 2003)

Summative assessment: a standardised assessment that grades the performance of the students. (Biggs, 2003)

Blend: to mix smoothly and inseparably together – *to blend the ingredients in a recipe*

To fit or relate harmoniously – *fusion music is a blend of different music genres* (www.dictionary.com)

Blended learning: Learning with different instructional strategies within a classroom (based on this research; see Learning and Instructional strategy below)

Computer: a machine that is capable of processing data and information electronically and digitally; requires physical components known as hardware and electronic components known as software to run or execute a set of well-defined instructions. (www.webopedia.com)

Communication: the act or vehicle that facilitates an exchange of ideas, views, opinions, etc. (www.dictionary.com)

Cooperative learning: Learning in small groups where students work together to achieve shared goals. (Johnson & Johnson, 2004)

Collaborative learning: Learning in groups where students create their own learning through dialogues and interactions among peers, other groups and tutors. (Laurillard, 2002)

Curriculum: a prescribed set of topics to be covered in a course of study with stated objectives and learning outcomes. (Biggs, 2003)

Education Technology (ET): harnessing technology namely IT, for more effective teaching (Biggs, 2003).

Information Technology (IT): a broad domain concerned with all aspects of managing and processing information with the use of computers. (www.webopedia.com)

Information and Communication Technology (ICT): the study or business of developing and using computer technology to process information and to promote communication.

Integrate: to incorporate separate parts into a combined, blended and unified whole. (www.dictionary.com)

Integrated curriculum: a set of topics that is derived from separate domains or fields to form the syllabus for a curriculum. (as applied in this research).

Internet: a massive network of networks, a networking infrastructure. It connects millions of computers together globally, forming a network in which any computer can communicate with any other computer as long as they are both connected to the Internet (www.webopedia.com).

Instructional strategy: technique that may be used to capture attention, increase motivation and provide cues to facilitate learning. (Laurillard, 2002)

Learning: to acquire knowledge of or a skill in a subject by study, by instruction or by experience. (www.dictionary.com)

Electronic learning (e-learning): learning through the use of computer technology, usually web based or hypertext based; content is delivered via audio/video/networked media e.g. CD, DVD or internet. (www.webopedia.com)

Online learning: learning through the use of computer technology connected to a networked environment, may be similar to e-learning, depending on context. (Jochems et al., 2004)

Pedagogy: the art or science of teaching; education; and instructional methods. (www.dictionary.com)

Software engineering: The computer science discipline concerned with developing large applications. Software engineering covers not only the technical aspects of building software systems, but also management issues, such as directing programming teams, scheduling, and budgeting. (www.webopedia.com)

Synchronous: Applied to communication that occurs simultaneously, where tutor and students are connected at the same time such as a telephone conversation. Contrast with asynchronous where communication is separate such as electronic mail. (Laurillard, 2002)

Teleconferencing: Any form of interactive person-to-person communication over a distance; allows many-to-many discussion. (Laurillard, 2002)

Virtual: As opposed to real or physical, implies a conceptual or simulated environment; e.g virtual learning environment refers to a classroom environment without physical desks or rooms. (www.webopedia.com)

Virtual reality: used more generally to refer to any virtual world represented in a computer, even if it is just a text-based or graphical representation. (www.webopedia.com)

Web log: also known as blog in short form. A personal journal of an individual that is publicly available; the author is known as a blogger and maintains updated information on the blog as frequently as warranted. (www.webopedia.com)

Web site: A set of interconnected formatted documents (web pages), usually including a homepage, generally located on the same server, and prepared and maintained as a collection of information by a person, group, or organization. (www.dictionary.com)

World Wide Web (WWW): A system of Internet servers that support specially formatted documents. The documents are formatted in a markup language called HTML (*HyperText Markup Language*) that supports links to other documents, as well as graphics, audio, and video files. (www.webopedia.com)

XML: short name for eXtensible Markup Language to enable Web documents to define, transmit, validate and interpret data between applications and organisations. (www.webopedia.com)

1.7 Assumptions and limitations

The focus of this study has evolved from the following:

1. Curriculum review and analysis is conducted for a first-year polytechnic IT diploma in the Principles of Computing (PrC) module. The results cannot be generalised to other types of post-secondary or tertiary certification by other institutions.
2. The results are confined to the field of computer programming and its effects cannot be generalised to other disciplines.
3. Data compiled for this study is retrospective and students' identities are not revealed or compromised in any way.
4. The results of this study are based on qualitative analysis of the curricula and quantitative analysis of students' past performance in the PrC module. As such, the study does not include the social, moral or motivational issues of the students or tutors.
5. The network and computer infrastructure that enabled the online activities in this study are assumed to be in working order and do not interfere with the data collection nor students' learning activities.
6. The study does not differentiate between students with personal notebooks and those with home personal computers nor between students without any computing equipment. Students are able to book computer usage within the school's premises and students are responsible for their own progress.
7. Students' absence and those missing out on conducted lessons are not measured as the online facilities are available during school week and online materials are made available for students to copy and to study at their own pace. Students are allowed to attend lessons in different module groups to make up for missed lessons.
8. This research does not involve classroom behaviour hence, it is important to note that tutors' teaching styles and classroom management are not being considered as part of the study.
9. Tutors involved in the delivery of the PrC module have received prior training on the programming language and teaching materials, and they have been involved in teaching first year computer programming to EI students for at least 3 years. Each tutor is an IT professional with at least 10 years of industry experience.

10. This study covers only the first semester of the EI diploma where the PrC module is being taught. Impact of the other core modules are not discussed in detail as their varied curricula is beyond the scope of this research.

1.8 Outline of the dissertation

Subsequent chapters in this report discuss the research activities carried out in this study. In chapter 2, the literature review explains the theoretical concepts and constructs covered in this study. Chapter 3 discusses the research design and methodology applied in this study. The comparative case study and the various qualitative and quantitative methods are described as well as their associated benefits and issues. Chapter 4 delves into the qualitative analysis and findings and is followed by the quantitative analysis and findings in Chapter 5. A summary of both analyses is made at the end of chapter 5. The subsequent discussion in chapter 6 interprets the findings in terms of past studies illustrated in chapter 2 and the research methods in chapter 3. Chapter 6 further includes the main contributions of this research and the recommendations for module improvements. Finally, chapter 7 concludes with the summary of implications of this study and suggestions for follow-up research.

Chapter 2 Literature review

By surveying the literature on concepts, frameworks and models covered in this study, the debate on what works, benefits and limitations are explored. This chapter can be divided into four main themes: a. sections 2.1 to 2.2 review the current discourse on integrated curriculum and explores how the learning of computer programming has evolved into an integrated curriculum; b. sections 2.3 to 2.5 explore the concepts and approaches of instructional strategies, online learning and blended learning respectively; particularly how blended learning is introduced to extend the efficacy of e-learning; c. section 2.6 discusses the role played by assessments in driving students' learning; and d. section 2.7 illustrates the various dimensions of integration and the challenges and issues involved in teaching and learning computer programming.

2.1 Significance of the integrated curriculum

Various studies have been conducted (as discussed by Mallery, 2000) to validate how an integrated curriculum can result in greater intellectual curiosity, improved attitude towards schooling, enhanced problem-solving skills, and higher achievement in college. Fink (2003) signifies that when students focus on problems worth solving, motivation and learning increase. Another premise supporting the move towards integrated curriculum is that the current system of discipline-based education is not as effective as it should be. The assumption is that most real world problems are multidisciplinary in nature and that the current curriculum is unable to engage students in real world situations. Thus, a discipline-based curriculum should be augmented with an integrated curriculum (Czerniak et al., 1999).

Some schools have used an integrated curriculum as a way to make education relevant and thus a way to keep students interested in school (Bean, 1995). In a traditional program, relevancy can be a problem. One of the most common questions in a mathematics class is, "Why are we learning this math?" And the common response is, "Because you will need to know it in your math class next year." This response seldom satisfies the learner. Schools report higher attendance rates when students are engaged in an integrated curriculum (Maurer, 1994). Having the opportunity to utilise knowledge and skills from several disciplines does offer increased opportunities for making the curriculum relevant. However, just because a curriculum is integrated does not automatically mean that it is relevant. As such, one of the greatest issues in integrated curriculum, as highlighted by Vars (1991), is how the interest in discipline-based topics could wax and wane as educators vacillated between subject matter and social problems.

Models of curriculum integration

Over the past decade, several models of curriculum integration have evolved. A review of the literature reveals that far more curriculum integration occurs at the primary levels of education than at the high school and college levels (Cushner, 2003). The emerging trend is for elementary schools to build interdisciplinary curriculum around themes (first submitted by Humphreys et al. 1981), whereas in secondary schools and colleges integrated curriculum are more likely to be based around problems (Savin-Baden et al., 2006). An example of a theme at the primary level could be "Our Community," which affords a relevant setting to specify distance, area, and quantities in the community; to read descriptions of the development and growth of the community; to interview and write about senior citizens who live in the community; to focus on the resources needed to sustain a community; to recognize the blend of ethnic influence on community life; to investigate community festivals and other cultural activities; and to engage in some of the technologies important to individual and community growth. On the other end of the spectrum, a university capstone course may involve students in solving a real world problem such as the design, development, and installation of automated tooling in a manufacturing plant. A solution to this problem would naturally lead the students into mathematical, scientific, and technological issues to be addressed and resolved.

Advantages of the theme-based model are that teachers can still identify with a given discipline (Humphreys et al., 1981); it is easier to connect the curriculum with national standards and state frameworks, and students are able to make connections among objectives from various disciplines. There may be a tendency, however, for a given theme and/or key concept to have little relationship with a specific discipline, causing the tendency for teachers to engage students in shallow or irrelevant learning.

In the interdisciplinary model (Fogarty, 1991), schools group traditional subjects into blocks of time, assign a given number of students to a team of teachers, and expect the teachers to deliver an interdisciplinary or integrated curriculum. For example, the core team may consist of four teachers who have approximately 110 students for a block of four periods a day. These teachers are given one hour of common planning time and another hour to learn on their own. The administration empowers them to use their block of time (approximately 175 minutes) in any way they wish. The most typical daily schedule involves groups of approximately 30 students rotating through the four disciplines. At least once a month, the teachers may introduce a new theme to the entire group at the same time or, they may take all of their students on a field trip. In practice, this model is being used with greater and greater frequency at the secondary school level.

This multi-discipline based model (Fogarty, 1991) offers several advantages i.e. teachers are given time to work together, they have a limited number of students, and this model can support a traditional curriculum while offering scheduling flexibility to the team. One disadvantage is that it is easy for teachers to simply continue doing what they have always done with little or no attention given to the interdisciplinary or integrated curriculum. The biggest disadvantage is that standards-based, integrated curriculum across the disciplines is scarce (Maurer, 1994), which means that teachers need to develop the curriculum on their own. Since the process of curriculum development is so time consuming, they are able to implement an integrated curriculum for only a small portion of the school year.

Another curriculum integration model involving collaborative learning is the problem-based model (Boud and Feletti, 1998). Ideally, this model places technology education at the core of the curriculum. Since we live in a highly technological society and technology is a human endeavour, this is a natural way to design the curriculum. With a technological problem at the centre, disciplines lend their support in helping to solve the problem. An example problem might be to determine how the waste produced in a community could be turned into an asset. In this instance, the social studies class can address the role of local government in collecting and disposing of waste; in science the emphasis could be on reducing materials to their basic elements and recombine them; and in mathematics one could study measurement, area, volume, and so forth. In technology education, the focus might be on the various technologies used to separate waste into categories as well as the transformation of waste into usable materials.

The problem-based model has been implemented in higher education with varied results (Savin-Baden et al., 2006). An advantage of this model of integration is that it offers high potential for the identification of relevant, highly motivating problems or scenarios. On the other hand, a disadvantage of this model is the difficulty of assuring that state frameworks and/or national standards are fully addressed in a given grade level.

There are other integrated models being implemented which are variations of the models mentioned above. From here, it can be readily inferred that researchers and practitioners must have a strong belief system in favour of the integrated curriculum if, in fact, they are to succeed in a sustained manner.

Implications of implementing an integrated curriculum

Past research had revealed that no matter which model was selected, there are several common factors that tend to emerge (Ornstein et al., 1999). Firstly, educators must

shift their belief system from one that is primarily didactic in nature to one that has a foundation in constructivism. Rather than asking students to follow the steps of procedure, memorize facts, or verify given principles or laws, educators have to encourage students to work together to discover knowledge, applying their knowledge as they solve real world problems.

Accordingly an extensive amount of professional development is essential for teachers to adapt to the integrated curriculum goals (Thorburn and Collins, 2003). This includes a significant intervention of two or three weeks of knowledge development in curriculum areas other than the one they are certified to teach. Also, this professional development must include extensive practice in the use of constructivist-oriented pedagogy. Another recommendation is that teachers become members of learning communities, working with one's peers to improve education. At another level, teachers work with their students in solving problems that have multiple answers.

Research has shown that integrated learning is enhanced when students learn to interact with one another (Angelo and Cross, 1993). As such, teachers need to become skilled in facilitating small group or collaborative learning in addition to incorporating experiential-oriented instruction. This includes inventorying and storing materials, the safe operation of instrumentation e.g. machines and equipment, and leading students toward efficient progress. Besides teaching and learning methods, teachers should employ authentic assessment strategies such as portfolios, performance exams, and rubrics to document student progress as opposed to standardised tests (Chapman and King, 2004). Authentic assessment as defined by Wiggins (1990) is based on performance where students demonstrate their knowledge and competencies identifying strengths and weaknesses. Methods applied are formative in nature and meant to coach students to apply and integrate their knowledge in real world contexts to gain authentic as in genuine understanding.

As discussed by Pinar (et al., 2005), stakeholders, administrators and school boards need to be oriented so that necessary resources and ongoing support can be provided to the teachers. Public information strategies have to be implemented in order to inform the community and parents that a new paradigm of education is being used. The expectation is for education to be provided as it has always been, and unless the public is informed of changes to be made, there is likely to be resistance. For an institution, changing to an integrated curriculum requires systemic reform (Posner, 2004). This includes the way teachers are prepared, certified, and assessed. Attention must also be given to state-wide assessment of students and the process whereby teacher credentials are renewed.

2.2 Integrated curriculum for computer programming

As a profession, programming is a curious blend of art, science and engineering (Lohr, 2001). The task of making computer software is still a remarkably painstaking, step-by-step endeavour. It involves more craftsmanship than machine magic, a form of creativity in the medium of software – just like chefs work with food, artists with paint, programmers work with code. Yet programming is a practical art form involving the engineering fascination with how things work and the inclination to build things.

According to Robin (et al., 2003), programming is almost always taught as a craft in the context of current technology (e.g. Java and its tools). How can we teach programming without being tied down by the limitations of existing tools and languages? Programming has been described by many authors as the new Latin of the school syllabus, a kind of mental whetstone for developing minds. It was falsely assumed that students would develop their general problem-solving skills through learning programming. However, reports from teachers of programming and results from some empirical studies (Van Roy et al., 2004) suggest that the teaching of programming has created significant difficulties for high-school and university students, and has failed to catalyze the development of higher order thinking skills. What has gone wrong?

The programmer's objective, for novice and expert alike, is first to specify a detailed plan that can be carried out (Abelson et al., 1996). That is, the programmer has to decompose the initial task. This is not trivial: Many people are quite unable to say how they perform certain tasks. For instance, many students in introductory programming classes are unable to explain how they are able to select the smallest of a series of integers. Next, the programmer must map this plan into the constructs of the target programming language. There are two points to be made about this mapping process. First, for the process to be "clean," the programmer needs to have a very clear idea of the abstract plan and of the constructs available in the programming language. One study of novice programmers conducted by Soloway (et al., 1989), showed that many novices had very fuzzy notions about a programming language – substantial misunderstandings had occurred with regard to virtually every construct in the language. Second, task decomposition and program coding are not as neatly decoupled as assumed. A simple example: If arrays are not available in the target programming language, then a plan that assumes this capability would be badly flawed. A thorough knowledge of the facilities provided by the programming language is needed even at the stage of formulating the task plan. Debugging a program is similarly complex and demands a variety of skills, including an ability to coordinate information derived from sources such as error messages, the program plan, the program specification, and the actual code.

To appreciate computer programming, students need to understand what is involved in the "task of programming" as discussed by Van Roy (et al., 2004). Firstly, programming is the act of extending or changing a system's functionality, i.e. for a software system, it is the activity that starts with a specification and leads to its solution as a program. Secondly, programming involves both (language-independent) architectural issues and (language-dependent) coding issues. Finally, to confound the issue of learning computer programming, different languages support different paradigms e.g.

- Java: object-oriented programming
- Fortran: functional programming
- Erlang: concurrent and distributed programming (for reliability)
- Prolog: logic programming

Do any of the paradigms require a student to study each computing language separately?

- New syntaxes to remember ...
- New semantics to understand ...
- New systems to develop and maintain ...

Hence, it is important to put programming on a solid foundation, otherwise students will have muddled thinking for the rest of their careers (Alexander et al., 2003). A typical mistake is confusing syntax and semantics. A simple semantics is important for predictable and intuitive behavior. The semantics should be simple enough to be used by programmers, not just by mathematicians.

Anderson et al. (1990) demonstrate that intelligent computer-assisted instruction (ICAI) technology can be a more effective way of teaching introductory programming courses – for certain populations. Specifically, the authors discuss the pedagogical effectiveness of a Lisp tutor developed at Carnegie-Mellon University. Soloway's (et al., 1989) idea of learning to program is equivalent to learning to construct mechanisms and explanations where his research challenges conventional wisdom by taking a fresh look at assumptions about the art of programming. Soloway advocates a more explicit approach to the teaching of problem-solving skills, which is based on the actual skills experienced programmers use in addressing real tasks. Cognitive experiments have suggested that the domain knowledge of experienced programmers is organized in a radically different way from the domain knowledge of novices; analogous results have also been reported for chess and music. In all cases, experts use larger chunks of knowledge. An important instructional question is how to bring novices up to the expert's level of domain knowledge. Aside from teaching details of the syntax and semantics of a particular programming language, Soloway argues, it is necessary to

explicitly and concurrently explain why and how programs work, the goal of any given program, what plan segments are, strategies for decomposing tasks, rules that well-formed programs adhere to, and design strategies. However to adhere to this approach would produce several radically different types of programming courses.

Abelson (et al., 1996) proposes another radical kind of computational medium named Boxer – one that would be highly customised, and able to accommodate a wide range of users, from a seven-year-old to an experienced non-professional. His research findings suggest that students' difficulties may have more to do with the nature of programming than with teaching per se. Boxer attempts to provide an environment for a wide spectrum of human activities. Its central notion is the metaphor of nested "boxes" organized in a hierarchy that gives novices access to explicit and detailed information about the computer environment, but allows proficient programmers to work at the highly abstract and implicit level that is natural to them. Another view is to allow novices to learn through games programming as explored by Leutenegger and Edgington (2007). The 'fun' element compels and motivates new programmers. In addition, the game presents a visual component for students to see their mistakes in the output or graphics. Whether the game concept can be extrapolated to other computing concepts such as database programming or web programming is yet unclear.

Based on the above arguments, course developers have to take into consideration various aspects and perspectives into the computer programming curriculum. Current discourse in the teaching-learning of computer programming emphasises the need to integrate curriculum, pedagogy and assessment in a continuous cycle to form a 'Constructivist Learning' environment (Jonassen, 1999) or an 'Active Learning' environment (Lavery et al., 2006). Communications skills have to be integrated too as the computing industry requires IT graduates who are effective programmers, team players and all-round problem solvers (Turner et al., 2003). For novice programmers, it is good that they are aware of the expectations of the profession so that their mindset is adjusted accordingly: being able to program is not the goal but rather knowing where that leads to. As discussed in this section, creating an integrated curriculum for computer programming has many dimensions. Invariably, these issues have significant implications on the pedagogy of teaching and learning computer programming

2.3 Instructional strategies for computer programming

In dealing with the issues raised in the previous section, several instructional strategies have been practiced or studied in the literature. An instructional strategy that underlines the pedagogy for computer programming can be defined as the way in which teachers present lesson content or how they facilitate learning (Burton et al., 2004). Computer

programming is at heart a practice oriented field (McAlister and Alexander, 2003); from the first computer architecture produced in 1945 by Von Neuman and binary programming concepts, the teaching-learning of computer programming is addressed from the educational, cognitive and/or psychological perspectives (Robins et al., 2003). One such perspective is constructivism – the notion that building new knowledge (construct) is based mainly on past experience (what the learner already knows) and a stimulus from the current situation (Burton et al., 2004). The theory of constructivism dates back to von Helmholtz (1866 referred by Burton et al., 2004) and is characterised by the following:

- What the learner knows
- Which concept the learner should engage in
- How to form effective construction of the new knowledge

These characteristics imply that learning is an active process and the teacher acts as a facilitator of the process. Fosnot (2005) contends that constructivism is a 'psychological theory of learning that describes how structures, language, activity, and meaning-making come about, rather than one that simply characterises the structures and stages of thought or one that isolates behaviours learned through reinforcement' (p. 34). Fosnot traces the works of Piaget (1950, 1977) and Vygotsky (1962,1978) to underscore two main perceptions of constructivism based on 1) the individual or cognitive constructivism; 2) the sociocultural effects on learning or social constructivism. Fosnot maintains that since humans are social beings, both perceptions are significant to the development of learning in the learner.

According to Posner (2004), a constructivist curriculum is akin to the "thinking curriculum" where to know is not only to receive but also to have interpreted the information and related it to other knowledge; to be skilled is not just to perform some action but also to know when to perform it and to adapt the performance to varied circumstances; thinking and learning becomes merged seamlessly integrating decision making, problem solving and judgements. Posner further cautions that the problem of understanding how much is retained by the learner and what the learner should do to learn makes a pure constructivist approach less meaningful. Other instructional strategies have been included in constructivist frameworks to improve its benefits to learners.

One such approach to assist novice learners is the instructional strategy of scaffolding (Wood et al., 1976). The metaphor implies a structure that is used to guide learners and the structure is reduced as the learners advance in their learning; in the same manner that a physical scaffold is placed to erect a building and its scaffold is removed level by level as the building is completed. Dennen (2004) has conducted extensive

research on scaffolding which includes computer programming. Although there is a debate as to whether scaffolding is teacher directed or learner directed, Dennen argues that scaffolding is able to support the learning of concepts, procedures, strategies and meta-cognitive skills. She further recommends three methods of scaffolding that is central and critical to learner success:

i. Zone of Proximal Development (ZPD) – to provide learning activities that are just beyond the learner's present ability. The ZPD is dynamic; moves as the learner progresses. The teacher has to assess the cognitive and emotional readiness of a learner and the appropriate scaffolding that supports the learner's motivation and confidence.
ii. Intersubjectivity – to enable shared understanding among learners in the classroom. Applying intersubjectivity, the teacher fosters a shared goal that removes conflicts of interest, participation and outcomes.
iii. Fading – occurs as the learner gains independence. This is a gradual process where the teacher gives feedback to the learner and allows the learner to take responsibility to proceed.

The three methods are stages in the scaffolding and can be repeated as a learner moves to the next level of complexity. Scaffolding techniques include questioning, summarising, clarifying and predicting in which text or concept is being discussed.

Collaborative and cooperative learning methods are known to have useful impacts on computer programming instruction. Collaborative and cooperative learning comprises a range of techniques from peer critiques to small writing groups (Johnson and Johnson, 1999). Its aim is to actively involve students in their own learning through sharing among groups (collaborate) and through completing tasks within a group (cooperate). The learning style is a cooperative approach which attempts to tap peer group influence and mobilise that influence in formal academic contexts. Collaborative skills such as praising others, disagreeing politely and listening attentively create a secure environment in which students feel they can experiment with ideas and build upon each other's ideas (Jacobs et al., 2002). If the group dynamics is managed well, collaborative and cooperative learning gives rise to higher-order knowledge which stimulates the group to learn more.

Teachers play a double role of instructor and facilitator and are obliged to develop skills to manage any number of the following problems as discussed by Jacobs (et al., 2002):

- class preparation: Decisions have to be made on class size, classroom arrangement, team members, gender or culture mix of team members, students' and group objectives and expectations.

- group management: How does the teacher manages the noise, delay and other disruptions in the group? What about non-participation or other behavioural problems?
- task division: Do the group members or the teacher decide on which task to assign to each member? Is there a time limit on each task? How much help ought a teacher give?
- assessment: What is the best mode of assessment? Should all members in a group have the same grade or different grade for specific task carried out by the member? Could students assess themselves or their members?

Problem-based learning (PBL, Boud and Felletti, 1998) works well with constructivist frameworks. The main instructional strategy in PBL is the small group tutorial or teams designed to encourage interactive learning among group members (Moesby, 2002). It has been successfully practiced in fields of study such as medicine and health sciences for the following reasons:

⇒ A shift from content-first to problem-first delivery.

⇒ Independent self-directed learning that is learner centric versus teacher-centric.

⇒ Active involvement in the problem solving process versus classroom learning.

⇒ Integrated approach to learning versus discipline based content.

⇒ Reflective learning leading to deep learning versus superficial rote-learning.

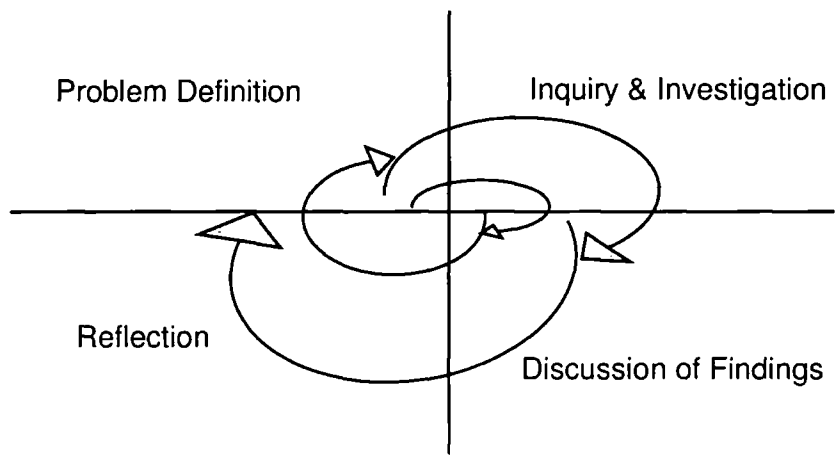


Figure 2.1 Problem-based learning cycle

In computer programming instruction, problem-based learning is applicable to project work where students work in teams to produce solutions to situated problems. As shown in figure 2.1, students go through different phases that are cyclical in nature. At the end of each phase, teams get together to present their results. Critical risks of PBL as explained by Cunningham and Cordeiro (2003) arise when:

- problems presented to learners are ill-structured and targets are unexplained. Learners must have a good grasp of problem-solving techniques and if the facilitator fails to address PBL skills, learners will not be able to learn meaningfully.
- group dynamics and disagreements result in deep divisions in the group. The experience gained is negative, wasting everybody's time and more importantly, thwarting learning. Unlike collaborative learning where tasks are clearly defined, PBL is more open-ended and exploratory. Learners without the experience of collaborative learning will require coaching in cooperative methods and team-building skills.
- negative behaviours such as laziness, prejudice and discrimination, will lead to a breakdown of communication and morale in the group. Learners need to cultivate the habit of active participation, giving positive feedback and constructive criticisms.

The instructional strategies listed in this section are applied in the curriculum of the PrC module of this study. Although these instructional strategies have been adapted to the computing learning and development environment, the strengths of these learning strategies support the learning of first-time computer programmers.

2.4 Online learning and pedagogy

Learning through the use of a computer is deemed to be electronic learning (Jochems et al., 2004) whether the internet is enabled, the materials are downloaded from a networked server or simply using the internet interface with the convenience of the familiar web-based look and feel. It is no wonder that many synonymous terms are used such as online learning, web-based learning, internet-based learning and distance learning. In this report, online learning and e-learning are used synonymously and education technology refers to applying e-learning in its various forms to engage students in learning as supported by Biggs (2003) and Ramsden (2003).

Computers offer the capability for integrating multiple media (multi-media) such as text, diagrams, pictures, sound, movies and animation in a single continuous presentation. In online learning, different media are used to enable interactivity, enhanced graphical user interfaces and animation. Laurillard (2002) argues that none of the media found today is 'developed as a response to a pedagogical imperative' and it is up to educators to 'fashion something academically respectable' from media. In table 2.1 Laurillard presents five categories of media where each category describes a media format and the respective forms of learning experiences.

Learning experience	Methods/technologies	Media forms
Attending, apprehending	Print, TV, video, DVD	Narrative
Investigating, exploring	Library, CD, DVD, Web resources	Interactive
Discussing, debating	Seminar, online conference	Communicative
Experimenting, practicing	Laboratory, field trip, simulation	Adaptive
Articulating, expressing	Essay, product, animation, model	Productive

Table 2.1 Five principal media forms with the learning experiences they support and the methods used to deliver them (from Laurillard, 2002, p.90)

The learning experiences reflected above are not mutually exclusive though they represent the best outcome in terms of the given media. More importantly, none of the media covers the full iteration between the interactions of the teacher in a classroom. In combination however, each media provides the benefits that the other lacks and in conjunction, there is a rippling effect that provides better coverage of the learning process.

In the discourse of online learning, it is perceived to generate the following benefits (Bastiaens and Martens, 2000 and Bates, 2005):

- ☑ Provides a context-rich learning environment since materials are presented in multiple forms;
- ☑ Improves effectiveness of learning since the human senses are engaged to digest information simultaneously;
- ☑ Enables learners to learn at their own pace, to control their own learning path, and to review as often as they wish;
- ☑ Allows learners the freedom to choose the place and time of study;
- ☑ Removes teacher's bias, prejudice and emotional quirks;
- ☑ Reduces teacher's load and involvement, the teacher is a facilitator;
- ☑ Replaces ineffective or potentially dangerous activities with simulations, animations and games; generates effective on-the-job training.

However, the above benefits have not been universally confirmed and even Bastiaens (et al., 2000) raises concern into the context and the implementation of online learning. The benefits can easily turn into disasters if the learning materials are poorly designed and organised; if teachers are not receptive to online learning and not certain how to act as facilitators; and if networking or communications services are not reliable.

Mason and Rennie (2006) have criticised the practice of putting traditional lecture or study materials on the web and calling the course e-learning. Online course content

should instead capitalise on the multimedia features as appropriate as to the context of the learners. Students need considerable support in order to see the advantages of online learning; based on her research, Mason (2001) claims that students perceive e-learning as:

- ☒ more work
- ☒ more expenses – require a computer with network connections
- ☒ more difficult to study or prepare for exams and tests
- ☒ often poorly supported; missing links and corrupted files add to the problems
- ☒ easier to copy and therefore not fair to original work
- ☒ teacher has abrogated the instructor role

Many issues affect the learner (Pintrick and Schunk, 2002) and when confronted with e-learning, students require support and scaffolding to adjust to the online environment. In a computer programming environment, students face another level of complexity in grappling with a computer language as well as the computing environment. E-learning becomes the essential tool to scaffold novice programmers (Boyle, 2005) such that learners can visualise computer programming terms and constructs in layman terms.

Many new technologies used in online learning allow for different pedagogies and instructional strategies to be created and integrated to support the learner (Littlejohn et al., 2007). Several examples are briefly explained here to appreciate the impact of online learning on communication and reflection e.g. chat, blogging, instant messenger; knowledge-sharing e.g. learning objects and file sharing; and data transmission e.g. streaming audio and video.

Internet relay chat (IRC), commonly called "chat", has existed for some time in text form which is basically synchronised written communication or synchronous e-mail (Bach et al., 2007). Its more popular cousin, instant messaging is a more dynamic technology that facilitate group communication by showing all group members when a user logs on resulting in close to synchronous text exchanges (Bach et al., 2007). Other features are its ability to incorporate voice chats, attachments, and its transportability – each user is able to login from multiple workstations (any computer with internet access), but will only receive information on the active computer. These technologies are fast replacing electronic discussion forums and the slower e-mails as they are able to facilitate immediate communication and interaction between learner-learner and learner-teacher.

Recently audio chat has become available, and point-to-point audio connections can be made between any two computers on the Internet. It is also possible to connect to

telephone over the Internet using voice over Internet protocol (VoIP), which is becoming very popular due to extremely cheap or even free calls. This technology has been effectively used to deliver synchronous teaching using an electronic blackboard along with VoIP in teleconferencing or more popularly known as webcasts. Web whiteboarding is another variation that uses similar technology more conveniently as a single tool allowing both teachers and learners to create, manipulate, review and update text and graphical information online while at the same time participating in a lecture or discussion. According to Bach (et al., 2007), these online capabilities have been used in teaching-learning activities such as brainstorming, inquiry and simulations.

Two other forms of online communication tools that are recently developed for educational use are web logs (also known as blogs or online diaries) and e-portfolios, and while these forms are not attempting synchronous communication, they enhance opportunities for more lengthy in-depth reflection that can be in either an individual or group mode. Essentially, blogs are web sites that are organized by time (Mason et al., 2006), consisting of commentary items that are posted in reverse chronological order. They are easy to use requiring little technical know-how since they are template-based, browser-edited and rely on database information. Blogs function effectively for knowledge sharing and community interaction since entries can be posted directly onto the web as the event unfolds. Mason (et al., 2006) describes the use of blogs in higher education courses at the Masters level, and Dennen (2004) presents a more informal use in computer mediated scaffolding of mentoring for pre-service teachers. In both cases, blogs are successful tools in educational environments for encouraging reflection, sharing of knowledge, and building and maintaining a networked community on the Internet.

E-portfolios extend the aspect of reflection, but concentrate more on evidence of the individual's achievements. There is an interactive element, but that is an optional element that can be added if the e-portfolio is intended for multiple reviewers. Mason (2001) describes an application of a multimedia tool that highlights its usage in assessment activities. Similar to the paper-based portfolio, the e-portfolio is a multimedia tool that facilitates the collection and selection of items and due to its hyper-functionality is much easier to handle than the paper-based portfolio; contents can be organized and sorted faster and easier, and hyperlinks make connections between multi-layers of experience possible along with continuous updating features. For educational purposes, e-portfolios have mainly been used in assessment, operating on the principle that 'reflection over time increases a learner's ability to make sense of concrete experience'. Mason calls for further exploration of e-portfolios and is confident in their benefit to learning environments.

E-learning objects are digital resources that can be reused to support learning. The term "learning objects" generally applies to educational materials designed and created in small chunks for the purpose of maximizing the number of learning situations in which the resource can be utilised (Boyle, 2005). Wiley (2002) defines learning objects as an electronic tool or resource that can be used, reused and redesigned in different contexts, for different purposes and by designers or educators. Boyle (2005) offers a detailed review of learning objects examining the characteristics that accelerates their extensive use in online learning – the potential for reusability, generativity, adaptability, and scalability. Furthermore, learning objects involve fully complete and discrete lessons, learning units and courses. Other technologies are available that facilitate faster and easier access to online documents making learning objects all the more attractive.

Contents on standard web sites require constant browsing for updates and developments; however "push" technology involves channel-based delivery that is "pushed" directly to the user's desktop (Yelland, 2007). Channels can be modified relating to interest groups and subdivided into folders containing further links. Push technology and data channels can be used to feed inexpensive and current news and information from relevant sites to instructors and students for learning and research purposes. File sharing offers another innovative tool for knowledge and information sharing between users that is not restricted to location, connection speed or a central server. Access to knowledge is promoted at a group level that is extremely valuable for team-projects, coursework, as well as collaboration at program or institutional level. Wiley (2002) recommends the use of learning object repositories or libraries that can be shared across communities of users in the manner that computer programming resource libraries are made available to IT developers.

A barrier that has been hard to overcome with online learning deals with internet connection speeds and the capability of transmitting large quantities of information without losing quality. This has especially been a problem with large audio and video files. Streaming media technology facilitates the transfer of audio and video files in a stream-like manner (Bach et al., 2007). The advantage of such technology is that the user does not have to wait until the transfer of data is complete – it can be used as soon as data starts arriving at the receiving computer. The data is converted into a format that is sent in a continuous stream of small segments which is played instantaneously; while the first data is played, the other incoming data is downloaded. Streaming technology is not dependent on fast connections, although typically faster connections provide greater quality, especially with video files. Streaming audio has given rise to better educational opportunities such as pre-recorded lectures, newscasts, broadcasts, projects and especially to facilitate e-learning objects download or viewing.

Streaming video offers equally attractive options to overcome the "page-turning" phenomenon of many online and virtual courses (Yelland, 2007).

All the authors mentioned here have unanimously placed a caveat on the prospects of online learning: it is difficult to meet learner's needs without the teacher's diagnosis and intervention. No matter how impressive technology or programmed instruction is – able to understand and react to different learner patterns, simulate complex processes visually, or provide timely feedback – online learning systems lack the ability to make subtle judgements that good teachers do. Higgins (et al. 2001, 2003) asserts that ICT is as useful as what teachers make of the technology: in how they select and organise resources and integrate ICT into their teaching instruction.

2.5 From online learning to blended learning

Computer programming being essentially an IT skill revolves around the successful use of ICT and its online integration in a networked environment for accessibility and availability. Online learning or electronic learning (e-learning) allows students to have access to learning materials on the internet or networked servers. The strength of e-learning in ready accessibility has speed up the information gathering process so much so that students tend to skim over the pages instead of bothering to understand the material presented. Students are adept at using the search tool to look for information and downloading the material with little thought as to its meaning and intention. Laurillard (2002) recommends that students have to be trained to be selective in choosing relevant material and to surf in a more productive and discriminating manner.

Students' learning from ICT can be distinguished between surface learning and deep learning (Biggs, 2003). Surface learning is limited by a selective, piece-meal approach that gets the task done without understanding whereas deep learning is preferred to not only cultivate understanding but also engagement and reflection. The current generation of students are more inclined towards visually-oriented learning and online learning materials can be used effectively to connect with students through animation, graphics, and interactive games and tests (Laurillad, 2002). Interactive media allow students to personalise their learning so as to gain ownership of that learning. The implementation of cognitive principles into the teaching-learning environment will stimulate students to interact with the materials in real-time and advance their learning to the next level.

Despite the obvious advantages of online learning, several studies conducted on the success of ICT and learning (Passey, 2006) revealed that students, who are left on their own without teacher intervention or support, hardly gained improvement in their learning. It is of no surprise that what matters most in the student learning is the face-

to-face interaction with the teacher or among peers. According to Passey, the shortcomings discovered by these studies failed to determine the significance of the study domain and its impact across each topic or within specific elements of each domain; failed to identify the level of involvement that have supported student learning; and subsequently unable to measure the limitations or gaps in student learning. Another misconception of online learning is the individual assumption that instruction should be customised to each student's learning style, habits, previous knowledge and motivation. Johnson (et al., 2004) argues that it is more productive for technology to be used interactively to promote cooperative learning among students.

A recent concept of integrating online learning with face-to-face interaction is the notion of blended learning. Several authors have laid claim to this term but with different interpretations. From the academic perspective, blended learning is the combination of e-learning with different pedagogical approaches (Driscoll, 2002); or with different types of media (Laurillard, 2002); or with different instructional design approaches and learning technologies (Mason et al., 2006). Oliver and Trigwell (2005) argue that the definition of blended learning is too wide and anomalous such that 'almost anything can be seen as blended learning'. They recommend a blend of learning where there is variation in the subject domain by integrating change from the perspective of the learner. This is where ICT and good teaching techniques are used in variation to allow different or changed perspectives for the learner.

Mason (et al., 2006) argues that in reality, any learning experience inevitably involves a combination of different inputs (reading, writing, thinking, talking) and styles or experiences. Various combinations of technologies, locations or pedagogical approaches have claimed to be blended learning:

* Applying asynchronous (e.g. e-mail or blogs) and synchronous (e.g. discussion forums, online chat) technologies in an online course;
* Combining formal learning (e.g. workshops or seminars) and informal learning (e.g. project discussions) in professional development;
* Accessing course material and resources from various locations e.g. learning centre, online libraries and other subject related databases;
* Using e-learning to substitute class attendance in a course.

Based on this understanding, Mason sees the relationship of blended learning as an extension of e-learning with face-to-face interaction as shown in figure 2.2. Distributed education refers to learning that is delivered across a wide geographical area such as different campuses or centres; distance education refers to open learning courses meant for students who are rarely at campus; e-learning refers to online forms of learning; and blended learning encompasses not only face-to-face and e-learning but

includes a range of teaching and learning resources and communication styles. Other alternative terms for blended learning are hybrid learning and flexible learning.

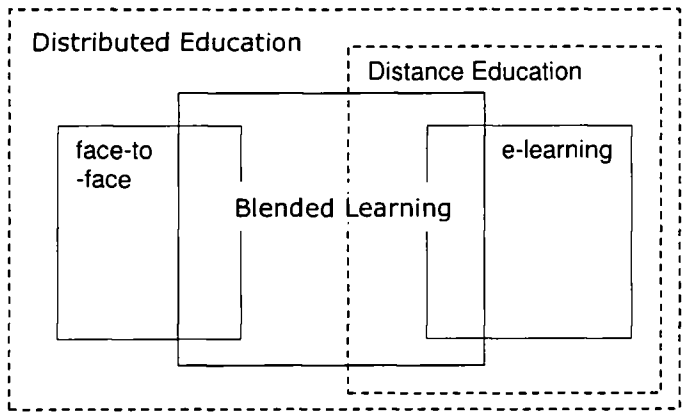


Figure 2.2 The relationship of e-learning to distributed learning (from Mason and Rennie, 2006, p.xvii)

In a recent publication, Littlejohn and Pegler (2007) introduce the concept of *blended e-learning* as applying different components of media resources, learner tasks, learning environments and various time-zones depending on the purpose of learning, the context of learning and the approaches to learning and teaching. They further identify four different blends of implementation as follows:

- Space blend: students meet face-to-face or through virtual learning environments;
- Time blend: learning tasks that are synchronous or asynchronous;
- Media blend: making use of different resources in various formats;
- Activity blend: orchestrating different learning activities to create a learning design.

To accommodate the different e-learning blends, Littlejohn and Pegler (2007) devise the LD_Lite tool based on Koper's (2006) Learning Design (LD) model for the Information Model Sponsor Global Learning Consortium (IMS/GLC), an integrated e-learning standards organisation. By doing so, the blended e-learning proposed by Littlejohn and Pegler is encroaching into the integrated e-learning domain. Koper (2006) has admitted that there are complexities in mapping learning objects with the teacher's pedagogical approaches amongst other issues with the LD framework. Furthermore, blended learning should not be confused with integrated e-learning. The latter regards the online environment as the primary medium where tasks and activities are posted, monitored and assessed online (Jochems et al., 2004) whereas in blended learning environments, e-learning plays a supportive, secondary role. As such, blended learning is more concerned with instructional strategies rather than instructional design models for e-learning.

From the e-learning professional or training perspective, blended learning is a combination of e-learning with other delivery methods sustained over a period (Gray, 2006):

> "The obvious advantage of the blended learning solution is that learning becomes a process, rather than an event. Blended learning puts training into the job environment, provides a forum for every learning style, includes reinforcement and coaching, and uses minimum effort and resources to gain maximum results."

A common ground held by both academics and professionals alike is that blended learning gives the tutor and the learner the flexibility to vary the learning experience using appropriate instructional strategies. The tutor's and the student's interaction plays a key role in managing the different teaching-learning activities. In order to manage the flexibility, appropriate assessments (Oliver and Trigwell, 2005) have to be in place to ensure that this mix-and-match approach does not get lost in the maze.

Integrated, blended learning models

Various pedagogic models have been integrated or blended into a learning environment to foster computer programming skills to the novice programmer. From the constructivist approach which creates a learning environment that promotes active student participation, Jonassen has developed the Mindtools (1996) and the Constructivist Learning Environment (1999). Mindtools function as logical statements to guide the learners and enable learners to teach the computer what the students have already learnt. Hence knowledge is built up not by the computer but by the student. The CLE as discussed by Jonassen (1999) applied the activity theory where the conceptual context of the learner plays a central role. By analysing each context and its relations, the kinds of conversation and collaborating tools will become apparent. The final analysis of the CLE involves the assessment of how components affect each other in this case the achievement of the learning outcomes.

An environment to blend constructivist learning in computer programming through integrating collaborative learning to promote active learner engagement is undertaken by the Active Learning in Computing (ALiC) (Sheridan-Ross et al., 2007, Hatch and Burd, 2006). Another blend of a learning framework is the Virtual Learning Environment (VLE) that combines constructivist and action learning pedagogies into Laurillard's Conversational Framework (Heinze and Procter, 2004). Both approaches have reported issues with students' performance such that students are not prepared to put in more effort than required to pass the module (Heinze et al., 2006, Sheridan-Ross et al., 2007, Hatch and Burd, 2006). Boyle (2005) has similarly implemented a blended learning approach through a VLE that holds e-learning objects to assist students in their understanding of Java computer programming. He claims that the blended

learning approach has gained popularity with the students though they do not prefer a particular learning method (lecture, text, graphic) over another.

Online portfolios (Higgs and Sabin, 2005) have been used to record students' assessments on programming exercises and team projects and to obtain feedback on students' progress. The authors find that online portfolios provide a natural assessment framework from which learning evidence can be recorded and continuous formative feedback from tutors can be given. Another form of recording concepts is proposed through the use of Anchor Concept (AC) graphs (Mead et al., 2006) where students can trace the inter-relations between concepts. The AC graphs highlight the connectivity between elements which can be functions, rules or objects with the intention to scaffold the cognitive understanding of the students as seen below:

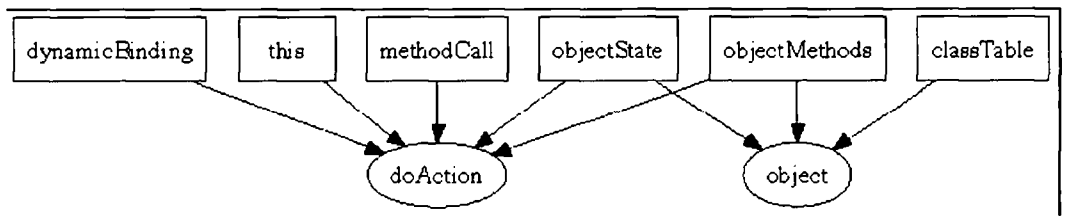


Figure 2.3 AC graph for object orientation – runtime perspective (Mead et al., 2006, pg 190)

Based on several cognitive and pedagogical constructs, the authors assert that the AC graphs provide the knowledge relevant to learning a collection of concepts within a domain and are able to determine the assessment strategies for that concept. However, the creation of the AC graphs is problematic and subjective and can be overwhelming for novice programmers.

Singapore with its emphasis on the IT master plan for its education environment (MOE, 2007d) have produced numerous online learning and teaching strategies and models. Notable amongst them is the Integrated Virtual Learning Environment (IVLE) (CDTL, 2005) which features discussion forums, feedback and a question bank for online assessments of seven generic type questions. Nanyang Polytechnic (NYP) Singapore, has established the Teaching Factory paradigm (Chung et al., 1999) to make learning relevant with industry practice. In addition, the e-learning strategy at NYP is aptly named the 'Integrated Technology Teaching and Learning' which is aimed at curriculum integration through the development of specialised laboratories and-or teaching-learning activities. This research is a product of NYP's learning environment and strategies. The results and findings (in later chapters) will reveal the learning points encountered in the Principles of Computing module of the EI diploma. Accordingly, the focus of this research is to explore the right blends of face-to-face interaction, e-learning and computing practice.

2.6 Learner centred assessment

> *What and how students learn depends to a major extent on how they think they will be assessed. (Biggs, 2003, p.140)*

The above statement implies the notion of students' perception of assessments which infers that assessments drive learning. In agreement with this, Haines (2004) recommends that in designing assessments, educators should be guided by the following principles:

- ∴ Learners are directed by their personal goals and intentions
- ∴ Learning improves with practice
- ∴ Learning improves with feedback

By considering these principles, assessments become relevant, inducing learners to adopt new goals which lead to the accomplishment of desired learning outcomes. Accordingly from this perspective, the assessment for novice computer programmers has to take into account the learners' goals and guide the learner's behaviour through scaffolding and other instructional strategies that improve the learner's competence. In addition, these assessments have to give feedback or meaning to be under the control of the learners so they could be tuned and optimised for individual intentions.

In a computing course, assessments normally include some forms of online assessments. As illustrated by Biggs (2003), online assessment or computer-assisted assessments have the following benefits: allows more than one attempt, supply hints, supply immediate feedback, can guide reading as a result of the test, and randomise questions if necessary. Online assessments allow formative and summative assessments at individual or group level. Students are able to post their answers and evaluation (self-assessment) in a web-based learning portfolio and subsequently invite comments from peers (peer assessments). Biggs defines the following assessment concepts as follows:

Formative assessment: the results of which are used for feedback during learning. Students and teachers both need to know how learning is proceeding. Feedback may operate both to improve the learning of individual students, and to improve teaching.

Summative assessment: the results of which are used to grade students at the end of a unit, or to accredit at the end of a programme.

Continuous assessment: the results of which contribute towards the summative final grade. Assessments are incremental to test the performance of the student over a sustained period of study rather than a final examination.

Self assessment: Getting a student to critique own work based on given criteria; to reflect on own strengths and weaknesses, and if a group project, own contributions.

Peer assessment: Getting a student to critique another student's work based on the same criteria. Self and peer assessment make students aware of the criteria for good performance; they learn to select good evidence and to judge a performance or product. Learning by questioning or critique is part of problem-based learning (refer to section 2.3) which plays an important role in effective professional learning.

Assessment is a theme running through most of the blended learning models mentioned earlier. Formative and summative assessments are integrated into the instructional process as well as individual and group assessments. The assessments are seamless and placed within the learner-environment interaction rather than using the individual or class as the unit of analysis. Laurillard (2002) recommends that effective assessments for ICT-based learning should be collaborative and performed in small groups. To ensure ICT materials are properly embedded into a course: (p. 207)

- Design assessment in terms of objectives
- Design questions to be open, non-technical and conceptual
- Ensure that learning through new media is assessed and accredited
- Design group assessment to fit objectives and modes of collaborative learning
- Involve students in the design of assessment and marking (authentic assessment)
- Reinterpret assessment criteria explicitly for learning from new media
- Use the productive media to test the new learning objectives that are being encouraged
- Communicate assessment requirements clearly

How do assessments relate to curriculum?

Teaching and learning take place in a curriculum system, which encompasses the classroom, the department and the institutional levels (Posner, 2004). In a poor system, in which the components are not necessarily integrated and tuned to support learning, only high achieving students spontaneously use higher-order learning processes. In an integrated system, on the other hand, all aspects of teaching and assessment are tuned to support high level learning. Constructive alignment is such a system (Biggs, 2003). It is an approach to curriculum design that optimises the conditions for learner centred learning.

What is constructive alignment?

According to Biggs (2003), the 'constructive' aspect refers to what the learner does, which is to *construct meaning* through relevant learning activities. The 'alignment' aspect refers to what the teacher does, which is to set up a learning environment that supports the learning activities appropriate to achieving the desired learning outcomes. The key is that the components in the teaching system (as shown in figure 2.4),

especially the teaching methods used, and the assessment tasks are *aligned* to the learning activities assumed in the learning outcomes.

To achieve constructive alignment in a course, four major steps are outlined as follows:

1. Defining the curriculum objectives
2. Choosing teaching/learning activities likely to lead to attaining the objectives
3. Assessing students' learning outcomes to see how well they match the learning outcomes
4. Arriving at a final grade

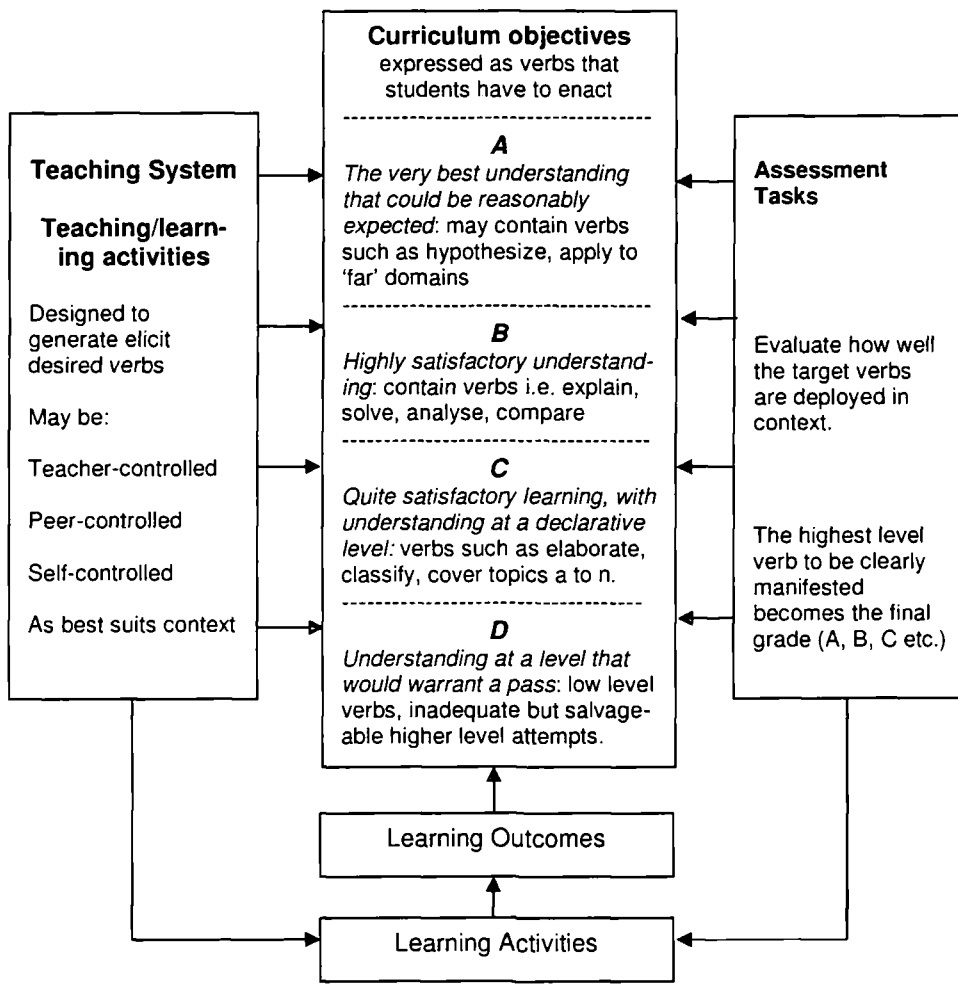


Figure 2.4 Constructive alignment model from Biggs (2003, p.28)

Next, teachers and subject experts have to develop a sound general framework for structuring levels of understanding of the topics and content appropriate to the learning outcomes. In order to evaluate how well a topic is understood by a learner, Biggs recommends the SOLO taxonomy, Structure of the Observed Learning Outcome (Biggs and Collis, 1982); 'provides a systematic way of describing how a learner's performance grows in complexity when mastering many academic tasks' (Biggs, 2003, p.38). Figure 2.5 shows the progressive levels of understanding, with some illustrative

verbs for each level, based on the SOLO Taxonomy. Those verbs become the markers throughout the system. They need to be embedded in the teaching-learning activities and in the assessment tasks so that there is a common track. The SOLO levels of understanding are:

L0. Prestructural – acquire information without understanding or making sense
L1. Unistructural – simple connections are formed
L2. Multistructural – more connections and logic are formed
L3. Relational – able to relate or generalise different parts in relation to the whole
L4. Extended abstract – able to generalise and transfer ideas within and beyond subject area

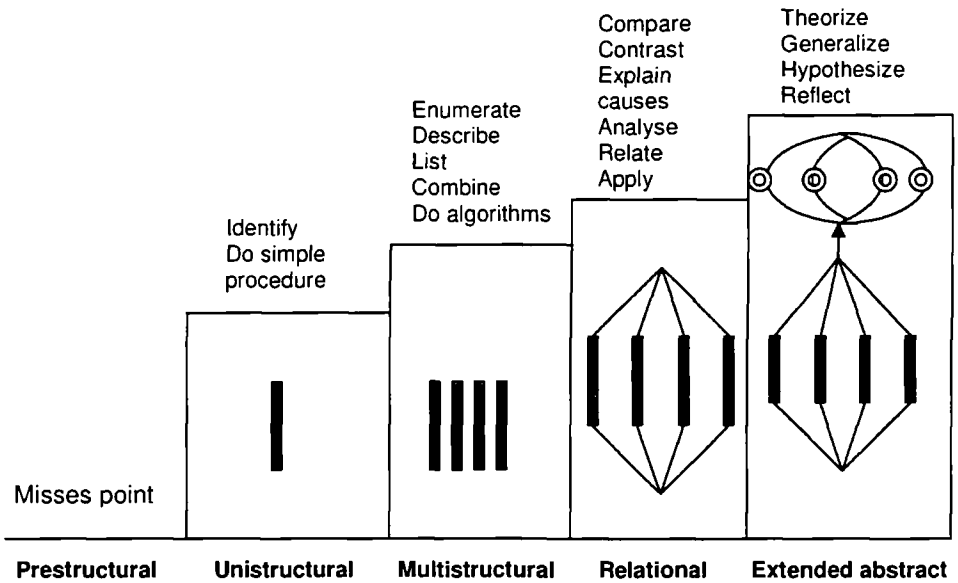


Figure 2.5 SOLO taxonomy from Biggs (2003, p.48)

Moseley et al. (2005) observe that the SOLO taxonomy is only concerned with students' performance without considering social interactions, interests or behaviour; students' understanding is expected to be predictable and moves in the stated progressive levels. However, Moseley concedes that the SOLO taxonomy has been successfully applied in a wide range of studies and at all levels of education which speaks for its practical value and effectiveness.

Another theoretical framework for constructive alignment is that proposed by Anderson and Krathwohl (2001) which revises the Bloom's taxonomy (1956) for cognitive domains i.e. knowledge, comprehension, application, analysis, synthesis and evaluation into six cognitive processes i.e. *remember, understand, apply, analyse, evaluate and create*. Similar to Bloom's and SOLO taxonomies, the revised taxonomy is hierarchical where the progress to the next level depend on mastery of the preceding levels. Again, the students' understanding takes precedence over other social

interactions. However, Anderson and Krathwohl introduce four knowledge types: factual, conceptual, procedural and meta-cognitive; and these knowledge types are separate from the six cognitive processes but together they form a two-dimensional table. Their framework is based on the constructivist approach where constructive alignment is achieved through the following questions:

- # The learning question: what is important for students to learn in the limited school and classroom time available?
- # The instruction question: how does one plan and deliver instruction that will result in high levels of learning for large numbers of students?
- # The assessment question: how does one select or design assessment instruments and procedures that provide accurate information about how well students are learning?
- # The alignment question: how does one ensure that objectives, instruction and assessment are consistent with one another?

Knowledge Dimension	Cognitive process dimension					
	Remember	Understand	Apply	Analyse	Evaluate	Create
	Recognising Recalling	*Interpreting Exemplifying Classifying Summarising Inferring Comparing Explaining*	*Execu-ting Implem-enting*	*Differenti-ating Organis-ing Attribu-ting*	*Checking Critiquing*	*Generating Planning Producing*
Factual knowledge	*Example assessment*					*Example activity*
Knowledge of terminology *Knowledge of specific details and elements*	Quiz on addition facts					Prepare and deliver a short talk about an aspect of a famous person's life

Table 2.2 Anderson and Krathwohl taxonomy (from Moseley et al. (2005) p. 106)

A sample of the two dimensional table is illustrated in table 2.2., Moseley et al. (2005) assert that the taxonomy is strongly focused on the cognitive domain and has grouped critical thinking and problem solving within the *understand* cognitive process. The taxonomy is a useful tool for teachers encouraging them to clarify and communicate what the learning outcomes and assessments are.

An alignment model that is widely used in north America is Webb's (2002) alignment model which recommends four alignment criteria as follows:

- ! Categorical concurrence: Are the same or consistent categories used in both curricular expectations and assessments?

- Depth-of-knowledge (DOK) consistency: To what extent are the cognitive demands of curricular aims and assessments the same?
- Range-of-knowledge (ROK) consistency: Is the span of knowledge reflected in curricular aims and assessments the same?
- Balance of representation: To what degree are different curricular objectives given equal emphasis on the assessments?

Webb defines alignment as the "degree to which expectations and assessments are in agreement and serve in conjunction with one another to guide the system toward students learning what they are expected to know and do" (2002, p2). In addition, he identifies four levels of cognitive demand to assess DOK consistency i.e. 1) Recall: the recollection of facts and-or information which requires a single step procedure; 2) Skill or Concept: use of information or conceptual knowledge to approach a problem in a fashion requiring two or more cognitive steps; 3) Strategic thinking: requires reasoning and the development of a plan or sequence of steps to use concepts in 2. in the solution of non-routine problems; and 4) Extended thinking: requires an investigation as well as time to process the multiple conditions of the problems being investigated where several connections in 2. and 3. are applied to solve a problem. In a recent journal, Webb (2007) has identified five issues that need to be addressed to reduce the subjectivity of alignment judgements: (these issues can be equally applied to Biggs' (2003) and Anderson's (et al., 2001) frameworks)

i. Acceptable level of categories for each subject;

ii. Different ways of considering what is an acceptable distribution of complexity in the DOK or cognitive demands of the assessment;

iii. The number and range of content that should be assessed or covered in one assessment.

iv. Finding the balance of representation of the objectives in the assessment

v. Accounting for changes in cognitive demands or DOK over the number of years of study

The three alignment frameworks discussed in this section underline the significance of finding correspondence between curriculum objectives and assessments to the knowledge and understanding of the learner which is acquired through the teaching-learning activities. Popham (2006) contends that even though 'alignment is a concept that is viewed differently by most of today's educators' (p.15:11), it is still a 'significant

factor in evaluating the quality of a test'; and thus in judging the validity of any inferences based on the application of the assessments.

In a competence based environment (as that of computing skills) alignment is relevant to raising students' proficiency (Keen, 1992). Pea (et al., 1983) raises the concern that there is no single evaluative scheme for assessing programming competency in computer programming due the wide and varied contexts in computer programming. The criteria and metrics will be different for different kinds of domain e.g. a games program needs to be evaluated on its compactness and speed, a business software has to be user-friendly and provides functional processes, a scientific program has to be fast, provable and accurate and a video software has to be able to pack the most functions in the smallest spaces and so forth.

2.7 Challenges posed by integrating curriculum, pedagogy and assessment in the teaching-learning of computer programming

The discussion thus far has shown how educators and researchers alike have worked and are still trying very hard to underpin the problems faced by first-time programmers and to propose learning and instructional strategies to enable students to master the skill and competence in computer programming. As discussed by Pea (et al., 1983), computer programming requires skills in analogical reasoning e.g. comparing and substituting, conditional thinking (e.g. repetitive loops, if-else), procedural thinking (e.g. building a model, following a map) and deductive/logical reasoning (cause-effect). Teaching-learning computer programming has to consider 3 main cognitive principles:

- Syntactics – the vocabulary of the language to represent variables, relations
- Semantics – the relations of the expressions of the syntax
- Pragmatics – the constraints in the user interaction with the language eg. The development environment, the debugging facilities, and the environment.

Furthermore, industry practice of IT software development involves team development in the design, coding and evaluation of computer programs (McGill, 2003). Advantages of teamwork remove the tyrannies and egocentricities of the individual and promote cooperative and collaborative learning. The nature of computer programming is evolutionary (Yelland, 2007); new goals emerge in tandem with new purposes for which programming activities are recognised as relevant. For example, web pages are not the sole proprietary of web designers. Action scripts allow simple yet powerful manipulation and redirection of web pages. With specialised formats such as XML tagging, data can be transferred across web pages without the need of databases. Similarly, spreadsheets and word processing documents incorporate programmable

instructions as well as other embedded applications such as statistical analysis, to run together in them.

Clements (2000) has debated on the complex questions of teaching computer programming as a discipline and teaching with computer programming as a means of reaching other goals e.g. mathematical achievement or language efficiency. He argues for the need for educators to take up the programming challenge so that they can learn first-hand the affects and effects of technology on their charges. Otherwise, educators will opt to produce and use mathematically simple but media enhanced solutions rather than mathematically richer programming environments. Thus, the next relevant issue is what students can do and learn after computer programming projects throughout their educational career.

These are the same challenges that this research aims to investigate. The main issues raised are not so much what to teach in terms of content but more of how to blend the content with the concept using a continuous, iterative dialogue (Laurillard, 2002, Hatch and Burd, 2006, and Heinze et al., 2006) within a learning environment that promotes incremental assessment and feedback (Bates, 2005). Constructivist approaches are well supported through e-learning (Mason et al., 2006) and in computing practice (Mead et al., 2006); hence, instructional strategies that employ constructivism are well-placed to promote the teaching and learning of computer programming.

Various perspectives on the disposition and purpose of education technology have been discussed as well as models for integrating curriculum, pedagogy and assessment in the teaching and learning of computer programming. It seems that computer programming is best taught using educational technology (McAllister and Alexander, 2003) rather than the old school chalk-and-board. Even with the help of ICT, the problems of the learner are not well documented and claims of success with the learning models are not easily validated.

The literature reviewed in this chapter has shown the significance of different aspects in the curriculum for teaching computer programming and the need for an integrated or blended approach. Although the learning contexts among the institutions reviewed were different, the opportunities and obstacles encountered provide common ground from which to understand and improve the education of computer programming. This research aims to contribute to this growing field of research in order to assist first-time computer programmers to grasp the essential concepts in computer programming.

Different dimensions of integration reviewed in this chapter form the basis of the investigation in chapters 4 and 5. The main focus is the blended learning framework that is the integration of different learning pedagogies: face-to-face interaction,

computing practice and e-learning. Next is the use of an integrated development environment for computer programming with e-learning support. Further, the teaching, learning and assessment systems are integrated into a constructive alignment model. In addition, various learning strategies i.e. collaborative learning, cooperative learning and problem-based learning are integrated into a formative performance based assessment. As discussed earlier in this chapter, one of the major challenges in the successful application of an integrated strategy is that it demands collaboration among different disciplines, flexibility in terms of implementation and richness in delivery.

Chapter 3 Research design & methodology

Crafting a case study research design that is sound and valid is one of the challenging tasks of this study. Boundaries on the data and methods that are applied in this study are discussed in this chapter starting with the context on which this study is focused (section 3.1), the research design that employs the comparative case study framework (section 3.2) and the research methods applied (sections 3.3 and 3.4). The research aims, objectives and questions stated in section 1.5 are revisited in section 3.5 to give direction to the analysis and in generating specific areas of investigation.

3.1 Context of the study

In chapter 1, this report has discussed the need and justification for the change in curriculum for the PrC module. Curriculum changes in the EI diploma are based on technology updates, ministry and industry needs and resource constraints (refer to section 1.2). The PrC module since its inception in 2003 up till 2005 has followed a structured, traditional approach. In 2006, a major revamp has been implemented which featured an integrated blended learning approach in the PrC course delivery. It is the intent of this research to examine the changes in order to reveal the implications to the teaching, learning and understanding of computer programming to first-time computer programmers who forms a large body of the PrC cohort.

In the first semester of the EI diploma, students take a total of 6 core modules one of which is the PrC module (refer to Appendix A for course structure). The students are divided into two study paths after which they are placed into module groups comprising at most 24 students. This means students from 6 module groups in study path A have the same set of 5 other core modules and students from the 5 other module groups in study path B have a different set of 5 other core modules. In the second semester, the students will swop the 5 core modules between the study paths except for the programming module which is extended into the Data Structures and Algorithms module. This study covers only the first semester of the EI diploma where the PrC module is being taught. Impact of the other core modules are not discussed as their varied curricula is beyond the scope of this research.

Data was collected for two semesters of the Principles of Computing (PrC) module, namely semester 2005-S1 which ran from May to September 2005 and semester 2006-S1 which ran from April to August 2006. Each period delineates a case study and there are similarities and differences between the two cases in terms of curriculum, teaching-learning activities and assessments. The main context of each case is illustrated as follows:

Case:	2005-S1	2006-S1
Duration per week	2-hr lecture, 3-hr practical and 1-hr tutorial	1-hr e-lecture, 3-hr practical
Lecture	Big group lecture (5/6 groups) in lecture theatre, after an hour students break out for e-learning on their own.	2/3 groups in customized computer laboratories with blended learning
Practical class	1 group of 20-24 students	Same but with blended learning
Tutorial class	1 group of 20-24 students using written assignments	None, students to do self-paced e-learning
Assessment	Individual practical test 20% Individual written test 40% Team project work 30% Class tutorials 10%	Individual practical test 50% Individual online assessment 20% Team project work 30%

Case study 2005-S1 applies e-learning as a self-paced mode within the lecture hours whereas case-study 2006-S1 employs a blended e-learning strategy where online learning is enhanced with face-to-face interactions. It is the intent of this research to compare the two case studies and to identify the gaps with respect to the curriculum, the learning environment and the students' performance in the PrC module.

Synopsis of cases

1. Students are accepted into the EI diploma based on the Singapore-GCE 'O' levels aggregate for five related subjects where the aggregate points are used as a measure of students' entry-level aptitude. The maximum cut-off aggregate points is 26. Those students who are accepted based on other certificates have attained higher achievement over and above the Singapore-GCE 'O' levels.
2. Each module group comprises at most 24 students and each module group attends the same lectures, practicals and tutorials. There are 11 module groups in 2005-S1 as well as in 2006-S1 student cohorts.
3. For students in 2005-S1 cohort: students attend weekly classes comprising 2-hour lecture, 3-hour practical and 1-hr tutorial. In the 2-hour lecture, one hour is delivered in the lecture theatre comprising 5-6 module groups and another hour for e-learning where students go to designated e-learning stations. Practical sessions are conducted in the computing laboratories and tutorial sessions are conducted in rooms without computing facilities for written assignments. Total hours covered is 90 hours in 15 study weeks.
4. For students in 2006-S1 cohort: weekly classes are reduced to an hour e-lecture comprising 2-3 module groups and a 3-hour practical session. All

assignments are submitted online. Total hours covered in 15 study weeks is 60 hours. Students are allowed to collaborate through online environments.

5. Content-wise: students in 2005-S1 are learning C++ with Microsoft Visual Studio 6 development environment whereas 2006-S1 students are learning C# with the integrated development environment from Microsoft Visual Studio 2005.
6. Assessments for students in 2005-S1 comprise individual practical tests (20%), written closed test (40%), written assignments (10%) and a simplified system submitted as project work (30%).
7. Assessments for students in 2006-S1 comprise individual practical tests (50%), online assignments and quizzes (20%) and a simplified system submitted as project work (30%). For students in 2006S1, all assessments are submitted electronically.

Cohort Composition

Case:	2005-S1	%	2006-S1	%
Total no. of students	232	100	247	100
No of males to females	142:90	61:39	139:108	56:44
Prior computing knowledge	31	13.3	29	11.7
GCE 'O' levels holders	201	86.7	218	88.3
GCE 'O' levels <= 20 points	37	16.0	30	12.2
GCE 'O' levels > 20 points	164	70.7	188	76.1

Table 3.1 Cohort composition for case 2005-S1 and case 2006-S1

The focus of this study involves students who have completed at least 10 years of study in primary and secondary schools. Students in secondary schools in Singapore spend about 30% of the curriculum time using computers (MOE, 2007a). These students are familiar with computers as users namely to surf the Internet for information, play computer games and send electronic mails or short messages in online 'chats'. As seen from table 3.1, the students who enrolled for the EI diploma composed of more than 86% Singapore-Cambridge GCE 'O' level school-leavers, whilst the rest may have computer programming experience prior to joining the EI diploma course. The 20-point GCE 'O' level aggregate denotes the cut-off for acceptance into the advanced GCE 'A' levels course; students who score 20 points or less and have chosen to enroll in the diploma course have a higher aptitude compared to those who scored above 20 points. This study will analyse the impact and differences of these students with respect to their performance in the PrC module.

3.2 Research design: Comparative case study in retrospective reconstruction

Based on the context of the preceding section, this research lends itself towards a comparative study of two cases where retrospective reconstruction is employed to derive the pros (what works) and the cons (what issues or challenges) of an integrated curriculum for first-time computer programmers in the Principles of Computing (PrC) module. Evidently, 2006-S1 represents the integrated curriculum case study whereas 2005-S1 is the traditional taught case study. By reconstructing the two cases in terms of the curriculum's learning objectives of computer programming, this research aims to raise the awareness to the implications faced by novice computer programmers. As supported by Yin (2003), by involving two cases, the research is able to contrast strategies for educational accountability which in turn 'represent a strong start towards theoretical replication' (p.54) and subsequently 'vastly strengthening the external validity' of the research findings.

Justifying the case study approach

Hammersley (et al., 2000) justly observes that:

> *In one sense, all research is case study: there is always some unit, or set of units, in relation to which data are collected and/or analysed. (p. 2)*

The qualitative mode of inquiry in a case study is not fixed or preset by the researcher and the information gathered and analysed covers a large number of features of each case. The main concern of the research is to explore the case in order to understand the 'hows' and 'whys' (Yin, 2003). As highlighted by Cohen (et al., 2007), case studies record effects and events in real contexts 'recognising that context is a powerful determinant of both causes and effects' (p.253). Quoting Hitchcock and Hughes (1995:322, p. 253), Cohen lists the hallmarks of case study as follows:

- ⌘ It is concerned with a rich and vivid description of events relevant to the case.
- ⌘ It provides a chronological narrative of events relevant to the case.
- ⌘ It blends a description of events with the analysis of them.
- ⌘ It focuses on individual actors or groups of actors, and seeks to understand their perceptions of events.
- ⌘ The researcher is integrally involved in the case.
- ⌘ An attempt is made to portray the richness of the case in writing up the report.

Those are the same characteristics apparent in this research and the research questions posed in this study aim to uncover the same purposes and intents.

Current discourse has raised several issues associated with the case study approach broadly categorised as follows (Hammersley et al., 2000):

Generalisability: case study being a recording of specific events cannot be generalised to a common theory or conclusion as those available in scientific experiments or statistical surveys.

Causal or narrative analysis: there is a lack of methodological rigour especially for single cases to deduce the causes and to determine the relationships that are contingent from the necessary.

Nature of theory: case study fails to embody a theoretical framework to give credence to its findings. As such, its findings are biased and cannot be deemed sound and consistent.

Authenticity and authority: a case unlike an experiment cannot be replicated and this bias also rejects any claims to authority. Claims to the uniqueness of a case and its representation of the unknown or neglected exacerbate the problem further.

In defence of case study research and to challenge the above issues, Flyvbjerg (2006) has raised the following counter arguments:

Issues	Resolution
Case study fails to provide general, theoretical (context-independent) knowledge in favour of concrete, practical (context-dependent) knowledge	*Predictive theories and universals cannot be found in the study of human affairs. Concrete, context-dependent knowledge is, therefore, more valuable than the vain search for predictive theories and universals.*
One cannot generalise on the basis of a single case and that the case study cannot contribute to scientific development	*One can often generalise on the basis of a single case, and the case study may be central to scientific development via generalisation as supplement or alternative to other methods. But formal generalisation is overvalued as a source of scientific development, whereas "the force of example" is underestimated.*
Case study method is claimed to be most useful for generating hypotheses in the first steps of a total research process, whereas hypothesis testing and theory building are best carried out by other methods later in the process.	*The case study is useful for both generating and testing of hypotheses but is not limited to these research activities alone.*
The case study contains a bias toward verification, understood as a tendency to confirm the researcher's preconceived ideas	*The case study contains no greater bias toward verification of the researcher's preconceived notions than other methods of inquiry. On the contrary, experience indicates that the case study contains a greater bias toward falsification of preconceived notions than toward verification.*
It is often difficult to summarise specific case studies into general propositions and theories,	*The problems in summarising case studies, however, are due more often to the properties of the reality studied than to the case study as a research method. Often it is not desirable to summarize and generalise case studies. Good studies should be read as narratives in their entirety.*

Maintaining research integrity

To remove the criticisms associated with case study research, researchers (Cohen et al., 2007, Yin, 2003, Gomm et al., 2000, Crossley et al., 1984) have recommended several approaches to test the validity and reliability of case study research, namely:

- Construct Validity: establish adequate operational measures for the concepts being studied; select specific changes that can be measured or verified through multiple sources, through a chain of evidence and/or through reviews from key informants.
- Ecological Validity: observe the behaviour and interactions of the human groups in their social and physical settings. By giving details of activities, the case study methods are able to identify important constraints and gaps within the empirical context.
- Internal Validity: identify causal relationships among variables and/or make inferences based on evidence collected. Analytic induction such as pattern matching, cross-case explanations, expert/peer review are recommended to provide the triangulation of evidence or data collected.
- External Validity: how transferable are the findings outside of the case being studied. Successful replication requires analytical generalisation from multiple case studies.
- Reliability: verify the findings to remove or to minimise errors and biases, and to justify the findings as accurate as possible. Reliability for case study research requires structured documentation where operational details are accounted for and can be easily audited.

The above tests are never done separately or individually but rather tend to overlap where one validity test supports the other to form a cohesive integrity of the research. For this research, two past cases are compared to reconstruct the teaching-learning activities in context supporting the ecological validity; students' performance during the assessments of these cases are measured and compared to verify evidence of competence in computer programming skills to support the construct validity; teaching-learning activities between the two cases as well as experts' review are included to provide triangulation as well as to strengthen the internal validity. Throughout the investigation of the two case studies, eliminative and analytic inductions are employed in the research design to strengthen the internal and external validities and its subsequent detailed documentation to improve the reliability of the research findings.

As explained by Hammersley (et al., 2000), comparative case study methods are validated through the twin processes of eliminative induction and analytic induction. Eliminative induction as prescribed by Mill (1974 as discussed in Hammersley et al.,

2000) is a method of agreement and difference: Examine cases to identify factors which always occur when a particular result is observed. This allows the researcher to search for necessary conditions of agreement and to reveal differences between two cases.

Cressey (1950, 1953, as discussed in Robinson, 2000) underlines his analytic induction method of theory development as follows:

1. Formulate a rough definition of the phenomenon to be explained.
2. Formulate a hypothetical explanation of the phenomenon.
3. Study one case to see if the hypothesis fits the facts of the case.
4. If not, either reformulate the hypothesis or redefine the phenomenon more precisely so as to exclude the facts of the case that defy explanations. The working hypothesis is maintained to enable new facts to fall within the case.
5. Practical certainty may be attained after a small number of cases are examined, but a single negative case requires a reformulation of the hypothesis or redefinition of the phenomenon.
6. The procedure continues until a universal relationship is established.
7. Finally, cases outside the area circumscribed by the definition are examined to determine whether or not the final hypothesis applies to them through eliminative induction.

The above methods require a step-by-step consideration of 2005-S1 and 2006-S1 cases and the building and testing of propositions as the cases proceed. This iterative process adds to the internal validity of this qualitative study without reducing its strength in external validity. The benefit of employing eliminative and analytic inductions for comparative case study is that the research leans towards a pragmatic goal-free evaluation (Patton, 2002). Goal-free evaluation allows the researcher to describe each case holistically in depth, in detail and in context with respect to the observed changes or situation. This alone forms a powerful argument for including an inductive approach in the data analysis.

The role of the researcher as key observer

Qualitative research is an inquiry process that occurs in the natural setting where the researcher is an instrument of data collection that explores a social or human problem (Creswell, 2003). The case study researcher uses multiple forms of data rich in context to build the in-depth case. A case study method is used when the researcher deliberately wants to cover contextual conditions that might be highly pertinent to the phenomenon of study (Yin, 2003). Creswell (2003) further identifies the role of the researcher as 'participatory and self-reflective' which are central to all qualitative studies with the following characteristics (summarised from pages 181-183):

i. the researcher has a close relationship with the natural setting where the research site is. Often the researcher is involved with the participants of the research and is part of the data collection process.

ii. qualitative research being fundamentally interpretive, implies that the researcher has to make a personal interpretation of the data and information being analysed.

iii. the value-laden aspect of qualitative inquiry allows the researcher to view social phenomena holistically; to build visual models of a process or phenomenon to appear as broad, panoramic views rather than micro-analyses.

iv. the researcher has to acknowledge his personal biography and how it shapes the study such that the personal-self becomes inseparable from the researcher-self.

v. the researcher uses complex reasoning that is multi-faceted, iterative and simultaneous throughout the research project where inductive reasoning is mainly employed alongside deductive processes.

The above characteristics imply that the qualitative researcher adopts one or more strategies of inquiry as a guide for the procedures used in the research.

This researcher has been attached to the EI diploma since October 2002 and has been teaching the PrC module since 2003. In 2006, this researcher is the module leader or convener for PrC which has led to the implementation of the integrated curriculum with a blended learning framework. Having been a tutor for the PrC module, this researcher is able to corroborate the findings of this research and to lend expert experience to the observations made. In addition, this researcher is able to justify and to explain the strategies behind the teaching-learning activities implemented in the integrated blended learning framework. In order to remove personal bias or prejudgement of the researcher (Cohen et al., 2007), findings from two other studies on the PrC module are being included. One is based on a six-sigma project undertaken in EI (NYP, 2005b) and another is a work improvement team project in EI (NYP, 2005a). The information from these two studies will serve as a triangulation method to verify the internal validity of this research.

Participant observation as performed in this study where the observer engages in the same activities being observed relies on the natural processes as they happen. Conversely in non-participant observation, the observer does not have any involvement with the activities being observed and stand aloof from the human representatives. Since this is a retrospective study, there is no possibility of manipulating the variables

and the unstructured, empirical evidence lends credence to the findings. Bailey (1994) has discovered the following benefits from participant observation:

- Evidence is superior to experiments and surveys when data are being collected on non-verbal behaviour.
- Investigators are able to discern ongoing behaviour as it occurs and are able to make appropriate notes about its salient features.
- Because case study observations take place over extended period of time, researchers can develop more intimate and informal relationships with those they are observing, in natural environments compared to the artificial settings of experiments and surveys.
- Case study observations are less reactive than other types of data gathering methods as data bias cannot be introduced in the very data that researchers are attempting to study unlike interviews and structured experiments.

In a retrospective study as in this study where innovative teaching-learning strategies are being compared over the traditional methods, the investigator provides evidence for any number of different hypotheses and has the flexibility to look for an interpretation consistent with the data. On the other hand, this very flexibility over the analysis is under debate especially on the reliability of the findings. Cohen (et al., 2007) has identified the following limitations arising from a retrospective or ex post facto study:

- There is a lack of control to manipulate the independent variable or to randomise the subjects. Hence, the investigator has to give directions on how the hypotheses can be subsequently tested by experimental or survey methods.
- Not able to ascertain whether the causative factor has been included or even identified. It may be the case that no single factor is the cause as a particular outcome may result from different causes on different occasions.
- When a relationship between the cause and the effect is determined, the possibility of its reverse must be considered. Similarly, the relationship between two factors does not establish cause and effect. Just because X precedes Y does not imply that X causes Y.
- Difficult to interpret or to single out cause and effect as events have multiple rather than single causes. Hence, the investigator has to introduce some measure of control in their investigation such as including statistical analysis of variance over the qualitative classifications.

The above limitations can be seen as risks that can be reduced during the analysis as long as the researcher is able to provide detailed operational measures in the study, much like the issues raised with case study design.

3.3 Research methods for qualitative analysis

Qualitative analysis can be regarded as three concurrent processes: continual reflection, data reduction and display, and conclusion drawing (Creswell, 2003). Continual reflection requires asking analytic questions about the data. Data reduction refers to the process of selecting, focusing, simplifying, abstracting, and transforming the data that appear in written-up field notes or transcriptions. Data display is an organized, compressed assembly of information that permits conclusion drawing and action. Conclusion drawing is deciding what facts and inferences mean, noting regularities, patterns, explanations, possible configurations, causal flows, and propositions. Conclusions are verified as analysis proceeds. Meanings emerging from the data have to be tested for plausibility, sturdiness, and confirming hypotheses – which leads to validity.

In terms of the qualitative comparative case study design of this research, analytic induction and participant observation are the twin methods applied to analyse the cases. These methods as discussed in the previous section allow the formulation of hypothesis and its subsequent confirmation or reformulation. In terms of documents, this research utilises the content analysis method to analyse the curriculum documents and study materials in conjunction with participant observation. Content analysis is a research method applied to written or visual materials for the purpose of identifying specified characteristics of the material (Fraenkel and Wallen, 2006). Content analysis is performed at two levels: the first level is a descriptive account of classification where units of analysis are defined followed by data coding and categorisation; the second level is an interpretive account where statistical tools are used to summarise frequencies, trends, correlations and so forth. Cohen (et al., 2007) argues for a more flexible approach to content analysis that is not fixed in quantitative analysis and theory generation; one that permits analytical induction and hypothesis confirmation. A qualitative content analysis method comprises the following stages (Cohen et al., 2007, p. 483-487):

1. Extract the interpretive comments that have been written on the data
2. Sort the data into key headings or areas
3. List the topics within each key area and put frequencies in which items are mentioned
4. Go through the list from 3. and put the issues into groups, avoiding category overlap
5. Comment on the groups or results from 4. and review their messages

Cohen (et al., 2007) contends that what is significant to the content analysis is that the researcher has to examine within and across categories for patterns, themes as well as exceptions; to decide which issue or concept to investigate or to discard; to report

evidence that confirms or rejects statements; and to account for the relationships and implications. Content analysis in this qualitative approach is suitable for this case study research. This researcher acts as the expert reviewer to categorise and classify different task areas of the curriculum. Patterns of interaction between different areas are compared and contrasted within the case and across the two cases. The evidence is subsequently applied to the hypothesis to affirm or to reformulate in view of further evidence.

3.4 Research methods for statistical analysis

Statistical methods are employed in qualitative case study research designs to add descriptive insight to explain causal complexity (Yin, 2003). Creswell (2003) explains that when qualitative and quantitative measures are integrated in the interpretation of research findings, the research design is said to apply one of the following approaches (p. 213-219):

Sequential – collection and analysis of quantitative and qualitative data are performed separately at different time period or phases, one after the other in the study.

- Explanatory Strategy: Quantitative analysis takes precedence over qualitative analysis. The qualitative analysis that follows is used to explain exceptional results from quantitative methods in more detail.
- Exploratory Strategy: Qualitative analysis takes precedence over quantitative analysis. Quantitative results are used to interpret qualitative findings to explore a phenomenon or emergent theory.
- Transformative Strategy: A theoretical perspective such as a conceptual framework or specific ideology, is used to guide the analysis using either quantitative or qualitative methods sequentially.

Concurrent – collection and analysis of quantitative and qualitative data are performed within the same phase of the study.

- Triangulation Strategy: Qualitative and quantitative measures are used to confirm, cross-validate or corroborate findings in a single study. The interpretation of the results either notes the convergence or explains divergence that has occurred, resolving discrepancies.
- Nested Strategy: Applies a lower priority method (quantitative or qualitative) embedded in a predominant method (qualitative or quantitative). The nesting is applied to address different question or level of inquiry. Conversely, it is used for mixed research methods e.g. an experimental research using case study method to evaluate quantitative and qualitative treatments.

Transformative Strategy: A specific theory is employed alongside the concurrent triangulation or nested strategies to define the problem, to identify the data sources and to analyse, interpret and report results throughout the research. The integration of the quantitative and qualitative data may occur during the analysis and-or the interpretation of the research.

In this study, the concurrent triangulation strategy is employed to add validity to the research findings. Both the qualitative data such as curriculum documents and teaching materials and the quantitative data such as students' module scores have been collected for each case during the same time period. The interpretation of the results are separately analysed in chapter 4 and 5 respectively and followed by a combined review towards the end of chapter 5. The inferences and implications are discussed as a whole in the concluding chapters 6 and 7.

According to Fraenkel (et al., 2006), statistics can be viewed in a descriptive or inferential manner. Descriptive statistics refer to grouped information of the data like the mean, the median, the standard deviation, the variance and so forth; allows graphical representation of values to reveal the spread and distribution. Inferential statistics derive or infer meaning from descriptive statistics such as significance or hypothesis testing, correlations, regressions and comparisons of means. Although descriptive statistics are useful, inferential statistics are powerful in giving meaning and thus more valuable to researchers.

In this study, descriptive statistics illustrate the cohort composition and student's performance in assessments. Inferential statistics allow further analysis on the differences revealed in the descriptive charts or tables such as the t-test for means and one-way analysis of variance (ANOVA). Since each case describes a different cohort of students, inferential statistics is performed for independent measurements. It is assumed that students' performance is normally distributed which enables parametric tests such as the independent t-test (comparison between two means). The level of significance, alpha (α), is 0.05 and the effect size for independent samples is based on Cohen's (1988) *d:* 0–0.20 = weak effect; 0.21–0.50 = modest effect; 0.51– 1.0 = moderate effect; greater than 1.0 = strong effect where effect size, $r = \sqrt{(t^2 / (t^2 + df))}$ where *t* is the t-test value and *df* is the degree of freedom derived from t-test.

Field (2005) claims that effect sizes provide an objective measure of the importance of an effect regardless of the significance of the test statistic. In addition, effect size allows inferences to be drawn on the likely effect in the entire population. Where the analysis of variance is concerned, the F-ratio is used to explain the variation between systematic to unsystematic variances (Field, 2005) and to see if this ratio is significance at $p < 0.05$. The software application employed in this study is the

Statistical Package for Social Science (SPSS). Tests for equal variances and homogeneity which confirm the normal distribution of the data, are derived from Levene's test (Field, 2005) that is available in the SPSS software.

3.5 Areas for investigation

There are three major areas of investigation in this study which correspond to the three research questions posed in section 1.5 i.e. the curriculum, the assessment and the students' performance.

Curriculum analysis

To give direction to the analysis of the curriculum, Posner (2004) proposes a comprehensive curriculum analysis framework as follows (p.19-22):

A. Curriculum Documentation and Origins
    i. How is the curriculum documented?
    ii. What situation resulted in the development of the curriculum?
    iii. What perspective if any, does the curriculum represent?
B. The Curriculum Proper
    i. What are the purposes and content of the curriculum?
    ii. What assumptions underlie the curriculum's approach to purpose or content?
    iii. How is the curriculum organised?
    iv. What assumptions underlie the curriculum's organisation?
C. The Curriculum in Use
    i. How should the curriculum be implemented?
    ii. What can you learn about the curriculum from an evaluation point of view?
D. Curriculum Critique
    i. What is your judgement about the curriculum?

Posner cautions that a complete and detailed analysis addressing the above issues is rarely achievable as most curriculum documents do not state theoretical and political commitments. However, for teachers and module conveners, a curriculum analysis is necessary to select and to adapt teaching-learning activities to fulfil curriculum objectives. He further likens curriculum analysis to detective work or literary analysis rather than clerical work or taking stock inventory; where inferences have to be made based on scattered evidence.

Accordingly, in this study, the curriculum analysis is carried out to investigate issues regarding its application to teaching and learning of computer programming to novice programmers. Thus, the issues raised in sets B and C of Posner's framework serve as probes in the analysis chapters 4 and 5. However the issues raised in all sets are discussed in chapters 6 and 7.

Q1: In what ways have the change in curriculum from a traditional structured approach to an integrated, blended learning framework influence the students' learning in computer programming?

In order to answer the first research question: The following hypotheses are generated:

H1: The change in curriculum from traditional structured approach to an integrated blended approach has improved the learning focus in computer programming.

H2: The change from traditional lecture to customised e-lectures has improved the learning focus in computer programming.

H3: The change from structured learning of C++ to blended learning of C# programming language has improved students' competence in computer programming.

H4: The teaching-learning activities found in the blended learning approach have improved students' learning focus in computer programming.

H5: The change from structured project work to problem-based project work has improved students' learning focus in computer programming.

These hypotheses cover various aspects of the curriculum and its delivery. The investigation aims to further the understanding of the computer programming constructs through new exploration and examination so as to evaluate the teaching-learning activities and to uncover effective teaching-learning strategies for first-time computer programmers. According to Posner (2004, p.261-263), an integrated based evaluation of a curriculum is focused on students' learning and has the following characteristics:

- Growth-Oriented: to strive for the growth and development of all students
- Student-Controlled: to give students a measure of control over their environment to increase the students' agency – to empower students to own the evaluation as a basis for self-improvement.
- Collaborative: to allow information to be shared between tutor and student and among peers from beginning to end. This way the distinction between evaluation and learning is seamless 'encouraging reflection, thinking and self-evaluation'.
- Dynamic: to measure students' progress through a continuous process of development, rather than on standardised tests.
- Contextualised: to make the learning realistic, with real-world examples so as to make it meaningful for students. In addition, students should be able to confront their weaknesses within the scope of the learning context in order to improve their competence.

- Informal: to enable teaching and evaluation to exist during the teaching-learning activities so that tutors are able to relate to individual students. This informal, one-to-one situation arises during or in close proximity to learning activities.
- Flexible and Action-oriented: to be experiential so that information gained during classroom interactions can be revised as and when 'teachable moments' arise.

While Posner (2004) asserts that 'few, if any, evaluations have all of the above characteristics', this study aims to examine how far the findings will match the above criteria. The innovative aspects of this comparative case study arise from an identified need for research on blended learning (Sharpe et al., 2006) to integrate different or varied learning experiences into an integrated curriculum for computer programming. It attempts to broaden the scope of online learning environments in order to incorporate tutor-facilitated instruction with various teaching-learning strategies. Another intent is to verify generalisations that have been made in relevant literature and to ascertain if they still hold true in blended learning environments. The design of this comparative case study research is not intended to provide inference of causality, nor to make broad generalisations applicable to other fields or domains of learning.

Assessment analysis

As discussed previously in section 2.6, assessments drive students' learning. This notion as Popham (2006) contends, treats assessments as 'curriculum clarifiers' – teachers should design assessments based on curriculum objectives and only then delve into the teaching-learning activities. This does not mean teaching to the items to be assessed but rather teaching towards the skills or knowledge identified in the curriculum; where the assessments represent those curriculum aims to be mastered by students. Similarly, in this study the assessment analysis seeks to evaluate the forms of assessments and the curriculum objectives supported by the assessments. This is outlined by the second research question as:

Q2: To what extent are the assessments affected by the traditional structured curriculum and the integrated, blended learning curriculum?

The above question examines the kinds of assessment undertaken by the students from each case and how these assessments influence the students' learning. The schedule for the assessments is as follows:

Case	2005-S1		2006-S1
Week 8	Practical test 1 (10%)	Week 6	Practical test 1 (25%)
Week 10	Project first draft (5%)		
Week 11	Practical test 2 (10%)	Week 11	Project first draft (5%)
Week 13	Written test (40%)	Week 12	Practical test 2 (25%)

Week 15	Project Presentation and submission (25%)	Week 15	Project Presentation and submission (25%)
Weeks 1 to 8, 10 to 12	Tutorial written assignments (10%)	Weeks 4 to 7, 11 to 16	Online assessments (20%)

By applying eliminative induction, it is apparent that the written test which constitutes 40% of the overall module assessment in case 2005-S1, is not present in case 2006-S1. The hypothesis to evaluate this difference is generated as follows:

H6: Closed individual written test is not aligned to the assessment of the students' learning focus in computer programming.

The notion of constructive alignment (as discussed in section 2.6) will be analysed in further detail through the next hypothesis:

H7: The blended learning approach is constructively aligned to the students' learning focus in computer programming.

The above hypotheses aim to bring together the curriculum and the assessment analysis into a holistic framework. In addition, the assessment analysis covers the higher level aim of this research i.e. to ascertain if an integrated blended learning environment engenders better quality learning from students and gives students more effective, meaningful feedback.

Performance analysis

As discussed by Biggs (2003), performance assessment 'requires students to perform tasks that mirror the objectives of the unit' (p. 184). Performance assessment should reflect the students' understanding and the assessment process should be formative and authentic of real-world situations. In a competence environment (as in a polytechnic), the student's performance has to be converted into a summative statement. Biggs recommends that the evaluation needs to distinguish between assessing the students' performance which is a qualitative continuous process and arriving at the results of that performance in a quantitative statement:

> *Quantifying performances that have been assessed holistically is simply an administrative device; there is no educational problem as long as it follows after the assessment process itself has been completed. (p. 200)*

In a computing environment, assessing students' computer programs and their project work is both formative and summative (McAllister and Alexander, 2003). When marking students' code for a practical test, marks are awarded not only for correct answers but also the structure and logic of the code; it is possible for a program code that generates

an incorrect answer to get a higher mark compared to an incorrect code that gives the right answer. Hence, the third research question duly states:

Q3: How have students performed in the computer programming module in the traditional structured environment compared to the integrated, blended learning environment?

Based on the above, the grades of the students' achievement in the PrC module are being statistically measured through the following hypotheses:

H8: The blended learning environment has increased students' module score in computer programming.

H9: The blended learning environment has improved students' project scores in computer programming.

H10: The blended learning environment has improved students' individual test scores in computer programming.

H11: There is no significant difference in the module performance across all module groups in each case.

Students' performance is relevant to this research as a measure of the students' competence in the PrC module and do not constitute towards any generalisation on computer programming scores. Statistical comparisons are made between the two cases on overall scores as well as detailed scores in individual tests and project work. The results will yield significant answers to the given hypotheses and will reveal any noteworthy similarities, exceptions or inconsistencies. The following variables which will be discussed in chapter 5 are described as follows:

- Entry-level aggregate points

  The entry-level aggregate points are recorded for students who are accepted into the EI course based on their Singapore-GCE 'O' level certificates. Other students who are accepted based on other certifications will be analysed as a separate group and given a score of 10 points which is the lowest point as these students have higher experience in IT. The entry-level aggregate is an independent variable of this study measured between 10 to 28 points. The lowest point shows the best aptitude entry-level student.

- Module score

  Given the research intent of this study to improve students' competence in computer programming, the module scores denote the final grades in the semester. This dependent variable is the end result of the semestral performance and a high score out of 100-point percentage indicates good performance. Other module

scores within the same semester are included as an observation of the students' performance.

- Project work score

  Students work in groups of two to produce a working solution based on a given problem. Students are allowed to discuss and to seek help among each other and those students who opt to work alone or in groups of threes are not distinguished separately in this study. The assessment for project work is open-ended and each group is assessed on the functionality and programming constructs included in the project. This variable constitutes 30% of the module score. Case study 2005-S1 requires a written report whereas case study 2006-S1 focused on students' concept learning and teamwork.

- Individual test score

  Individual tests are closed tests that relate the students' self-competence. In 2005-S1, individual tests made up 60% of the module score: 20% belonged to two practical tests and 40% made up a written test. In 2006-S1, individual tests made up 70% of the module score: 50% for two practical tests and 20% for incremental online assessments. This variable is a dependent variable of the module score and forms a major fraction of the module score. It will determine the students' individual achievement and understanding of the PrC module.

- *Module group*

  Students from each cohort are placed into module groups of 20-24 students. The course coordinator allocates students based on their entry level aggregate points and ensures that each group has a fair mix of students. However, there may be last minute changes from late registrations. The results of the module score across the groups are analysed to look for group performance and to see how much variation there is across groups compared to the overall cohort's scores in each case.

Overview diagram

As seen in figure 3.1, each research question has a set of hypotheses that covers different aspects of the research area. The sequence shows the process of evaluation rather than dependencies or relationships; H9 and H10 are shown as subcomponents of module score and are evaluated in the illustrated order. The relationships of the research questions and hypotheses is inferred after the analysis and discussed in chapter 6.

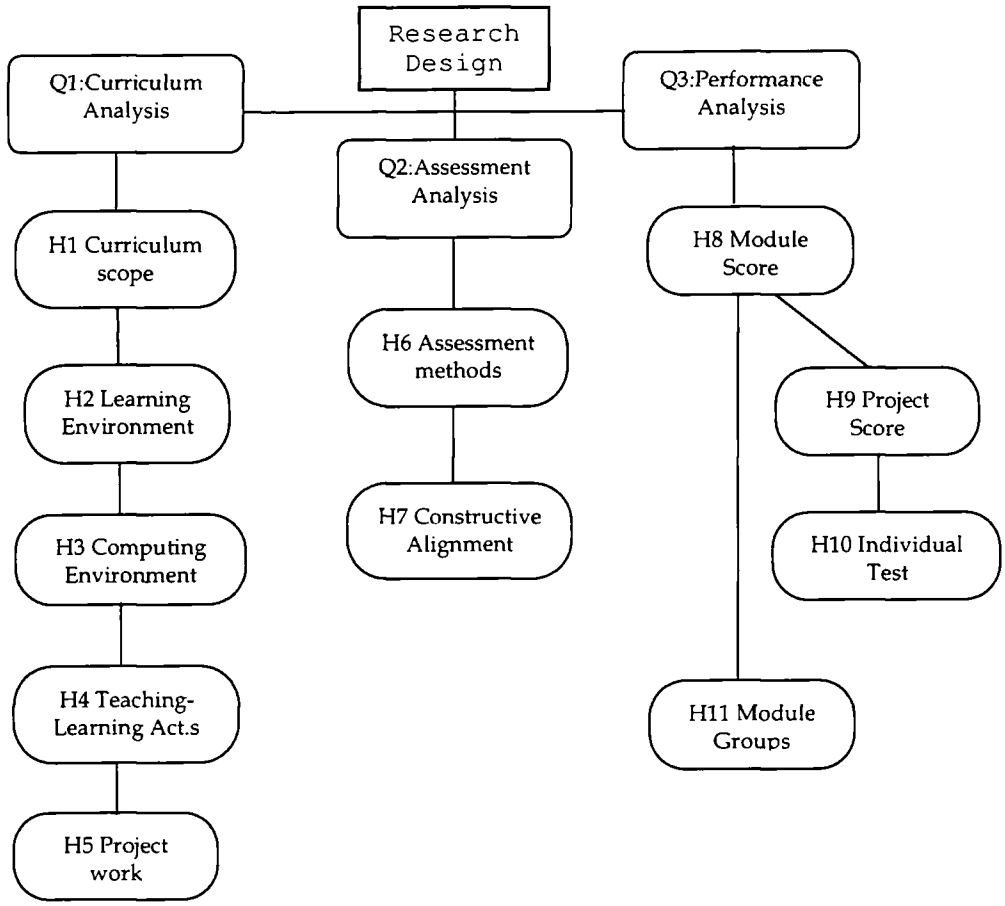


Figure 3.1 Comparative case study research design

Chapter 4 Findings I : Curriculum & assessment analysis

Delivering the findings of the qualitative analysis of this comparative case study research involves a combination of research methods i.e. analytic induction, content analysis and expert observation. In this chapter, the current study's empirical evidence on the two cases is examined comparatively in retrospective reconstruction. The evidence in each case is analysed qualitatively with the aim of determining the factors and attributes of the traditional versus the blended learning framework. Each assumption is presented along with the analysis to arrive at the justification to confirm the hypothesis or to reformulate the hypothesis based on naturalistic, contextualised evidence. Towards the end of the analysis, a constructive alignment framework is proposed which augments the validity of this comparative case study.

4.1 Reviewing the curriculum for Principles of Computing (PrC)

H1: The change in curriculum from traditional structured approach to an integrated blended approach has improved the learning focus in computer programming.

Content analysis of the curriculum documents is employed at the start of the analysis so as to evaluate the learning objectives of each case and their impacts on the teaching-learning activities. As discussed in chapter 3, the purpose of the empirical analysis is to compare the case studies and to explore the mechanisms that generate different learning approaches for each case. The main curriculum document is the module syllabus comprising the module aims and objectives and detailed topics. The module syllabus is drawn up by the module convener and supervisor and is approved through the course management process (refer to section 1.2). The curriculum aims and module syllabus for case 2005-S1 and 2006-S1 can be found in Appendix B.

The learning objectives for PrC module are shared in both cases, with the common aim to introduce students to computer programming fundamentals. The cognitive skills (derived from Pea et al., 1983) are inferred from the learning objectives giving rise to the following assumptions:

	Learning Objective	Cognitive Skill
1)	understand the concept of software systems	factual knowledge
2)	write algorithms to describe the logic of a computer program	design knowledge
3)	use variables to store and retrieve data in a computer program	input-output logic: analogical knowledge
4)	use decision-making and program flow control constructs to represent the logic of a program	if-else logic, repetition control logic: conditional knowledge

5) understand the concept of structured programming with functions and subroutines	procedural and causal logic: procedural knowledge, deductive knowledge

Factual knowledge is acquired through the understanding of concepts in the study material such as how the computer accepts and processes information. Design knowledge describes the processing steps required to find a solution with the help of input or output variables and messages. Analogical knowledge is concerned with the substitution of variables during the input, processing and output process; students need to know how different types of variables are applied and their interaction in the computer program. Conditional knowledge is gained when students are able to apply decision-making instructions using if-else constructs or repetitive loops to control the computer logic. Understanding structured programming implies that students are able to develop the sequential step-by-step instructions required to solve given problems. By applying subsets of code known as functions, subroutines or methods, students have gained procedural knowledge and by validating inputs and producing reliable outputs, students have demonstrated causal or deductive logic. Hence, a summary of the applied knowledge for the PrC module is determined as follows:

a) factual knowledge
b) design knowledge
c) analogical knowledge
d) conditional knowledge
e) procedural knowledge
f) deductive knowledge

Since the knowledge is acquired from the topics covered in the PrC module, a matrix can be drawn against the topics as illustrated in table 4.1.

	Knowledge to be gained from Learning Objectives					
Topics	Factual	Design	Analogical	Conditional	Procedural	Deductive
Computer Software Systems	√					
Computer Algorithms and Program Design	√	√				
An Introduction to C /C++ (or C#) Programming Language	√		√√			
Basic Data Types and Variables		√	√			
Computing Operators and Expressions		√	√√			
Program Flow Control and Decision making		√	√√	√√	√	
Functions (Methods in classes)		√	√	√	√√	√
Arrays		√	√√	√		

Table 4.1 Curriculum topics and the relevant knowledge required where each tick denotes significance of the knowledge

There is little difference between the topics in both cases; Other than the programming language specifics, which are C++ and C# respectively, the topics between the two cases are equivalent. A summary of the topics covered and durations is shown in table 4.2.

Topics	Case 2005-S1 Total Duration: 90 hours Lecture = 30, Practical = 45 Tutorial = 15	Lecture	Practical	Tutorial	Case 2006-S1 Total Duration: 60 hours Lecture = 15, Practical = 45	Lecture	Practical
1	Computer Software Systems	1	0	1	Computer Software Systems	1	0
2	Computer Algorithms and Program Design	5	9	2	Computer Algorithms and Program Design	1	1
3	An Introduction to C /C++ Programming Language	1	2	1	An Introduction to C# Programming Language	1	5
4	Basic Data Types and Variables	1	1	0	Basic Data Types and Variables	1	1
5	Computing Operators and Expressions	2	3	1	Computing Operators and Expressions	2	2
6	Program Flow Control and Decision making	10	15	5	Program Flow Control and Decision making	3	12
7	Functions	6	9	3	Class Methods	3	15
8	Arrays	4	6	2	Arrays	3	9

Table 4.2 Curriculum topics covered in the PrC module

The removal of the tutorial sessions from case 2006-S1 as well as the reduction of the lecture hours have caused a shift in the instruction focus of the topics in 2006-S1. Topic 2 in case 2005-S1 covers definition of a computing algorithm, flow-charting and pseudo-code. This topic is completed in the first three weeks of the semester. In case 2006-S1, the emphasis on flow-charting and pseudo-code is only factual; the focus is more on the computing algorithm technique as it is relevant to the rest of the topics. Moreover, the concepts learned in topic 2 are applied and reinforced in all subsequent topics and further in the project work. As reflected in table 4.1, design knowledge is applied through most of the topics.

For topic 6, case 2005-S1 covers all the forms of program flow control, whereas case 2006-S1 concentrates on basic if-else and while loops and covers switch-case in topic 7 and for loop in topic 8. This way the program flow control is context related. The curriculum hours for topics 7 and 8 remains the same in case 2006-S1 to indicate the significance of these topics to learning computing skills; whilst that for topic 3 is increased to include more pragmatic skills in debugging computer programs. Thus, case 2006-S1 has made up for the difference of 30 instruction hours by applying the learning objectives and the respective knowledge forms to the curriculum topics.

These findings have confirmed that the integrated curriculum of case 2006-S1 is more focused on the learning objectives of the curriculum (as reflected in table 4.1) and has made up for the reduction in curriculum hours this way. Hence by analytic induction, the hypothesis H1 is confirmed.

The next stage of the curriculum analysis reviews the differences in the curriculum. As seen in table 4.1, the curriculum duration for the semester is 90 hours in case 2005-S1 but is reduced to 60 hours in case 2006-S1 where the major reduction is the removal of tutorial session (15 hours) and the lecture hours being reduced to 15 hours. To evaluate the impact of these changes, the study investigates the teaching-learning environment and the spread of the curriculum time over a regular weekly period.

4.2 Inspecting the learning environment

H2: The change from traditional lecture to customised e-lectures has improved the learning focus in computer programming.

Past evidence of learning research (Ramsden, 2003, Fry et al., 2003) highlights the importance of physical environment and how it affects students' focus and understanding. Table 4.3 below lists the facilities used for the PrC module:

Facility	Capacity	Features
Lecture theatre	180 to 300 students	Fitted with lecturer's computer, large projector screen and whiteboard
e-Learning plaza	wide laboratory with at least 100 computers	Meant for individual self-paced learning. Student may request for head phones.
e-Lecture	Customized laboratory with 75 computers	Fitted with 2 tutor computers and attached projectors. Microphone + speakers included. Two whiteboards on wheels.
Practical	Laboratory with 25 computers	Whiteboard and tutor's computer has attached projector and microphone.
Tutorial	Classroom for 30 students without computer or projector	Whiteboard and standalone overhead projector available.

Table 4.3 Teaching facilities for PrC module

Weekly lessons for case 2005-S1 comprise an hour of lecture, e-learning and tutorial and three hours of practical computing; there is a minimum of fifteen study weeks in a semester and the lecture and e-learning hours make up the total lecture hours of thirty. For case 2006-S1, there is only an hour of e-lecture followed by three hours of practical computing. Table 4.4 below shows the sequence of weekly lessons for each case:

Teaching mode	**Time**	**Delivery**
Case 2006-S1	in hours	
1. Lecture	1	Lecturer goes through concepts in study materials for 120-144 students. May switch to programming mode to demonstrate computer programs.

Teaching mode	Time	Delivery
2. e-Learning	1	Tutor takes attendance and answers questions from 72-96 students. Student reviews online learning objects in a self-paced manner. Online quizzes allow student to gauge own understanding of concepts.
3. Practical	3	Tutor guides at most 24 students through worked examples. Students practice computing concepts in a programming development environment.
4. Tutorial	1	Tutor reviews answers to tutorial questions with at most 24 students.
Case 2006-S1		
1. e-Lecture	1	Lecturer goes through concepts in study material on one computer and e-Learning material on the other computer. May switch to programming mode on one of the computers. 48 to 72 students walk through the programming concept with lecturer and work through exercises on their computers.
2. Practical	3	Tutor goes through concepts similar to that of e-lecture with 24 students and spends time going through programming examples and problems in the study materials.

Table 4.4 Weekly distribution of curriculum time

Problems faced by case 2005-S1

Lecture:

Students are passive and are unable to appreciate demonstrations made by the lecturer. Tendency for students to talk, sleep, leave or play with their mobile phones. Lecturer wastes time trying to get every student's attention and participation.

e-Learning:

Many students go through the online material quickly and complete in fifteen minutes or less. Instead of finding out answers to online quizzes, students tend to copy from each other. Students surf other unrelated websites or chat online. Similar to the lecture, tutor has to prevent students from talking, leaving or playing with their mobile phones.

Tutorial:

Students are supposed to attempt the questions before the session. Instead many students come unprepared; students copy the answers from friends or simply wait for the tutor to give the answers.

Practical sessions take place in the same location for both cases. Hence, it is the e-lecture that replaces the lecture, e-learning and tutorial sessions. The e-learning materials for case 2006-S1 are more context-driven and practice-oriented with simulated video-clips of main exercises such as setting up the project environment and

debugging C# programs. Where relevant, tutorial questions from case 2005-S1 are included but formulated in an interactive quiz format.

The e-Lecture is delivered to a smaller group of students compared to the lecture and e-learning sessions but the problems still persist. There is not enough evidence to substantiate the H2 hypothesis. Seeking new evidence finds that the face-to-face interactions between tutor and students provided in case 2006-S1 is the same number of hours as that of case 2005-S1. In addition, case 2006-S1 is giving students more time for hands-on computing practice compared to case 2005-S1, where students spend 3 out of 6 hours a week on hands-on practice.

Per week	**Case 2005-S1**	**Case 2006-S1**
Face-to-face:	4 hours (practical + tutorial)	4 hours (e-lecture + practical)
Hands-on :	3 hours (practical)	4 hours (e-lecture + practical)

In view of the new evidence, the H2 hypothesis is reformulated as follows:

H2A: The reduction in curriculum time from 6 hours per week to 4 hours per week has increased hands-on practice for students and has not affected face-to-face interaction between students and tutors,

It can be concluded that the students in case 2006-S1 have not been adversely affected by the reduction in curriculum hours but have gained more hands-on practice as well.

Computer programming development environment

H3: The change from structured learning of C++ to blended learning of C# programming language has improved students' competence in computer programming.

C++ was the computer programming language for the PrC module from 2003 to 2005. In 2006, C# was nominated as the PrC's computer programming language to cater for the developments in Microsoft's .NET[1] environment. C++ is an object-oriented computer programming language developed by AT&T Bell Laboratories (Stroustrup, 1993) whereas C# is a component-oriented programming language developed by Microsoft (Gunnerson, 2005) as the premier language for its .NET platform. The programming development environment for C++ for case 2005-S1 was the Microsoft Visual Studio 6, whilst case 2006-S1 was upgraded to the rapid application development environment of Microsoft Visual Studio 2005. Table 5.2 shows the main features between the two environments and how these differences affect students' learning.

[1] .NET is the Microsoft Web services strategy to connect information, people, systems, and devices through software. Integrated across the Microsoft platform, the .NET technology enables businesses to build, deploy, manage, and use connected, secured solutions with Web services. (www.microsoft.com/net/basics.mspx)

Feature	Visual Studio 6 C++ (case 2005-S1)	Visual Studio 2005 C# (case 2006-S1)
Editing	No guidance	Code completion; colour coded objects and variables; spell-checking
Syntax checking	Only when code is compiled.	In edit mode, syntax prompts e.g. missing semicolon or unpaired parenthesis In build or execution mode, error messages appear.
Debugging	Error messages are not easy to understand e.g. students encounter "Fatal Error" for unpaired braces.	Error messages are clear with help guide.
	Limited debugging facilities	Able to debug in different modes.

Table 4.5 Comparison of features in Visual Studio 6 versus Visual Studio 2005

The evidence here indicates that the development environment for C++ is not helpful to novice programmers who are new to computer programming syntax. However, C++ is also available in Visual Studio 2005. Compared to Visual Studio 6, the Visual Studio 2005 development environment is more user-friendly with customizable tool windows and hide/view capabilities; it is easier for students to spot and correct errors and to obtain results.

Online learning support

Case 2005-S1 provides e-learning objects for every topic covered in the curriculum, as seen in figure 4.1.

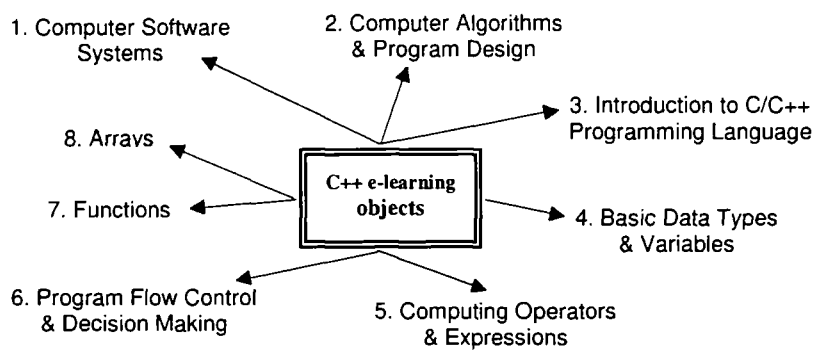


Figure 4.1 e-Learning objects organised according to topics

Case 2006-S1 improves on the e-learning materials as displayed in figure 4.2 where learning objects are organised in three areas. Besides topics, pragmatics includes simulated clips of starting and using the C# programming development, and debugging C# codes. Sample codes on text and numeric formatting, searching, date and time manipulation are meant to support student practice and projects.

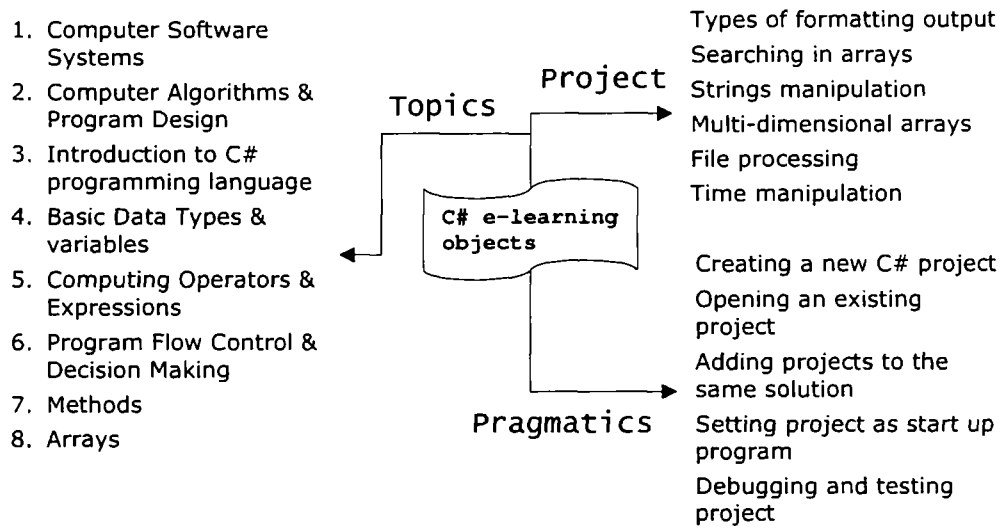


Figure 4.2 e-Learning objects organised according to topics, project and pragmatics

In view of the above improvements made to the programming development environment, H3 is reformulated to support the current evidence:

H3A: The changes in the programming development environment for case 2006-S1 have improved students' competence in computer programming.

4.3 Examining teaching-learning activities in PrC

The analysis at this stage returns to the main difference between the traditional structured approach and the blended learning approach. In order to make up for the reduction in curriculum time, the teaching-learning activities in case 2006-S1 have been revamped to the blended approach. The current hypothesis under review is:

H4: The teaching-learning activities found in the blended learning approach have improved students' learning focus in computer programming.

How are teaching-learning activities organised?

Case 2005-S1:

Lecture materials are organised to last for an hour, as such a lecture session may include more than one topic. Subsequent e-learning, practical and tutorial materials are organised to explain and support the concepts covered in the lecture. The student printed handbook is organised in the same manner, divided into three sections: lecture presentation slides, practical exercises and tutorial questions (see Appendix D for sample material). The thrusts of the structured approach:

- Lecture allows students to listen and to receive concepts;
- e-learning provides self-paced learning and exploration of concepts;
- Practicals provide students with hands-on computing practice;
- Tutorials enable students to work through concepts by thinking and reflecting.

Figure 4.3 depicts the flow of activities as conducted in each respective case.

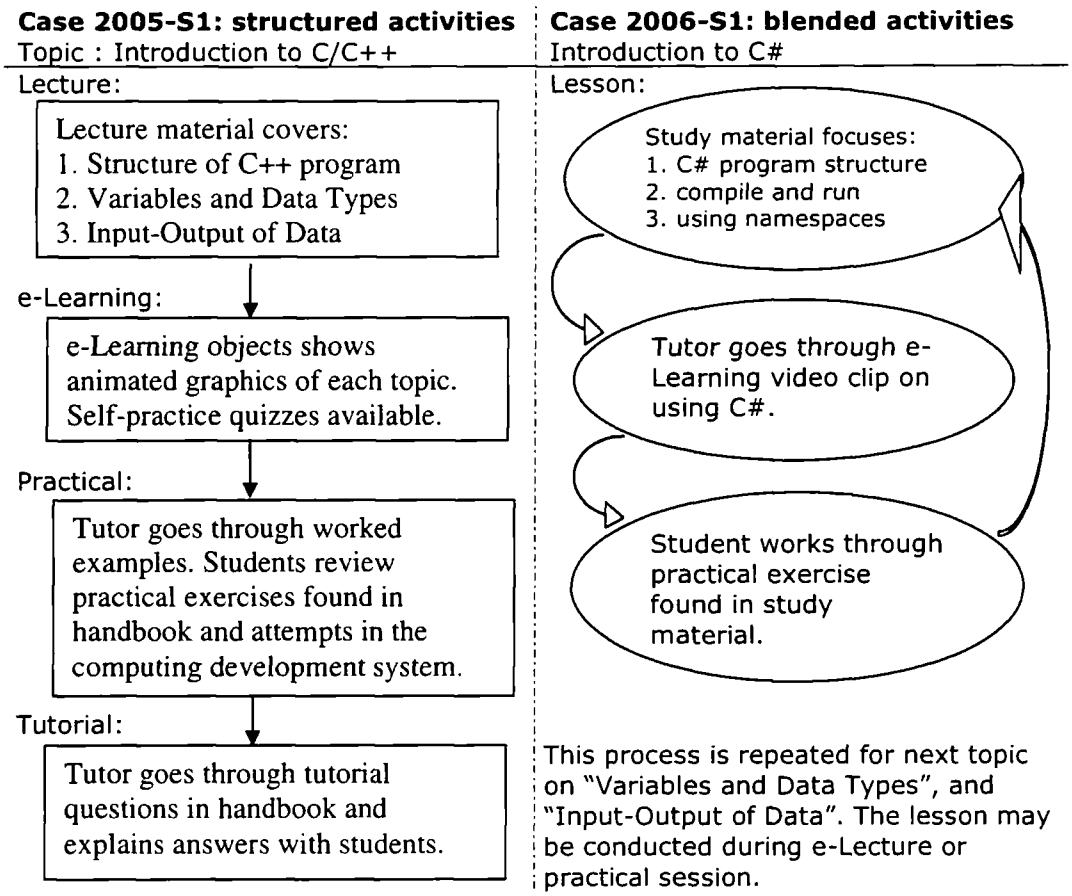


Figure 4.3 Teaching-learning activity structure (refer to Appendix D for study material)

The repetition of concepts engenders students to remember and assimilate learning and understanding of computing concepts. On the other hand, students may choose to attend selective sessions or may decide to do other things during the lesson.

The sequence of the activities is important as it builds the students' comprehension. However due to resource constraints, half of the module groups did not enjoy this nominated sequence. The worst situation is for the practical to begin ahead of the other sessions. Once the flow of the delivery is not in sequence, students experience disruption and this inevitably leads to confusion.

Since the lecture covers several concepts, it is not easy for students to relate theory to practice and students cannot infer which practice example or exercise is related to the concept behind it. The problem is compounded when students try to revise materials on their own.

Students in one module group have the same tutor for practical and tutorial sessions. However for lectures and e-learning, students may see a different lecturer or tutor. In the worst scenario, a module group has three different tutors. This may be good as students get a varied delivery but it could also be bad if the tutors' deliveries are diverse such that students become disoriented.

Case 2006-S1:

Study materials are organised around a main concept and categorized into weekly activities to help students to remember the chronological order. There is no printed handbook and students download online materials which they may choose to print according to own preferences. Online materials also allow students who own portable computers to work without cumbersome handbooks.

The blended learning approach integrates the theory, the e-learning and the practical around a main concept allowing students to relate and to understand the concept better and more naturally. The topic can be covered in the e-Lecture or the practical session; hence it does not matter if a practical session comes before e-lecture. It is easier for students to revise and to repeat the study materials on their own.

Students have the benefit of face-to-face interaction in both e-Lecture and practical sessions which allow them to clarify doubts immediately. Due to resource constraints, half of the module groups may not have the same tutor for e-Lecture and practical sessions. This may cause confusion to some students.

The evidence presented here highlights that the main benefit of the blended learning over the traditional structured learning is the context focus. Hence, the H4 hypothesis is refined as:

H4A: The teaching-learning activities found in the blended learning approach are context-driven and holistically integrated which improve students' learning focus.

How is project work carried out?

The aim of the project work is to allow students to integrate all the concepts covered in the module into a working solution. Project work starts in week 7 of the semester and students are to find partners to form a team to develop a working solution. Tutors will guide students in the practical sessions and each team has to present the solution towards the end of the semester in week 15. The current hypothesis under review is as follows:

H5: The change from structured project work to problem-based project work has improved students' learning focus in computer programming.

Case 2005-S1:

Module convener releases 4 different projects to all students. In each module group, students form teams of 2 or 3 persons and choose one of the 4 projects as illustrated in figure 4.4. The tutor decides on the number of teams allowed to work on a particular project. Specifications of each project and the allocation of marks to the project are made known to the students. The specifications serve as the

structured plan for students to develop their solutions; marks are likewise structured around the project functions (see Appendix E for project specification).

The tutor of the module group guides the students towards the completion of the project during the practical sessions. Tutor must ensure that the work developed by students are their own and not plagiarized. With 10 or 12 groups and 3 different projects, the tutor has to manage the project supervision over the normal coverage of the syllabus and the preparation of tests.

Students have to develop their project based on the given specifications and refer to program codes from the practical sessions. Students may consult tutors or their peers and have to struggle with errors in the program code as well as the program logic. As novice programmers, students have great difficulty finding the correct programming construct to implement in their solutions.

The projects are common across the cohort and students from different module groups tend to copy programs from one another. It is difficult for a tutor to detect copied solutions unless it comes from the module groups within the tutor's charge. Finally, in a team, one member may be doing all the work. Since the marks are structured, it does not recognize effort and team work into the assessment.

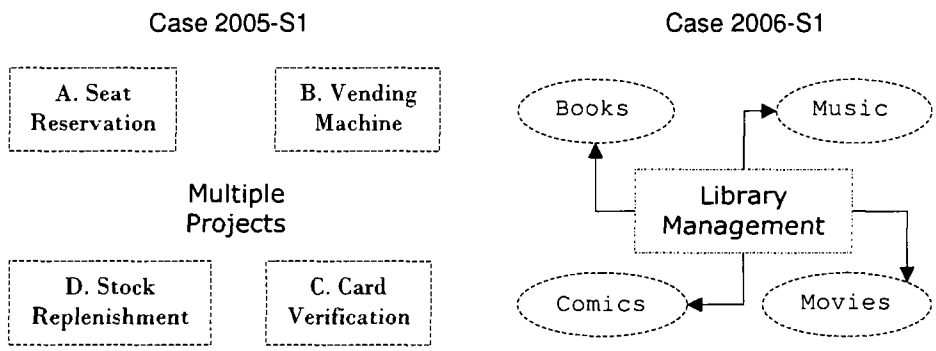


Figure 4.4 Projects for cases 2005-S1 versus 2006-S1

Case 2006-S1:

Each module group is assigned one project specification where each team is allowed to pick a domain or suggest its own (refer to figure 4.4). For example, the project specification is to build a library management system; domains can include a library of books, a library of rental comics, a library of music compilation or a library of movies and so forth. Using the problem-based approach, the specifications are open-ended and include a set of questions to guide the tutor and the students on the project development. Teams are limited to 2 students each and only one team will comprise one or three members; each domain can have at most 3 groups. Project criteria are formative in nature and projects are examined towards the last 3 weeks before submission for progressive development (see Appendix E for project criteria).

Tutor is able to guide the class as every team is working on the same problem and to explain programming constructs that are meaningful to the problem. The project questions promote collaborative and cooperative team development; marks are allocated for contribution and team work. Tutors find it easier to mark the projects as there is a main focus.

Students are motivated to collaborate across teams as they are solving a common problem. Within the team, students cooperate on the functionality of the solution knowing that their contributions are taken into account. Peer assessment is mainly formative in nature as marks are not allocated for it but students are encouraged to review each other's programs to learn from each other and to ensure that the programs are different and not a copy of the other.

Based on the evidence found, the hypothesis is reaffirmed that case 2006-S1 has provided a better project development approach over case 2005-S1.

4.4 Validating assessment

Assessments drive students' learning (as reviewed in section 2.6) and thus, have the important task of motivating students to study and to perform reasonably well to pass the module. The assessment components found in the PrC module is shown in table 4.6 below:

case	Assessment	Contribution	Type
2005-S1	Practical test	20%	Individual
	Written test	40%	Individual
	Project work	30%	Team
	Tutorial assignment	10%	Individual
2006-S1	Practical test	50%	Individual
	Project work	30%	Team
	Online quiz	20%	Individual

Table 4.6 Assessment components

Individual components make up 70% and team work contributes 30% in both cases. The highest individual component for case 2005-S1 is the written test (40%) whereas for case 2006-S1, the practical test takes precedence (50%). Based on the curriculum aim and objectives discussed in section 4.1, PrC module is a practice-oriented computing course; it follows that the next hypothesis to be reviewed is:

H6: Closed individual written test is not aligned to the assessment of the students' learning focus in computer programming.

Case 2005-S1:

Practical test and project work are solely practice-oriented assessments where students develop computer programs using the computing development

environment; practical test is a closed individual test but project work is a team effort. The written test is a closed individual test where a computing development environment is absent. Tutorial assignment is another written assessment but it is an open individual continuous assessment. It is clear from table 4.6, that case 2005-S1 has placed equal percentage (50%) on written work and practical work. In order to evaluate the alignment of the assessment with the curriculum, a matrix of the assessment type with the curriculum is drawn in table 4.7. It is noted that since the curriculum gives 50% of its delivery on practical classes that the same percentage of the assessments are practice-oriented. However, written test which is based on lecture materials contributes 40% to assessment but lectures cover only 16.6% of the curriculum. In addition, written test not being practice-oriented does not support the learning aims and objectives of the curriculum. Tutorial assignments are based on materials found in the e-learning and tutorial questions from the study handbook. Tutorial assignment is written work where submissions are mandatory and contribute 10% to the module assessment. In terms of curriculum coverage, the tutorial accounts for more than 33% i.e. 16.6% of lecture and 16.6% of e-learning materials. Hence, it appears that assessments in case 2005-S1 are not proportionately tied to the curriculum material nor does it fully support the curriculum's aims and learning objectives.

	Curriculum materials				
Case 2005-S1	Lecture	e-Learning	Practical	Tutorial	Total
Duration in hours	15	15	45	15	90
Duration in percentage	16.6	16.6	50.0	16.7	100
Written test	*		x		40
Project work			*		30
Practical test			*		20
Tutorial assignment		*		*	10

Table 4.7 Curriculum versus assessment components
* denotes significance and x denotes inapplicable

Case 2006-S1:

Similar to case 2005-S1, practical test and project work are practice oriented where the main difference is that practical test contributes 50% to the overall assessment. The online quiz is a continuous assessment and a closed individual test covering computing concepts found in the online study materials. As confirmed by earlier hypotheses, case 2006-S1's blended learning approach is 100% practice-oriented

and the computer development environment employed by case 2006-S1 supports the student learner. This means that during the closed individual test, students are able to resolve errors and find solutions to the test questions. Similarly, the blended learning environment supports students in their project work as well as continuous online assessments. It follows that the assessments found in case 2006-S1 fully supports the curriculum's aims and learning objectives.

More evidence can be found when the assessment schedule is presented to seek out how the assessments from each case support the learning of the students.

Examining assessment schedule

The semestral schedule for case 2005-S1 starts on May 30th 2005 whereas case 2006-S1 starts on April 17th 2006. The change is due to an organisational shift to bring forward the polytechnic's semestral date in line with those in junior colleges.

Semestral schedule for Prc module

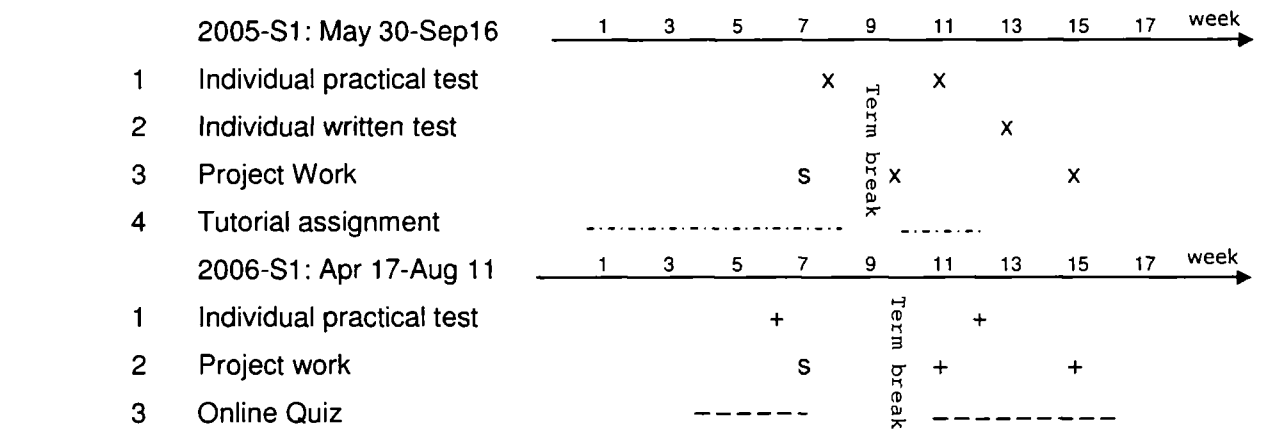


Figure 4.5 Assessment schedule: s marks start of project, x and + mark test date for each case

Case 2005-S1:

Term 1: May 30 – July 8, break: July 23 – 31, Term 2: Aug 1 – Sep 16 (wk16)

Semester term 1 is for 8 weeks followed by 1 week of term break. Term 2 is from week 10 to week 16.

Practical tests are conducted in weeks 8 and 11, each contributing 10% to the overall assessment. The written test comes at week 13. Not enough time to recover or learn from previous mistakes or to prepare for next tests.

Project starts in week 7 and first draft is submitted at the start of term 2 in week 10. Students are supposed to work on own time and to submit and present final project in week 15.

Lectures, e-learning and tutorials end after week 13 to allow students to work on their projects where tutorial assignments are submitted at the end of the tutorial class.

Problems: Students cannot start on their project as they are preparing for first practical test in week 8. This means students have less than 2 weeks before first submission due in week 10. Students have one week to prepare for second practical test in week 11. Following that students have the major written test in week 13. Students are not given enough time to review mistakes and prepare for tests. When the assessment schedule is taken into consideration (refer to figure 4.5), the written test is conducted as the last test which adds extra strain on the students' study load and stress.

Students could only start completing their project after week 13 as they are busy preparing for individual tests. Within two weeks students are not able to improve on the quality of the project solutions. Week 16 is reserved for students who fail their projects or could not submit their projects due to extenuating circumstances.

Case 2006-S1:

Term 1: April 17 – June 9, term break: June 10 – 25, Term 2: June 26 – Aug 11 (wk 17)

First practical test is in week 6 and second practical test is in week 12. There is a 6 week gap which allows tutors to conduct remedial lessons for weak students and for students to learn from their errors and to prepare for the next test.

Project work starts in week 7 and first submission is in week 11 after the term break. After the second practical test in week 12, students can focus on their project which is 3 weeks' before the final submission in week 15. Students whose project did not meet requirements have until week 17 to resubmit.

Online quiz starts in week 4 and continues every 2 weeks till week 16. The incremental continuous assessment is designed to give feedback to students on their understanding of computing concepts. It is a mixture of multiple-choice questions and short structured questions. It is conducted during the e-lecture sessions and is employed to encourage students to be attentive during the e-lecture group sessions.

The evidence analysed thus far indicates that the written test in case 2005-S1 does not contribute to the students' programming competence and has prevented students from putting more effort in improving their programming competence. In addition, the tutorial assignments too do not support the students' learning of computer programming. The hypothesis H6 is redefined to reflect the analysis as follows:

H6A: Closed individual written test and tutorial assignments are not aligned to the assessment of the students' learning focus in computer programming.

Hence, the removal of the closed written test and tutorial assignment from the assessment component is justified in case 2006-S1. By adding the online quiz, case 2006-S1 has provided additional student learning support and has incorporated an

assessment feedback to students. In order to show that case 2006-S1 has cultivated the assessment for learning quality through the blended learning approach and that the teaching-learning activities are constructively aligned, the next hypothesis is put forth:

H7: The blended learning approach is constructively aligned to the students' learning focus in computer programming.

4.5 Alignment framework for integrated curriculum

The constructive alignment model as developed by Biggs (2003) has been adapted in this research to evaluate the integrated curriculum in terms of Posner's (2004) characteristics (as given in chapter 3). As shown in figure 4.6, the curriculum characteristics become the main objectives to be met by the Teaching system, the Learning system and the Assessment system. The Teaching system defines the activities and these activities can be mapped and evaluated to one or more of the curriculum characteristics. Similarly, the Assessment system defines various tasks where students' performance is assessed. The Learning system specifies the students' learning outcomes to see if the learning outcomes map to the same characteristics defined by the Teaching system and the Assessment system. Judgement is based on the context of the activities, tasks and outcomes and by applying the SOLO taxonomy (refer to section 2.6).

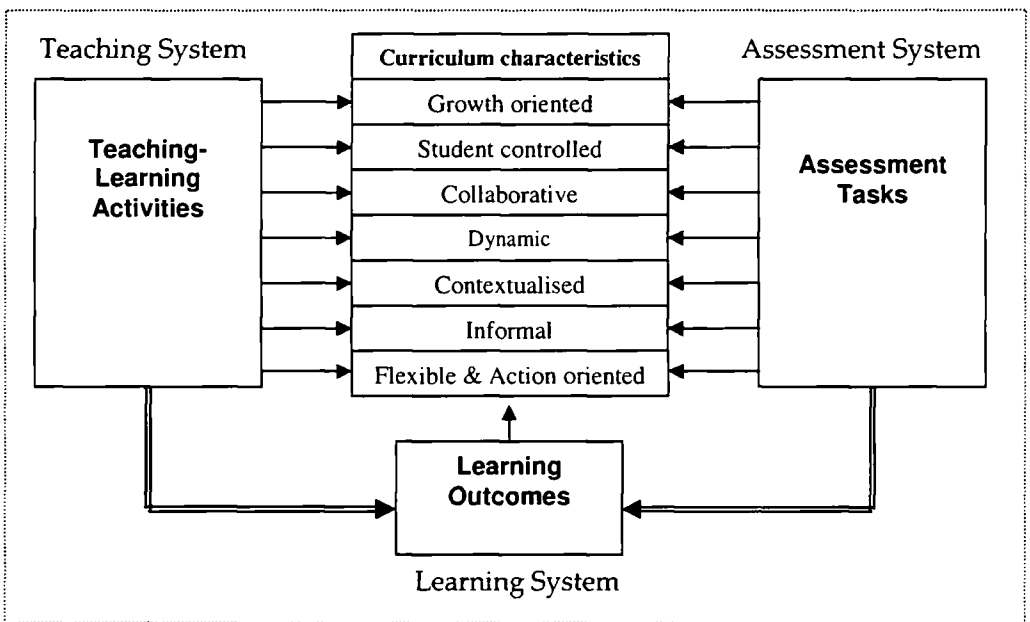


Figure 4.6 Aligning integrated curriculum with teaching, learning and assessment[2]

As recommended by Biggs (2003 p.30), an aligned system of instruction would integrate the teaching-learning activities with the curriculum objectives (as specified in the characteristics above) and are embedded in the assessment tasks so that students

[2] The model in Figure 4.6 is adapted and revised from Biggs'(2003, p. 28) constructive alignment model and Posner's (2004, p.261-263) integrated curriculum evaluation.

would engage with the learning outcomes. Students' learning are primed or guided by the teaching and the assessment systems and the students' own priorities are being measured by their performance in the learning outcomes.

e-Lecture or Practical

Teaching-learning activities as shown in figure 4.7 are determined as verbs in the teaching system. These verbs are translated to learning outcomes in the learning system where students are able to repeat steps I to IV on different examples of the same concept. The assessment system specifies the desired level of attainment which is used by the tutor as a guide to the accomplishment of the learning objectives. The cycle can be repeated as the activities progress which is denoted by the circular arrows at the centre of figure 4.7.

By employing the blended learning approach, the e-lecture or practical lesson incorporates contextualised, flexible and action-oriented characteristics. The online quiz assessment is growth oriented and dynamic ensuring a continuous feedback to students. The practical test can be considered as growth oriented as students are allowed to use the development environment to code and test their programs before submission

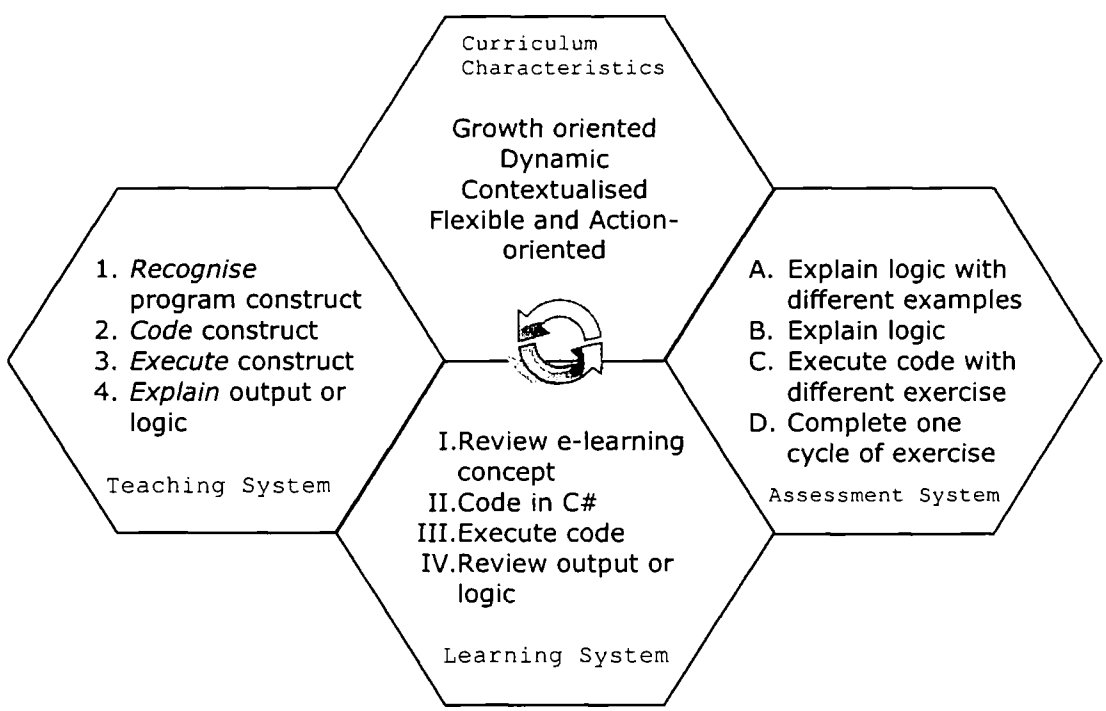


Figure 4.7 A specimen of the alignment framework to teach a computing concept in an e-lecture or practical indicating the curriculum characteristics supported by the integrated model

Project work

The teaching system instructs the tutor to identify the project's tasks (as shown in figure 4.8) which are found in the specifications of the project. The tutor guides the students to achieve the learning outcomes and at the same time to develop the solution for the project. The assessment system specifies the project criteria with respect to the working solution; student contribution and team work are reflected as plus factors in the overall assessment.

The project development supports student-controlled and collaborative characteristics of the integrated curriculum. Project specifications are open-ended which allow students greater ownership and exchange of ideas. Progressive monitoring and feedback of the students' projects makes the project work dynamic, informal, flexible and action-oriented. Students work on real-life domains which contextualised the project for students' learning.

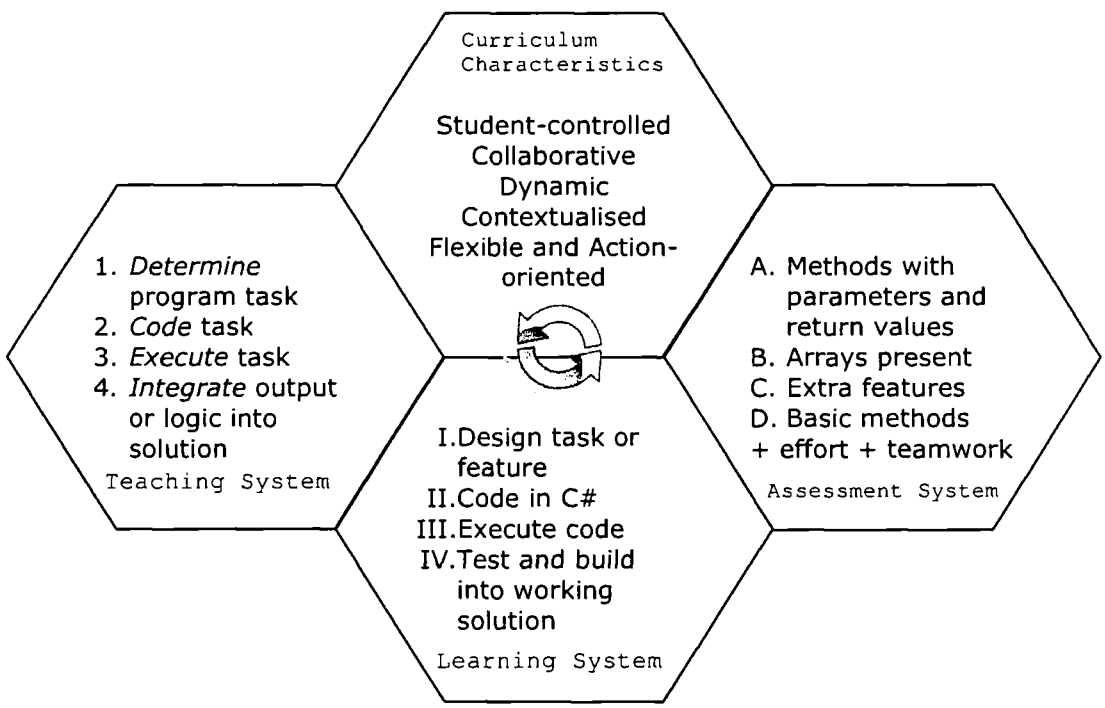


Figure 4.8 A specimen of the alignment framework to develop a solution during project work indicating the curriculum characteristics supported by the integrated model

The above analysis using the alignment framework reaffirms hypothesis H7 implying that the blended learning approach enables far more congruence of the curriculum, the learning objectives and the assessment compared to the traditional structured approach.

The evidence collected and verified through the hypotheses in this chapter indicates a positive, cohesive and continuous picture of the blended learning framework and the integrated curriculum for computer programming. Consistent with eliminative and

analytic induction, each hypothesis is confirmed or reformulated in light of evidence found in the analysis. By employing the proposed alignment framework, this research establishes the reliability of this study. To maintain the validity of this study, the next chapter (chapter 5) analyses the students' performance in the PrC module. As a means of triangulation, comments and feedback from students and tutors are presented at the chapter 5 but these data are only indicative and are not meant to reaffirm or refute the hypotheses.

Chapter 5 Findings II : Statistical Analysis

Extracting statistical evidence from the students' performance in the module enables this study to evaluate the students' competence in the computer programming module. This chapter investigates statistical information obtained from the comparative case study i.e. the students' module score in the PrC module and its sub-components, the project score and the individual score. In keeping with the eliminative and analytic inductions of the comparative case study, the statistical analysis does not generalise but rather confirms or reforms hypothesis based on the statistical evidence. Subsequently, the hypotheses from both findings (chapters 4 and 5) are put together to highlight those that reaffirms the assumptions regarding blended learning. Triangulation with student and tutor comments and other related past reports are furnished to support the validity of this study.

5.1 Demographics of cases

To understand the composition of each case, descriptive statistics were performed to see if there are significant independent variables common across the 2 cases as shown in table 5.1.

Case	*Independent Variable*	Overall Frequency	Percent	GCE'O' Levels	Percent	Higher Certs	Percent
	Gender						
2005-S1	Female	90	38.8	73	36.3	17	54.8
	Male	142	61.2	128	63.7	14	45.2
	Total	232	100.0	201	100.0	31	100.0
2006-S1	Female	108	43.7	98	45.0	10	34.5
	Male	139	56.3	120	55.0	19	65.5
	Total	247	100.0	218	100.0	29	100.0
	Age						
2005-S1	17+	92	39.7	92	45.8		
	18+	79	34.1	79	39.3		
	19+	53	22.8	26	12.9	27	87.1
	20+	4	1.7	3	1.5	1	3.2
	21+	1	0.4	1	0.5		
	22+	1	0.4			1	3.2
	23+	1	0,4			1	3.2
	24+	1	0.4			1	3.2
	Total	232	100.0	201	100.0	31	100.0
2006-S1	17+	101	40.9	101	45.0		
	18+	88	35.6	88	40.4		
	19+	42	17.0	26	11.9	16	55.2
	20+	1	0.4	1	0.5		
	21+	10	4.0	2	0.9	8	27.6
	22+	2	0.8			2	6.9
	23+	1	0.4			1	3.4
	24+	1	0.4			1	3.4
	27+	1	0.4			1	3.4
	Total	247	100.0	218	100.0	29	100.0
	Race						
2005-S1	Chinese	185	79.7	161	80.1	24	77.4
	Indian	15	6.5	15	7.5		

Case	*Independent Variable*	Overall Frequency	Percent	GCE'O' Levels	Percent	Higher Certs	Percent
	Malay	29	12.5	23	11.4	6	19.4
	Others	3	1.3	2	1.0	1	3.2
	Total	232	100.0	201	100.0	31	100.0
2006-S1	Chinese	203	82.2	179	82.1	24	82.8
	Indian	10	4.0	7	3.2	3	10.3
	Malay	32	13.0	30	13.8	2	6.9
	Others	2	0.8	2	0.9		
	Total	247	100.0	218	100.0	29	100.0
	Nationality						
2005-S1	Singapore	213	91.8	184	91.5	29	93.5
	Malaysia	5	2.2	5	2.5		
	Indonesia	5	2.2	5	2.5		
	China	5	2.2	4	2.0	1	3.2
	India	3	1.3	3	1.5		
	Vietnam	1	0.4			1	3.2
	Total	232	100.0	201	100.0	31	100.0
2006-S1	Singapore	235	95.1	208	95.4	27	93.1
	Malaysia	5	2.0	4	1.8	1	3.4
	Indonesia	1	0.4	1	0.5		
	China	4	1.6	4	1.8		
	India	2	0.8			1	3.4
	UK	1	0.4	1	0.5		
	Total	247	100.0	218	100.0	29	100.0

Table 5.1 Descriptive statistics of gender, age, race & nationality

The female to male gender ratio for case 2005-S1 is 2:3 whereas for case 2006-S1 it is almost equal, 0.8:1. When the cohort for case 2005-S1 separates those with standard GCE 'O' levels and those with higher certificates, the distribution evens out. Case 2005-S1 has more females with higher certificates and case 2006-S1 has more males with higher certificates. The dependent variable is the module score that the student achieved at the end of the PrC module. When the means of the module score are computed for male and female students, the values are similar across males and females in both cases as seen in table 5.2. It seems that males are performing slightly better than females as computer programming is technical oriented. An independent samples test or better known as t-test was performed to see if gender difference is a significant independent variable to this comparative study.

Dependent Variable	Case	SEX	Mean	Std. Error	95% Confidence Interval	
					Lower Bound	Upper Bound
MODULE SCORE	2005-S1	F	66.733	1.382	64.018	69.449
		M	69.444	1.100	67.282	71.605
	2006-S1	F	67.065	1.261	64.586	69.544
		M	69.540	1.112	67.355	71.724

Table 5.2 Module score distribution across gender

Based on Levene's test (recommended by Field, 2005) as seen in table 5.3, the result for case 2005-S1 module score is $F(1, 230) = 4.798$, where significance value of 0.030 indicates $p < 0.05$, meaning the variances are not homogeneous which is acceptable since the distribution for case 2005-S1 has more males. As such equal variances are not assumed; the t-test result $t(230) = 1.599$ is read where significance value is 0.111

i.e. $p > 0.05$, proves that the difference in the means is not significant. Case 2006-S1 module score is $F(1,245) = 2.076$ where significance value is 0.151, denotes $p > 0.05$; variances are homogeneous and roughly equal. The t-test results $t(245) = 1.518$ shows that significance value is $p > 0.05$; similar to case 2005-S1, the difference in the means is not significant. This implies that the gender variable will not be a major consideration to this study.

Module Score By case		Levene's Test Equality of Variances		t-test for Equality of Means						
		F	Sig.	t	df	Sig. (2-tailed)	Mean Diff	Std. Err Diff.	95% Confidence Interval of the Diff	
									Lower	Upper
2005-S1	A	4.798	.030	1.488	230	.138	2.710	1.822	-.880	6.300
	¬A			1.599	226.391	.111	2.710	1.695	-.630	6.051
2006-S1	A	2.076	.151	1.518	245	.130	2.475	1.630	-.736	5.686
	¬A			1.540	240.343	.125	2.475	1.607	-.691	5.640

A: Equal variances assumed; ¬A: Equal variances not assumed

Table 5.3 Independent samples test (or t-test) for module score based on gender variable

The age variable represents the number of years, a student takes to achieve the aggregate to be accepted into the course, the younger the student the better off he/she is. In this study, the number of students who achieved the acceptable aggregate at 17 is 45.8% for case 2005-S1 which is almost the same as that for case 2006-S1, 45.0%. In the second group of students with higher certificates, starts at two years above and the distribution is about the same across case 2005-S1 (31 students) and case 2006-S1 (29 students). It appears that case 2006-S1 has a higher proportion of students aged 21+: 8 with higher certificates and 2 with GCE 'O' levels. From the data particulars, the 8 students comprised 5 male students who have completed their National service and 3 female students who were formerly in full-time employment; the 2 female students with GCE 'O' levels are from China and had spent a few years to master the English language. The age variable denotes the maturity of the students but since the distribution across both cases (except for 21+) is almost similar, age is not of consideration in this study.

The race variable is a consideration for ethnicity and cultural diversity. Both cohorts have more than 80% Chinese and the distribution of other races is similar across the Indians, Malays and others. In this situation, the Chinese cultural influence will predominate over the other races. As such this variable is not a part of the analysis. The same goes for the nationality variable. The polytechnic being a government funded

institution has more than 90% Singapore citizens and it is assumed that the other nationalities will have only minor influences.

What remains in this study is the entry level variable which denotes students without computer programming experience i.e. the GCE 'O' levels students and those with prior knowledge i.e. students with higher certificates. Based on Pearson's correlation statistics (recommended by Field, 2005 p.127) as seen in table 5.4, the results prove that entry level plays a significant role in the module score which reflects the students' competence in computer programming at the end of the semester.

		2005-S1 MODULE SCORE	2005-S1 Entry level	2006-S1 MODULE SCORE	2006-S1 Entry level
MODULE SCORE	Pearson Correlation	1	-.322(**)	1	-.364(**)
	Sig. (1-tailed)		.000		.000
	N	232	232	247	247
GCE 'O' LEVEL Attainment	Pearson Correlation	-.322(**)	1	-.364(**)	1
	Sig. (1-tailed)	.000		.000	
	N	232	232	247	247

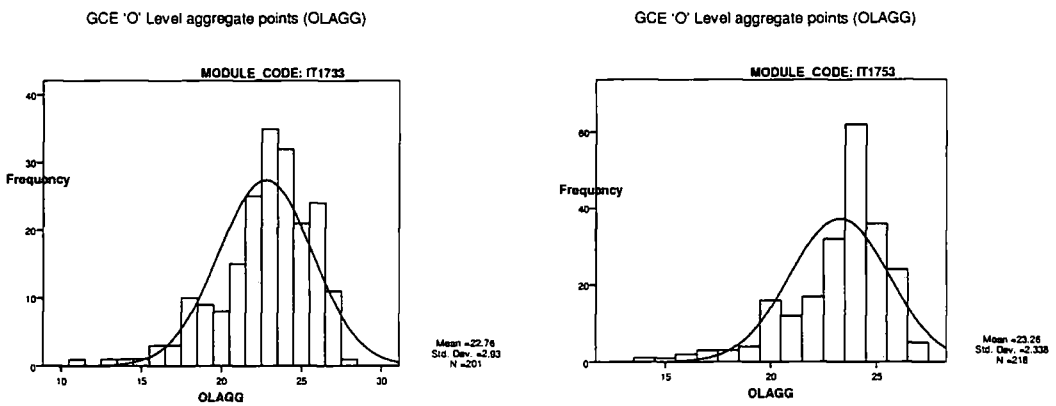
** Correlation is significant at the 0.01 level (1-tailed).

Table 5.4 Results of Pearson correlation of module score and entry level variables

Case	2005-S1	2006-S1
Total students	201	218
Mean	22.76	23.26
Std. Error of Mean	.207	.158
Median	23.00	24.00
Mode	23	24
Std. Deviation	2.930	2.338
Variance	8.583	5.466
Std. Error of Skewness	.172	.165
Std. Error of Kurtosis	.341	.328
Range	17	13
Minimum	11	14
Maximum	28	27
Skewness	-.989	-1.226
Kurtosis	1.320	1.818

Table 5.5 GCE 'O' levels aggregate points distribution

Looking at the statistics distribution for students with GCE 'O' levels only as shown in table 5.5, cohort 2005-S1 has a lower mean but higher standard deviation and variance compared with cohort 2006-S1. However, based on the values of skewness and kurtosis from both cohorts, the respective values are well below 1.96 for significance at $p < 0.5$ and below 3.29 at $p<0.001$ which indicates a normal distribution. In addition, the histogram diagram (refer to figure 5.1) with normal curve, further confirms that both distributions are normal. Hence parametric tests can be applied across the two groups.



2005-S1 2006-S1

Figure 5.1 Histograms with normal curves for GCE 'O' levels aggregate

For GCE 'O' levels students, their aggregate points are used to measure an initial aptitude as the selection into the EI diploma course is based on the students' choice and GCE 'O' levels aggregate points. Those with higher certificates are selected through direct application from each student. From the means and mode of the GCE 'O' levels aggregate of the two cases, case 2005-S1 is a higher aptitude cohort compared with case 2006-S1. The module code for PrC in case 2005-S1 is IT1733 and that in case 2006-S1 is IT1753 (as reflected in the module syllabus, appendix B).

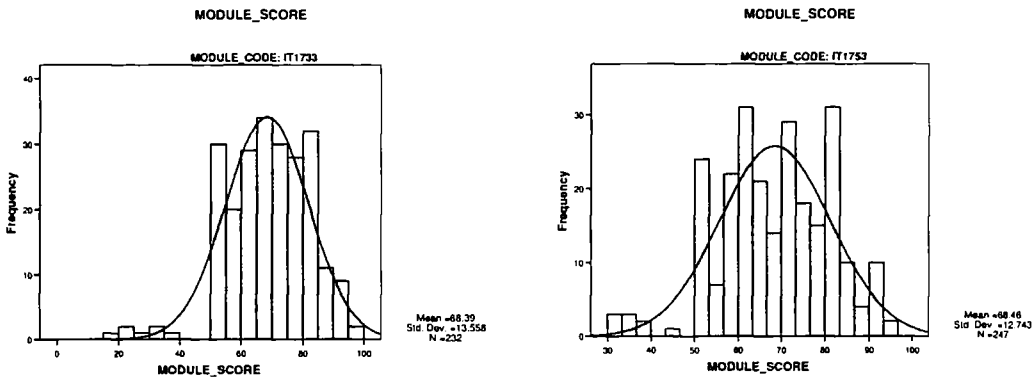
5.2 Dependent variable 1: Module score

H8: The blended learning environment has increased students' module score in computer programming.

The statistics distribution for the module score across case 2005-S1 and 2006-S1 are shown in table 5.6 with the respective histograms in figure 5.2. There seems to be only slight increase (0.07) of case 2006-S1 over case 2005-S1. Moreover, both histograms have more than one mode.

Case	2005-S1	2006-S1
Total students	232	247
Mean	68.39	68.46
Std. Error of Mean	.890	.811
Median	69.00	68.00
Mode	51	58
Std. Deviation	13.558	12.743
Variance	183.832	162.379
Skewness	-.617	-.313
Std. Error of Skewness	.160	.155
Kurtosis	1.031	.086
Std. Error of Kurtosis	.318	.309
No. who scored < 50	7	9
Minimum	19	31
Maximum	96	96

Table 5.6 Module score distribution between cases 2005-S1 and 2006-S1



2005-S1 2006-S1

Figure 5.2 Histograms with normal curves for module score

Module Score	Levene's Test for Equality of Variances		t-test for Equality of Means						
	F	Sig.	t	Df	Sig. (2-tailed)	Mean Diff	Std. Error Diff	95% Confidence Interval of the Difference	
								Lower	Upper
Equal variances assumed	.221	.638	-.054	477	.957	-.065	1.202	-2.427	2.296

Table 5.7 Results of independent t-test for module score between cases 2005-S1 and 2006-S1

Based on Levene's test (table 5.7), the result for module score is $F(1, 477) = 0.221$, where significance value of 0.638 indicates $p > 0.05$, meaning the variances are homogeneous and are roughly equal. This implies that the independent t-test is appropriate since the variances are roughly equal and the scores come from different students in the cases. Another assumption of the t-test is that the data are acceptable normally distributed which are shown in figures 5.1 and 5.2. Based on table 5.7, the t-test result $t(477) \approx -0.054$, $p > 0.05$, ascertains that the difference in the means is not significant. Hence, although case 2006-S1's has a higher mean compared to case 2005-S1, it does not prove conclusively that the students' performance has improved over case 2005-S1. On the other hand, since the GCE 'O' levels distribution in table 5.4 indicates that case 2006-S1 has weaker students, the module score results for case 2006-S1 indicates sustained performance over case 2005-S1. This implies that H8: The blended learning environment (case 2006-S1) has increased students' module score in computer programming can be refined as follows:

H8A: The blended learning environment has sustained students' module score in computer programming.

Using eliminative induction (as explained in chapter 3), the analysis at this point investigates whether students with higher certificates and those with GCE 'O' levels

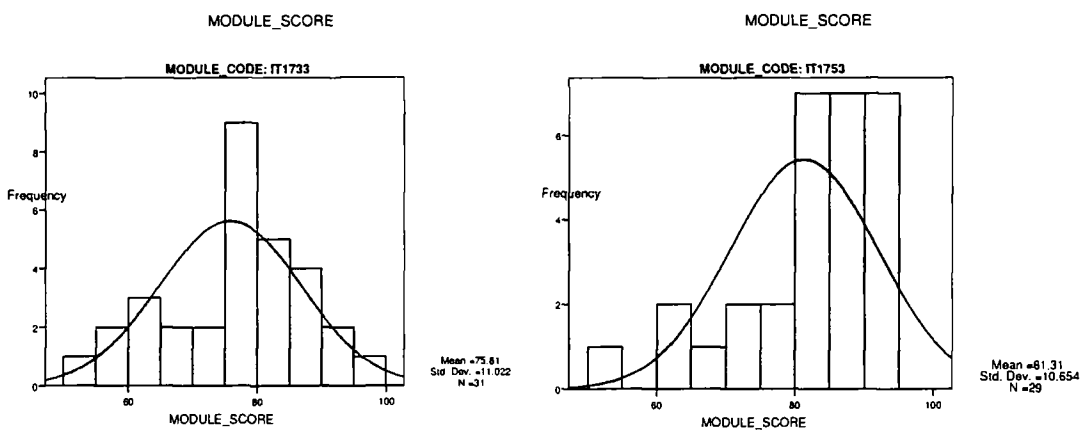
benefit from the blended learning environment. A new hypothesis H8B is added as such: *The blended learning environment has improved students' module score for different groups in the cohort,* i.e. those with GCE 'O' levels entry level (novice computer programmers) and those with higher certificates (prior computing knowledge).

Based on the new hypothesis, the statistics distribution is recomputed as follows:

Case	2005-S1 Overall	2006-S1 Overall	2005-S1 GCE'O'	2006-S1 GCE'O'	2005-S1 HigherCert	2006-S1 HigherCert
Total students	232	247	201	218	31	29
Mean	68.39	68.46	67.25	66.75	75.81	81.31
Std. Error of Mean	.890	.811	.957	.814	1.980	1.978
Median	69.00	68.00	68.00	66.00	77.00	83.00
Mode	51	58	51	58	75	85
Std. Deviation	13.558	12.743	13.575	12.017	11.022	10.654
Variance	183.832	162.379	184.268	144.420	121.495	113.507
Skewness	-.617	-.313	-.616	-.383	-.379	-1.193
Std. Err of Skewness	.160	.155	.172	.165	.421	.434
Kurtosis	1.031	.086	1.109	.343	-.501	1.214
Std. Err of Kurtosis	.318	.309	.341	.328	.821	.845
No. who scored < 50	7	9	7	9	0	0
Minimum	19	31	19	31	54	51
Maximum	96	96	95	96	96	95

Table 5.8 Module score distribution with sub-groups GCE 'O' levels holders and higher certificate holders

There is an improvement for the higher certificate group in case 2006-S1 over case 2005-S1. The histograms further add proof to this observation:



2005-S1 Higher Certificate 2006-S1 Higher Certificate

Figure 5.3 Histograms with normal curves of module score for higher certificate holders

Based on Levene's test (table 5.9), the result for module score is F(1, 58) = 0.110, where significance value of 0.741 indicates p > 0.05, meaning the variances are homogeneous and roughly equal. The t-test value for t(58) is -1.964, where p<0.05 (i.e. 0.054/2=0.027), proves that the difference in the means is significant. The effect size r, sqrt((-1.964)^2 / (-1.964)^2 + 58)) is 0.25, which is a modest effect of the total variance. Hence hypothesis H8B is true for students with higher certificates.

Module score	Levene's Test for Equal Variance		t-test for Equality of Means (equal variances assumed)						
	F	Sig.	T	df	Sig. (2-tailed)	Mean Diff	Std. Err. Diff	95% Confidence Interval of the Difference	
								Lower	Upper
Higher Certs	.110	.741	-1.964	58	.054	-5.504	2.802	-11.113	.105
GCE 'O' Levels	1.602	.206	.401	417	.689	.501	1.250	-1.957	2.959

Table 5.9 Results of independent t-test for module score of students with higher certificates and those with GCE 'O' levels (between cases 2005-S1 and 2006-S1)

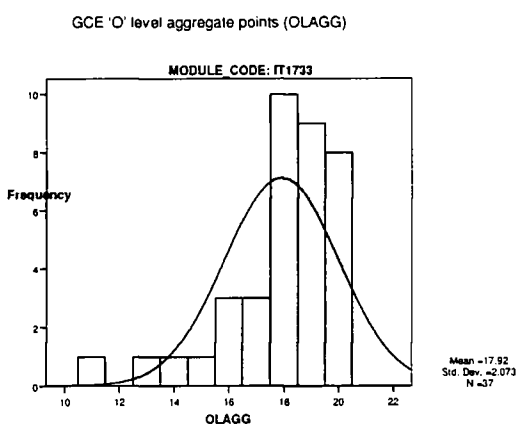
For GCE 'O' level holders, the mean is slightly lower for case 2006-S1, 0.5 less than that for 2005-S1 (see table 5.8); however, the standard deviation and variance are lower for case 2006-S1 over 2005-S1. Looking at the independent t-test as seen in table 5.9, the t-test value, t(417) is 0.401 where p > 0.05 denotes that the mean is non-significant, which implies that even though the mean module score for GCE 'O' level holders are lower in case 2006-S1, it does not prove that students in case 2006-S1 have performed worse than case 2005-S1.

Since H8B is not tenable for novice computer programmers, it is still possible to add another hypothesis to see if blended learning has helped weaker students from the GCE 'O' level holders. Another sub-group can be made to separate those who scored more than 20 aggregate points over those who scored 20 or less. 20 denotes the cut-off aggregate point for students to enrol in junior colleges to take up the GCE 'A' levels. Moreover from the demographics, 18.4% of 2005-S1 students scored 20 or less and only 13.4% of 2006-S1 students scored the same.

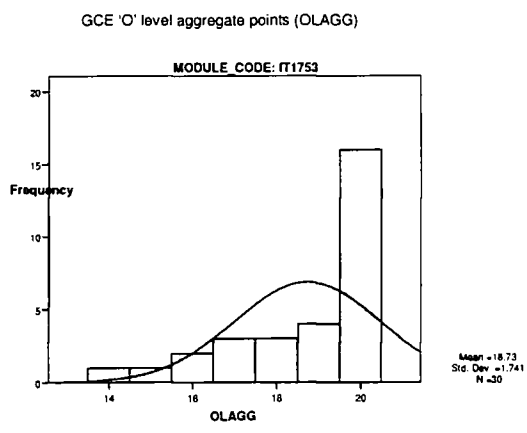
Another sub-hypothesis is added, H8C: *The blended learning environment has improved students' module score for weaker students of GCE 'O' level holders of more than 20 aggregate points.* The statistics distribution for GCE 'O' level holders sub-groups is computed in table 5.10 and the corresponding histograms are displayed in figure 5.4.

Case	2005-S1	<= 20	> 20	2006-S1	<= 20	> 20
Total students	201	37	164	218	30	188
Mean	22.76	17.92	23.85	23.26	18.73	23.98
Std. Error of Mean	.207	.341	.137	.158	.318	.105
Median	23.00	18.00	24.00	24.00	20.00	24.00
Mode	23	18	23	24	20	24
Std. Deviation	2.930	2.073	1.749	2.338	1.741	1.435
Variance	8.583	4.299	3.058	5.466	3.030	2.059
Std. Error of Skewness	.172	.388	.190	.165	.427	.177
Std. Error of Kurtosis	.341	.759	.377	.328	.833	.353
Range	17	9	7	13	6	6
Minimum	11	11	21	14	14	21
Maximum	28	20	28	27	20	27
Skewness	-.989	-1.566	.178	-1.226	-1.283	-.202
Kurtosis	1.320	2.755	-.859	1.818	.715	-.321

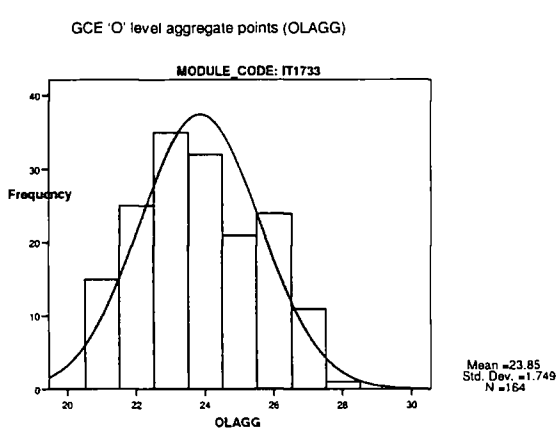
Table 5.10 GCE 'O' levels aggregate points distribution with sub-groups of 20 or less and above 20



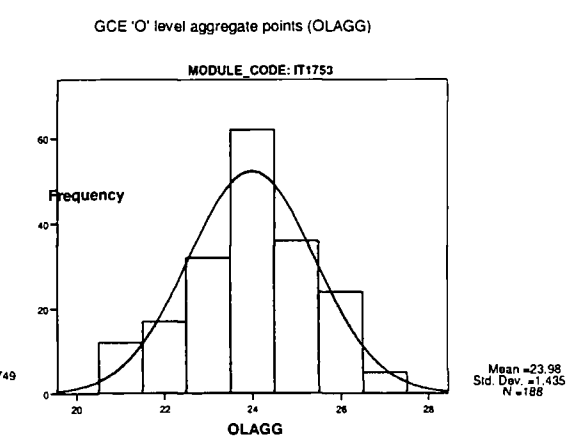
2005-S1, 20 or less aggregate



2006-S1, 20 or less aggregate



2005-S1, above 20 aggregate



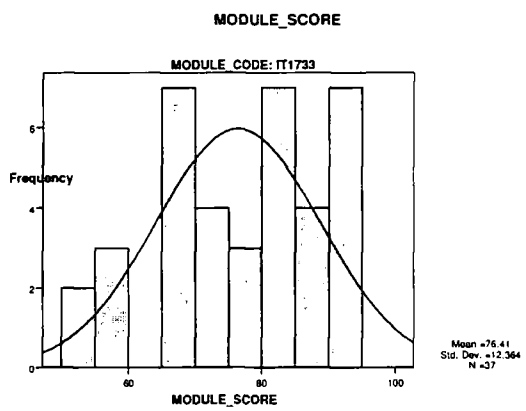
2006-S1, above 20 aggregate

Figure 5.4 Histograms with normal curves for GCE 'O' levels aggregate subgroups

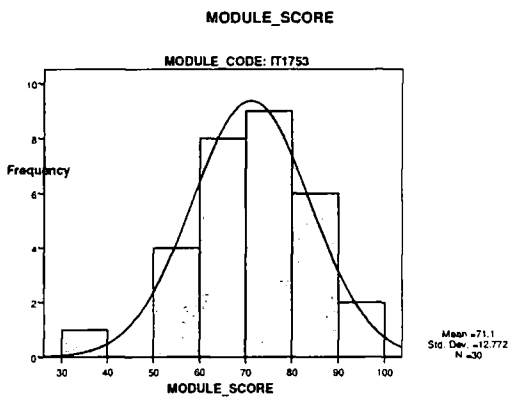
Based on the additional sub-groups, the distribution for module score is recomputed:

Case	2005-S1 GCE'O'	2006-S1 GCE'O'	2005-S1 20 or less	2006-S1 20 or less	2005-S1 above 20	2006-S1 above 20
Total students	201	218	37	30	164	188
Mean	67.25	66.75	76.41	71.10	65.18	66.05
Std. Error of Mean	.957	.814	2.033	2.332	1.015	.859
Median	68.00	66.00	79.00	72.00	66.00	66.00
Mode	51	58	67	66	51	63
Std. Deviation	13.575	12.017	12.364	12.772	13.000	11.780
Variance	184.268	144.420	152.859	163.128	169.009	138.767
Skewness	-.616	-.383	-.312	-.901	-.826	-.338
Std. Err.of Skewness	.172	.165	.388	.427	.190	.177
Kurtosis	1.109	.343	-.931	2.096	1.467	.241
Std. Err. of Kurtosis	.341	.328	.759	.833	.377	.353
No. who scored < 50	7	9	0	1	7	8
Minimum	19	31	53	31	19	32
Maximum	95	96	95	93	93	96

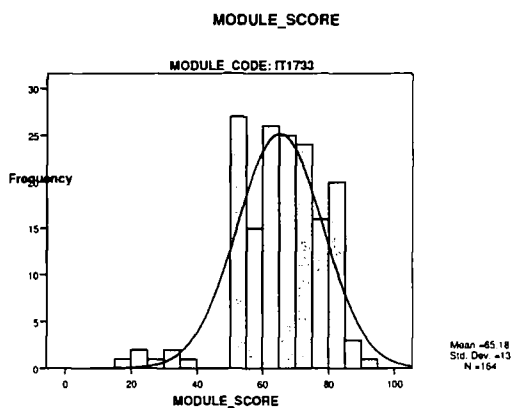
Table 5.11 Module score distribution with GCE 'O' levels sub-groups <=20 and >20



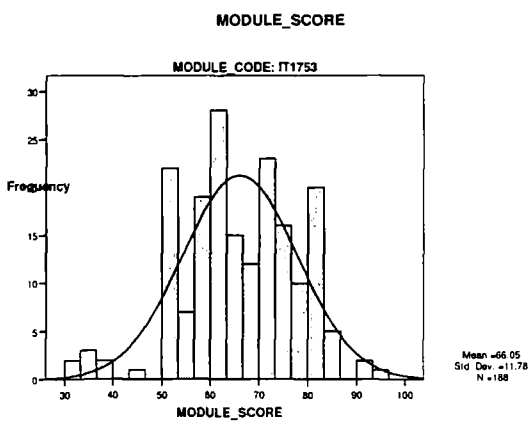
2005-S1, 20 or less aggregate



2006-S1, 20 or less aggregate



2005-S1, above 20 aggregate



2006-S1, above 20 aggregate

Figure 5.5 Histograms with normal curves of module score, GCE 'O' levels subgroups

The analysis at this juncture shows that even though case 2006-S1 has started with a weaker cohort compared to case 2005-S1 (as seen from table 5.10 and fig. 5.4), students from 2006-S1 do not perform any worse than case 2005-S1. Looking at the module score means in table 5.11, where case 2005-S1 is concern, there is more than 10 points difference between the 20 or less group and the above 20 group (76.41, 65.18); however in case 2006-S1, there is only a 5 point difference (71.10, 66.05) between the same groups. Further analysis on the independent t-test as shown in table 5.12, reveal that the difference in the means between the cases across the GCE 'O' levels groups are not significant (where $p > 0.05$). This implies that the hypothesis holds for H8A as students in case 2006-S1 has shown sustained performance over the better 2005-S1 group of novice programmers and H8C is withdrawn.

Module score	Levene's Test Equality Var.		t-test for Equality of Means (equal variances assumed)						
	F	Sig.	T	df	Sig. (2-tailed)	Mean Diff	Std. Error Diff	95% Confidence Interval of Difference	
								Lower	Upper
GCE less = 20	.280	.599	1.721	65	.090	5.305	3.083	-.851	11.462
GCE > 20	.598	.440	-.659	350	.510	-.870	1.321	-3.468	1.728

Table 5.12 Results of independent t-test for module score of groups with GCE 'O' levels points 20 and less and points above 20 (between cases 2005-S1 and 2006-S1)

Comparing the means for students with prior programming as shown in table 5.8, with those in the 20 or less group in case 2005-S1, there is a slight 0.6 point drop (75.81, 76.41); however in case 2006-S1, students with prior programming has a 10 point advantage (81.31,71.10). Apparently, blended learning has sustained the performance of case 2006-S1 across all groups in general and is able to assist students with prior programming knowledge in particular.

It is noted here that there are a number of students who scored less than 50 in this module. Each cohort has its fair share of weak students who did not pass this module and although learning analysis is beyond the scope of this study, the module group analysis in section 5.5 will review how students perform in their respective groups.

To investigate the discrepancies that arise from the module score, the analysis reviews the main components of the module score i.e. the project score and the individual test score.

5.3 Dependent variable 2: Project score

H9: The blended learning environment has improved students' project scores in computer programming.

The project work involves at least two students working together to produce a working solution. Whether a student chooses to work in a group of three or individually is left to the discretion of the tutor. This component contributes 30% to the module score for both cases 2005-S1 and 2006-S1. The module score comprises different components that make up the overall score as seen in tables 5.13 and 5.14,

Case 2005-S1	MODULE SCORE	Writtn40%	Lab20%	Proj30%	Tut10%
N	232	232	232	232	232
Mean	68.39	68.96	71.55	63.5552	80.91
Std. Error of Mean	.890	1.283	1.391	1.04555	.270
Median	69.00	69.00	75.00	64.4100	80.00
Mode	51	55(a)	92(a)	68.33	80
Std. Deviation	13.558	19.537	21.187	15.92532	4.114
Variance	183.832	381.712	448.906	253.616	16.926
Skewness	-.617	-.517	-.492	-.974	4.403
Std. Error of Skewness	.160	.160	.160	.160	.160
Kurtosis	1.031	-.170	-.498	2.674	17.686
Std. Error of Kurtosis	.318	.318	.318	.318	.318
Range	77	100	96	100.00	20
Minimum	19	0	4	.00	80
Maximum	96	100	100	100.00	100

a Multiple modes exist. The smallest value is shown

Table 5.13 Module score distribution and its assessment components for case 2005-S1

Case 2006-S1	MODULE SCORE	Lab50%	Proj30%	Quiz20%
N	247	247	247	247
Mean	68.46	64.35	67.0830	81.03
Std. Error of Mean	.811	.985	.85330	.813
Median	68.00	65.22	68.0000	84.38
Mode	58(a)	49	65.00	78
Std. Deviation	12.743	15.475	13.41073	12.773
Variance	162.379	239.472	179.848	163.151
Skewness	-.313	-.339	.090	-1.869
Std. Error of Skewness	.155	.155	.155	.155
Kurtosis	.086	-.311	-.421	4.690
Std. Error of Kurtosis	.309	.309	.309	.309
Range	65	77	74.50	71
Minimum	31	19	24.00	28
Maximum	96	96	98.50	99

Table 5.14 Module score distribution and its assessment components for case 2006-S1

A common component for cases 2005-S1 and 2006-S1 is the project work which contributes 30% to the module score. The mean is improved for case 2006-S1 over case 2005-S1 but since the difference between the variances is high (253.616 – 179.848 = 73.768), the independent t-test is carried out. Based on Levene's test, the result for module score is $F(1, 477) = 0.221$, and for project score is $F(1,477) = 0.114$ where significance values are greater than 0.05, meaning the variances are

homogeneous and are roughly equal. This implies that the independent t-test is appropriate since the variances are roughly equal and the scores come from different students in the cases. Another assumption of the t-test is that the data are normally distributed which was already shown in figures 5.1 and 5.2. Based on table 5.15, the t-test result $t(477) = -2.628$, $p < 0.05$, further proves that the difference in the means is significant. This means the hypothesis H9: The blended learning environment has improved students' project scores in computer programming is tenable. This implies that the difference in the cases represents a genuine improvement of the project work where cohort 2006-S1 performs better because of the blended learning environment.

Based on Project score	Levene's Test Equality Var.		t-test for Equality of Means						
	F	Sig.	t	df	Sig. (2-tailed)	Mean Difference	Std. Err. Diff.	95% Confidence Interval of Difference	
								Lower	Upper
Equal variances assumed	.114	.736	-2.628	477	.009	-3.52782	1.34237	-6.16551	-.89014

Table 5.15 Results of independent t-test for project work between cases 2005-S1 and 2006-S1

The effect size r. is computed as follows: $r = \text{sqrt}[(-2.628)^2/((-2.628)^2+477)] = 0.12$, which is a small effect of the total variance of the population.

5.4 Dependent variable 3: Individual test score

H10: The blended learning environment has improved students' individual test scores in computer programming.

The individual tests are different for case 2005-S1 and 2006-S1. In the traditional approach, case 2005-S1 has a final written test paper which contributes 40% of the module score and two practical tests which contribute a total of 20% to the module score. Case 2006-S1 has removed the written test in line with the integrated curriculum approach and focuses on the students' practical computer programming skills. As such, there are two practical tests contributing a total of 50% to the module score and an incremental online quiz/assessment that contributes 20% of the module score. In both cases, the practical tests are conducted in the computer laboratory where students are to work on the answers individually. It is a closed test but students are allowed to make use of the language development environment to help them with the syntax and compilation checks before submitting their own answers.

As seen in table 5.16, case 2005-S1 cohort performs better in the practical tests (mean=71.55) compared to case 2006-S1 (mean=64.35). On the other hand, students fared worse in the written tests for case 2005-S1 (mean=68.96) compared to the

incremental online quiz for case 2006-S1 (mean=81.03). When the individual tests are combined, the means is only a difference of 0.7102 (69.8218 – 69.1116).

	2005-S1 Lab20%	Written40%	2006-S1 Lab50%	Quiz20%	2005-S1 Indv60%	2006-S1 Indv70%
N	232	232	247	247	232	247
Mean	71.55	68.96	64.35	81.03	69.8218	69.1116
Std. Error of Mean	1.391	1.283	.985	.813	1.17201	.85677
Median	75.00	69.00	65.22	84.38	70.6667	70.2643
Mode	92(a)	55(a)	49	78	56.67(a)	58.34(a)
Std. Deviation	21.187	19.537	15.475	12.773	17.85148	13.46518
Variance	448.906	381.712	239.472	163.151	318.675	181.311
Skewness	-.492	-.517	-.339	-1.869	-.477	-.678
Std. Err. of Skewness	.160	.160	.155	.155	.160	.155
Kurtosis	-.498	-.170	-.311	4.690	-.315	.780
Std. Err. of Kurtosis	.318	.318	.309	.309	.318	.309
Range	96	100	77	71	85.33	72.59
Minimum	4	0	19	28	14.67	23.27
Maximum	100	100	96	99	100.00	95.86

Table 5.16 Individual test score distribution and its contributing components

Further tests on variances proved that the variances are not homogeneous as shown in table 5.17; the Levene's test result for individual test score is $F(1, 477) = 23.591$, where the significance value is zero, $p < 0.05$, meaning the variances are not homogeneous. Taking the reading for equal variances not assumed from table 5.17, the t-test result $t(477)= 0.489$, $p > 0.05$, further proves that the difference in the means is not significant. This implies that the hypothesis H10 is reformulated to reflect that:

H10A: The blended learning environment has sustained students' individual test scores in computer programming.

	Levene's Test Equality Var.		t-test for Equality of Means						
	F	Sig.	t	df	Sig. (2-tailed)	Mean Diff	Std. Err. Diff	95% Confidence Interval Difference	
Based on Individual test score								Lower	Upper
Equal variances	23.591	.000	.493	477	.622	.7102	1.43931	-2.11798	3.53837
Not Equal variances			.489	428.856	.625	.7102	1.45178	-2.14328	3.56368

Table 5.17 Results of independent t-test for individual test score

Tutorial assignments

The tutorial assignment score is a measure of weekly submissions made by the students, with regard to participation rather than assessment. This is why the tutorial assignment score in case 2005-S1 is either 80% or 100% and its contribution is only 10% to the module score. Hence, the tutorial assignment component is not relevant to

the performance analysis of this study. In case 2006-S1 this component is removed as tutorials are no longer conducted in this cohort and has been replaced with e-learning.

5.5 Performance across module groups

The investigation so far has analysed the students' performance based on the cohort in each case. To see if the students' performance is widely different from the module group performance within each cohort or case, the following null hypothesis is put forth:

H11: There is no significant difference in the module performance across all module groups in each case.

The above implies that what have been observed (H8A to H10A) applies to the module groups at the tutor/module level. There are 11 module groups in case 2005-S1 as well as case 2006-S1 and the mean module score of each group is shown in table 5.18. It seems that there is equal number of groups with mean module score below the overall mean; five groups highlighted in each case.

Case 2005-S1					Case 2006-S1				
Module Group	N	Mean	StdErr Mean	StdDev	Module Group	N	Mean	StdErr Mean	StdDev
Overall	232	68.39	.890	13.558	Overall	247	68.46	.811	12.743
EI0501	21	66.48	2.550	11.686	EI0601	21	68.33	2.960	13.566
EI0502	22	71.77	2.648	12.421	EI0602	23	71.26	2.402	11.522
EI0503	21	68.62	2.853	13.075	EI0603	23	61.696	2.610	12.517
EI0504	22	73.18	2.384	11.181	EI0604	22	71.77	2.888	13.543
EI0505	19	72.16	3.011	13.124	EI0605	22	71.46	2.533	11.884
EI0506	22	63.45	3.139	14.725	EI0606	22	66.00	2.263	10.614
EI0507	20	67.25	3.441	15.389	EI0607	24	75.38	2.516	12.328
EI0508	21	62.81	3.073	14.081	EI0608	24	66.92	2.451	12.007
EI0509	23	73.65	2.492	11.949	EI0609	22	58.46	1.723	8.081
EI0510	20	62.35	2.332	10.429	EI0610	23	71.696	2.279	10.927
EI0511	21	69.95	3.581	16.409	EI0611	21	69.67	3.244	14.867

Table 5.18 Module groups' mean module score in PrC module (further details can be seen in Appendix F, tables FI.1-3 for case 2005-S1, and FII.1-3 for case 2006-S1)

Case 2005-S1 has a lowest group module mean score of 62.35 whereas case 2006-S1 has a lowest of 58.46; highest group module mean score for case 2005-S1 is 73.65 whereas for case 2006-S1 is 75.38. With such wide discrepancies, the Analysis of Variance (ANOVA) is performed for each case to see if the module means are statistically significant and the results summary are shown in table 5.19

MODULE SCORE			Sum of Squares	Df	Mean Square	F	Sig.
2005-S1	Between Groups	(Combined)	3738.290	10	373.829	2.133	.023
	Within Groups		38727.016	221	175.235		
	Total		42465.306	231			
2006-S1	Between Groups	(Combined)	5483.565	10	548.357	3.755	.000
	Within Groups		34461.738	236	146.024		
	Total		39945.304	246			

Table 5.19 One-way ANOVA on module means score of 11 module groups

There is a significant difference in the module score means across module groups for case 2005-S1: $F(10,221) = 2.133$, $p<0.5$ sig = 0.023; To calculate the effect size where ω = sqrt[(SSm – dfm *MSr) /(SSt + MSr)], obtains ω = sqrt((3738.29 -10*175.235)/ (42465.306 + 175.235)) = 0.216; indicates a modest effect. Similarly for case 2006-S1: $F(10,236) = 3.755$ $p < 0.5$; ω = (5483.565 – 10*146.024)/(39945.304+146.024)) = 0.317 (modest effect). Hence both cases have shown significant differences in the module mean score across the module groups with modest effect. (For detailed results of the ANOVA see Appendix F tables FI.4–5 and FII.4–5).

To see which groups' mean module scores are statistically different in each case, the comparison test is applied using the exploratory post-hoc procedure since the data are not from planned experiments but rather natural settings. Since the module groups' sizes are different and not equal, the Games-Howell procedure is utilised (Field,2005) for multiple comparisons; the homogeneous groupings to see which module groups are similar utilises the Garbriel's and the Hochberg's GT2 pairwise test. Details of the results of multiple comparisons can be seen in Appendix F, table 6. Table 5.20 displays the results of the Gabriel's test and the subsets in both cases clearly show significance of $p > 0.05$. However, it is clear that case 2005-S1 has less variation compared to case 2006-S1. Conversely, in case 2006-S1, group 9's module mean score being the lowest is significantly different from 5 other groups.

Case 2005-S1	Group	N	Subset for alpha = .05
			1
Gabriel(a,b)	10	20	62.35
	8	21	62.81
	6	22	63.45
	1	21	66.48
	7	20	67.25
	3	21	68.62
	11	21	69.95
	2	22	71.77
	5	19	72.16
	4	22	73.18
	9	23	73.65
	Sig.		.280

Case 2006-S1	Group	N	Subset for alpha = .05		
			1	2	3
Gabriel(c,b)	9	22	58.45		
	3	23	61.70	61.70	
	6	22	66.00	66.00	66.00
	8	24	66.92	66.92	66.92
	1	21	68.33	68.33	68.33
	11	21	69.67	69.67	69.67
	2	23		71.26	71.26
	5	22		71.45	71.45
	10	23		71.70	71.70
	4	22		71.77	71.77
	7	24			75.38
	Sig.		.109	.263	.414

a Uses Harmonic Mean Sample Size = 21.035.
b The group sizes are unequal. The harmonic mean of the group sizes is used. Type I error levels are not guaranteed.
c Uses Harmonic Mean Sample Size = 22.411.

Table 5.20 Means for groups in homogeneous subsets

To investigate other influences on the group performance, the covariates – entry level aggregate and gender are included in the ANOVA analysis and the summarised results are shown in table 5.21 (detailed results are shown in Appendix F tables FI.9–16 and FII.9–16).

Source of variance	Sums of squares	Df	Mean Square	F-ratio	Sig
Case 2005-S1:					
Covariate (entry level aggregate)	4443.355	1	4443.355	28.513	.000
Main effect (module group)	3782.498	10	378.250	2.427	.009
Residual Error	34283.661	220	155.835		
Covariate (gender)	817.753	1	817.753	4.746	.030
Main effect (module group)	4151.387	10	415.139	2.409	.010
Residual Error	37909.262	220	172.315		
Case 2006-S1:					
Covariate (entry level aggregate)	5393.220	1	5393.220	43.601	.000
Main effect (module group)	5572.330	10	557.233	4.505	.000
Residual Error	29068.518	235	123.696		
Covariate (gender)	838.830	1	838.830	5.863	.016
Main effect (module group)	5950.171	10	595.017	4.159	.000
Residual Error	33622.908	235	143.076		

Table 5.21 Analysis of covariance (ANCOVA) summary table

The covariate, entry level aggregate, is significantly related to the module score:

2005-S1: $F(1,220) = 28.513$, $p < 0.05$, $r = \text{sqrt}((-5.340)^2 / ((-5.340)^2 + 220)) = 0.339$

where $t = -5.340$ is obtained from Appendix F, table FI.12.

2006-S1: $F(1,235) = 43.601$, $p < 0.05$, $r = \text{sqrt}((-6.603)^2 / ((-6.603)^2 + 235)) = 0.396$

where $t = -6.603$ is obtained from Appendix F, table FII.12.

The covariate, gender, is significantly related to the module score:

2005-S1: $F(1,220) = 28.513$, $p < 0.05$, $r = \text{sqrt}((2.178)^2 / ((2.178)^2 + 220)) = 0.145$

where $t = 2.178$ is obtained from Appendix F, table FI.15.

2006-S1: $F(1,235) = 43.601$, $p < 0.05$, $r = \text{sqrt}((2.421)^2 / ((2.421)^2 + 235)) = 0.156$

where $t = 2.421$ is obtained from Appendix F, table FII.15.

The covariates have a significant effect on the module scores for both cases; the effect for entry level aggregate is modest whereas that for gender is small effect. This implies that students' aptitude based on prior computing skills or entry level aggregates have a higher impact on module scores compared to gender. After controlling for the effect of the covariates, there is significant variation on the module scores across both cases: 2005-S1: $F(10, 220) = 2.427$, $p < 0.05$; and 2006-S1: $F(10, 235) = 4.505$, $p < 0.05$.

Based on the above evidence, the hypothesis H11 is reformulated as follows:

H11A: There are significant differences in the module performance across all module groups in both the traditional structured and the blended learning environments.

Since there are significant differences, it is worth investigating the performance of the module groups in the project work as well as the individual tests. This explores the students' competence in terms of developing working solutions (project work) and assessing individual learning (individual test). Each case is reviewed separately to examine group dynamics within the cohort. As affirmed by Johnson and Johnson (2004) through their extensive research on cooperative learning, cooperative groups using computer based problem-solving and instruction perform better than competitive groups and individuals, leading to positive reinforcement.

Case 2005-S1 module groups:

In parallel with the module mean score, ANOVAs of the project score and the individual test are performed and the results are shown in table 5.22; detailed results are shown Appendix F tables FI.4-8. Both project and individual test scores are significant where $p < 0.05$.

2005-S1			Sum of Squares	Df	Mean Square	F	Sig.
Project	Between Groups	(Combined)	6585.309	10	658.531	2.799	.003
	Within Groups		51999.909	221	235.294		
	Total		58585.219	231			
Individual Test	Between Groups	(Combined)	8327.736	10	832.774	2.819	.003
	Within Groups		65286.233	221	295.413		
	Total		73613.969	231			

Table 5.22 One-way ANOVA for project score and individual test score for groups in 2005-S1

Effect size for project: ω = sqrt((6585.309 -10*235.294)/ (58585.219 + 235.294)) = 0.268;

Effect size for individual test: ω = sqrt((8327.736 -10*295.413)/ (73613.969 + 295.413)) = 0.2696.

Hence there is modest effect in the total variances for both project and individual test scores across the groups in case 2005-S1. Next, the homogeneous subsets are compared as shown in table 5.23, to see how the groups differ. The subsets are not significantly different, $p>0.05$, however, it is interesting to note that group 2 which has the lowest project mean score conversely has the highest individual test mean score. This is a 10-group and almost -25 points mean difference (54.7 – 79.4 = -24.7). Similarly, group 11 has a 6-group and -15 points mean difference (59.4 – 74.7 = -15.3). Only group 10 which has the lowest module mean, shows a positive 6 points mean difference (64.6 – 58.6 = 6); the only group which has a project score higher than individual test. There are 7 students who failed this module; 1 from groups 1, 6, 7, 9 and 11; and 2 from group 8. These failed students may lower the group mean scores yet none are from group 2. Hence, the module group's mean performance in case

2005-S1 do not correctly reflect the module group's performance in terms of the students' project work and individual tests.

2005-S1: Project mean: 63.5552			
		Subset for alpha = .05	
Group	N	1	2
2	22	54.6559	
8	21	55.4743	
11	21	59.3610	59.3610
6	22	61.2095	61.2095
1	21	62.2981	62.2981
10	20	64.6460	64.6460
3	21	65.8305	65.8305
7	20	66.4980	66.4980
9	23	68.7287	68.7287
5	19	69.2068	69.2068
4	22		71.5886
Sig.		.120	.425

Individual test mean: 69.8218			
		Subset for alpha = .05	
Group	N	1	2
10	20	58.6333	
6	22	63.0909	63.0909
8	21	64.8254	64.8254
7	20	66.7000	66.7000
1	21	67.6825	67.6825
3	21	68.7302	68.7302
5	19	72.8772	72.8772
4	22	73.9394	73.9394
11	21	74.6667	74.6667
9	23	76.1739	76.1739
2	22		79.3939
Sig.		.058	.120

Uses Harmonic Mean Sample Size = 21.035.
The group sizes are unequal. The harmonic mean of the group sizes is used. Type I error levels are not guaranteed.

Table 5.23 Project vs individual test means for 2005-S1groups in homogeneous subsets

What can be inferred is that students who are good at the individual tests are not able to extend their conceptual knowledge to their project work or do not bother to expend effort in the project work. This could be attributed to the tight assessment schedule (refer to figure 4.4), written test and lack of context focus in the traditional structured environment. There is an inconsistent learning behaviour between students who focus on performing well in tests yet not on projects and those who could not perform in tests focus on project. Hence, the results do not meet learning objectives.

Case 2006-S1 module groups:

2006-S1			Sum of Squares	Df	Mean Square	F	Sig.
Project	Between Groups	(Combined)	6056.277	10	605.628	3.743	.000
	Within Groups		38186.271	236	161.806		
	Total		44242.549	246			
Individual Test	Between Groups	(Combined)	6532.421	10	653.242	4.050	.000
	Within Groups		38070.075	236	161.314		
	Total		44602.496	246			

Table 5.24 One-way ANOVA for project score and individual test score for groups in 2006-S1

ANOVAs of the project score and the individual test are performed as shown in table 5.24 which highlights significant differences in their means, $p < 0.05$. Effect size for project: ω = sqrt((6056.277 -10*161.806)/ (44242.549 + 161.806)) = 0.316; effect size for individual test: ω = sqrt((6532.421 -10*161.314)/ (44602.496 + 161.314)) = 0.332. Both project and individual test mean scores indicate modest effect in total variance.

2006-S1: Project mean: 67.0830				
		Subset for alpha = .05		
Group	N	1	2	3
9	22	56.4545		
6	22	59.5682	59.5682	
5	22	65.2727	65.2727	65.2727
8	24	66.4792	66.4792	66.4792
3	23	66.6957	66.6957	66.6957
1	21	67.4762	67.4762	67.4762
11	21	68.5476	68.5476	68.5476
10	23		69.7826	69.7826
2	23		70.6087	70.6087
4	22		72.2500	72.2500
7	24			74.0208
Sig.		.086	.052	.692

Individual test mean: 69.1116			
		Subset for alpha = .05	
Group	N	1	2
9	22	59.3495	
3	23	59.6641	
8	24	67.1206	67.1206
6	22	68.8257	68.8257
1	21	68.8364	68.8364
11	21	70.1614	70.1614
4	22	71.6300	71.6300
2	23	71.6673	71.6673
10	23		72.6153
5	22		74.0711
7	24		76.0280
Sig.		.070	.650

a Uses Harmonic Mean Sample Size = 22.411.
b The group sizes are unequal. The harmonic mean of the group sizes is used. Type I error levels are not guaranteed.

Table 5.25 Project vs individual test means for 2006-S1 groups in homogeneous subsets

The homogeneous subsets are not statistically significant, $p > 0.05$ as indicated in table 5.25. Again group 9 has the lowest mean score for project and individual test though the difference is less than -3 points (56.5 – 59.4 = -2.9) and its significantly different with 4 groups in project score and 3 groups in individual test score. There are 3 failed students in group 9 with 2 students who were debarred because of non-attendance i.e. their scores are not considered for this module. This indicates that group 9 has low cooperative group processing (Johnson and Johnson, 2004) which does not facilitate competence in computer programming. There are 9 other students who failed this module: 2 from groups 1 and 3 and each from group 4, 5, 6, 7 and 11. It is not within the scope of this study to examine why students failed and their reasons for non-attendance, non-submission or non-engagement in the PrC module.

Comparing case 2005-S1 and case 2006-S1 in terms of the module groups' performance, there is a gap in terms of each cohorts' competence in computer programming. The mean score difference between the project and individual test for every group is computed to verify this discrepancy. As seen in table 5.26, it is apparent that case 2006-S1 has lower difference compared to case 2005-S1. Only 3 groups in case 2006-S1 has more than 3 points difference (groups 6, 5 and 3) compared to 7 groups in case 2005-S1. This implies that module groups in case 2006-S1 has performed consistently in the project work and individual test compared to module groups from case 2005-S1.

2005-S1 Group	Project – Individual Test	Project – Individual Test	2006-S1 Group
2	-24.738	-9.2575	6
11	-15.3057	-8.7984	5
8	-9.3511	-2.895	9
9	-7.4452	-2.8327	10
1	-5.3844	-2.0072	7
5	-3.6704	-1.6138	11
3	-2.8997	-1.3602	1
4	-2.3508	-1.0586	2
6	-1.8814	-0.6414	8
7	-0.202	0.62	4
10	6.0127	7.0316	3

Table 5.26 Project less individual test mean difference in ascending order

Thus, another hypothesis is generated to reflect the new evidence on module group performance:

H11B: Module groups in the blended learning environment exhibit consistent performance compared to those in the traditional structured environment.

5.6 Summary of hypotheses

The quantitative statistical analysis in this chapter supports the qualitative empirical analysis in chapter 4. Clearly, students with prior computing experience are able to perform in the blended learning approach over those in the traditional structured learning. For novice computer programmers, there is sustained performance in the blended learning approach. Putting together the hypotheses from the previous and the current chapter, table 5.27 shows the summary of hypotheses and the corresponding evidence.

		Hypothesis Description	Evidence
1.	H1 ✓	The change in curriculum from traditional structured approach to an integrated blended approach has improved the learning focus in computer programming.	Curriculum learning objectives, section 4.1, tables 4.1 & 4.2
2.	H2A ✓	The reduction in curriculum time from 6 hours per week to 4 hours per week has increased hands-on practice for students and has not affected face-to-face interaction between students and tutors.	Curriculum delivery, section 4.2, table 4.4
3.	H3A ✓	The changes in the programming development environment for case 2006-S1 have improved students' competence in computer programming.	Computing environment, sec. 4.2, table 4.5, fig. 4.1 & 4.2
4.	H4A ✓	The teaching-learning activities found in the blended learning approach are context-driven and holistically integrated which improve students' learning focus.	Teaching-Learning programming concepts, sec. 4.3, fig. 4.3

		Hypothesis Description	Evidence
5.	H5 ✓	The change from structured project work to problem-based project work has improved students' learning focus in computer programming.	Teaching-Learning project work, sec. 4.3, fig. 4.4
6.	H6A ✓	Closed individual written test and tutorial assignments are not aligned to the assessment of the students' learning focus in computer programming.	Assessments, section 4.4, tables 4.6 & 4.7, fig. 4.5
7.	H7 ✓	The blended learning approach is constructively aligned to the students' learning focus in computer programming.	Constructive alignment, section 4.5, figs. 4.6 – 4.8
8.	H8A ≈	The blended learning environment has sustained students' module score in computer programming.	Module score, section 5.2, tables 5.6, 5.7, 5.10 – 5.12
9.	H8B ✓	The blended learning environment has improved students' module score for those with higher certificates (prior computing knowledge).	Module score, section 5.2, tables 5.8 & 5.9
10.	H9 ✓	The blended learning environment has improved students' project scores in computer programming.	Project score, section 5.3, tables 5.13 – 5.15
11.	H10A ≈	The blended learning environment has sustained students' individual test scores in computer programming.	Individual test score, section 5.4, tables 5.16 & 5.17
12.	H11A	There are significant differences in the module performance across all module groups in both the traditional structured and the blended learning environments.	Module group analysis, section 5.5, tables 5.18 – 5.21
13.	H11B ≈	Module groups in the blended learning environment exhibit consistent performance compared to those in the traditional structured environment.	Module group analysis, section 5.5, tables 5.22 – 5.26

Table 5.27 Summary of hypothesis and its supporting evidence
(✓ denotes supported and ≈ denotes promising)

The final analysis in this comparative study indicates that 9 out of the 13 hypotheses explored support the integrated curriculum of case 2006-S1 (shown with ✓). Two hypotheses show promise and opportunity for improvement (marked with ≈) and the remaining hypothesis H11A indicates that there are gaps in the group performance. Although the analysis may seem highly positive, 9 of the hypotheses have been reformulated to reflect the scope of the evidence and only 4 of the original hypotheses are supported without modifications. A comparative table of case 2005-S1 and 2006-S1 (table 5.28) is produced to summarise the analysis in this study.

Analysis \ Case	2005-S1	2006-S1	Evidence
Curriculum scope	Topics covered do not correspond to learning objectives	Topics are covered according to learning objectives	Curriculum learning objectives, section 4.1, tables 4.1 & 4.2
Learning Environment	90 hours comprising lecture, e-learning, practical, tutorial	60 hours comprising e-lecture and practical	Curriculum delivery, section 4.2, table 4.4
Computing environment	C++ using visual studio 6, e-learning objects based on topics	C# using visual studio 2005, e-learning support for topics, project & pragmatics	Computing environment, sec. 4.2, table 4.5, fig. 4.1 & 4.2
Teaching-Learning activities	Lecture focus, printed handbook, study materials organised according to lecture	Context driven, practice and e-learning designed in a single context or concept	Teaching-Learning programming concepts, section 4.3, fig. 4.3
Project work activities	Structured specifications	Open ended specifications, problem-based approach	Teaching-Learning project work, section 4.3, fig. 4.4
Assessment methods	Written test, practical test, project work and tutorial assignments	Practical test, project work and online assessment	Assessments, section 4.4, tables 4.6 & 4.7, fig. 4.5
Constructive alignment	Teaching and learning are focused on lecture materials, as well as written test. Not align with learning objectives.	Teaching, learning and assessment systems are reinforcing one another, results in congruence of topics.	Constructive alignment, section 4.5, figs. 4.6 – 4.8
Module performance	Unequal across novice programmers and those with prior computing	Sustained performance for novices and improved for those with prior computing	Module score, section 5.2, tables 5.6 – 5.12
Project performance	Students perform below module performance	Students perform better than module performance	Project score, section 5.3, tables 5.13 – 5.15
Individual performance	Unequal distribution of marks and unequal variances	Sustained performance and variances	Individual test score, section 5.4, tables 5.16 & 5.17
Group performance	Significant differences in means across groups, and inconsistent performance across project and individual scores.	Significant differences in means across groups, yet consistent performance across project and individual scores.	Module group analysis, section 5.5, tables 5.18 – 5.26

Table 5.28 Comparative table for analysis of curriculum, teaching-learning activities and assessment

5.7 Supplementary data

Student and tutor feedback

The following information is obtained through the polytechnic's semestral feedback process. Students are required to submit online feedback on every module they attend in study week 14 of the semester.

	Student Feedback question	Case 2005-S1	Case 2006-S1
1	Provision of module materials	1.77	2.09
2	Provision of laboratory equipment and facilities	1.67	2.05
3	Use of good quality and effective teaching aids (e-Learning)	1.73	2.09
4	Presentation of topics/lectures	1.72	2.09
5	Explanation of topics/lectures	1.71	2.18
6	Conducting of practical sessions	1.75	2.06
7	Tutorials conducted	1.64	Nil
8	Access to/availability of lecturers/tutors for discussions/on module consultations	1.77	2.08
9	Overall rating of course delivery	1.74	2.12

Average Rating = ((Excellent x 1) + (Good x 2) + (Marginal x 3) + (Poor x 4))/Total of Respondents

Table 5.29 Results of student feedback

As seen in table 5.29, case 2006-S1's overall rating has decreased slightly by 0.38 points (1.74, 2.12). It is interesting to note that case 2005-S1 cohort has checked the tutorial highest of all the items on the questionnaire. Lowest score (1.77) goes to the study material which comes mainly in a printed handbook and the access to/availability of tutors which means students seek more face-to-face interaction. For case 2006-S1, the lowest score (2.18) goes to the explanation of topics and similar to case 2005-S1, students rate face-to-face sessions highly where conducting of practical session rate 2.06.

Empirical evidence indicates that all tutors are new to C# programming language and to the Visual Studio 2005 development environment. In addition, this is the first time that the blended learning approach is stipulated by the department's course management on the PrC module. Out of five tutors, two tutors are part-time and may not have benefited from the extra training received by full-time staff. The two part-time staff are briefed by the module convener and guided by the study materials. Based on tutor meetings, the following comments were gathered –

A: I prefer the traditional method as we are given time to repeat and reinforce the concepts. I like tutorials best; it gets the students thinking without the distraction of the computers.

B: The blended learning is more focused, but time management is a challenge. We should not decrease the curriculum time – students need more hands-on practice.

C: Blended learning takes up too much time and effort. I cover all the presentation material for an hour and let the students complete the exercises in the remaining time.

D: I don't get enough time to spend with students (using the blended learning approach). I get disoriented with the e-learning and the presentation materials being integrated this way.

E: The e-lecture is a nightmare; I spend half the time getting the students to stop chatting or surfing and to focus on the study materials.

It seems that tutors need more support and time to get used to the blended learning approach as well as to gain more exposure so that they are comfortable with the technology and the pedagogy. Until they do, students will not gain the full benefit from the blended learning environment.

Other related studies

Before blended learning is implemented for cohort 2006-S1, preliminary studies have been conducted two years before in the polytechnic's department. These studies (NYP, 2005a, 2005b) have been audited by external parties to the department where this study is being conducted. Recommendations for improvement to the computing course were as follows:

- Tutors provide illustrative examples of programming code
- Supplement teaching activities with e-learning material
- More hands-on practice for students, make free access laboratories available for students
- Continuous assessment is to be stipulated and endorsed so as to give students more topical assessments and to encourage self-evaluation and feedback for students.
- Reinforce the student buddy system to get student leaders to support weaker students.
- Tutors to monitor students' progress closely so that timely remedial action can be provided for weak students.

The recommendations above have shaped the teaching-learning materials developed for case 2006-S1 and are meant to improve students' learning of computer programming. A number of resources and investment have gone into this study's innovative strategy. The next chapter will discuss the impacts and trade-offs of the blended learning approach over the traditional structured approach.

Chapter 6 Discussion

Findings from the analyses completed in chapters 4 and 5 are discussed with respect to the theoretical constructs and frameworks raised in earlier chapters. As much as possible, the discussion attempts to reveal where and how these findings provide a rich context for deeper analysis and understanding of integrated curriculum and blended learning. Initially, the discussion examines this present study's findings with past studies (section 6.1) which were briefly reviewed in chapter 2, to see if there is new information, similarities or gaps to be addressed. Subsequently, the discussion inquires into the research questions (section 6.2) to see what answers are revealed and further inferences can be made; how well the research methodology has been applied (section 6.3); the strengths or contributions (section 6.4) and the limitations of the study (section 6.5). A list of recommendations is suggested at the end of this discussion (section 6.6) in view of the evidence to support the student and the tutor to this new challenging initiative of blended learning in computer programming.

6.1 Findings and related literature

Teaching introductory computer programming modules with blended learning has been successfully applied at the London Metropolitan University (Boyle, 2005, et al., 2003); the programming language taught is Java[3]. The blended learning approach comprises lectures, small-group tutorials of 15 students and e-learning objects made available in a virtual learning environment. The online learning is supplementary and as reported by Bradley and Boyle (2004), only 10% of the students do not find the online learning useful or do not make use of it at all. Student questionnaires are used to evaluate the course and students' e-learning sessions are logged by the system.

> One conclusion that can be drawn, is that although our student population is diverse in respect of its range of abilities and previous experiences, it demonstrates a range of use of the available resources, and shows that students are adopting learning patterns to suit their individual needs and goals. (Bradley and Boyle, 2004)

Although the researchers claim a pass rate increase of 12-23% after the first year, it is not apparent how the students have been assessed with the blended learning approach. When compared with the current study, the passing rate of the PrC module

[3] A computer programming language developed by Sun Microsystems (www.webopedia.com). Java is a general purpose programming language with a number of features that make the language well suited for use on the World Wide Web.

is above 96% (derived from table 5.8). Furthermore, since 20% of the online assessments are based on the study and online materials, all students have to refer to the online materials of the PrC module.

Similarly, Sayer et al. (2004) has applied e-learning objects into an introductory Java programming course. Online assessments are available to help students with self assessment. It is not clear how the e-learning is incorporated into the course delivery and if the e-learning is optional or compulsory. Student survey shows highly positive feedback and students' performance is measured against the previous cohort that does not have online learning. It is discovered that the students' performance in the final exams has increased by 2.3% over the earlier cohort but the students' continuous assessment score has reduced by 10%; the number of students in the present cohort has increased by 36.5%. The authors explain the discrepancies as:

> Perhaps, however, we now have a more accurate picture of the programming ability of the students who were assessed using WebCT, since it has been found in studies of non on-line assessment strategies that there are often significant inconsistencies in the actual programming skills of some students and their coursework scores, with high scores related to students with weak programming skills. (Sayers et al., 2004)

It appears that although the students' performance is not impressive with online learning, there are intangible benefits to be gained namely in students' motivation and the competence of programming skills. Their analysis between test scores and programming skills is a reflection of this study's findings with respect to module groups' performance in project versus individual test (refer to section 5.5). Thus the notion of constructivist instructional strategies like scaffolding and collaborative learning plays an important role in building students' competence in computing. Combined with constructive alignment of the learning objectives, the aim is to enable students to excel in all forms of assessments and not targeting one at the expense of the other.

In the research by Heinze et al. (2004, 2006), blended learning is employed for a part-time IT course integrating a virtual learning environment and face-to-face interactions. Based on Laurillard's Conversational Framework (2002), iterative learning cycles are designed to allow the student to communicate with the teacher in action and feedback modes. However, the modules covered by Heinze et al. are Project Management and Systems Analysis and Design; students attend a face-to-face session each week and are supported between sessions with online discussion tasks. Nevertheless, Heinze et al. report that the teachers' willingness to incorporate blended learning and the

students' willingness to engage in the conversations are the main issues that require further investigation.

> Blended Learning means that less time will be spent face to face in the classroom, which distances the Learner and the Teacher and unless there is effort being put into interacting with each other, there is a chance that there may be no dialogue between the learner and the teacher. (Heinze, et al. 2006, p.11)

The findings reported by Heinze are collected from student and tutor focus groups, but since the teaching-learning activities and learning objectives are not reported by Heinze, there is no way to see if the blended learning is constructively aligned to the students' learning focus. However, the caveat raised on the lack of dialogue between learner and teacher in blended learning highlights the importance of reflection and flexibility (refer to section 2.5). The teacher has to actively seek dialogues through scaffolding or problem based questions; however, in a classroom situation whilst attempting to complete the lesson, this is easier said than done. Tutor comments from this study correspond to Heinze's observations.

Another pedagogical strategy for teaching computer programming to novice programmers can be found in the Active Learning in Computing (ALiC) paradigm (Sheridan-Ross et al., 2007, Lavery et al., 2006 and Hatch and Burd, 2006) where constructivist approaches promote higher levels of learner engagement. A holistic learner environment is created not only with online learning tools but include a specialised Techno-Café (Hatch and Burd, 2006). The instructional strategies applied by the authors such as peer group support and collaborative project work are similar to those implemented in this study. Although the researchers report high student and tutor satisfaction, quantitative measures on students' achievement or learning involvement are not available. This could be attributed to the project's Phase 1 stretching between 2005 and 2007. Statistical analysis of the studies found in the literature is either sparse or non-existent. A possible explanation could be that rigorous statistical analysis involving t-tests or analysis of variance is resource intensive especially with respect to time and data. Current discourse in blended learning is more concerned with qualitative evaluations (as supported by Sharpe et al., 2006) and theoretical frameworks as discussed in chapter 2.

Numerous studies have been conducted incorporating online learning or learning technology into regular subjects or course modules (Savenye et al., 2004, Jochems et al., 2004, and MOE, 2007a) through the use of anchor concept maps, online portfolios and virtual learning environments (as discussed in sections 2.4 and 2.5). However, a majority of these past studies do not give a full account of the curriculum, the learning

objectives, the teaching activities and the assessments; claims on student satisfaction and learning gains are cursory and lack validity. Many institutions have no problem collecting cases of blended learning and evaluating their merits or limitations (Sharpe et al., 2006). In addition, most case studies are stored in a virtual learning environment that is readily accessible. However as noted by Sharpe, there is little evidence of academic staff applying the cases to help themselves without mediation or direction from management or dedicated e-learning teams. Few studies are clear about how the students' have improved their learning based on the e-learning integration or interaction. Davies and Graff (2005) highlight the issue that the association of online activity to students' final grades is not linear; higher levels of participation do not necessarily lead to higher grades. This current study has shown similar results, however, it is differentiated from previous studies in that the constructive alignment strategy has been applied to evaluate the effectiveness of the teaching-learning activities in the blended learning environment.

Introductory courses to computer programming tend to cover the same fundamental knowledge (Irons et al., 2004) and in some institutions, the same course is provided for all students at degree and post-graduate levels (Boyle, 2005). In this study, it is revealed that the syllabus for the PrC module is the same for both courses but since each case utilises a different programming language, the study materials differ. What emerges from the analysis is the way the integrated blended learning approach is able to focus on context and constructive alignment of the curriculum, teaching-learning activities and assessment. This is supported by studies on transforming study materials to reflect constructive alignment (Biggs, 2003, Webb, 2002 and Anderson et al., 2001). In this manner, students are able to develop holistic learning strategies to mastering computing programming constructs (as proposed by Robins et al., 2003). Knowledge levels identified in this study (refer to section 4.1) is derived from the cognitive skills of computing instruction which combines cognitive theory from Pea (et al., 1983) and knowledge theory from Anderson (et al., 2001). This derivation of factual, design, analogical, conditional, procedural and deductive knowledge forms differs from previous studies on teaching computing that identifies general levels of cognitive and meta-cognitive skills (Robins et al., 2003), those that focus on software engineering knowledge (Irons et al., 2004) and those that are concerned with programmed instruction or instructional design (Burton et al., 2004). The knowledge forms have generated clearer understanding of the learning outcomes and its alignment with the assessments and teaching-learning activities.

The theory of constructivist learning supports the notion of the learner constructing meaning from a combination of previous knowledge and the current knowledge

(Fosnot, 2005). Central to the constructivist theme is its learner centredness; the fact that it allows other instructional strategies such as scaffolding, collaborative learning and problem-based learning to be combined makes it a transformative and innovative solution (Dennen, 2004). In this study, constructivist instructional strategies have been incorporated into the teaching-learning activities and the e-learning objects of the blended learning environment. Bach (et al., 2007) reports on various studies of blended learning that employ online exercises to develop analytical analysis, collaborative learning and problem solving methodology. The tutor acts as moderator and the discussions involve both synchronous and asynchronous modes. In this study, the online activities that are facilitated by the tutor are asynchronous but peer-to-peer discussions and collaboration allow for both modes. Although peer-to-peer assessments are not measured, the evidence shows a better project performance for students in the blended learning environment. Similar to this study, Bach finds that student feedback indicates a preference for tutor guidance and explanations in the online activities. This revelation of students' unwillingness to venture into independent self directed learning is further supported by Dennen (2004). Dennen proposes that tutors act as mentors and coaches to provide scaffolding to online learners.

In summary, the blended learning approach has benefited over the traditional structured approach by applying various theoretical and pragmatic frameworks to the curriculum objectives, teaching-learning activities and assessment tasks. Hence, the blended learning approach has engendered constructive alignment by integrating 1) cognitive knowledge forms, 2) SOLO taxonomy to assess students' understanding and competence, 3) constructivist learning approaches in teaching-learning sessions, 4) interactive computing development environment and 5) rigorous e-learning support.

6.2 Research questions – how well answered

The aim of this current study is to evaluate the efficacy of the integrated curriculum which combines or blends various pedagogies and instructional strategies with online technology in the teaching of computer programming in a polytechnic or competence based environment (refer to section 1.2). To maximise the evaluation in this study, the integrated curriculum is compared to its predecessor i.e. the traditional, structured approach.

The main research questions as stated in section 1.5 are:

Q1. In what ways have the change in curriculum from a traditional structured approach to an integrated, blended learning framework influence the students' learning in computer programming?

Q2. To what extent are the assessments affected by the traditional structured curriculum and the integrated, blended learning curriculum?

Q3. How have students performed in the computer programming module in the traditional structured environment compared to the integrated, blended learning environment?

Initially the hypotheses were generated from the research questions (as seen in section 3.5); as the analytic induction progresses, those hypotheses that are not reaffirmed have been modified to reflect the evidence found in the analysis.

		Hypothesis Description	Evidence
Q1	H1	The change in curriculum from traditional structured approach to an integrated blended approach has improved the learning focus in computer programming.	Curriculum learning objectives, section 4.1, tables 4.1 & 4.2
Q1	H2A	The reduction in curriculum time from 6 hours per week to 4 hours per week has increased hands-on practice for students and has not affected face-to-face interaction between students and tutors.	Curriculum delivery, section 4.2, table 4.4
Q1	H3A	The changes in the programming development environment for case 2006-S1 have improved students' competence in computer programming.	Computing environment, sec. 4.2, table 4.5, fig. 4.1 & 4.2
Q1	H4A	The teaching-learning activities found in the blended learning approach are context-driven and holistically integrated which improve students' learning focus.	Teaching-Learning programming concepts, sec. 4.3, fig. 4.3
Q1	H5	The change from structured project work to problem-based project work has improved students' learning focus in computer programming.	Teaching-Learning project work, sec. 4.3, fig. 4.4
Q2	H6A	Closed individual written test and tutorial assignments are not aligned to the assessment of the students' learning focus in computer programming.	Assessments, section 4.4, tables 4.6 & 4.7, fig. 4.5
Q2	H7	The blended learning approach is constructively aligned to the students' learning focus in computer programming.	Constructive alignment, section 4.5, figs. 4.6 – 4.8
Q3	H8A	The blended learning environment has sustained students' module score in computer programming.	Module score, section 5.2, tables 5.6, 5.7, 5.10 – 5.12
Q3	H8B	The blended learning environment has improved students' module score for those with higher certificates (prior computing knowledge).	Module score, section 5.2, tables 5.8 & 5.9
Q3	H9	The blended learning environment has improved students' project scores in computer programming.	Project score, section 5.3, tables 5.13 – 5.15
Q3	H10A	The blended learning environment has sustained students' individual test scores in computer programming.	Individual test score, section 5.4, tables 5.16 & 5.17
Q3	H11A	There are significant differences in the module performance across all module groups in both the traditional structured and the blended learning environments.	Module group analysis, section 5.5, tables 5.18 – 5.21
Q3	H11B	Module groups in the blended learning environment exhibit consistent performance compared to those in the traditional structured environment.	Module group analysis, section 5.5, tables 5.22 – 5.26

Table 6.1 Summary of research question, hypothesis and its supporting evidence

Table 6.1 gives a summary of the hypotheses that have answered the research questions respectively. The first question, Q1, concerns the curriculum and the hypotheses that answered Q1 are H1 to H5 as seen in table 6.1. There are five different areas where the integrated curriculum has improved over the traditional structured curriculum. What can be inferred from these improvements is that the alignment of the curriculum objectives with the learning objectives improves the learning focus; the teaching-learning activities are designed based on the learning focus; hence the teaching-learning activities become context driven which assists students in their understanding and practice. The integrated curriculum based on blended learning has raised the learning focus for computer programming compared to the traditional structured curriculum. Figure 6.1 shows the relationships of the areas as discovered in the analysis. The learning environment and the computing environment influence one another as shown by the curved arrows; similarly the teaching-learning activities and the project work support the students symbiotically. As an integrated whole, H1 to H5 promotes a holistic learning environment for the learner where one area enriches the other.

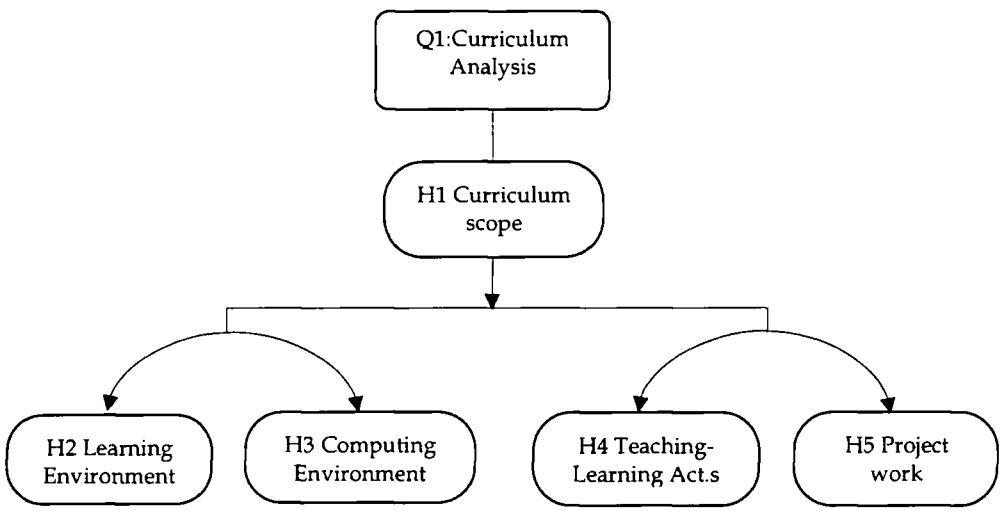


Figure 6.1 Q1 – Curriculum and its influence on teaching-learning areas

The second question, Q2, explores how well the assessments correspond or are aligned to the curriculum and learning objectives. This study highlights the way blended learning has enabled the teaching, the learning and the assessments to be constructively aligned; and ensures that students obtain continuous feedback which encourages students to improve their learning. In the traditional structured system, a linear process begins from the curriculum objectives to the teaching system where the assessments are derived from the teaching material (refer to table 4.7); whereas in the blended learning system, the assessments were derived from the curriculum objectives

and the teaching system is developed based on assessment requirements; evaluation and feedback completes the synergistic cycle, highlighted in figure 6.2.

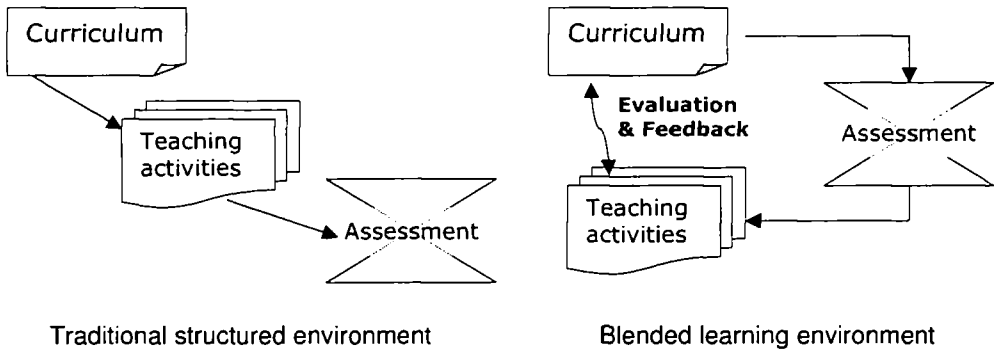


Figure 6.2 Q2 – Assessment and its relation to teaching and learning

Finally, the third question, Q3, examines how students have performed and see if there is any marked improvement between the two cases. According to this study, the results of the module score for all students are equivalent across both cases (H8A). Comparisons across the cases proceeded at different levels as shown in figure 6.3 to examine how different groups of students fared. Differences in module mean scores between GCE 'O' level aggregates are examined but are not statistically significant; thus, denoted by dotted arrows. It is revealed that students with prior computing knowledge perform better than novices and in particular the blended learning approach has significantly assisted the former students over the traditional structured approach (H8B).

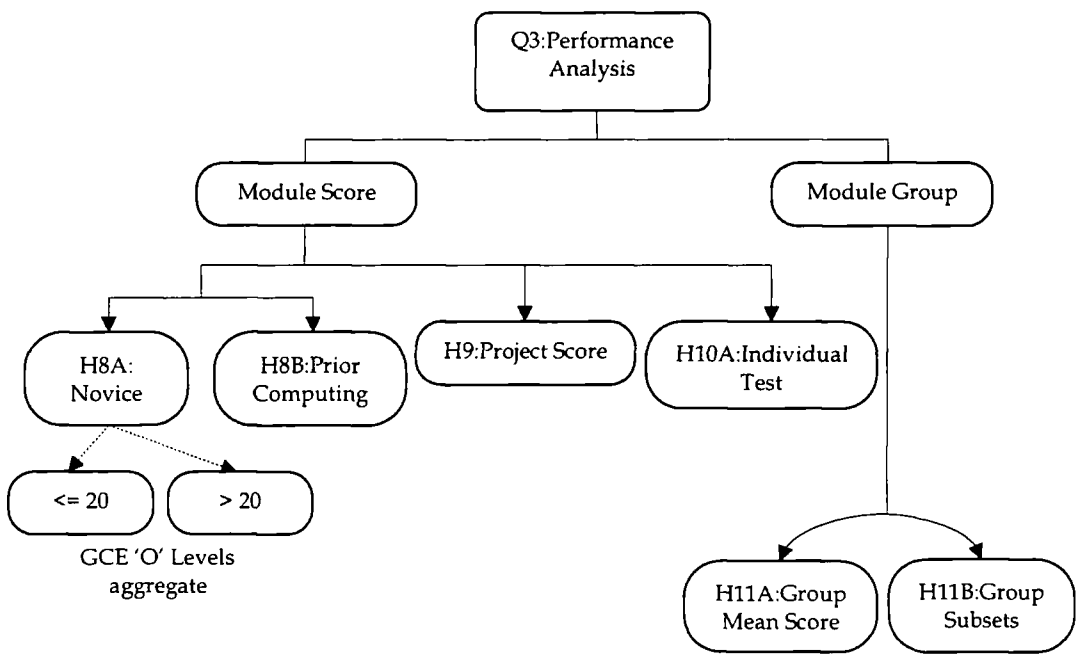


Figure 6.3 Q3 – Students' performance in the PrC module based on statistical analysis

In addition, the analysis discovers that students in the blended learning approach score better in project work (H9). Switching to the problem-based project work (H5) has improved students' project score (H9). Similarly, students' module score (H8A) and individual test score (H10A) have been sustained as a result of the way assessments have been aligned with the learning objectives. This implies that assessments drive the students' learning and since the assessments are supporting the learning objectives, students' achievements are tied to the curriculum objectives as well. In terms of group performance, it is discovered that students' aptitude measured through GCE 'O' level aggregate and prior computing knowledge affect the students' performance (H8A, H8B).

Variations in group performance show similar impact of entry level aggregate (H11A) across both cases. Further investigation on the subsets of module means within each case reveals that the students in the blended learning environment (case 2006-S1) have displayed consistent performance over those in the traditional structured environment (H11B). Hence the group dynamics (Johnson and Johnson, 2004) contribute to the group mean score, illustrated as curved arrows in figure 6.3.

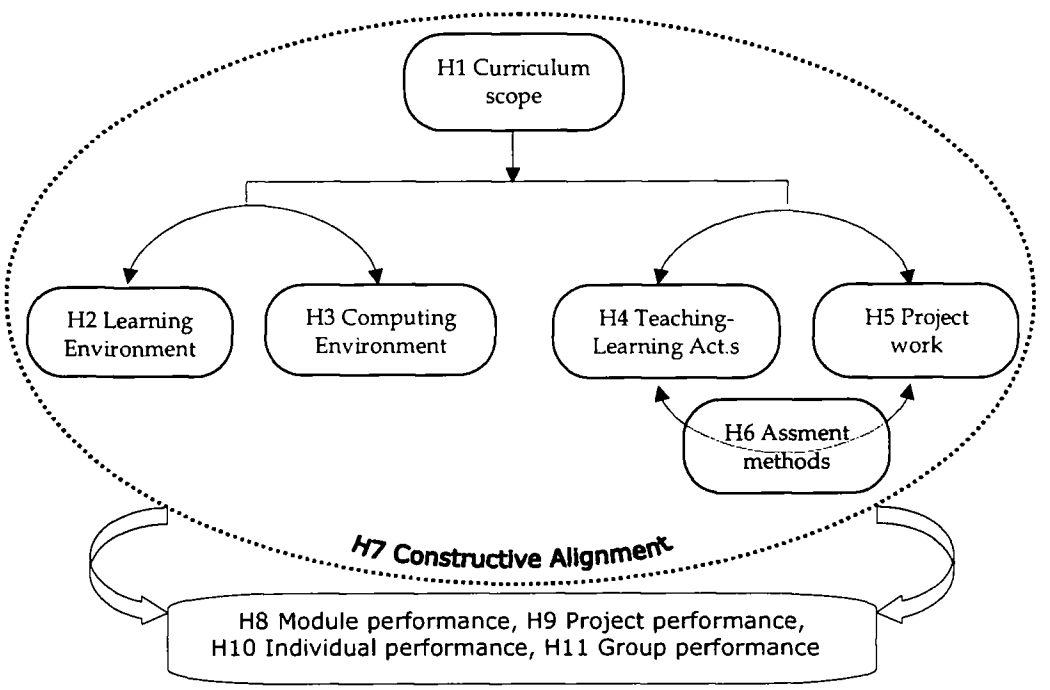


Figure 6.4 Conceptual map of the integrated blended learning environment based on hypotheses covered

Based on the hypotheses that have been affirmed as a result of the investigation of the research questions, a summary of the overall findings of the integrated blended learning environment is presented in figure 6.4; the holistic learning environment is sustained by the constructive alignment which in turn translates to the students'

performance. In the integrated curriculum as revealed in this study, 80% of the assessment is practice oriented and this makes hands-on practice significant. It can be inferred that the practical assessments induce students to spend more time with hands-on practice. Furthermore, students are encouraged to learn not just from their tutor but also from each other in a cooperative, collaborative manner. Further evidence from student and tutor feedback reveal that students find tutors' explanation lacking in the integrated curriculum and that tutors are new to the integrated curriculum as well as the blended learning environment. To reflect on the evidence, it is inferred that module mean score is maintained despite tutors and students being new to the blended learning environment.

At the beginning of this study (in section 2.2), it is argued that computer programming is a skill or competence based task and is more appropriately viewed not as a subject matter but more of the integration of domains such as communications, information processing and programmed instructions and so forth. Hence the teaching of computer programming should be seen as an integrated curriculum where real-world, meaningful learning activities combine theory, practice and online learning. Accordingly (in section 2.5), blended learning is put forth as the most appropriate learning pedagogy which also supports an integrated learning environment of face-to-face interactions with online learning. Through the comparative case studies of the traditional structured approach to that of the blended learning approach, it is found that the traditional structured approach has not constructively aligned its curriculum, teaching, learning and assessment objectives in the way that blended learning has managed to perform. In this respect, the answers provided to the first and second research questions have been successfully answered. Answers to the third question has revealed several tensions namely in the way tutors and students engage in the blended learning environment. Since issues of motivation and engagement are not within the scope of this study (see section 1.7), information regarding these gaps are not collected. Nonetheless, the findings has allowed this research to form new understanding and contribute new insights into the implementation of the integrated curriculum in a competence based environment in general and that of the blended learning for computer programming in particular.

6.3 Research methodology – how well applied

The comparative case study methodology employed by this study has afforded a more intuitive analysis into the capabilities of the integrated curriculum as compared to the traditional structured curriculum. Yin (2003) defines a case study as a research strategy that *'investigates a contemporary phenomenon within its real-life context, especially when the boundaries between phenomenon and context are not clearly*

evident.' (p. 13). As discussed in section 3.2, the comparative case study approach lends validity to the various findings of this study. As such, it has facilitated evidence to be thoroughly investigated in chapters 4 and 5, and results to be duly explained in the preceding section 6.2. Nevertheless, this section serves to evaluate the current study in terms of its research methodology as a means to provide a richer exploration of the issues arising from the case study research and to provide possible solutions to overcome any limitations. As observed by McMillan and Wergin (2005), the aim of a methodology is to help the researcher to understand the research process rather than the products of the scientific inquiry.

In a special report on blended e-learning research and practices, Sharpe et al. (2006) reveal that a majority of current research in Europe including United Kingdom, America and even Australia employ case study methodologies. This finding is not surprising and is in line with the observation made by Gomm (et al., 2000) that innovations especially that involving education or online technologies are most suited to the case study methodology. Hence, this research is well placed to gain the benefits of the case study methodology.

In order to evaluate the robustness of this study's research findings as well as the integrity of the research in terms of validity and reliability, the following criteria (as discussed earlier, refer to section 3.2) are now applied to the evidence as follows:

- Construct Validity: From the content analysis of the curriculum documents, the information is summarised in tables 4.1 to 4.7. Actual documents of curriculum aims, objectives and topics are available in appendix B. Applying participant observation and supported by the related theories, a chain of evidence is explained for all the constructs found in hypotheses.
- Ecological Validity: Documents used in this study (found in appendices B to E) are controlled documents within the polytechnic and verified by the internal audit team and the course management committee. Student information and module timetable are gathered from the controlled student management database within the polytechnic. This study is based on two actual cohorts that have gone through the PrC module in their first semester in the first year of their course of study. The students' general and entry level data are captured in student management database as well as their module scores.
- Internal Validity: By applying eliminative and analytic inductions, each hypothesis is being matched with the evidence. Quantitative analysis is performed on the students' performance in the PrC module as a form of triangulation to add to the internal validity and to support the construct validity of this study.

- External Validity: The constructive alignment model (figure 4.5 in section 4.5) is derived from the theoretical framework of Biggs (2003) and the curriculum characteristics from Posner (2004). It lends theoretical validity to this research and may be adapted to other subject matter applying similar alignment concepts.
- Reliability: The threat of researcher bias is mitigated by providing supplementary information (see section 5.7) of student and tutor feedback and recommendations from previous studies performed within the researcher's department. The statistical analysis performed in chapter 5 reduces the reflexivity of this study's findings.

Many past case studies included surveys on student satisfaction, focus groups of students and-or tutors and interviews (as researched in Bradley et al., 2004, Heinze et al., 2005 and Sheridan-Ross et al., 2007). Although some knowledge can be gained from these data, Sharpe et al. (2006) note that such studies fail to gather meaningful evidence of improvement in student learning and achievement; it also suffers the implication of some penalty such as the withdrawal of the learning environment or loss in self-esteem. More effective case studies are those that perform triangulation with students' performance in graded assessments (Sayers et al., 2004), server log files of students' online clicks (Bradley et al., 2004) and-or online assessments (Morris and Walker, 2006). This study has included students' performance where online assessment is an integral component of the student performance.

The research aims and objectives of this study have focused on the integrated curriculum and its implications on a competence based environment, in particular the introductory computer programming module. For the same reasons raised earlier that students' satisfaction do not correlate to better learning, understanding or performance, intensive surveys were not undertaken in this study. In terms of students' online activity, the e-learning materials provided to students in the PrC module can be downloaded to the students' portable computers. This is the recommended practice as opening the e-learning materials from the network server slows down the students' computers; another reason is to allow students who do not have internet access to review the materials. Thus, students' online activity cannot be measured from the network. This is not to deny that for a follow-up evaluation for development of the blended learning environment, student surveys (as subscribed by Scriven, 1995) and online activities should be measured (Huntley-Moore and Panter, 2006). Online activities can be extended to e-portfolios or blogs (Creanor et al., 2006)

Focus groups and interviews are found to be effective for studies that aim to evaluate the students' experience and-or learning as well as those of the tutors (Heinze, 2005, Creanor et al., 2006). The methods applied by the LEX (Learner's Experience of e-

learning) research study (Creanor et al., 2006) are comprehensive yet resource intensive as follows:

- pilot interviews and focus groups, using a semi structured interview format in conjunction with interview plus (presenting interviewees with their e-portfolio or diary or online assessments to prompt discussions)
- two methodology workshops for interviewers on the Interpretative Phenomenological Analysis (IPA) approach. The first workshop focused on questioning techniques and the second on IPA coding and analytical techniques.
- revised guidelines for interview handling in the light of the IPA investigation.
- piloting the rigorous IPA analytical approach and developing a teamwork approach to analysis.

Other forms of evaluating and analysing students' and tutors' engagements with the learning environments are ethnographic methods (McConnell, 2005) and active learning or action research methods (Sheridan-Ross et al., 2007). Due to resource constraints, these methods are not included in this study, thus, the research context of this study has been narrowed to the curriculum analysis. The issue of students' and tutors' perceptions has been duly noted in sections 5.6 and 6.2.

Statistical analysis undertaken in this study has applied parametric t-tests and analysis of variance. Assessments are subjective as marks are awarded by tutors and online test questions are subject to interpretations by students (McMillan and Wergin, 2005). As discovered by this study, statistical findings cannot determine the efficacy of students' learning and understanding of computer programming. A more effective method to explain causes from statistical results is to conduct interviews or focus groups as explained above. Nevertheless, by comparing the statistical findings with the qualitative findings (as discussed in sections 6.2), this study has successfully applied the concurrent triangulation strategy (refer to section 3.4). Issues with regard to group dynamics and students' motivation are duly raised; as well as those issues that concern the validity of the assessments on which the statistical analysis is based on are further discussed in section 6.5.

The comparative case study research design has been fully applied in this study to yield interesting results not only for blended learning but that for structured learning as well. The retrospective reconstruction of the cases has garnered realistic inferences and conclusions through reflective, introspective observation. With the combined analyses of curriculum and content, categories and summaries are raised and evaluated. Finally, the eliminative and analytic inductions of the hypotheses have

generated acceptable deductions and have resolved complex evidence found in this study.

6.4 Strengths of the current study: Contributions to research and practice

It is evident from the discussion that this research has benefited from the comparative case study design to illuminate the contributions to the pedagogy of teaching computer programming. This is a study that combines analysis of curriculum, learning outcomes, pedagogy and assessment – in itself an integration of different dimensions – to the promotion of transferring computing skills to novice programmers. According to Savenye and Robinson (2004), studies involving these dimensions are known as 'culture four' research and they find that there are fewer culture four research in education technology compared to instructional design and development research. Another contribution of this study is that the data and the analysis are based on empirical evidence in naturalistic settings. Thus the contexts drawn from this study, although not generalisable to wide populations, are relevant for comparability and translatability to other settings and cultures in similar situations (Cohen et al., 2007). With respect to software education research, this study has derived a curriculum-knowledge model (refer to table 4.1) based on cognitive skills in computing that can be applied for future curriculum alignment or verification. In addition, the constructive alignment model applied in this study (refer to figures 4.6 to 4.8) can be similarly adapted for evaluation or design purposes of an integrated curriculum.

This study has evaluated the integrated curriculum and its impact on the students' learning focus with respect to the curriculum aims, objectives and intended learning outcomes, the curriculum delivery and the computing environment. This has led to the evaluation of a context driven and holistic teaching-learning activities as well as e-learning objects for programming concepts as well as project work. This research has shown how the alignment of the teaching, the learning and the assessment systems has been successfully integrated into the curriculum to cultivate an interactive, progressive and contextualized learning environment (refer to figure 6.4). A summary of the key findings of this study is given in table 5.28; from eleven areas investigated, thirteen hypotheses which affirmed significant observations have been noted (refer to table 6.2). Further improvements are recommended at the end of this chapter based on significant observations made.

The blended learning approach has been found (refer to sections 4.3 and 6.1) to assist students in their project work; students are able to apply programming constructs and code examples from e-learning objects to their project work. This discovery adds to the growing literature on blended learning with the inclusion of collaborative and cooperative learning as well as learning from peers and continuous feedback. When

blended learning is employed alongside the integrated curriculum, it gives rise to a learning environment that exhibits congruence and positive reinforcement. Hence, this evidence adds to the discourse in instructional strategy in general and blended learning in particular.

A secondary opportunity afforded by the constructive alignment framework of this study is its application for course evaluation and course design, as well as for producing teaching plan for tutors and teaching-learning activities. Understanding the impact of assessments on learning, allow course managers or module conveners to plan assessment where the students' learning context and loading is appropriate. As shown by the evidence in this study, the assessments in the integrated curriculum has been planned to give maximum learning or revision for students in between assessments. As a consequence of the findings of the present study, the results may be of relevance to the application of learning technologies as well as the study of computer science education in general and software education in particular.

6.5 Limitations of the current study: Opportunities and risks

The evidence in this study indicates that assessments drive student learning. As observed by Popham (2006), to perform a thorough inventory of the types of assessment and the questions within each assessment and to relate the assessments to the students' learning and performance is the work of a doctoral thesis. Accordingly, the findings to such a study will add substance and validity to the curriculum analysis investigated in this study. Assessment evaluation according to Popham, comprises the procedures, the formative and summative components, the positive and negative effects, the performance rubrics as well as the reliability and the validity of the tests. Another consideration for computing tests is the impact of automated assessment tools and online question banks. This study has given some evidence of collaborative assessment in project work. An extended scope is to evaluate peer assessment and its usefulness in the students' performance. Topping (1998) has identified 17 variables that affect peer assessment ranging from curriculum to expected reward and 8 quality implementation factors for consideration from clarifying expectations to evaluating feedback.

The theoretical framework for computing knowledge that emerges from this study as well as its constructive alignment model has been described as the strengths of this study earlier; however, each has been derived separately. The constructive alignment model (refer to figure 4.6) has not shown how the teaching, learning and assessment systems correspond with the knowledge types or cognitive skills for computing (refer to table 4.1) i.e. factual, design, analogical, conditional, procedural and deductive. Further investigation is required to see if it is worthwhile to integrate a knowledge/skill

dimension into the constructive alignment model. The purpose of doing so is to enable educators to create teaching instructions (or learning activities) across the cognitive domain which amplify the constructive coherence of the instruction, learning and assessment constructs. This is an extension of the direction of Anderson and Krathwohl (2001) model and Webb (2002) alignment model. Table 6.2 illustrates a representation of the proposed taxonomy. Instead of ticks, the teacher may add a quantitative value or the exercise index into the box. The multi-dimensional approach lends a congruent thinking framework to the teaching instruction design model; hence adding shape to the alignment model which tends to be limited to a generalised linear or curve fitting paradigm.

	Knowledge types or cognitive skills in computing					
Alignment dimensions	Factual	Design	Analogical	Conditional	Procedural	Deductive
Teaching System						
Recognise if-else syntax	✓					
Code if-else construct		✓				
Execute code	✓					
Explain output and logic			✓	✓		
Show how to debug					✓	✓
Learning System						
Review e-learning	✓	✓	✓			
Code in C#				✓		
Execute code	✓					
Check output	✓		✓			
Examine logic				✓		✓
Assessment System						
D: Complete one exercise	✓					
C: Execute with different exercise				✓		
B: Explain logic				✓		✓
A: Explain with debugging or different examples			✓	✓	✓	✓

Table 6.2 Computing cognitive knowledge with constructive alignment taxonomy

A case study can be designed to see if the above table is meaningful and able to scaffold the novice programmer. It could also be extended to other subject domains and other assessment models such as a performance rubric to promote authentic assessment. As Stevens and Levi (2005) recommend, an effective rubric is one that not only scores students' work but also helps a teacher to conduct better instructions.

It is beyond the scope of this study to explore all the different facets and perspectives that contribute to the teaching and learning of computer programming. What the evidence reveals is that the relationships between peers and student-tutor are important and affect students' performance and learning. This is not surprising as the students have known teacher directed learning in their mainstream education and once

they have enrolled in higher education, these students now have to cope with a more open-ended, learner-based or self-directed learning. Cornford (2002) finds that good teachers can help to organise learning situations such that 'individuals (students) are forced to consider their own personal strengths and weaknesses, reflect on these, and learn from these experiences' (p. 361).

Based on the evidence and the literature discussed in this report, the following risks ought to be taken into consideration when implementing the blended learning framework:

⊗ Pedagogy before technology

Watson (2001) highlights the danger of assuming that 'technology will be the catalyst to create change' which is like placing the cart before the horse. In deciding which programming language to teach to novice programmers, the fundamental rule is to go for simplicity rather than the latest programming language which is unassured (Lippman et al., 2005). Similarly, in blending e-learning activities within teaching-learning activities, educators or tutors need to consider the aspects of knowledge analysis, evaluation and synthesis that can be gained from technology.

⊗ Tutor engagement

Tutors may not be aware of the blended learning styles and its benefits; or they may be unwilling to adapt to new techniques. Even for those who support the new changes, tutors may not be aware of their personal perceptions or prejudice and may confuse students with their delivery (James and Pedder, 2006). Tutors have to ensure that students communicate effectively, collaborate or share their knowledge, and that students receive fair and equal treatment even with different assessments (as championed by Gregory and Chapman, 2006).

⊗ Student engagement

Students are using trial and error to understand programming constructs or asking peers who are equally unclear; a case of the blind leading the blind. Students are not willing to focus on their learning and may be distracted by other interests and experiences. Students expect answers from tutors without trying on their own or seek answers from peers without understanding. Students may plagiarise and submit the work of others (Irons et al., 2004).

⊗ Management expectation

Management has to be aware of the efficient use of staff and student contact time and how this relates to the effectiveness of the blended learning approach. Resources and training have to be made available for the smooth operation of

the learning environment (Bates, 2005). Performance metrics and student achievements have to be gauged from understanding the other risks mentioned here.

6.6 Recommendations for improvement in practice

The discussion so far has shown that the integrated curriculum may be effective but lacks efficiency; blended learning framework is viable but needs long-term commitment to achieve sustainable performance. The following recommendations can be implemented for the PrC module in particular and for similar introductory computer programming modules in higher education.

1. Practice workshop

   One of the tensions noted in earlier sections, is the lack of tutorials that enables students to reflect and to raise concerns regarding their learning. One way around this issue is to develop a series of learner support workshops to allow students more face-to-face interaction with their tutors and to encourage 'learning how to learn' computing skills (Lavery et al., 2006). Being a workshop, lessons can be staggered fortnightly for 4 or 5 hourly sessions in the study semester. To ensure that students attend every session, students can be asked to submit their reflections or worked examples, online or written, at the discretion of the tutors.

2. Free access laboratory

   Students need more hands-on computing practice in order to master the computing constructs. Although they may be able to do this at their own homes, students can focus better in a laboratory environment. At Durham University, a customised laboratory was constructed with cubicles to allow for both group and individual work (Hatch and Burd, 2006). As photographed in figure 6.5, the customised laboratory contains booths that seat 6 to 8 persons, with recessed lights to reduce glare. Each booth is fitted with a 42-inch plasma display screen accompanied by a touch-screen interactive overlay; a tablet computer and another notebook computer, with network capabilities are attached and both computers are connected to the display screen. Feedback from students has been excellent plus it is the only laboratory where coffee is allowed. Although the total cost of more than £277,000 (above $680,000 Singapore currency) is prohibitive, a scaled down version will be able to provide similar benefits to students.

Figure 6.5 Techno-café at computer science department of Durham University (Hatch & Burd, 2006, p.4)

3. Continuous feedback

The evidence in this study indicates that feedback for the PrC module is only performed once in a semester (see section 5.7). Hounsell (2003) recommends that for evaluation of new technologies, feedback is best taken at periodic intervals so that students may recall their experiences and course designers or tutors have enough time to respond to issues raised in the feedback. Similarly in the blended learning environment, it is worthwhile to obtain an initial survey at the start of the term, an investigative survey in mid-term in week 8 and an overall survey in week 15. Each survey may be similar but should emphasise different aspects of the learning experience so that better corrective action can be taken into account.

4. Tutor collaborative support

According to a study performed by McKenzie (2001), teachers need to experience for themselves the differences between teacher-focused and student-focused approaches. A successful method is for tutors to observe one another in the classes that each tutor teaches; the observer acts like a student and experiences the active learning from the tutor. Constructive feedback and exchange will help tutors to improve in their learning how to teach in the blended learning environment. Tutors are also encouraged to record their experiences in some form, be it an online journal or written diary to reflect on their experiences. Network resources can be made available to allow tutors to reflect on best practices and to exchange ideas.

5. Personal development plan (PDP)

First year students being new to higher education are not certain about planning their learning and managing their time and work load. Although the scope of the PDP is outside of the PrC module, its implementation can only benefit all tutors and students. With the help of personal tutors, students create own PDP and each PDP is reviewed at monthly meetings which are arranged to ensure that students are on track and managing well. It is up to the personal tutors to cultivate positive group dynamics at these meetings to ensure that students remain committed to their learning and stay focused on their studies. Stevenson (2006) has reported highly positive results on PDP where personal tutors are able to provide academic guidance based on a holistic view of their students throughout their three-year study programme.

Chapter 7 Conclusion

Great opportunities are expected with technology advancements. With that in mind, this research moves into its final reflection to summarise the implications of the integrated curriculum for an introductory computer programming module (section 7.1). Critical reflections are shared (section 7.2) to summarise the observation-cum-evaluation process of this study. Finally, suggestions for follow-up research (section 7.3) are given to add different perspectives on the findings found in this study as well as to extend the ideas proposed within this study.

7.1 Summary of implications

A major implication of this study is its potential application in integrated curriculum development and evaluation for the course manager and the module convener. As stated at the beginning of this report (see section 1.4), one of the main significance of this research is to gain insight and understanding about ways in which course managers and developers can provide means and measures for supporting students towards learner success in blended learning environments. The integrated curriculum has many advantages but it could also lead to confusion if the curriculum objectives are not synchronized or aligned with the learning objectives. Where computer programming is concerned, the integrated curriculum is a necessary strategy in order to give students real-world needs of the IT industry.

This study has identified three dimensions to the blended learning approach for computer programming i.e. face-to-face (f2f) interactions, computing practice or pragmatics and e-learning. When these dimensions are integrated, a learning space ensues as shown in figure 7.1.

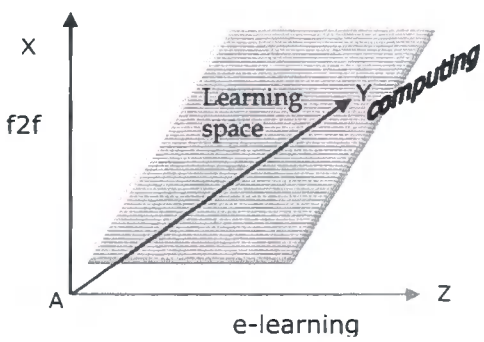


Figure 7.1 Learning space for computer programming

Learning with only face-to-face interactions (along the AX axis) implies the traditional learning of chalk and board without computers and without computing practice. This was how programming was taught in the 1980's and earlier when computers were expensive and software were not only expensive but also difficult to install, run and

maintain. With minimal or zero pragmatic knowledge, it takes at least six months to train a fresh IT graduate (Lohr, 2001). On the other hand, learning with only e-learning (along the AZ axis) is inefficient. This could be the situation of the distance learner who is not able to install the software for computing practice and he learns through correspondence. His fate is similar if not worse to that the pure f2f leaner. Finally, learning with only computing practice (along the AY axis) is a case of trial and error. Unless the learner is a computing genius, the learning curve is steep and fraught with unknown and unnecessary risks.

Having two out of the three dimensions deprive the learner of the rich context and real-world application of computer programming. Learning with f2f and computing means the learner has to develop self visualisation of technical concepts and constructs. As McAllister (et al., 2003) aptly points out, novice programmers are easily disheartened by the intricacies of computer logic causing many to fail or to drop out of introductory programming courses. Conversely, having f2f with e-learning cheats the learner of the pragmatics of computer programming leading to incompetence. Whereas e-learning and computing without f2f lacks scaffolding and motivation as discussed in the preceding chapters.

It is up to the teacher or instructor to find an optimum mix of face-to-face interaction, computing practice and e-learning, and to exercise the flexibility to adapt the learning space according to the needs of the class or even individual students. An effective learning environment requires all three dimensions to be aligned to the curriculum objectives. It is observed in this study that basic tenets of computer programming remain the same despite changes in the software engineering curriculum. Thus, applying the blended learning approach promotes the transferability of learning skills and increases the competence of learners.

This study seeks to give a reflective approach towards the integrated curriculum by:

- Identifying the strengths and limitations of the integrated curriculum for computer programming using the blended learning framework;
- Considering the risks associated with the limitations and how best to reduce the risks;
- Adapting the integrated curriculum to maximize its benefits and strengths for future students or future courses.

Technology perspective

Soloway and Spohrer (1989) have recognised several issues faced by novice programmers 18 years ago which are mainly related to the semantics of programming language constructs and the pragmatics of computing; in particular identifying and correcting programming errors or bugs. These issues have been greatly reduced by the

present generation of computing development environments. In a computing environment as discovered in this study, the development environment plays a key role to promote learner understanding which in turn translates to programming efficiency.

The next major issue raised by Soloway and Spohrer (1989) has to do with plan composition i.e. how do novice programmers put together a working solution. The challenges as discussed in preceding chapters of this study, to convey the cognitive and meta-cognitive skills of programming to the novice programmers require vigilance and perseverance. Implicit in this study is the role of learning technologies in simplifying and helping learners to visualise the concepts and creating awareness of learners' deficiencies. Designers of teaching-learning materials, e-learning objects and online assessments have to be aware of the instructional strategies that best integrates context with the learning objectives (Laurillard, 2002). With time and other resource constraints, IT schools may also have to look into automated assessment tools and methods (Ala-Mutka, 2005) not only for consistency and feedback but also to detect plagiarism.

Student perspective

The evidence in this study implies that the blended learning approach is a strongly learner-centric mode which supports engaged and motivated learners. For low-ability or unmotivated or disinterested learners, online learning serves to confuse or to distract them and creates a sense of boredom (Abbey, 2000). Professional counseling is able to assist to some extent in helping poor learners but a more useful step is to introduce a personal development plan (PDP) for students at the start of the semester as recommended in section 6.6. As learners, students are generally not geared towards maximizing their learning (Ramsden, 2003). Another implication to the success of the blended learning approach is the students' willingness to be engaged learners and to take responsibility for their achievements and shortcomings. Although motivational strategies are beyond the scope of this study, the students' performance is the key to the success of the blended learning approach. Hence, despite their complaints and being dissatisfied, the students that have gone through the blended learning environment are able to produce better projects and higher understanding of computing constructs. The implication is to press on and when students are used to self-learning and self-engagement, learning becomes a habit. At the very least, students who are comfortable with blended learning are geared towards life-long learning, flexible delivery and shared knowledge.

Tutor perspective

The blended learning framework implies more tutor involvement, monitoring and collaboration. There is also an increase in designing and adapting teaching-learning

activities to suit students' needs and in carrying out performance based and continuous assessments. James and Pedder (2006) reveal that assessment for learning encourages teachers to think about the purposes of assessment and work smarter not harder to achieve learning objectives. However, they have observed that teachers employ their expertise within the classroom environment to assimilate their values in line with their practices. Similarly, in the blended learning environment, tutors need to be aware of the constructive alignment framework in order to implement the teaching-learning activities and to voice their concerns so that they can be supported appropriately.

As educators, tutors have to appreciate their influence on students' learning and attitudes (Higgins & Moseley, 2001). Change is never easy and adapting self values to achieve a bigger goal is not immediate. It is implied in this study that tutors need support in order to act out the blended learning approach to encourage students to perform. Blended learning is resource intensive and requires more attentiveness and monitoring for the tutor (Mason et al., 2006). Not only must the tutor prepare for the lesson but the tutor has to be alert to students' learning dynamics and allow for flexibility and adaptability during lessons and focus on teachable moments. In addition, the role of tutors on the development of the blended learning is critical in providing guidance and support rather than obstacles and criticisms.

7.2 Reflections of the researcher as key observer

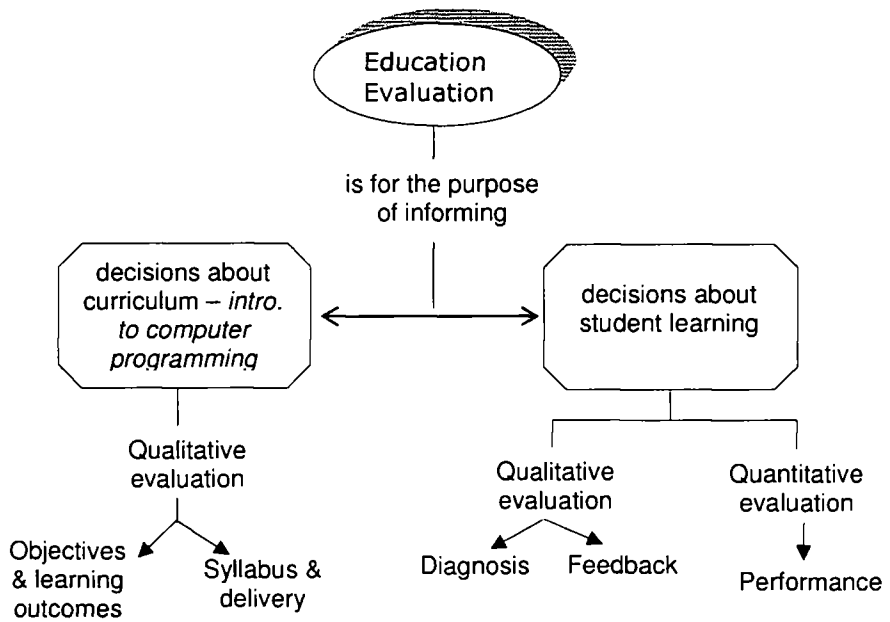


Figure 7.2 Purpose and roles of evaluation adapted for this study

This study is fundamentally an evaluation of teaching and learning methods and as Oliver (2000) observes, evaluation is the process by which the researcher makes value

judgements where these judgements concern the educational value of innovations, and/or the pragmatics of innovative teaching techniques and resources. Whilst going through the research and evaluation of the study, this researcher has referred to Posner's (2004, p.240) model for education evaluation as illustrated in figure 7.2 which is adapted for this study.

Critical introspection and reflection are constantly applied to ensure fair judgement of decisions made during the research process. With the aid of the adapted model, this researcher has attempted to avoid tunnel vision as advised by Posner: by applying the curriculum's objectives and learning outcomes to the student's learning; by identifying the issues that the curriculum addresses, subjugates or ignores; by determining which particular issues that support the curriculum and those that undermine the curriculum. These findings have been duly reported and discussed in this report to the expert knowledge of this researcher.

The model shown in figure 7.2 is substantiated by this study but does not generalise to every curriculum evaluation study. Oliver (2000) has cautioned against providing a model or checklist for evaluation studies as the checklists or models reduce the quality of the results and may lead to unintended or wrong decisions. Accordingly in this study, this researcher has employed other theoretical perspectives of curriculum evaluation in the diagnosis and feedback of student learning with regard to constructive alignment (Biggs, 2003) and constructivist approaches (Jonassen, 1999, Dennen, 2004, and Mead et al., 2006). By applying multiple perspectives, this researcher has critically examined the issues raised in this study and has explored their alternatives. The qualitative analysis serves to provide formative evaluation of the concepts and frameworks evaluated; the case study design enables the process and events to be evaluated and the quantitative analysis allows the outcome of the students' learning to be evaluated. The impacts of the findings are evaluated in the discussion of this research (see chapter 6). At the end of this study (see section 6.6), recommendations are made to improve and to give continuing support to the blended learning approach in computer programming. In the following section, suggestions to extend the research are put forth as a serious extension and expansion of the different aspects of teaching and learning in computer programming.

As long as technology is dependent on computers, the demand for good software engineers with solid computer programming skills will remain high. Although this study does not assume to produce IT graduates with excellent computing skills, it aims to provide insight on the correct form of learning environment that cultivates effective computing skills in novice programmers.

7.3 Suggestions for post research

The findings from this study offer new insights into several areas of educational research i.e. curriculum integration, blended learning, learning technologies, assessments and constructive alignment. Issues and questions are raised in this study which creates opportunities for follow-up research and consequently new areas of research.

In blended learning, Sharpe et al. (2006) call for a longitudinal case study that tracks learners over their whole study programme. This is an excellent way to study the factor of time perspectives and to evaluate the learner's conception of the learning process over time and the learner's experiences. This study has only analysed information for the first semester of first-year students; to gain more value added information, the study can be extended to monitor students' progress throughout their programming modules in the three-year diploma programme. A wider perspective can be gained if a cross-sectional case study is used to track students from other IT courses as performed by Boyle (2005). In a polytechnic, there are different IT courses specialising in various tracks and it will be interesting to reveal how these courses differ in the same respects as of this study. This study is focused on students from a polytechnic in Singapore. Since one of the limitations of this study is its generalisability (as mentioned in section 6.3), it is of interest to education research to replicate this study for other similar institutions in other parts of the world and compare similarities and differences. With more cases to validate the blended learning environment, the blended learning approach may gain precedence over other online learning approaches.

Another research question that can be answered from more research: is whether blended learning is sustainable as a learning strategy that encourages learners to collaborate and interact with peers in the pursuit of learning? Such findings are extremely significant for the continuation of blended learning as an effective learning framework, and its extended use at all levels of education and in various programmes of study.

As technological advancement continues in education technology, education research must also move forward to evaluate and to examine the effects of learning and operating in new learning environments. This study has applied blended learning specifically in the domain of computer programming and the findings observed raised further questions regarding broader fields of learning and simulation. Shaffer (2006) has delved into the area of 'epistemic games' for developing authentic simulation of professional practices.

> In play, we participate in a simulation of a world we want to inhabit,
> and epistemic play is participation in a thickly authentic simulation

> that gives learners access to the epistemic frame of a community of practice. When it succeeds, it is fun, not because fun is the immediate goal, but because interest—linked to identity, understanding, and practice—is an essential part of an epistemic frame, and thus of an epistemic game. (Shaffer, 2006 p.5)

Shaffer has applied his research to various domains such as urban ecology and planning, journalism, law and engineering. An epistemic frame is created by analysing the cognitive knowledge of professionals; next, the epistemic game depends on developing appropriate simulation technologies which constitutes the game engine; finally, appropriate teaching-learning system of activities is designed that utilise the game engine. Shaffer claims that using the epistemic games paradigm encourages students to learn through participation of valued reflective practices. Perhaps a new study can be undertaken to determine whether epistemic games is a new perspective of online learning or whether it is another blend of learning in real-world situations.

Back to the future

Clarity occurs through multiple perspectives dealing with similar factors applied in different domains and contexts of learning. The teaching of computer programming remains a complex and multi-faceted task as agreed in the research literature and supported by the findings of this study; as long as new technology is being invented, the demand for efficient computer programmers remains high. Learning to program computer instructions is the threshold to information technology learning (Robins et al., 2003). Efforts to encourage children to be familiar with cognitive skills in computing have seen the proliferation of robotics and IT clubs in after-school programs. Maloney (et al., 2004) claims that creating a collaborative computing environment for youths will create a pool of keen programmers:

> Our working hypothesis is that, as kids work on personally meaningful Scratch projects such as animated stories, games, and interactive art, they will develop technological fluency, mathematical and problem solving skills, and a justifiable self-confidence that will serve them well in the wider spheres of their lives. (Maloney et al., p. 1)

Software programs such as MicroWorlds[4] and LEGO MindStorms[5] have made similar attempts to encourage youths to learn computing skills in informal, active learning,

[4] MicroWorlds is a multi-media programming environment, refer to http://www.microworlds.com
[5] MindStorms is a robotics toolkit from Lego company that incorporates a graphical programming language, refer to http://mindstorms.lego.com

constructivist settings. The fact that these settings do not attempt to lay the computing fundamentals in an organized manner may be a paradox to the learner development. As observed by McDougall and Boyle (2004), there is a mixture of formal and informal learning in this environment and it is not clear if solutions provided by students are the result of conceptual understanding or gained merely through trial and error. It remains to be seen if these children and youths will register for formal computer programming courses after completing mainstream education and whether any amongst them will become the next Dennis Ritchie, Bjarne Stroustrup or James Gosling[6]. Hence, another useful area of research is to examine effective learning strategies for computer programming that are constructively aligned to the learner's personal, motivational interests.

Is computer programming an art or a science?
Programming is an art, debugging is a science. - *researcher's post script.*

[6] Dennis Ritchie is the creator of C, Bjarne Stroustrup is the creator of C++ and James Gosling is the creator of Java programming languages, refer to http://www.gotw.ca/publications/c_family_interview.htm

References and readings

Abbey, B. (2000). *Instructional and Cognitive Impacts of Web-Based Education.* London: Idea Group, UK.

Abelson, H. & Sussman, G.J. (1996). *Structure and Interpretation of Computer Programs.* Second edition, MIT Press, USA.

Ala-Mutka, K.M. (2005). *A Survey of Automated Assessment Approaches for Programming Assignments.* Computer Science Education, 15(2), p. 83-102.

Alexander, S., Clark, M., Loose, K., Amillo, J., Daniels, M., Boyle, R., Laxer, C. and Shinners-Kennedy, D. (2003). *Case studies in admissions to and early performance in computer science degrees.* Working group reports from ITiCSE on Innovation and technology in computer science education, Thessaloniki, Greece, p. 137-147.

Anderson, J.R., Boyle, C., Corbett, A., & Lewis, M.W. (1990). *Cognitive Modelling and Intelligent Tutoring,* Artificial Intelligence, 42, p.7-49.

Anderson, L.W. & Krathwohl, D.R. (2001). *A Taxonomy for Learning, Teaching and Assessing: a revision of Bloom's taxonomy of educational objectives.* New York: Longman, USA.

Angelo, T.A. & Cross, K.P. (1993). *Classroom Assessment Techniques.* Jossey-Bass Inc, California, USA.

Bach, S., Haynes, P. & Smith, J.L. (2007). *Online Learning and Teaching in Higher Education.* Berkshire, Open University Press, UK.

Bailey, K.D. (1994). *Methods of social research.* Fourth edition, Maxwell Macmillan International, USA.

Bastiaens, T.J. & Martens, R.L. (2000). *Conditions for web-based learning with real events*, in Instructional and Cognitive Impacts of Web-Based Education. London: Idea Group, UK.

Bates, A.W. (2005). *Technology, e-learning and distance education.* London: Routledge, UK.

Bean, J.A. (1995). *Curriculum integration and the disciplines of knowledge.* Phi Delta Kappan, 76, p. 616-622.

Biggs, J. (2003). *Teaching for Quality Learning at University.* Second edition, Open University Press, UK.

Biggs, J.B. and Collis, K.F. (1982). *Evaluating the Quality of Learning: SOLO Taxonomy.* New York: Academic Press, USA.

Bloom, B.S. (1956). *Taxonomy of Educational Objectives: the classification of educational goals. Handbook 1: cognitive domain.* New York, McKay, USA.

Boud, D., & Feletti, G. (1998). *The Challenge of Problem-based Learning.* St Martin's Press, NY, USA.

Boyle T. (2005) *A dynamic, systematic method for developing blended learning.* Education, Communication and Information, Special Issue on Blended Learning, Volume 5 Issue 3, p. 221-232.

Bradley, C., & Boyle, T. (2004). *Students' use of learning objects.* Interactive Multimedia Electronic Journal of Computer-Enhanced Learning, 6(2), Wake Forest University, USA. Available from http://imej.wfu.edu/articles/2004/2/01 /index.asp 23 August 2007.

Burton, J.K., Moore, D.M., Magliaro, S.G. (2004). *Behaviourism and Instructional Technology* in Handbook of research on educational communications and technology, edited by Jonassen, D. (Second edition), Lawrence Erlbaum Assoc, Inc. Publishers, Mahwah, NJ, USA,p.3-36.

Chan, C.B. (2002). *Heart Work* (collection of 36 articles/memoirs from various authors). Singapore Economic Development Board.

Chapman C. & King R.S. (2004). *Differentiated Assessment Strategies: One Tool Doesn't Fit All.* Corwin Press, California, USA.

Chiang, M. (1998). *From Economic Debacle to Economic Miracle.* Ministry of Education, Singapore.

Chung, Y.W., Woon, A. & Lee, N. (1999). *E-learning @ Nanyang Polytechnic: Harnessing IT to support core educational philosophies.* Available from http://www.moe.gov.sg/edumall/mpite/edtech/papers/b5_2.pdf 17 August 2007

Clements, D.H. (2000). *From exercises and tasks to problems and projects – Unique contributions of computers to innovative mathematics education.* The Journal of Mathematical Behaviour Vol 19, issue 1, p. 9-47.

Cohen, J. (1988). *Statistical power analysis for the behavioral sciences,* Second edition. Hillsdale, N.J.: Lawrence Erlbaum, USA.

Cohen, L., Manion, L., & Morrison, K. (2007). *Research methods in education.* Sixth edition, London: RoutledgeFalmer, UK.

Cornford, I.R. (2002). *Learning-to-learn strategies as a basis for effective lifelong learning.* International Journal of Lifelong Education, Vol. 21, No. 4 (July–August 2002), p. 357–368.

Creanor, L., Gowan, D., Howalls, C. & Trinder, K. (2006). *Learner's Experience in e-learning (LEX) Project Report* http://www.jisc.ac.uk/uploaded_documents/ L%20E%20X%20project%20report% 20February%202006.doc 23 August 2007

Cressey, D. (1950). *Criminal Violation of Financial Trust.* American Sociological Review, 15, p. 738-42.

Cressey, D. (1953). *Other People's Money.* Glencoe, Free Press, USA.

Creswell, J.W. (2003). R*esearch Design: Qualitative, Quantitative and Mixed Approaches.* Second edition, Sage Publications, California, USA.

Crossley, M. & Vulliamy, G. (1984). *Case-Study Research Methods and Comparative Education*, Comparative Education, Vol. 20, No. 2, p. 193-207.

Cunningham, W.G. & Cordeiro, P.A. (2003). *Educational Leadership: A Problem-based Approach.* Second Edition, Pearson Education Inc, Allyn and Bacon, USA.

Cushner, K. (2003). Human Diversity in Action: Developing Multicultural Competencies for the Classroom, Second edition. New York, NY; McGraw-Hill, USA.

Czerniak, C.M., Weber, W.B., Sandmann, A., & Ahern, J. (1999) A Literature Review of Science and Mathematics Integration. School Science and Mathematics, 99, p. 421-430.

Davies, J. & Graff, M. (2005). *Performance in e-learning: Online participation and student grades.* British Journal of Educational Technology. 36 (4), p. 657-663.

Dennen, V.P. (2004) *Cognitive apprenticeship in Educational Practice: Research on Scaffolding, Modeling, Mentoring and Coaching as Instructional Strategies* in Handbook of research on educational communications and technology, edited by Jonassen, D. (Second edition), Lawrence Erlbaum Assoc, Inc. Publishers, Mahwah, NJ, USA, p. 813-828.

Driscoll, M. (2002). *Blended learning: Let's get beyond the hype.* LTIMagazine. Available from http://elearningmag.com/ltimagazine/article/articleDetail.jsp?id=11755 23 August 2007

Field, A.P. (2005). *Discovering Statistics using SPSS.* Second edition. Sage Publications, UK.

Fink, L.D. (2003). *Creating Significant Learning Experiences: An Integrated Approach to Designing College Courses.* John Wiley & Sons, California, USA.

Flyvbjerg, B. (2006). *Five Misunderstandings About Case-Study Research.* Qualitative Inquiry Volume 12 Number 2 April 2006 p. 219-245

Fogarty, R. (1991). *Ten ways to integrate curriculum.* Educational Leadership, 47(2), p.61-65.

Fosnot, C.T. (2005). *Constructivism: theory, perspectives, and practice,* Second edition. New York: Teachers College Press, USA.

Fraenkel, J.R. & Wallen, N.E. (2006). *How to design and evaluate research in education,* Sixth edition. McGraw-Hill Higher Education, NY, USA.

Franklin, B.M. (2000) *Curriculum & Consequence: Herbert M. Kliebard and the promise of schooling.* Teachers College, Columbia University. USA.

Fry, H., Ketteridge, S. & Marshall, S. (2003). *A handbook for teaching and learning in higher education: enhancing academic practice.* Second edition (eds.), London: Kogan Page, UK.

Gregory, G.H. & Chapman, C. (2006). *Differentiated Instructional Strategies: One Tool Doesn't Fit All.* Corwin Press, California, USA.

Gomm, R., Hammersley, H. & Foster, P. (2000). *Case Study Method: Key issues, key texts.* (Eds), Sage Publication Ltd, UK

Gunnerson, E. (2005). *A Programmer's Introduction to C# 2.0,* Third Edition. Berkeley, CA: Apress, USA.

Gray, C. (2006). *Blended Learning: Why everything old is new again – but better.* Learning Circuits, March 2006, Available from http://www.learningcircuits.org /2006 /March/gray.htm 17 August 2007

Haines, C. (2004). *Assessing students' written work: Marking essays and reports.* RoutledgeFalmer, UK.

Hammersley, M., Gomm, R. & Foster, P. (2000). *Introduction and Case Study and Theory,* in Case study method: key issues, key texts. Edited by Gomm R. Hammersley M. and Foster P. Sage Publications Ltd, p. 1-16, 234-258.

Hatch, A., & Burd L., (2006). *Creating a Working Environment for Group-work: Techno-Café Experience Report,* 7th Annual Conference of the Subject Centre for Information and Computer Science, Trinity College, Dublin, 29th- 31st August 2006.

Heinze, A. & C. Procter (2004). *Reflections on the Use of Blended Learning.* Education in a Changing Environment conference proceedings 13th-14th September 2004, University of Salford, UK.

Heinze, A., Proctor, C. & Scott B. (2006). *The Theory and Practice of the Conversational Framework: Proposed Amendments and Enrichments.* Education in a Changing Environment 12th-13th January 2006 Conference Proceedings. University of Salford, UK.

Higgins, S. (2003). *Does ICT improve learning and teaching in schools?* Nottingham: British Educational Research Association. Monograph.

Higgins, S. & Moseley D. (2001). Teachers' thinking about ICT and learning: beliefs and outcomes. Teacher Development, Vol. 5(2), p. 191–210.

Higgs, B. & Sabin M. (2005). *Towards Using Online Portfolios in Computing Courses.* SIGITE'05, October 20–22, 2005, Newark, New Jersey, USA. p.323-328.

Hitchcock, G. and Hughes, D. (1995). *Research and the teacher: a qualitative introduction to school-based research,* Second edition. London: Routledge, UK.

Hounsell, D. (2003). *The evaluation of teaching,* in A handbook for teaching and learning in higher education: enhancing academic practice. Second edition edited by Fry H., Ketteridge S. & Marshall S. London: Kogan Page, UK.

Humphreys, A., Post, T. & Ellis, A. (1981). *Interdisciplinary Methods: A Thematic Approach.* Goodyear Publishing Co., CA. USA.

Huntley-Moore, S. & Panter, J.R. (2006). *A Practical Manual for Evaluating Teaching in Universities.* Dublin: All Ireland Society for Higher Education. Available from http://www.aishe.org/readings/2006-1/aishe-readings-2006-1.html 23 August 2007

Irons, A. & Alexander, S. (2004). *Effective learning and teaching in computing* (eds). London: RoutledgeFalmer, UK.

Jacobs, G.M., Power, M.A. & Loh, W.I. (2002). *The Teacher's Sourcebook for Cooperative Learning: Practical Techniques, Basic Principles, and Frequently Asked Questions.* Corwin Press, Inc. USA.

James M. & Pedder D. (2006). *Beyond method: Assessment and Learning practices and values.* The Curriculum Journal, Vol 17, No.2 p.109-138.

Jochems, W., van Merriënboer, J. & Koper R. (2004). *Integrated e-learning: implications for pedagogy, technology and organization*, (eds). London : RoutledgeFalmer, UK.

Johnson, D.W. & Johnson, R.T. (1999). *Learning Together and Alone: Cooperative, Competitive, and Individualistic learning*, 5th Ed. Pearson Education Inc. Allyn and Bacon, Boston, USA.

Johnson D.W, & Johnson, R.T (2004). *Cooperation and the Use of Technology* in Handbook of research on educational communications and technology, edited by Jonassen, D. (Second edition), Lawrence Erlbaum Assoc, Inc. Publishers, Mahwah, NJ, USA, p.785-811.

Jonassen, D.H. (1996). *Computers in the classroom: Mindtools for critical thinking.* Columbus, OH: Merrill/Prentice-Hall.

Jonassen, D.H. (1999), *Designing constructivist learning environments*, in Reigeluth, C.M. (Eds),Instructional Theories and Models: A New Paradigm of Instructional Theory, Second edition, Lawrence Erlbaum Associates, Mahwah, NJ, p.215-39.

Keen, K. (1992). *Competence: What is it and how can it be developed?* In J. Lowyck, P de Potter & J. Elen (Eds.), Instructional design: Implementation issues (pp.111-122). Brussel: IBM International Education Center.

Kirschner, P.A. (2005). Learning in innovative learning environments. *Computers in Human Behavior, 21* , 547-554.

Koper, R. (2006). *Current Research in Learning Design.* Educational Technology & Society, Vol. 9 (1), p. 13-22.

Laurillard, D. (2002). *Rethinking university teaching: a conversational framework for the effective use of learning technologies.* London: RoutledgeFalmer, UK.

Lavery, J., Burd, L., & Hodgson B., (2006). *Getting Sustainable Development into the Curriculum*, 7th Annual Conference of the Subject Centre for Information and Computer Science, Trinity College, Dublin, 29th- 31st August 2006

Leutenegger, S. and Edgington, J. (2007). *A games first approach to teaching introductory programming.* ACM SIGCSE Bulletin , Proceedings of the 38th SIGCSE technical symposium on Computer science education SIGCSE '07, Volume 39 Issue 1, p. 115-118.

Lippman, S.B., Lajoie, J. & Moo, B.E. (2005). *C++ Primer*, Fourth edition. Upper Saddle River, NJ: Addison-Wesley, USA.

Littlejohn, A. & Pegler, C. (2007). *Preparing for Blended e-Learning.* Routledge, UK.

Lohr, S. (2001). *Go To: The story of the math majors, bridge players, engineers, chess wizards, maverick scientists and iconoclasts – the programmers who created the software revolution.* Basic Books, Perseus Books Group, New York. USA.

Mallery, A.L. (2000). *Creating a catalyst for thinking: the integrated curriculum.* Allyn & Bacon, USA.

Maloney, J., Burd, L., Kafai, Y., Rusk, N., Silverman, B. & Resnick, M. (2004). *Scratch: A Sneak Preview.* Second International Conference on Creating, Connecting and Collaborating through Computing (C5'04), c5, p. 104-109.

Mason, R. (2001). *E-learning: What have we learnt?* Proceedings of the 2001, 9th International Symposium, Improving student learning: improving student learning using learning technology edited by Chris Rust, Oxford: Oxford Centre for Staff & Learning Development, p.27-34.

Mason, R. & Rennie, F. (2006). *Elearning : the key concepts.* London: Routledge, UK

Maurer, R. (1994). *Designing Interdisciplinary Curriculum in Middle, Junior High and High Schools.* Allyn & Bacon Inc. USA.

McAllister, G. & Alexander, S. (2003). *Key aspects of teaching and learning in information and computer sciences,* in A handbook for teaching and learning in higher education: enhancing academic practice. Second edition edited by Fry H., Ketteridge S. & Marshall S. London: Kogan Page, UK.

McConnell, D. (2005) *Examining the dynamics of networked e-learning groups and communities.* Studies in Higher Education. 30 (1), p. 25-42.

McDougall, A. & Boyle, M. (2004). *Student strategies for Learning Computer Programming: Implications for Pedagogy in Informatics.* Education and Information Technologies 9:2, p.109-116.

McGill, T. (2003). *Current Issues in IT Education,* (Eds). IRM Press, UK.

McKenzie, J. (2001). *Variation in ways of experiencing change in teaching, the development and use of learning technologies and the likely consequences for student learning.* Proceedings of the 2001, 9th International Symposium, Improving student learning: improving student learning using learning technology edited by Chris Rust, Oxford: Oxford Centre for Staff & Learning Development, p. 163-173.

McMillan, J.H. and Wergin, J.F. (2005). *Understanding and Evaluating Educational Research.* Third edtion, Prentice Hall, Upper Saddle River, N.J., USA.

Mead, J., Gray, S., Hamer, J., James, R., Sorva, J., St. Clair, C. and Thomas, L. (2006). *A cognitive approach to identifying measurable milestones for programming skill acquisition.* ACM SIGCSE Bulletin , Working group reports on ITiCSE on Innovation and technology in computer science education ITiCSE-WGR '06, Volume 38 Issue 4, p. 182-194.

Mill, J.S. (1974). *A System of Logic, Ratiocinative and Inductive: Being a Connected View of the Principles of Evidence and the Methods of Scientific Investigation.* Toronto: University of Toronto Press, Canada.

Moesby, E. (2002). *The process towards implementing Problem-based Learning – "Pieces for the puzzle".* The National Workshop on Problem Based Workshop (PBL) held at Republic Polytechnic, Singapore September 16th 2002 .

Moseley, D., Baumfield, V., Elliott, J., Gregson, M., Higgins, S., Miller, J. & Newton, D. (2005). *Frameworks for thinking: a handbook for teaching and learning.* Cambridge: Cambridge University Press, UK.

Morris, L. & Walker, D. (2006). *CAA sparks chemical reaction: Integrating CAA into a learning and teaching strategy.* Evaluation of the use of the virtual learning environment in higher education across Scotland. QAA Scotland. Retrieved from www.enhancementthemes.ac.uk/documents/flexibleDelivery/Flexible_ delivery_QAA_128.pdf 23 August 2007

NYP, (2005a). *To improve the first attempt Module Passing Rate of SIT(EI) year 1 students*, Work and Improvement Team Skills (WITS) project completed in June, 2005. Unpublished report submitted to Nanyang Polytechnic EXCEL committee, Singapore.

NYP, (2005b). *Reduction of Failure Performance Rate of Year-1 Students from ITDF04 Diploma Course,* Six-sigma project completed in December, 2005. Unpublished report submitted to Productivity and Standards Board, Singapore.

Oliver, M. (2000). *An introduction to the Evaluation of Learning Technology.* Educational Technology & Society 3(4) 2000 http://ifets.massey.ac.nz/ periodical/vol_4_2000/intro.html last retrieved on 02 August 2007

Oliver, M. & Trigwell, K. (2005). *Can 'Blended Learning' be redeemed?* E-Learning, Vol 2(1), p.17-26.

Ornstein, A.C. & Behar-Horenstein, L.S. (1999). *Contemporary issues in curriculum.* Second edition, Allyn & Bacon, USA.

Passey D. (2006). *Technology enhancing learning: Analysing uses of information and communication technologies by primary and secondary school pupils with learning frameworks.* The Curriculum Journal, Vol 17, No.2 p.139-166.

Patton, M.Q. (2002). *Qualitative research & evaluation methods.* Third edition, Sage Publications, USA.

Pea, R.D. & Kurland, D.M. (1983). On the Cognitive Prerequisites of Learning Computer Programming, Technical report 18. New York: Center for Children & Technology, Bank Street College of Education,

Piaget, J. (1950). *The Psychology of Intelligence.* London: Routledge and Kegan Paul, (1977). *Equilibrium of cognitive structures.* New York: Viking. USA.

Pinar, W.F. & Irwin, R.L. (2005). *Curriculum in a new key: The collected works of Ted T. Aoki.* Lawrence Erlbaum Associates, Inc. USA.

Pintrick, P.R. & Schunk, D.H. (2002). Motivation in Education: Theory, Research and Applications. Second edition. Pearson Education, Inc. USA.

Popham, W.J. (2006). *Mastering assessment: a self-service system for educators.* New York: Routledge, Taylor and Francis, USA.

Posner, G.J. (2004). *Analysing the Curriculum.* Third edition, McGraw-Hill, USA.

Ramsden, P. (2003). *Learning to teach in higher education.* Second edition, London: RoutledgeFalmer, UK.

Robins, A., Roundtree J. & Roundtree, N. (2003). *Learning and Teaching Programming: A Review and Discussion,* Computer Science Education, Vol. 13, 2, pp.137-172.

Robinson, W.S. (2000). *The Logical Structure of Analytic Induction*, in Case study method: key issues, key texts. Edited by Gomm R. Hammersley M. and Foster P. Sage Publications Ltd, UK.

Savenye, W.C. and Robinson, R.S. (2004) *Qualitative Research Issues and Methods: An Introduction for Educational Technologists.* in Handbook of research on educational communications and technology, edited by Jonassen, D. (Second edition), Lawrence Erlbaum Assoc, Inc. Publishers, Mahwah, NJ, USA, p.1045-1071

Savin-Baden, M. & Wilkie, K. (2006). *Problem-based learning online,*(eds). Maidenhead: Open University Press, UK.

Sayers, H. M., Nicell, M. A. & Hagan, S. J. (2004). *Supporting and assessing first year programming: The use of WebCT.* Italics e-journal. 3 (1), Available from www.ics.heacademy.ac.uk/italics/Vol3-1/sayers/WebCT.pdf 23 August 2007

Scriven, M. (1995). *Student Ratings Offer Useful Input to Teacher Evaluation*, ERIC/AE Digest available from http://eric.ed.gov/ERICWebPortal/recordDetail?accno =ED398240 23 August 2007

Shaffer, D. W. (2006). *How computer games help children learn.* New York: Palgrave Macmillan, USA.

Sharpe, R., Benfield, G., Roberts, G., Francis, R. (2006) *The undergraduate experience of blended e-learning: a review of UK literature and practice.* Special report from The Higher Education Academy, UK. Available from http://www.heacademy.ac.uk/assets/York/documents/ourwork/research/ literature_reviews/blended_elearning_full_review.pdf 23 August 2007

Sheridan-Ross, J., Harrison, G., Gray, J. (2007). *Groupwork: strategies adopted by students,* ITALICS e-journal, May 2007, Volume 6 Issue 2.

Soloway, E. & Spohrer, J.C. (1989). *Studying the Novice Programmer.* (Eds), Lawrence Erlbaum, Hillsdale, NJ, USA.

Stevens, D.D & Levi, A.J. (2005). *Introduction To Rubrics: An Assessment Tool To Save Grading Time, Convey Effective Feedback And Promote Student Learning.* Stylus Publishing, VA, USA.

Stevenson, N. (2006). *Integrating personal tutoring with personal development planning.* University of Westminster, Higher Education Academy. Available from http://www.heacademy.ac.uk/assets/York/documents/ourwork/tla /personal_tutoring/web0150_eCasebook_intergrating_personal_tutoring_with_ personal_development_planning.pdf 23 August 2007

Stroustrup, B. (1993). *A History of C++: 1979-1991.* Proceedings from ACM History of Programming Languages conference (HOPL-2). ACM Sigplan Notices. March 1993, Vol 28 No 3, pp 271-298.

Thorburn, M. & Collins D. (2003). *Integrated curriculum models and their effects on teachers' pedagogy practices.* European Physical Education Review, 9(2), p. 185-209.

Topping, K. (1998). *Peer Assessment between Students in Colleges and Universities.* Review of Educational Research, 68(2), p. 249-276.

Turner, R. & Lowry, G. (2003). *Education for a Technology-Based Profession: Softening the Information Systems Curriculum*, in Current Issues in IT Education, McGill, T (eds), IRM press, London, UK, p. 153-172.

Van Roy, P. & Haridi, S. (2004). *Concepts, Techniques, and Models of Computer Programming.* Massachusetts Institute of Technology, USA.

Vars, G.F. (1991). I*ntegrated curriculum in historical perspective.* Educational Leadership, 49(2), p.14-15.

Vygotsky, L.S. (1962). *Thought and Language.* Cambridge, MA: MIT Press, USA. (1978). *The Mind in Society: the development of higher psychological processes.* Cambridge, MA: Harvard University Press, USA.

Watson, D.M. (2001). *Pedagogy before Technology: Re-thinking the Relationship between ICT and Teaching*, Education and Information Technologies, Vol. 6, No. 4, pp. 251-266.

Webb, N. L. (2002). *Alignment study in language arts, mathematics, science, and social studies of state standards and assessments for four states.* Washington, DC: Council of Chief State School Officers, USA.

Webb, N.L. (2007). *Issues Related to Judging the Alignment of Curriculum Standards and Assessments,* Applied Measurement in Education, 20(1). p. 7-25

Wiggins, G. (1990). *The Case for Authentic Assessment.* Practical Assessment, Research & Evaluation, 2(2). Available from http://PAREonline.net/getvn.asp?v=2&n=2 23 August 2007

Wiley, D.A. (2002). *Connecting learning objects to Instructional design theory: A definition, a metaphor, and a taxonomy.* In D.A.Wiley (Ed.). The Instructional Use of Learning Objects. Bloomington, Indiana: Agency for Instructional Technology and Association for Educational Communications of Technology, USA.

Wood, D.J., Bruner, J.S. and Ross, G. (1976). *The Role of Tutoring in Problem Solving.* Journal of Child Psychology and Psychiatry, Vol 17, p. 89-100.

Yelland, N. (2007). *Shift to the future : rethinking learning with new technologies in education.* New York: Routledge, UK.

Yin, R.K. (2003). *Case study research: design and methods.* Third edition, Sage Publications, CA USA.

Web references

- ACM, (2007). http://www.acm.org/education/curricula.html last retrieved 23 August 2007
- CDTL, (2005). *Learning with Technology.* (3 articles on using the Integrated Virtual Learning Environment). http://www.cdtl.nus.edu.sg/brief/V8n3/default.htm last retrieved 23 August 2007
- HEA, (2007). Academy Updates. http://www.heacademy.ac.uk/resources /publications/academyinfo/academyupdate last retrieved 23 August 2007
- IDA, (2007a). Infocomm Adoption. http://www.ida.gov.sg/Infocomm%20Adoption /20060405174006.aspx last retrieved 23 August 2007
- IDA, (2007b). *Number of Infocomm Professionals Reached New High in 2005.* http://www.ida.gov.sg/News%20and%20Events/20050705104523.aspx ?getPagetype=20 last retrieved 23 August 2007
- MOE, (2007). *Ministry of Education, Singapore.* Refer to http://www.moe.edu.sg last retrieved 23 August 2007
- MOE, (2007a). *Masterpan II*, http://www.moe.gov.sg/edumall/mp2/mp2.htm last retrieved 23 August 2007
- MOE, (2007b). *From Secondary to Post-Secondary Education.* http://www.moe.gov.sg/corporate/eduoverview/Sec_sectopost.htm last retrieved 23 August 2007
- MOE, (2007c). *Polytechnics.* http://www.moe.gov.sg/corporate/post _secondary.htm#poly last retrieved 23 August 2007
- MOE, (2007d). *ICT integration.* http://www.moe.gov.sg/edumall/tl/it_integration.htm last retrieved 23 August 2007
- NYP, (2007a). School of Information Technology http://www.nyp.edu.sg/SIT /sit.html last retrieved 23 August 2007
- NYP, (2007b). *Nanyang Polytechnic Annual Report 2005/2006.* http://www.nyp.edu.sg/aboutNYP/publications_sub_page.html last retrieved 23 August 2007

Appendix A: Course structure for school of IT, Engineering Informatics diploma in the first semester of first year

CASE 2005-S1

Core modules	
IT1744	Internet Computing
IT1734	Business Information Systems
IT1733	**Principles of Computing**
IT1742	Data Structures & Algorithms
IT1732	Electronic Resource Processing
IT1736	Semestral Project 1
IT1746	Semestral Project 2
IT1735	Creativity and Productivity
IT1743	Manufacturing Processes
IT1731	Computing Mathematics 1
IT1741	Computing Mathematics 2
IT1745	Communication Skills 1

Programme A for groups 1 to 6:
IT1731, IT1732, IT1733, IT1734, IT1735 and IT1736

Programme B for groups 7 to 11:
IT1741, IT1743, IT1733, IT1744, IT1745 and IT1746

CASE 2006-S1

Core modules	
IT1755	Internet Computing
IT1754	Business Information Systems
IT1753	**Principles of Computing**
IT1762	Data Structures & Algorithms
IT1764	Fundamentals of Networking
IT1756	Web Design and Multimedia Project
IT1766	Innovation Project
IT1752	Digital Electronics
IT1763	Manufacturing Processes
IT1751	Computing Mathematics 1
IT1761	Computing Mathematics 2
IT1765	Communication Skills 1

Programme A for groups 1 to 5 and 11:
IT1751, IT1752, IT1753, IT1754, IT1755 and IT1756

Programme B for groups 6 to 10:
IT1761, IT1763, IT1753, IT1764, IT1765 and IT1766

Appendix B: Module syllabus for Principles of Computing (PrC)

Case 2005-S1

NANYANG POLYTECHNIC
SCHOOL OF INFORMATION TECHNOLOGY

DIPLOMA IN ENGINEERING INFORMATICS
MODULE SYLLABUS

MODULE CODE : IT1733

MODULE NAME : Principles of Computing

AIM(S) : To introduce to students the programming fundamentals such as algorithms, logic, representation of information in a computer, variables and data types.

OBJECTIVE(S) : On successful completion of this subject module, the students will be able to:

1) understand the concept of software systems
2) write algorithms to describe the logic of a computer program
3) use variables to store and retrieve data in a computer program
4) use decision-making and program flow control constructs to represent the logic of a program
5) understand the concept of structured programming with functions and subroutines.

PRE-REQUISITES : None

MODULE TYPE : Core / ~~Prescribed Elective~~ ~~/ Complementary Elective~~ ~~/ Special Elective~~

HOURS / CREDIT : 90 Hours / 6 points

MODE OF TEACHING: Lecture (30) Practical (45) Tutorial (15) Test (0)

MODE OF ASSESSMENT: (Total : 100%)		
	Examination	(Not Applicable)
	Written Test	(40%)
	ICA1 (Tutorial)	(10%)
	(Practical test)	(20%)
	ICA2 (Mini-project)	(30%)
	Examination Duration	(N.A.)
	Supplementary Assessment	(N.A.)

NANYANG POLYTECHNIC
SCHOOL OF INFORMATION TECHNOLOGY

DIPLOMA IN ENGINEERING INFORMATICS
MODULE SYLLABUS

SUBJECT CODE : IT1733

SUBJECT NAME : Principles of Computing

SYNOPSIS : This module introduces to the students the programming fundamentals such as algorithm, logic, computer representation of information, variables and data types. Students also learn to plan and describe program logic using flowcharts and pseudo-code.

TEXT REFERENCES

1) Simple Program Design, Lesley Anne Robertson, CT, 2000, B.
2) Problem Solving using C: Structured Programming Techniques, Yuksel Uckan, PE, 1995, B.
3) Problem Solving & Program Design in C, Jeri R. Hanly, Eliiot B. Koffman, TM, 2002, B.
4) The C Programming Language, Brian W. Kernighan, Dennis M. Ritchie, PH, 1988, B.

PUB :

AW	Addison-Wesley	CT	Course Technology	JW	John Wiley
MH	McGraw-Hill	PH	Prentice Hall	PE	Pearson Education
TS	Thomson	TM	Times Mirror Higher Education Group		
OT	Others				

MAT :

B	Book	M	Magazine	V	Video
C	CBT	U	User Manuals	O	Others

NANYANG POLYTECHNIC
SCHOOL OF INFORMATION TECHNOLOGY

DIPLOMA IN ENGINEERING INFORMATICS
MODULE SYLLABUS

SUBJECT CODE : IT1733

SUBJECT NAME : Principles of Computing

No	TOPICS (L:P:T) / SUBTOPICS	INSTRUCTIONAL OBJECTIVES (What You want students to achieve)	BIB. REF
1.	**Computer Software Systems** (L:T:P = 1 : 1 : 0)	(L : 0 hr self-directed learning) (P : 0 hr self-directed learning)	
1.1	The Computer System	• Know computer system components. • Know how a computer works.	1-3
1.2	Computer Hardware	• List the main components of computer hardware.	1-3
1.3	Computer Software	• List the classifications of computer software.	1-3
1.4	Programming Languages	• Explain the terms Machine Language, Assembly Language and High Level Language.	1-3
1.5	Computer Program	• Explain the concept of the computer program (source program, object program).	1-3
2.	**Computer Algorithms and Program Design** (L:T:P = 5 : 2 : 9)	(L : 2 hr self-directed learning) (P : 3 hr self-directed learning)	
2.1	Program Development Process	• Describe the steps in program development process.	1-3
2.2	Algorithms	• Describe what an algorithm is. • Describe guidelines for good algorithm design. • List two methods of presenting an algorithm.	1-3
2.3	Pseudo code	• List common words and keywords used in writing pseudo code. • Describe the six basic operations which a computer performs. • Describe operations using pseudo code.	1-3
2.4	Flowcharting	• Represent an algorithm using a	1-3

No	TOPICS (L:P:T) / SUBTOPICS	INSTRUCTIONAL OBJECTIVES (What You want students to achieve)	BIB. REF
		flowchart. • Describe symbols used for flowcharting.	
2.5	Developing an Algorithm	• Describe the various steps in designing and checking a solution algorithm.	1-3
2.6	Modular Program Design	• Explain the steps in modularisation. • Explain the use of structure charts.	1-3
3.	**An Introduction to the C/C++ Language** (L:T:P = 1 : 1 : 2)	(L : 0 hr self-directed learning) (P : 1 hr self-directed learning)	
3.1	Creating a C/C++ Program	• Explain the steps of creating a C/C++ program. • Understand the concept of computer variables.	4
3.2	Computer Variables	• Understand the concept of computer variables. • Know how to define variables.	4
3.3	Statements and Expressions	• Explain the concepts of C/C++ statements and expressions.	4
3.4	Basic Input and Output	• Be familiar with the basic input and output.	4
4.	**Basic Data Types and Variables** (L:T:P = 1 : 0 : 1)	(L : 0 hr self-directed learning) (P : 1 hr self-directed learning)	
4.1	Variable Naming	• Express C variable naming convention and style	4
4.2	Basic Data Types	• Understand integer, character and floating point data types.	4
4.3	Type Conversions	• Know auto-data conversion.	4
4.4	Derived Data Types	• Know the user derived /defined data types.	4
5.	**Computing Operators and Expressions** (L:T:P = 2 : 1 : 3)	(L : 1 hr self-directed learning) (P : 1 hr self-directed learning)	
5.1	Introduction	• Know tokens and classification of operators.	4
5.2	Binary Operators	• Know binary operators, such as arithmetic operators, relational operators, logical operators and assignment operator. • Know the logical operators, cast operator, increment and decrement	4

No	TOPICS (L:P:T) / SUBTOPICS	INSTRUCTIONAL OBJECTIVES (What You want students to achieve)	BIB. REF
		operators.	
5.3	Unary Operators	• Know how to apply unary operators.	4
5.4	Operator Precedence	• Understand operator precedence and evaluation order.	4
5.5	Comments	• Know how to use comments to make the program more readable and clear.	4
6.	**Program Flow Control and Decision Making** (L:T:P = 10 : 5 : 15)	(L : 5 hr self-directed learning) (P : 5 hr self-directed learning)	
6.1	Introduction to Control Structures	• List four types of flow control structures.	4
6.2	Selection Structures	• Use if and if/else statements for decision making. • Use a switch/case statement for multiple selections.	4
6.3	Looping Structures	• Use for, while, and do/while statements to control the number of repetitions for a group of statements.	4
7.	**Functions** (L:T:P = 6 : 3 : 9)	(L : 3 hr self-directed learning) (P : 3 hr self-directed learning)	
7.1	Introduction	• Understand the basic concept of functions.	4
7.2	Function Definition	• Know how to define functions with or without arguments.	4
7.3	Function Prototypes	• Know how to declare a function (prototype).	4
7.4	Function Call	• Understand how to call a function and how to pass arguments to a function.	4
7.5	Function Communication	• Know how functions communicate with each other. • Know how to return value to the calling function and how to make use of the returned value.	4
8.	**Arrays** (L:T:P = 4 : 2 : 6)	(L : 3 hr self-directed learning) (P : 4 hr self-directed learning)	
8.1	Why Arrays are Used	• Understand why and when to use arrays.	4
8.2	Declaration of Arrays	• Understand the definition of an array.	4
8.3	Array Manipulation	• Know how to refer to individual elements of an array. • Be able to store values into an array	4

No	TOPICS (L:P:T) / SUBTOPICS	INSTRUCTIONAL OBJECTIVES (What You want students to achieve)	BIB. REF
		and read values out from an array. • Be able to search lists and tables of values stored in arrays.	
8.4	Multi-Dimensional Arrays	• Know how to define and use multi-dimensional arrays.	4

ADDITIONAL NOTES:

No of Assignments to set	One (1), Mini-project
No of Tests to set	Two (2), Practical test and One (1), Written test
No of Supplementary Test to set	0
Duration of each test	1 hr (Practical test 1), 1 hr (Practical test 2), 2 hr (Written test)
Test to cover which topics	Topics 1- 8
No of Examination questions to set	0

Case 2006-S1

NANYANG POLYTECHNIC
SCHOOL OF INFORMATION TECHNOLOGY

DIPLOMA IN ENGINEERING INFORMATICS
MODULE SYLLABUS

MODULE CODE : IT1753

MODULE NAME : Principles of Computing

AIM(S) : To introduce to students the programming fundamentals such as algorithms, logic, representation of information in a computer, variables and data types.

OBJECTIVE(S) : On successful completion of this module, the students will be able to:

1) understand the concept of software systems
2) write algorithms to describe the logic of a computer program
3) use variables to store and retrieve data in a computer program
4) use decision-making and program flow control constructs to represent the logic of a program
5) understand the concept of structured programming with functions and subroutines.

PRE-REQUISITES : None

MODULE TYPE : Core / ~~Prescribed Elective~~ / ~~Complementary Elective~~ /~~Special Elective~~

HOURS / CREDIT : 60 Hours / 4 points

MODE OF TEACHING : Lecture (15) Practical (45) Tutorial (0) Test (0)

MODE OF ASSESMENT: (Total : 100%)		
	Examination	(Not Applicable)
	Practical Tests	(50%)
	ICA1 (Online Assessment)	(20%)
	ICA2 (Mini-project)	(30%)
	Examination Duration	(Not Applicable)
	Supplementary Assessment	(Not Applicable)

NANYANG POLYTECHNIC
SCHOOL OF INFORMATION TECHNOLOGY

DIPLOMA IN ENGINEERING INFORMATICS
MODULE SYLLABUS

SUBJECT CODE : IT1753

SUBJECT NAME : Principles of Computing

SYNOPSIS : This module introduces to the students the programming fundamentals such as algorithm, logic, computer representation of information, variables and data types. Students also learn to plan and describe program logic using flowcharts and pseudo-code.

TEXT REFERENCES

5) Simple Program Design (4th Ed), Lesley Anne Robertson, CT, 2004, B.
6) Problem Solving & Program Design in C (4th Ed), Jeri R. Hanly, Eliiot B. Koffman, TM, 2004, B.
7) C# Complete, SYBEX, 2003.
8) Simply C#, Harvey M. Dietel, PH, 2004, B and C.
9) C# Programming, Jesse Liberty, OR, 2005, B.

PUB :

AW	Addison-Wesley	CT	Course Technology	JW	John Wiley
MH	McGraw-Hill	PH	Prentice Hall	PE	Pearson Education
OR	O'Reilly	TM	Times Mirror Higher Education Group		
OT	Others				

MAT :

B	Book	M	Magazine	V	Video
C	CBT	U	User Manuals	O	Others

NANYANG POLYTECHNIC
SCHOOL OF INFORMATION TECHNOLOGY

DIPLOMA IN ENGINEERING INFORMATICS
MODULE SYLLABUS

SUBJECT CODE : IT1753

SUBJECT NAME : Principles of Computing

No	TOPICS (L:P:T) / SUBTOPICS	INSTRUCTIONAL OBJECTIVES (What You want students to achieve)	BIB. REF
1.	**Computer Software Systems** (L:T:P = 1 : 0 : 0)		
1.1	The Computer System	• Know computer system components. • Know how a computer works.	1-3
1.2	Computer Hardware	• List the main components of computer hardware.	1-3
1.3	Computer Software	• List the classifications of computer software.	1-3
1.4	Programming Languages	• Explain the terms Machine Language, Assembly Language and High Level Language.	1-3
1.5	Computer Program	• Explain the concept of the computer program (source program, object program).	1-3
2.	**Computer Algorithms and Program Design** (L:T:P = 1 : 0 : 1)		
2.1	Program Development Process	• Describe the steps in program development process.	1-3
2.2	Algorithms	• Describe what an algorithm is. • Describe guidelines for good algorithm design. • List two methods of presenting an algorithm.	1-3
2.3	Pseudo code	• List common words and keywords used in writing pseudo code. • Describe the six basic operations which a computer performs. • Describe operations using pseudo code.	1-3
2.4	Flowcharting	• Represent an algorithm using a flowchart.	1-3

No	TOPICS (L:P:T) / SUBTOPICS	INSTRUCTIONAL OBJECTIVES (What You want students to achieve)	BIB. REF
		• Describe symbols used for flowcharting.	
2.5	Developing an Algorithm	• Describe the various steps in designing and checking a solution algorithm.	1-3
2.6	Modular Program Design	• Explain the steps in modularisation. • Explain the use of structure charts.	1-3
3.	**An Introduction to the C# Language** (L:T:P = 1 : 0 : 5)		
3.1	Creating a C# Program	• Explain the steps of creating a C/C++ program. • Understand the concept of computer variables.	3-5
3.2	Computer Variables	• Understand the concept of computer variables. • Know how to define variables.	3-5
3.3	Statements and Expressions	• Explain the concepts of C# statements and expressions.	3-5
3.4	Basic Input and Output	• Be familiar with the basic input and output.	3-5
4.	**Basic Data Types and Variables** (L:T:P = 1 : 0 : 1)		
4.1	Variable Naming	• Express C variable naming convention and style	3-5
4.2	Basic Data Types	• Understand integer, character and floating point data types.	3-5
4.3	Type Conversions	• Know auto-data conversion.	3-5
4.4	Derived Data Types	• Know the user derived /defined data types.	3-5
5.	**Computing Operators and Expressions** (L:T:P = 2 : 0 : 2)		
5.1	Introduction	• Know tokens and classification of operators.	3-5
5.2	Binary Operators	• Know binary operators, such as arithmetic operators, relational operators, logical operators and assignment operator. • Know the logical operators, cast operator, increment and decrement operators.	3-5

No	TOPICS (L:P:T) / SUBTOPICS	INSTRUCTIONAL OBJECTIVES (What You want students to achieve)	BIB. REF
5.3	Unary Operators	• Know how to apply unary operators.	3-5
5.4	Operator Precedence	• Understand operator precedence and evaluation order.	3-5
5.5	Comments	• Know how to use comments to make the program more readable and clear.	3-5
6.	**Program Flow Control and Decision Making** (L:T:P = 3 : 0 : 12)		
6.1	Introduction to Control Structures	• List four types of flow control structures.	3-5
6.2	Selection Structures	• Use if and if/else statements for decision making. • Use a switch/case statement for multiple selections.	3-5
6.3	Looping Structures	• Use for, while, and do/while statements to control the number of repetitions for a group of statements.	3-5
7.	**Functions** (L:T:P = 3 : 0 : 15)		
7.1	Introduction	• Understand the basic concept of functions.	3-5
7.2	Function Definition	• Know how to define functions with or without arguments.	3-5
7.3	Function Prototypes	• Know how to declare a function (prototype).	3-5
7.4	Function Call	• Understand how to call a function and how to pass arguments to a function.	4
7.5	Function Communication	• Know how functions communicate with each other. • Know how to return value to the calling function and how to make use of the returned value.	4
8.	**Arrays** (L:T:P = 3 : 0 : 9)		
8.1	Why Arrays are Used	• Understand why and when to use arrays.	4
8.2	Declaration of Arrays	• Understand the definition of an array.	4
8.3	Array Manipulation	• Know how to refer to individual elements of an array.	4

No	TOPICS (L:P:T) / SUBTOPICS	INSTRUCTIONAL OBJECTIVES (What You want students to achieve)	BIB. REF
		• Be able to store values into an array and read values out from an array. • Be able to search lists and tables of values stored in arrays.	
8.4	Multi-Dimensional Arrays	• Know how to define and use multi-dimensional arrays.	4

ADDITIONAL NOTES:

No of Assignments to set	Two (2),1 Quiz + 1 Mini-project
No of Tests to set	Two (2), Practical tests
No of Supplementary Test to set	0
Duration of each test	1 hr (Practical test 1), 1 hr (Practical test 2)
Test to cover which topics	Topics 1- 8
No of Examination questions to set	0

Appendix C: PrC module delivery (teaching) plan

Case 2005-S1: GUIDELINES FOR IT1733 MODULE

IT1733 Principles of Computing introduces students to fundamentals of computer programming technology, such as concepts of computer software systems, algorithm design, modular program design, decision making and program flow control. At the end of the module, students must be able to

- Design programs using pseudo codes and modular design principles
- Implement these programs using C/C++
- Write C/C++ programs involving decision making and flow control
- Write functions in C/C++

There is NO exam for this module. Please note following schedule for assignments and tests:

Wk	Description	Topics Tested	%	Session	Assessed By	Remarks
3	e-Quiz	e-Learning material (Weeks 1-3)	-	e-learning	Self-practice	MCQ Questions 20 minutes
7	Interim report submission	Algorithms & Modular Design	5	Lab	MT	-
8	Lab Test1	Basic Data Types Operators & Expression Program Flow Control (Week 3 – 6)	10	Lab	MT	2 Questions. 45 minutes
8	e-Quiz	e-Learning material (Weeks 4-8)	-	e-learning	Self-practice	MCQ Questions 20 minutes
11	Lab Test1	Functions Program flow control	10	e-learning	MT	
13	Common Test (written)	Algorithm design Operators & Expressions **Program Flow Control Functions (without parameters) 1 D Array** (Week 1 – 11)	40	Lecture	MT	Structured question
15	Submission of final report	Project presentation	25	Lab	MT	
16	e-Quiz	e-Learning material (weeks 1-16)	-	e-learning	Self-practice	MCQ Questions 30 minutes
16	Disc. Forum	Decision Making	-	e-learning	Self-practice	
1-12	Tutorial Work	All tutorial classes	10	Lab	MT	Assignment submission

MINI PROJECT

The mini project will commence on Week 7. A project briefing will be conducted during week 7 lecture by the tutors. Students will be told to access the student drive to access the project specs. A copy of the student project briefing and project specs is available in the MT file.

The tutors of the respective groups will primarily be responsible for the assessment of this component.

ATTENDANCE

Please use Student Attendance System to update students' attendance for those sessions you are teaching before end of each week.

COURSE MATERIAL

All the students will be told during the first lecture to purchase the course material (lecture notes, tutorials & practical) from the print shop.

Case 2005-S1: Student Learning Plan for IT1753

Week	Lecture Topics	Lab Topics
1	W1A: Computer Software Systems	W1B: Algorithms and Program Design
2	W2A: Structure of C# program	W2B: Variables & Data Types W2C: Input and Output
3	W3A: Operators and Expressions – Binary Operators	W3B: Unary Operators W3C: Relational Operators
4	W4A: Flow Control and Decision Making: Conditional IF-else statement	W4B: IF-else conditions using Operators ***eQuiz 1 (W1 to W2)***
5	W5A: Modular Programming using Methods	W5B: Methods without Parameter Passing
6	W6A: Flow Control and Decision Making: WHILE loops	W6B: DO-WHILE loops ***Labtest 1 (W1 to W4)***
7	W7A: Single Dimensional Arrays ***eQuiz 2 (W5 to W6)***	W7B: Arrays Manipulation ***Project start,***
8	W8A: Flow Control and Decision Making: Switch-Case	W8B: Review of Loops, Project Coding
9 – 10	Term Break	
11	W11: Methods with Parameter Passing	Lab Assignment ***First project submission***
12	W12: Methods with Return Values	Lab Assignment, Project Coding ***Labtest 2 (W1 to W8)***
13	W13: Strings & String Methods	Project Coding & Review
14	W14: Multi-dimensional Arrays	Project Coding & Review
15	W15: File Input-Output Processing ***eQuiz 3 (W1 to W12)***	***Final project presentation& submission***
16	eLearning: Review Case study	***Final Quiz (W1 to W12)***
17	Self-Directed Learning: Review all materials	

Appendix D: Specimen teaching material

Case 2005-S1

Principles Of Computing

- Introduction to C/C++
- Variables and Data Types
- Input/output

Topics

- Structure of a C/C++ program
- Variables and Basic Data Types
- Input/Output

Objectives

- Be able to understand a basic C++ program
- Be able to define what is a variable
- Explain the various data types
- Be able to understand input and output

Structure of a C/C++ Program

- Let's start by examining a simple program that displays "Good day! " on the computer

```
1  #include <iostream>
2  using namespace std;
3  /* my first program */
4  int main(void)
5  {
6     cout << "Good Day!";
7     return 0;
8  }
```

Pre processor directive
This is a comment
Default module to be executed
Module must start and end with braces
Display "Good Day"

Listing : myFirstProg.cpp

Structure of a C/C++ Program

- **Line 1**
  - This is known as a pre-processor directive. A pre-processor is part of the C++ compiler.
  - This line is an instruction to include the file **iostream** inside the C++ program.
  - **iostream** contains all the basic declarations of input/output, defined in the 'std' namespace.
- **Line 2**
  - This tells the computer that we are using everything which is defined is 'std'.
- **Line 3**
  - This is a comment included in the program for documentation and clarification purpose.
  - Comments are ignored by the C++ compiler during compilation

Structure of a C/C++ Program

- **Line 4**
  - Recall that as part of program design, major tasks are broken into smaller tasks called modules. Each module is assigned a name for identification.
  - In C/C++ the default module (that is, the module which is executed when the program runs) is the **main()** module.
  - The keywords **int** and **void** will be explained in the lecture on Functions.
  - The statements under main() are enclosed inside { and } (on lines 5 and 8).

Structure of a C/C++ Program

- **Line 6**
  - The message enclosed in double quotation marks "Good Day!" is called a string.
  - The << notation is to send the message to cout
  - cout is the command to use for displaying information on the screen (computer monitor).
- **Line 7**
  - This line tells the computer that we have finished running the module.
  - The 'return' means go back to where you came from.
  - The '0' means take back the value 0.
  - This will be explained in the lecture on Functions.

Structure of a C/C++ Program

- Example 1 : What is wrong with the following Program?

```
#include <iostream>
using namespace std;
/* my first program */
int main(void)
{
   cout << "Hi there";
   return 0;
}
```

Variables

Variables - What are variables?

Variables are names assigned by you, the programmer, to store data in your program.

```
Examples of variables (Pseudo code ):
Read max_temp, min_temp
```

For example, the pseudo code above will read in 2 values from the keyboard, and store them into the variables max_temp and min_temp.

In C/C++, variable names must follow certain naming conventions which we will look at now.

Variable naming convention

- Cannot be a C keyword
- Must begin with a letter or an underscore
- Can be followed by letters, underscore or digits.
- Cannot have special characters (such as control chars, space)
- Only the first 32 characters of the identifiers are significant.
- Case sensitive

Variable naming convention

Examples of invalid variable names

void	reserved C keyword
3days	must begin with letter or underscore
two fold	cannot have space

Examples of valid variable names

- Total
- TOTAL
- end_of_file
- _good

Variable naming convention

- Examples of invalid variables names. Why are they invalid?

say_what?	Cannot contain ?
31day	Must start with letter or underscore
stop!	Cannot contain !
stop there	Cannot contain space(s)

Variables & Data Types

- In most programming languages including C and C++, variables must be **declared** before they can be used.
- For example, in C/C++ it will be as follows:

```
int main(void)
{
    int max_temp, min_temp;
    ...
    return 0;
}
```

- Variable declarations tell the compiler
  - What the names of the variables are, and
  - What kind of information (data type) is stored in the variables

Variables & Data Types

```
int main(void)
{
    int max_temp, min_temp;
    ...
    return 0;
}
```

- In the above example, **int** is an integer (whole number) data type. This means that the max_temp and min_temp variables can only store values which are whole numbers.
- We will now examine the more common basic data types in C/C++.

Variables & Data Types

- **int**
  - Represents whole numbers
  - Range between -32767 and 32767
  - Example:

```
int max_temp = 41;
```

- **long**
  - Represents whole numbers
  - Range from 0 to4294967295
  - Example:

```
long population = 3100000L;
```

Variables & Data Types

- ***float and double***
  - Represents floating point numbers (ie numbers with decimal points)
  - Examples:

```
float height = 1.75;
double sensitivity = 0.00456;
```

When to use float or double?

	float	double
Precision	6 digits	15 digits
Magnitude	between about 10^{-38} and 10^{38}	between about 10^{-308} and 10^{308}

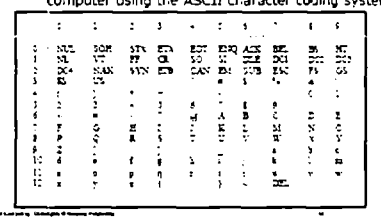
Variables & Data Types

- **char**
  - Represents a character value – letter, digit, or special symbol
  - Each character must be enclosed in single quotes.
  - Examples:

```
char letter = 'A';
char digit = '2';
char asterisk = '*';
```

Variables & Data Types

- **char**
  - Character values are represented internally in computer using the ASCII character coding system.



More than one character

- Recall that a character variable can only store a single character.

```
EXAMPLE:
char letter ='A';
```

- If we want to represent a sequence of characters, for example "John", we need to use an array of characters.

Strings

- What are strings?
  A string is a sequence of characters enclosed by double quotes.

```
Example :
cout << "This is a character string."
cout << "I can count 1 2 3 4 5 6 7 8 9 10"
```

Declaration of strings

- C **does not** support strings as a data type
- C++ **does** support strings as a data type
- Strings are represented by arrays of type char, terminated by a null character , '\0'.

```
Examples :
In C or C++ we do this:
  char full_name[30]; /* this is an array of characters */
  char job_title[20]; /* this is another array of chars */
or in C++ we can do this:
  string full_name ; /* this is a string variable */
  string job_title; /* this is another string variable */
```

Initialization of strings

- We can use a string constant to initialise an array of characters.

```
Example :
char job_title[10] = "student";
```

s	t	u	d	e	n	t	\0	?	?
[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]

Initialization of strings

- We can also initialise a string variable without declaring the size of the array.

```
Example :
string job_title = "student";
```

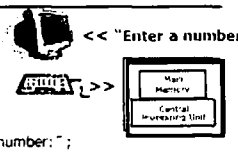
s	t	u	d	e	n	t	\0
[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]

Input and Output

- In C++, input and output is done using streams.
- A *stream* is a flow of data, usually characters.
- The following streams can be used:
  - cin — used for input (typically from keyboard)
  - cout — used for output (typically to monitor)
- These basic streams are defined inside <iostream>
- When we do **#include <iostream>** the computer lets us use them.
- Note that the full names are std::cin and std::cout.
- When we type **using namespace std;** we can shorten the names to cin and cout.
- The operators << and >> are used to put data into a stream.

Input and Output

: Example :




```
#include <iostream>
using namespace std;
int main(void)
{
        int number;
        cout << "Enter a number: " ;
        cin  >> number;
        cout << "Number = " << number;
        return 0;
}
```

Example output:
Enter a number : 12
Number = 12

Input and Output

: Calls to << and >> can be put together.
: Example :

```
#include <iostream>
using namespace std;
int main(void)
{
  cout << "sum=" << 7+3 << endl;
  cout << "End of program";
  return 0;
}
```

Output:
sum=10
End of program

NB : endl means go to next line, that is, it instructs the cursor to move to the next line on the display

Input and Output

: Calls to << and >> can be put together.
: Example :

```
#include <iostream>
using namespace std;
int main(void)
{
  int x, y;
  cout << "Enter 2 numbers:";
  cin >> x >> y;
  return 0;
}
```

NB : x is read in before y.
Example input:
Enter 2 numbers :12 13
12 will be assigned to x
13 will be assigned to y

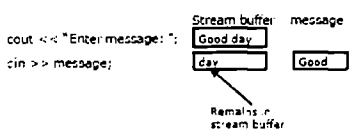
Input and Output

: A restriction of cin is that it treats white space as a delimiter
: Example :

```
#include <iostream>
#include <string>
using namespace std;
int main(void)
{
  string message;
  cout << "Enter message: ";
  cin >> message;
  cout << "Display: " << message << endl;
  return 0;
}
```

Example input & output:
Enter message: **Good day!**
Display: **Good**

Input and Output



Input and Output

: If reading a string is required, including white space, use getline()
: Syntax:

std::getline(std::istream_Istr, std::string _Str, char _Delim)

: Reads characters from the input stream _Istr until the delimiter _Delim is reached.
: Characters read are stored into the string _Str
: The delimiter is removed from the input stream buffer and not actually stored in the string.

Input and Output

- Example:

```
#include <iostream>
#include <string>
using namespace std;
int main(void)
{
   string name;
   cout << "Enter name:";
   getline(cin, name, '.');   // reads until a '.' is found
   cout << "Name : " << name;
   return 0;
}
```

```
Enter name: John Tan, [Enter]
Name : John Tan
```

Input and Output

- Example: The default delimeter is the enter key '\n'

```
#include <iostream>
#include <string>
using namespace std;
int main(void)
{
   string name;
   cout << "Enter name:";
   getline(cin, name);   // reads until a '\n' is found
   cout << "Name : " << name;
   return 0;
}
```

```
Enter name: John Tan[Enter]
Name : John Tan
```

Input and Output

- cin.ignore
  - Is used to skip over a number of characters or up till and including when the delimiter character is reached.
  - Syntax :

cin.ignore(int num, char delimiter)

- Skips over num characters or up till and including when delimiter is reached.

Input and Output

- Example :

```
OUTPUT:
Enter age: 18[Enter]
Enter name: De Hui[Enter]
Age=18
Name= De Hui
```

	Stream buffer	name	age
cout << "Enter age: ";	18[Enter]		
cin >> age;	[Enter]		:8
cout << "Enter name: ";	[Enter]'De Hui[Enter]		
getline(cin, name, '\n');	De Hui[Enter]		

NB the [Enter] key is usually represented as '\n' in a computer program.

Input and Output

- Example :

```
OUTPUT:
Enter age: 18[Enter]
Enter name: De Hui[Enter]
Age=18
Name= De Hui
```

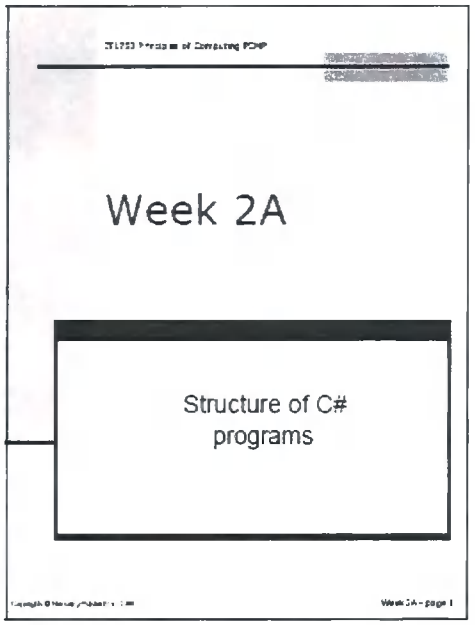
	Stream buffer	name	age
cout << "Enter age: ";	18[Enter]		
cin >> age;	[Enter]		:8
cin.ignore();			
cout << "Enter name: ";	De Hui[Enter]		
getline(cin, name, '\n');		De Hui	

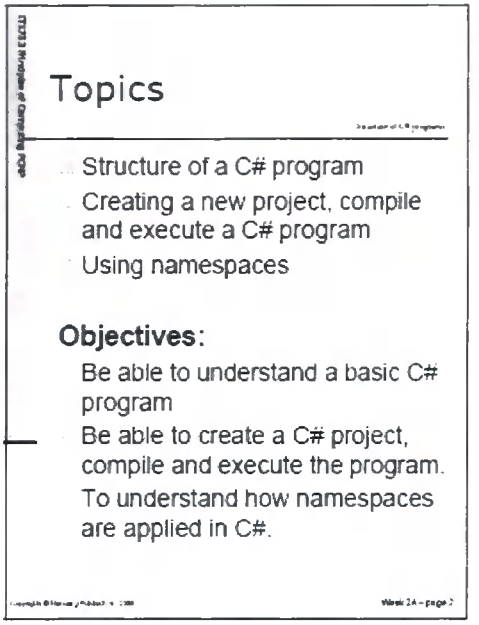
NB the [Enter] key is usually represented as '\n' in a computer program.

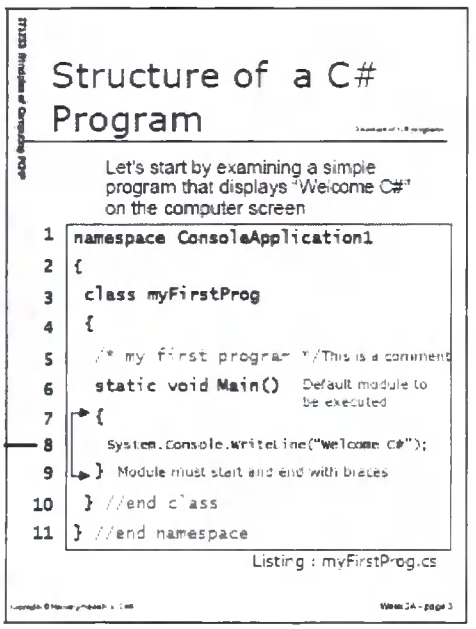
Summary

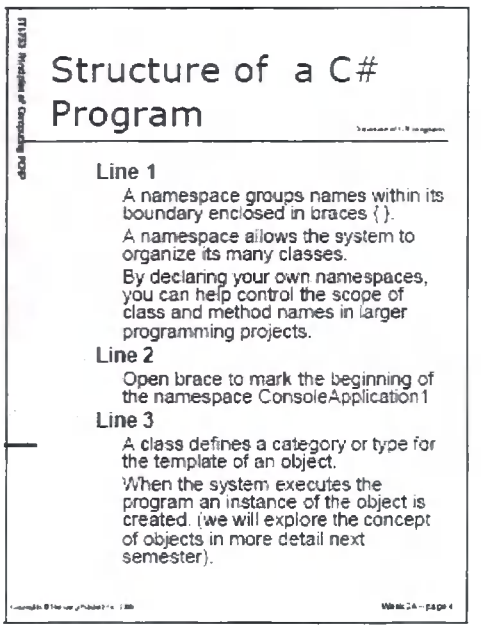
- Structure of a C++ program
- How to define proper variable names
- Basic data types in C++
- Basic Input/Output

Case 2006-S1


Week 2A
Structure of C# programs


Topics
Structure of a C# program
Creating a new project, compile and execute a C# program
Using namespaces
Objectives:
Be able to understand a basic C# program
Be able to create a C# project, compile and execute the program.
To understand how namespaces are applied in C#.


Structure of a C# Program
Let's start by examining a simple program that displays "Welcome C#" on the computer screen
1 namespace ConsoleApplication1
2 {
3 class myFirstProg
4 {
5 /* my first program */ //This is a comment
6 static void Main() Default module to be executed
7 {
8 System.Console.WriteLine("Welcome C#");
9 } Module must start and end with braces
10 } //end class
11 } //end namespace
Listing : myFirstProg.cs


Structure of a C# Program
Line 1
A namespace groups names within its boundary enclosed in braces { }.
A namespace allows the system to organize its many classes.
By declaring your own namespaces, you can help control the scope of class and method names in larger programming projects.
Line 2
Open brace to mark the beginning of the namespace ConsoleApplication1
Line 3
A class defines a category or type for the template of an object.
When the system executes the program an instance of the object is created. (we will explore the concept of objects in more detail next semester).

Structure of a C# Program

- **Line 4**
  - Open brace to mark the beginning of the class myFirstProg.
- **Line 5**
  - This is a comment or remarks that explain the program code.
  - Comments are ignored by the C# compiler during compilation.
- **Line 6**
  - Recall that as part of program design, major tasks are broken into smaller tasks called modules. Each module is assigned a name for identification.
  - In C#, the default module (that is, the module which is executed when the program runs) is the **Main()** module.
  - Modules in C# are known as methods.
  - static void defines the behaviour of the Main method. (Further explanation will be given in the Methods topic later.)

Structure of a C# Program

- **Line 7**
  - Open brace to mark the beginning of the Main() method.
- **Line 8**
  - The message enclosed in double quotation marks "Welcome C#" is called a string.
  - The Console is an object that maps to your computer screen. It is defined in the class Console within the System namespace: System.Console
  - WriteLine() is a method of the Console class that accepts the string "Welcome C#".
- **Lines 9,10,11**
  - Closing brace for each open brace respectively.

Structure of a C# Program

Example 1 : What is wrong with the following Program?

```
using System;
/* my first program
Void Main(void)
{
    Writeline("Hello my FRIEND")
}
```

Rewrite the corrected program:

Using Namespace

Compare this program with the one in the previous example (p.3)

```
using System;
namespace ConsoleApplication1
{
 class myFirstProg
 {
  /* my first program */
  static void Main()
  {
  // System.Console.WriteLine() is recoded to:
     Console.WriteLine("Welcome C#");
  }
 } //end class
} //end namespace
```

System namespace contains many built-in C# classes. For this module, we are using the Console class and its methods like WriteLine.

Summary

- Understand how a C# program is structured
- How to create a C# project, compile and execute C# application

Review the eLearning material on how to start Microsoft Visual Studio 2005 (MSVS 2005).

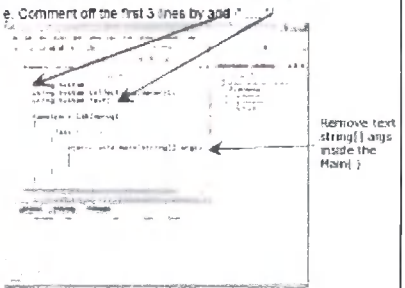
Practical 2A

1. How to create a C# Console Application on MSVS 2005.
   a. On the desktop, click on the MSVS 2005 icon.
   b. On the **File** menu, point to **New** and then click **Project**.

   c. On the **Templates:** box, select **Console Application**.
   d. Select **Location** field, and type the following `C:\temp\06nnnnA` (enter your admin no.), in the **Name** field, type **Lab2Aprog1**, and click **OK** button.

Practical 2A

1. How to create a C# Console Application on MSVS 2005.
   e. Comment off the first 3 lines by add /* ... */

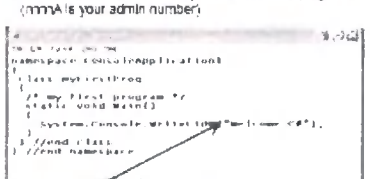


   f. Type the following inside the Main() method:
      `System.Console.WriteLine("Welcome C#");`
   g. Press Ctrl F5 to compile and run your application.
      A text message is displayed: Congratulations! You've just written your first C# application.

Remove the comments /*...*/, see what happens, refer to slide 8.

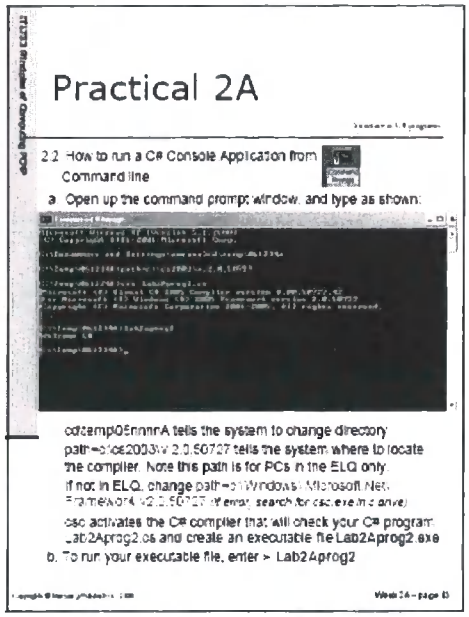
Practical 2A

2. How to create a C# Console Application on Notepad.
   a. On the desktop, click on the Notepad icon.
   b. When the Notepad editor is open, cut and paste the program in slide 3 and save as `C:\temp\06nnnnA\Lab2Aprog2.cs` (nnnnA is your admin number).



   c. Change "*.txt" to "*.*", save the file (select **File** > **Save**).
   d. Check the filename, it must be Lab2Aprog2.cs

Next we will see how we compile and execute the text file Lab2Aprog2.cs from the command line.



Practical 2A

2.2 How to run a C# Console Application from Command line

a. Open up the command prompt window, and type as shown:

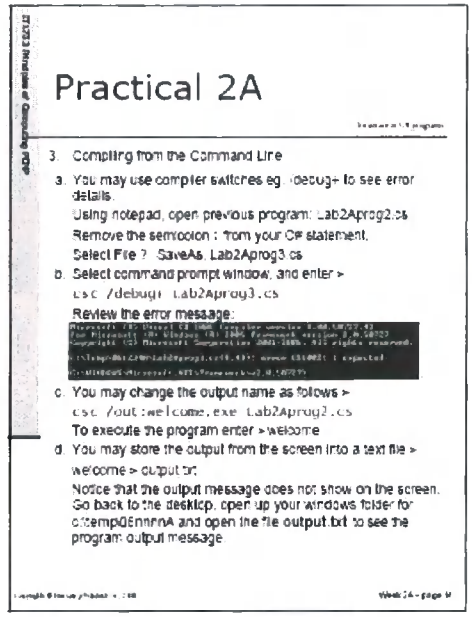
c:\temp05nnnnA tells the system to change directory

path=c:\csc2003\v 2.0.50727 tells the system where to locate the compiler. Note this path is for PCs in the ELQ only.

If not in ELQ, change path=c:\Windows\Microsoft.Net\ Framework\v2.0.50727 (if error, search for csc.exe in c drive)

csc activates the C# compiler that will check your C# program Lab2Aprog2.cs and create an executable file Lab2Aprog2.exe

b. To run your executable file, enter > Lab2Aprog2



Practical 2A

3. Compiling from the Command Line

a. You may use compiler switches eg. /debug+ to see error details.

Using notepad, open previous program: Lab2Aprog2.cs

Remove the semicolon ; from your C# statement.

Select File ? SaveAs. Lab2Aprog3.cs

b. Select command prompt window, and enter >

```
csc /debug+ Lab2Aprog3.cs
```

Review the error message:

c. You may change the output name as follows >

```
csc /out:welcome.exe Lab2Aprog2.cs
```

To execute the program enter >welcome

d. You may store the output from the screen into a text file >

welcome > output.txt

Notice that the output message does not show on the screen. Go back to the desktop, open up your windows folder for c:\temp05nnnnA and open the file output.txt to see the program output message.

Appendix E: Specimen project specifications

Case 2005-S1

IT1733 Mini Project

PROJECT OBJECTIVE

- The objective of the mini project is to integrate concepts and knowledge learned in this module, and apply it in a real life application.
- The project specifications given here are somewhat simplified, but should nevertheless, give you a flavor of how, given a problem, to define a problem, break down the solution in a structured manner, design the algorithms, implement the algorithms in C/C++, and finally test your program.

PROJECT OVERVIEW

- Project comprises 25% of the overall assessment marks
- Project is to be undertaken in groups of 2 students.
- Students are to submit the names of members in their group to the tutors by week 7.
- There are in all 4 project titles to choose from
  - **A: Seat Reservation System**
  - **B: Drinks Vending Machine**
  - **C: Card Issue Verification**
  - **D: Stock Sales Order System**
- There shall be ***no more than 3 groups*** in any class having the same project title.

PROJECT OVERVIEW

- If project is undertaken by only **1 student per group**, the project requirements will be reduced as follows :

Project	Requirements to remove:
A	-View waiting List -Transaction report generation
B	- Expiry date entry and checking -Transaction report generation
C	-Blacklist configuration & checking -Report generation
D	-Replenishment of items -Report generation

PROJECT MILESTONES

Wk	Description	Remarks
4	*Project starts with briefing by lab MT*	
5	Project title and name of group members to be submitted to MT	
10	Interim report to be submitted to MT	See submission requirements. 5%
15	Final report to be submitted to MT *Project presentation*	See submission requirements. 20%

GUIDELINES

GENERAL

- Before you start discuss with your MT on how to proceed with your project.
- When in doubt, always asks your MT
- Never wait until the last minute to submit your work

REGULATIONS

- Project **MUST BE ORIGINAL**
- No copying of others' solutions. Students found copying or allowing others to copy will get zero marks

ASSESSMENT CRITERIA

20%	Report includes all contents as specified in the report format
10%	Program Compiles and Runs
10%	Meaningful variable names, comments and indentation of code
5%	Be able to answer questions during presentation that shows you understand: Functions (without parameter passing us ng global variables)
5%	If-else statements
5%	While loops
5%	1 Dimensional Arrays
5%	For Loops
5%	Switch Case Statements
5%	2 Dimensional Arrays
5%	Functions with parameter passing
20%	All the functionality in the project completed and/or extra "creative" functionality has been added

PROJECT SUBMISSION

INTERIM REPORT SUBMISSION

- The interim report should comprise of a print-out of your program in week 10. The program does not need to be complete

FINAL REPORT SUBMISSION

- The final report comprises of a print-out of your completed program in week 15.

PRESENTATION

- The presentation in week 15 will involve you explaining your program for 5 minutes and your MT asking questions to verify that you have written the code.

Nanyang Polytechnic

MINI-PROJECT A

Course: Diploma in Engineering Informatics
Module: IT1733 – Principles of Computing

Time Allowed: 9 weeks

1. Introduction

The objective of this project is to develop an Airline Reservation System.

Green Dot airline has just purchased a computer for its new automated reservation system. You have been asked to program the new system. You are to write a program to assign seats on each flight of the airline's only plane. Assume the plane has 6 rows with 3 seats in each row.

When your program starts, a system menu will be displayed:

```
GREEN AIRLINE ON-LINE RESERVATION SYSTEM
(A)   Reserve a seat
(B)   Cancel a reservation
(C)   Display Seating arrangement
(D)   View Waiting List
(E)   Transaction Report
(F)   Quit
>> Enter Choice :  XX
```

- **Reserve a Seat** adds a person to a flight or waiting list
- **Cancel a Reservation** removes a passenger from a flight or waiting list
- **Display Seating arrangement** displays the seats occupation and the customer ID occupying it.
- **View waiting list** allows viewing of customer IDs who are placed on waiting list
- **Transaction Report** allows generation of a report containing all transactions taken place

2. Functional Requirements

2.1 Reserve a seat

When a customer wants to reserve a seat, you should do following:

a) Input a passenger ID
b) Check if any seats are available. If yes, allocate a seat.
c) If seats are not available, check if the customer wants to be put in the waiting list. If yes, assign the passenger ID to waiting list.

d) Enter time and date of transaction.

For each transaction, a log is generated. The log data will comprise of the following:

Field	Description	Format
Transaction Code	A code to identify type of transaction	1 - Assign seat 2 – Put on waiting List 3 – Remove from waiting List 4 - Cancel reserved seat 5- Cancel from waiting List
Passenger ID	A unique ID which identifies the passenger	1-9999
Time	Time of transaction	HHMM
Date	Date of transaction	DDMMYY

A maximum of 100 transactions can be stored.

2.2 Cancel a reservation

When a customer wants to cancel a reservation, the program will display the following.:

```
Cancel Reservation
(A)   Cancel Reserved seat
(B)   Cancel queue in waiting list
(C)   Return to previous screen
>> Enter Choice :  XX
```

a) Request for the passenger's ID
b) Search for the passenger's ID and delete it.
c) For (A), if the waiting list is empty, update the seating array so that the seat is available for the other bookings
d) For (A), if the waiting list is not empty, get the first person in the waiting list who opted for the same seating category, and allocate the seat to them. The system can accommodate of maximum of 10 waiting lists.

2.3 Display Seat arrangement

Create your own output screen design to show the seating arrangement.

```
          Seat1      Seat2      Seat3
Row 1:    [1223]     [3232]     []
Row 2:    [1211]     [1121]     []
Row 3:    []         []         []
Row 4:    []         [3411]     []
Row 5:    []         []         []
Row 6:    []         []         []
```

2.4 Display the waiting list

Design your output to show the waiting list. The following is an example:

```
Waiting List [3]:
=================
1. 1231             140203
2. 1341             140203
3. 2313             150203
```

2.5 Transaction Report

A transaction report will be generated when this option is selected. The following will be displayed: the transaction code, passenger ID, and date and time of transaction. For example :

Transaction Report

Transaction Code	Passenger ID	Time	Date
1	1223	1130	140203
2	1231	1230	140203
.....			

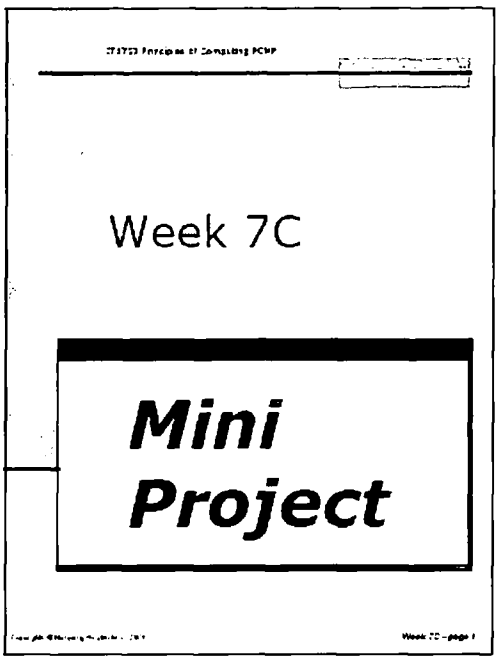
3. Hints

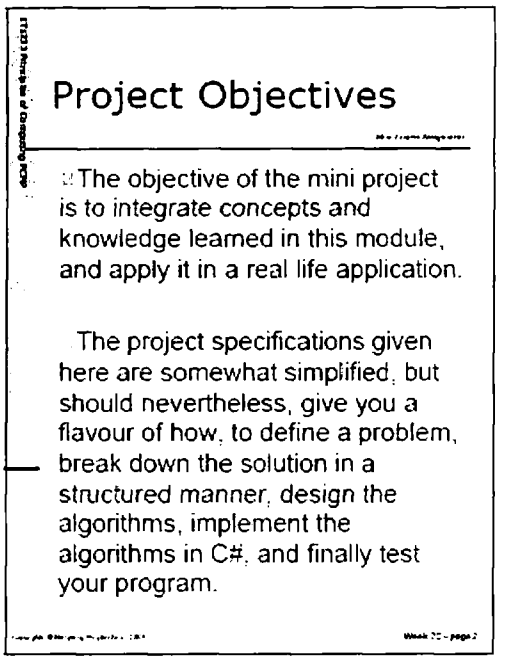
- Array declaration
  To keep records of your data, you will need to declare the following 1 Dimensional arrays to store the following information : passenger ID , waiting list ID, and transaction information to capture the transaction code, passenger ID, date and time of transaction.

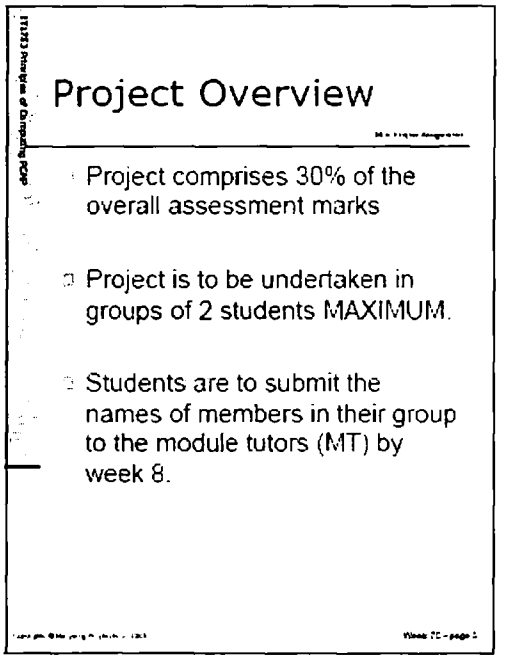
- Modular Program Design
  Your program should be designed in a modular manner, using a Structure Chart. As a guideline, each module that you develop should not comprise more than 30 lines of pseudo code.

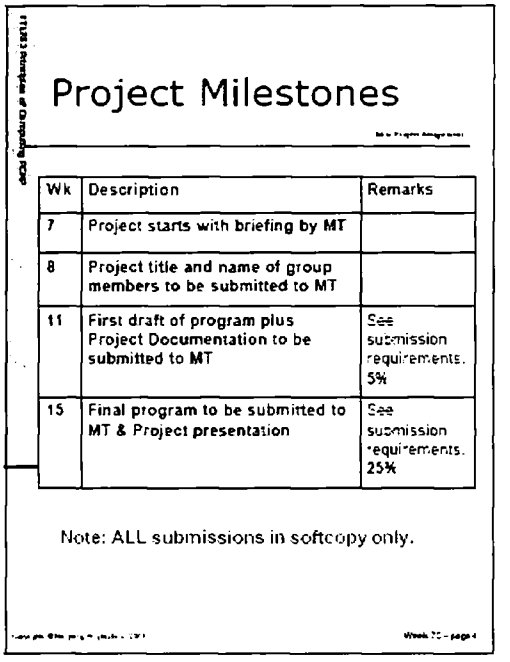
- Decision Making
  You are free to make use of If/else, switch/case, while loops, arithmetic, relational and logical operators in implementing decision making.

Case 2006-S1


Week 7C
Mini Project


Project Objectives
The objective of the mini project is to integrate concepts and knowledge learned in this module, and apply it in a real life application.
The project specifications given here are somewhat simplified, but should nevertheless, give you a flavour of how, to define a problem, break down the solution in a structured manner, design the algorithms, implement the algorithms in C#, and finally test your program.


Project Overview
Project comprises 30% of the overall assessment marks
Project is to be undertaken in groups of 2 students MAXIMUM.
Students are to submit the names of members in their group to the module tutors (MT) by week 8.


Project Milestones
Wk
Description
Remarks
7
Project starts with briefing by MT
8
Project title and name of group members to be submitted to MT
11
First draft of program plus Project Documentation to be submitted to MT
See submission requirements. 5%
15
Final program to be submitted to MT & Project presentation
See submission requirements. 25%
Note: ALL submissions in softcopy only.

Guidelines

GENERAL

Before you start, discuss with your MT on how to proceed with your project.

When in doubt, always asks your MT

Never wait until the last minute to submit your work.

From week 13, you must show progress in your program coding.

REGULATIONS

Project **MUST BE ORIGINAL**

No copying of others' solutions. Students found copying or allowing others to copy will have to resubmit their projects.

Assessment Criteria

15%	First Draft program should have menu options and at least two methods. Report must have defining table, structure charts and pseudo-code.
20%	Progress of Program Development in week 13 and week 14
10%	Meaningful variable names, comments and indentation of code.
40%	Be able to answer questions during presentation that shows you understand the following: Methods (without parameter passing with class variables)
	If-else statements
	While or Do-while loops
	1 Dimensional Arrays
	For Loops
	Switch-Case Statements
	Custom Methods with parameter passing
	Custom Methods with return value
15%	All the functionality in the project completed and/or extra "creative" functionality has been added

Week 11 SUBMISSION

1. First Draft PROGRAM
2. INTERIM REPORT SUBMISSION

The interim report comprises the following:

1. Introduction
   Give a brief introduction of the project background
2. Objective
   State the objectives of the program
3. Problem Definition
   Define the problem using a defining table
4. Identification of main processing tasks
5. Structure chart
   Draw up a structure chart
6. Pseudo codes of modules
   Include pseudo codes of at least 5 modules

Week 15: FINAL PRESENTATION

Project PRESENTATION (40%)

Demonstrate the capabilities of your application

Show how well your application is able to handle errors

Document your program by adding comments

For groups of 2 persons, indicate the contribution of each person to the application.

Nanyang Polytechnic

MINI-PROJECT / ASSIGNMENT

Course: Diploma in Engineering Informatics
Module: IT1753 – Principles of Computing

Submission: Week 15

Chickadee Fast Food

1. Introduction

Objective : Write a computer program to simulate an ordering system for a fast food company.

Overview : A fast food company is interested in setting up a computer system to improve their service to their customers. They would like to keep information on their customers for special offers and promotions. They also need a daily report on customers' favourite food.

When your program starts, a system menu will be displayed:

```
CHICKADEE FAST FOOD SYSTEM
(A)   Enter Food Menu
(B)   Enter Customer Information
(C)   Update Customer Order
(D)   Generate Customer Details
(E)   Display Order Transaction Report
(F)   Quit
>> Enter Choice :  XX
```

- **Enter Food Menu** adds a food details to the system.
- **Enter Customer Information** adds a customer details to the system.
- **Update Customer Order** allocates customer details and payment.
- **Generate Customer Details** shows all customers in the system and their favourite food.
- **Display Order Transaction Report** shows all order transactions.

2. Functional Requirements

2.1 Enter Food Menu

The system should allow the user to enter each food or drink item as follows:

Field	Description	Format
Food Code	A code to identify unique food	F0001
Food Name	Eg. Chicken Burger, Lemon Lime	text
Cost	amount per item	99.99
Promotion	discount given, if 0 means full price	99%

A maximum of 20 food items can be stored. You may include meal sets and extra value meals to provide more variety.

2.2 Enter Customer Information

The system should allow the user to enter the customer details as follows:

Field	Description	Format
Customer Code	A code to identify unique customer	C0001
Customer Name	Who the customer is	text
Contact no.	Handphone or resident no.	99999999
Date of birth	keep track of birthdays	DD/MM/YYYY

A maximum of 50 customers can be stored.

2.3 Update Customer Order

The system should allow the user to enter the customer order details as follows:

Field	Description	Format
Order Code	A running no of orders	1-99
Food Code	What is the food item	F0001
Customer Code	Which customer is this	C0001
Quantity	no. items ordered	99
Trans Date	Date item was served	DD/MM/YYYY
Payment mode	C – Cash N – Nets V – Gift voucher R – Credit Card	

A maximum of 100 orders can be stored in the system.

2.4 Generate Customer Details

The system will be able to display the customer information and their corresponding orders.

```
                    Chickadee Customer Information

I: Customer Details
      Customer Name     Code       Contact       Birth Date    Age
       Phua Chu Kang    C0001      91830951        5 Jan       45
       Phua Ah Beng     C0003      95601802       18 Mar       35
       Arnold Ang       C0005      64648585       31 Jul       10
       Kenny Rogers     C0007      65152525       28 Feb       22

II: Favourite Food Details
 Customer  Food Name          Qty    First Date    Last Date
 C0001     Chicken Burger      20    1/12/2004    21/01/2005
 C0001     Onion Rings         50     1/01/2005    31/01/2005
 C0005     Kids Meal            5    15/10/2004    31/12/2004
```

2.5 Display Order Transaction Report

The system will be able to display the following information:

```
                Chickadee Order Transactions

I: Cash
 Order   Food     Customer    Cost $    Qty     Trans Date
   1     F0001    C0006       5.50      1        8/01/2005
   3     F0003    C0001       2.50      2       18/01/2005
   8     F0001    C0007       5.50      1       28/01/2005

II: Nets
 Order   Food     Customer    Cost $    Qty     Trans Date
   2     F0001    C0002       5.50      1        8/01/2005
   4     F0002    C0003       6.50      1       18/01/2005
   5     F0002    C0004       6.50      3       28/01/2005

III: Gift Voucher
 ......................
 ......................
```

3. Recommendation

- Solution

  Select a specific domain e.g. fast food, sandwich bar, ice-cream parlour. You may determine your own solution domain. Inform your tutor before you start on your project.

- Modular Program Design

  Your program should be designed in a modular manner, using a Structure Chart. As a guide, each module corresponds to each item on your main menu.

- Menu control

  The user is allowed to choose the menu in any order and as long as he or she wants. Ensure that error messages are given for invalid input.

- Decision Making

  You are free to make use of If/else, switch/case, while loops, arithmetic, relational and logical operators in implementing decision making.

- Array declaration

  To keep records of your data, you will need to declare arrays and apply loops to control your data.

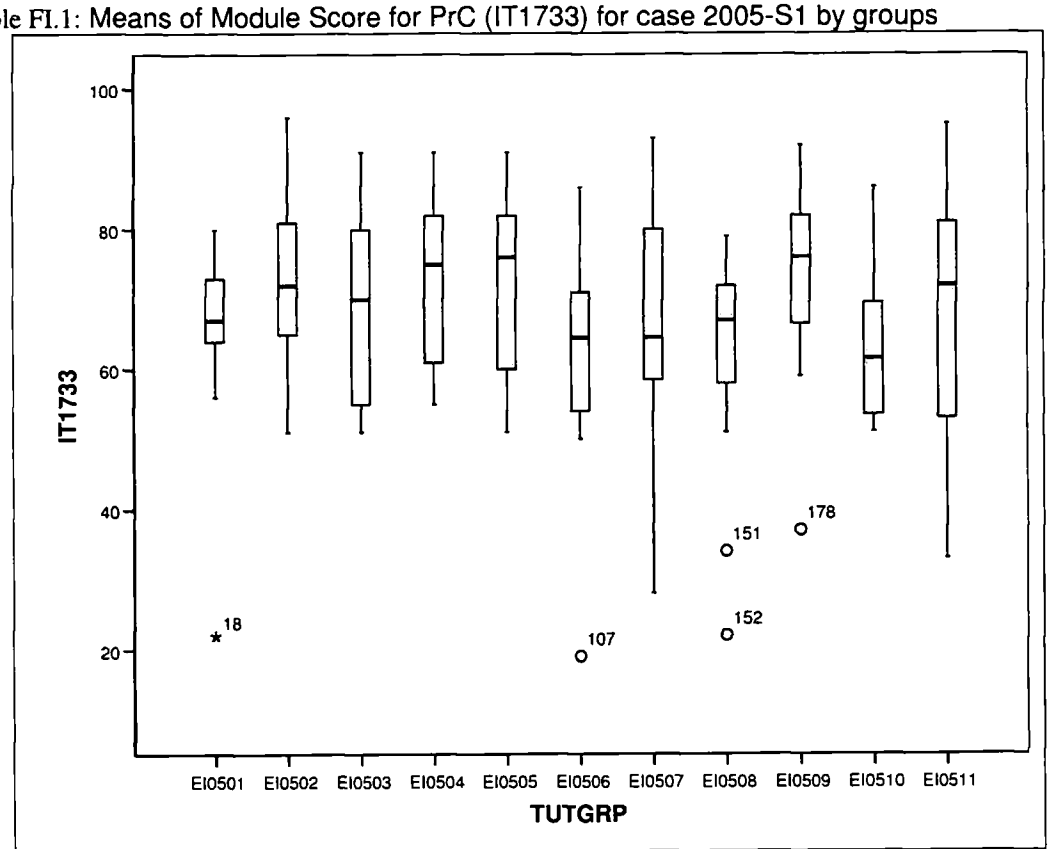
Sample codes are available in the e-Learning materials. You are highly advised to make use of the sample codes to enhance the solutions you produce.

Note:
You may change the data items and report layouts in agreement with your tutor. Please do this before working on your first draft.

Appendix F: Module group statistics

Note: Case 2005-S1: Tables FI, Case 2006-S1: Tables FII

Table FI.1: Means of Module Score for PrC (IT1733) for case 2005-S1 by groups



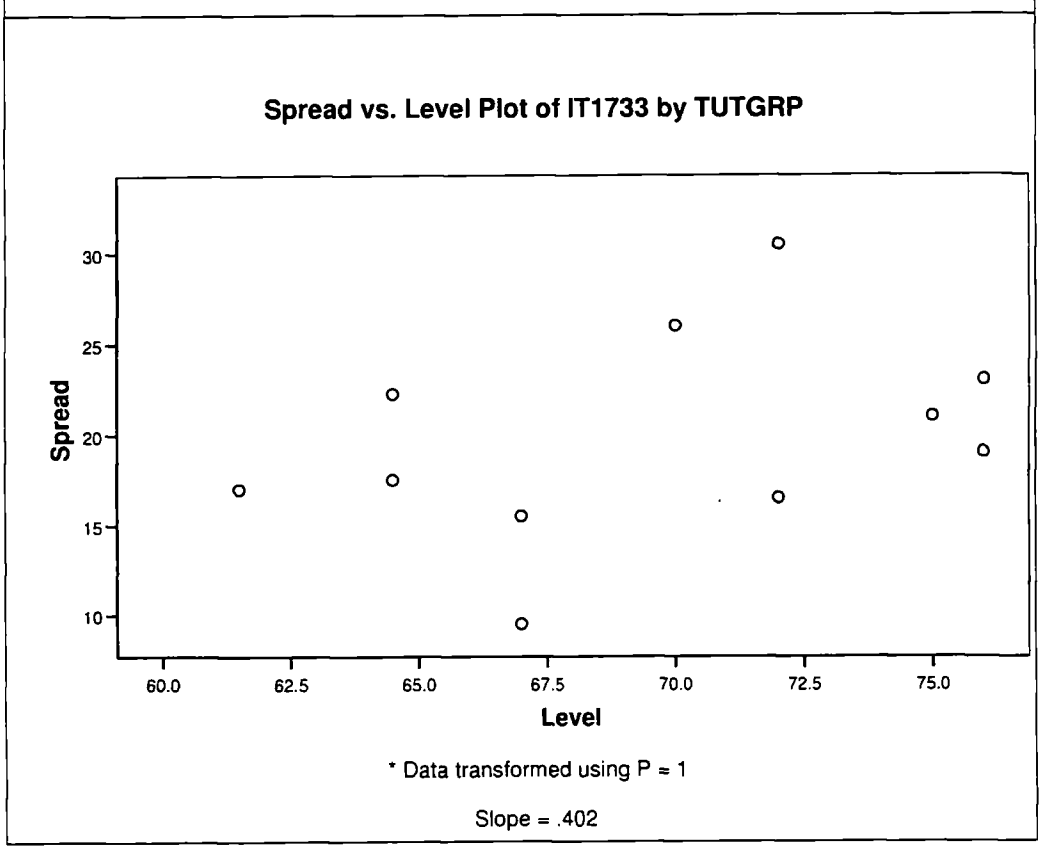


Table FI.2:Case 2005-S1: Descriptives

	Group	N	Mean	Std. Deviation	Std. Error	95% Confidence Interval for Mean		Minimum	Maximum
						Lower Bound	Upper Bound		
MODULE SCORE	1	21	66.48	11.686	2.550	61.16	71.80	22	80
	2	22	71.77	12.421	2.648	66.27	77.28	51	96
	3	21	68.62	13.075	2.853	62.67	74.57	51	91
	4	22	73.18	11.181	2.384	68.22	78.14	55	91
	5	19	72.16	13.124	3.011	65.83	78.48	51	91
	6	22	63.45	14.725	3.139	56.93	69.98	19	86
	7	20	67.25	15.389	3.441	60.05	74.45	28	93
	8	21	62.81	14.081	3.073	56.40	69.22	22	79
	9	23	73.65	11.949	2.492	68.48	78.82	37	92
	10	20	62.35	10.429	2.332	57.47	67.23	51	86
	11	21	69.95	16.409	3.581	62.48	77.42	33	95
	Total	232	68.39	13.558	.890	66.64	70.15	19	96
Project	1	21	62.2981	13.37926	2.91959	56.2079	68.3883	10.00	73.33
	2	22	54.6559	16.52837	3.52386	47.3276	61.9842	28.33	95.83
	3	21	65.8305	12.17677	2.65719	60.2877	71.3733	35.00	92.50
	4	22	71.5886	12.19079	2.59908	66.1835	76.9937	43.33	89.16
	5	19	69.2068	14.07166	3.22826	62.4245	75.9892	43.33	89.16
	6	22	61.2095	16.37119	3.49035	53.9510	68.4681	10.00	86.66
	7	20	66.4980	19.70533	4.40624	57.2756	75.7204	10.00	100.00
	8	21	55.4743	19.19226	4.18809	46.7381	64.2105	.00	72.50
	9	23	68.7287	11.28691	2.35348	63.8479	73.6095	35.00	84.16
	10	20	64.6460	9.28124	2.07535	60.3022	68.9898	50.00	83.33
	11	21	59.3610	20.27317	4.42397	50.1327	68.5892	28.33	95.83
	Total	232	63.5552	15.92532	1.04555	61.4951	65.6152	.00	100.00
IndvTest	1	21	67.6825	15.07407	3.28943	60.8209	74.5442	20.00	88.67
	2	22	79.3939	16.64559	3.54885	72.0137	86.7742	37.33	99.33
	3	21	68.7302	21.43922	4.67842	58.9711	78.4892	27.33	94.00
	4	22	73.9394	14.47627	3.08635	67.5210	80.3578	44.67	98.67
	5	19	72.8772	17.75561	4.07342	64.3193	81.4351	35.33	94.00
	6	22	63.0909	18.30783	3.90324	54.9737	71.2081	14.67	90.67
	7	20	66.7000	16.90078	3.77913	58.7902	74.6098	30.00	96.00
	8	21	64.8254	17.06706	3.72434	57.0566	72.5942	25.33	92.67
	9	23	76.1739	16.19105	3.37607	69.1724	83.1754	32.67	99.33
	10	20	58.6333	15.21230	3.40157	51.5138	65.7529	35.33	91.33
	11	21	74.6667	18.97249	4.14014	66.0305	83.3028	28.67	100.00
	Total	232	69.8218	17.85148	1.17201	67.5126	72.1310	14.67	100.00

Table FI.3: Test of Homogeneity of Variances

	Levene Statistic	df1	df2	Sig.
MODULE_SCORE	.968	10	221	.472
Project	1.388	10	221	.187
IndvTest	.862	10	221	.570

Table FI.4: ANOVA case 2005-S1 for Module Score

				Sum of Squares	df	Mean Square	F	Sig.
MODULE SCORE	Between Groups	(Combined)		3738.290	10	373.829	2.133	.023
		Linear Term	Unweighted	183.072	1	183.072	1.045	.308
			Weighted	158.111	1	158.111	.902	.343
			Deviation	3580.179	9	397.798	2.270	.019
		Quadratic Term	Unweighted	.338	1	.338	.002	.965
			Weighted	1.999	1	1.999	.011	.915
			Deviation	3578.181	8	447.273	2.552	.011
	Within Groups			38727.016	221	175.235		
	Total			42465.306	231			
Project	Between Groups	(Combined)		6585.309	10	658.531	2.799	.003
		Linear Term	Unweighted	.180	1	.180	.001	.978
			Weighted	.503	1	.503	.002	.963
			Deviation	6584.806	9	731.645	3.109	.002
		Quadratic Term	Unweighted	873.988	1	873.988	3.714	.055
			Weighted	870.786	1	870.786	3.701	.056
			Deviation	5714.020	8	714.252	3.036	.003
	Within Groups			51999.909	221	235.294		
	Total			58585.219	231			
IndvTest	Between Groups	(Combined)		8327.736	10	832.774	2.819	.003
		Linear Term	Unweighted	484.258	1	484.258	1.639	.202
			Weighted	436.736	1	436.736	1.478	.225
			Deviation	7891.000	9	876.778	2.968	.002
		Quadratic Term	Unweighted	237.152	1	237.152	.803	.371
			Weighted	278.692	1	278.692	.943	.332
			Deviation	7612.308	8	951.538	3.221	.002
	Within Groups			65286.233	221	295.413		
	Total			73613.969	231			

Table FI.5: Post Hoc Procedure: Multiple comparisons case 2005-S1

Dependent Variable		(I) Grp	(J) Grp	Mean Diff (I-J)	Std. Error	Sig.	95% Confidence Interval	
							Lower Bound	Upper Bound
MODULE SCORE	Games-Howell	1	2	-5.297	4.039	1.000	-18.84	8.24
			3	-2.143	4.085	1.000	-15.84	11.55
			4	-6.706	4.039	.995	-20.25	6.83
			5	-5.682	4.191	1.000	-19.73	8.37
			6	3.022	4.039	1.000	-10.52	16.56
			7	-.774	4.136	1.000	-14.64	13.09
			8	3.667	4.085	1.000	-10.03	17.36

Appendix F: Module group statistics

Dependent Variable	(I) Grp	(J) Grp	Mean Diff (I-J)	Std. Error	Sig.	95% Confidence Interval	
		9	-7.176	3.995	.980	-20.57	6.22
		10	4.126	4.136	1.000	-9.74	17.99
		11	-3.476	4.085	1.000	-17.17	10.22
	2	1	5.297	4.039	1.000	-8.24	18.84
		3	3.154	4.039	1.000	-10.39	16.69
		4	-1.409	3.991	1.000	-14.79	11.97
		5	-.385	4.146	1.000	-14.28	13.51
		6	8.318	3.991	.868	-5.06	21.70
		7	4.523	4.090	1.000	-9.19	18.23
		8	8.963	4.039	.766	-4.58	22.50
		9	-1.879	3.948	1.000	-15.12	11.36
		10	9.423	4.090	.690	-4.29	23.13
		11	1.820	4.039	1.000	-11.72	15.36
	3	1	2.143	4.085	1.000	-11.55	15.84
		2	-3.154	4.039	1.000	-16.69	10.39
		4	-4.563	4.039	1.000	-18.10	8.98
		5	-3.539	4.191	1.000	-17.59	10.51
		6	5.165	4.039	1.000	-8.38	18.70
		7	1.369	4.136	1.000	-12.50	15.24
		8	5.810	4.085	1.000	-7.89	19.51
		9	-5.033	3.995	1.000	-18.43	8.36
		10	6.269	4.136	.999	-7.60	20.14
		11	-1.333	4.085	1.000	-15.03	12.36
	4	1	6.706	4.039	.995	-6.83	20.25
		2	1.409	3.991	1.000	-11.97	14.79
		3	4.563	4.039	1.000	-8.98	18.10
		5	1.024	4.146	1.000	-12.87	14.92
		6	9.727	3.991	.563	-3.66	23.11
		7	5.932	4.090	1.000	-7.78	19.64
		8	10.372	4.039	.440	-3.17	23.91
		9	-.470	3.948	1.000	-13.71	12.77
		10	10.832	4.090	.371	-2.88	24.54
		11	3.229	4.039	1.000	-10.31	16.77
	5	1	5.682	4.191	1.000	-8.37	19.73
		2	.385	4.146	1.000	-13.51	14.28
		3	3.539	4.191	1.000	-10.51	17.59
		4	-1.024	4.146	1.000	-14.92	12.87
		6	8.703	4.146	.857	-5.19	22.60
		7	4.908	4.241	1.000	-9.31	19.13
		8	9.348	4.191	.756	-4.70	23.40
		9	-1.494	4.104	1.000	-15.24	12.25
		10	9.808	4.241	.682	-4.41	24.03
		11	2.206	4.191	1.000	-11.84	16.25
	6	1	-3.022	4.039	1.000	-16.56	10.52
		2	-8.318	3.991	.868	-21.70	5.06
		3	-5.165	4.039	1.000	-18.70	8.38
		4	-9.727	3.991	.563	-23.11	3.66
		5	-8.703	4.146	.857	-22.60	5.19
		7	-3.795	4.090	1.000	-17.50	9.91
		8	.645	4.039	1.000	-12.90	14.19
		9	-10.198	3.948	.427	-23.43	3.04

Appendix F: Module group statistics

Dependent Variable	(I) Grp	(J) Grp	Mean Diff (I-J)	Std. Error	Sig.	95% Confidence Interval	
		10	1.105	4.090	1.000	-12.60	14.81
		11	-6.498	4.039	.997	-20.04	7.04
	7	1	.774	4.136	1.000	-13.09	14.64
		2	-4.523	4.090	1.000	-18.23	9.19
		3	-1.369	4.136	1.000	-15.24	12.50
		4	-5.932	4.090	1.000	-19.64	7.78
		5	-4.908	4.241	1.000	-19.13	9.31
		6	3.795	4.090	1.000	-9.91	17.50
		8	4.440	4.136	1.000	-9.43	18.31
		9	-6.402	4.047	.998	-19.96	7.16
		10	4.900	4.186	1.000	-9.14	18.94
		11	-2.702	4.136	1.000	-16.57	11.16
	8	1	-3.667	4.085	1.000	-17.36	10.03
		2	-8.963	4.039	.766	-22.50	4.58
		3	-5.810	4.085	1.000	-19.51	7.89
		4	-10.372	4.039	.440	-23.91	3.17
		5	-9.348	4.191	.756	-23.40	4.70
		6	-.645	4.039	1.000	-14.19	12.90
		7	-4.440	4.136	1.000	-18.31	9.43
		9	-10.843	3.995	.319	-24.24	2.55
		10	.460	4.136	1.000	-13.41	14.33
		11	-7.143	4.085	.987	-20.84	6.55
	9	1	7.176	3.995	.980	-6.22	20.57
		2	1.879	3.948	1.000	-11.36	15.12
		3	5.033	3.995	1.000	-8.36	18.43
		4	.470	3.948	1.000	-12.77	13.71
		5	1.494	4.104	1.000	-12.25	15.24
		6	10.198	3.948	.427	-3.04	23.43
		7	6.402	4.047	.998	-7.16	19.96
		8	10.843	3.995	.319	-2.55	24.24
		10	11.302	4.047	.262	-2.26	24.86
		11	3.700	3.995	1.000	-9.69	17.09
	10	1	-4.126	4.136	1.000	-17.99	9.74
		2	-9.423	4.090	.690	-23.13	4.29
		3	-6.269	4.136	.999	-20.14	7.60
		4	-10.832	4.090	.371	-24.54	2.88
		5	-9.808	4.241	.682	-24.03	4.41
		6	-1.105	4.090	1.000	-14.81	12.60
		7	-4.900	4.186	1.000	-18.94	9.14
		8	-.460	4.136	1.000	-14.33	13.41
		9	-11.302	4.047	.262	-24.86	2.26
		11	-7.602	4.136	.972	-21.47	6.26
	11	1	3.476	4.085	1.000	-10.22	17.17
		2	-1.820	4.039	1.000	-15.36	11.72
		3	1.333	4.085	1.000	-12.36	15.03
		4	-3.229	4.039	1.000	-16.77	10.31
		5	-2.206	4.191	1.000	-16.25	11.84
		6	6.498	4.039	.997	-7.04	20.04
		7	2.702	4.136	1.000	-11.16	16.57
		8	7.143	4.085	.987	-6.55	20.84
		9	-3.700	3.995	1.000	-17.09	9.69

Appendix F: Module group statistics

Dependent Variable		(I) Grp	(J) Grp	Mean Diff (I-J)	Std. Error	Sig.	95% Confidence Interval	
			10	7.602	4.136	.972	-6.26	21.47
Project	Games-Howell	1	2	7.64219	4.57620	.841	-7.9674	23.2518
			3	-3.53238	3.94774	.998	-17.0043	9.9395
			4	-9.29054	3.90887	.406	-22.6190	4.0379
			5	-6.90875	4.35267	.878	-21.8212	8.0037
			6	1.08855	4.55045	1.000	-14.4307	16.6078
			7	-4.19990	5.28574	.999	-22.4404	14.0406
			8	6.82381	5.10531	.955	-10.7100	24.3576
			9	-6.43060	3.75006	.819	-19.2338	6.3726
			10	-2.34790	3.58205	1.000	-14.6503	9.9545
			11	2.93714	5.30052	1.000	-15.3037	21.1780
		2	1	-7.64219	4.57620	.841	-23.2518	7.9674
			3	-11.17457	4.41342	.319	-26.2600	3.9109
			4	-16.9327(*)	4.37868	.016	-31.8982	-1.9673
			5	-14.55093	4.77904	.118	-30.8756	1.7738
			6	-6.55364	4.95985	.960	-23.4245	10.3173
			7	-11.84209	5.64204	.585	-31.1672	7.4830
			8	-.81838	5.47336	1.000	-19.5009	17.8642
			9	-14.07279	4.23751	.064	-28.5962	.4507
			10	-9.99009	4.08958	.371	-24.0921	4.1119
			11	-4.70504	5.65589	.999	-24.0360	14.6259
		3	1	3.53238	3.94774	.998	-9.9395	17.0043
			2	11.17457	4.41342	.319	-3.9109	26.2600
			4	-5.75816	3.71697	.894	-18.4198	6.9035
			5	-3.37637	4.18119	.999	-17.7332	10.9805
			6	4.62093	4.38671	.992	-10.3696	19.6114
			7	-.66752	5.14545	1.000	-18.4971	17.1620
			8	10.35619	4.95992	.593	-6.7389	27.4513
			9	-2.89822	3.54958	.999	-14.9909	9.1945
			10	1.18448	3.37161	1.000	-10.3645	12.7334
			11	6.46952	5.16063	.970	-11.3576	24.2966
		4	1	9.29054	3.90887	.406	-4.0379	22.6190
			2	16.93273(*)	4.37868	.016	1.9673	31.8982
			3	5.75816	3.71697	.894	-6.9035	18.4198
			5	2.38179	4.14450	1.000	-11.8464	16.6100
			6	10.37909	4.35176	.402	-4.4902	25.2484
			7	5.09064	5.11569	.994	-12.6472	22.8285
			8	16.11435	4.92903	.076	-.8824	33.1111
			9	2.85994	3.50630	.999	-9.0610	14.7809
			10	6.94264	3.32601	.593	-4.4210	18.3063
			11	12.22768	5.13096	.407	-5.5071	29.9625
		5	1	6.90875	4.35267	.878	-8.0037	21.8212
			2	14.55093	4.77904	.118	-1.7738	30.8756
			3	3.37637	4.18119	.999	-10.9805	17.7332
			4	-2.38179	4.14450	1.000	-16.6100	11.8464
			6	7.99730	4.75439	.835	-8.2431	24.2377
			7	2.70884	5.46229	1.000	-16.0973	21.5150
			8	13.73256	5.28789	.287	-4.4032	31.8683
			9	.47815	3.99507	1.000	-13.2796	14.2359
			10	4.56084	3.83780	.979	-8.7515	17.8732

Appendix F: Module group statistics

Dependent Variable	(I) Grp	(J) Grp	Mean Diff (I-J)	Std. Error	Sig.	95% Confidence Interval	
		11	9.84589	5.47660	.774	-8.9633	28.6551
	6	1	-1.08855	4.55045	1.000	-16.6078	14.4307
		2	6.55364	4.95985	.960	-10.3173	23.4245
		3	-4.62093	4.38671	.992	-19.6114	10.3696
		4	-10.37909	4.35176	.402	-25.2484	4.4902
		5	-7.99730	4.75439	.835	-24.2377	8.2431
		7	-5.28845	5.62117	.997	-24.5476	13.9707
		8	5.73526	5.45185	.992	-12.8777	24.3483
		9	-7.51915	4.20968	.780	-21.9421	6.9038
		10	-3.43645	4.06074	.999	-17.4339	10.5610
		11	1.84859	5.63507	1.000	-17.4162	21.1134
	7	1	4.19990	5.28574	.999	-14.0406	22.4404
		2	11.84209	5.64204	.585	-7.4830	31.1672
		3	.66752	5.14545	1.000	-17.1620	18.4971
		4	-5.09064	5.11569	.994	-22.8285	12.6472
		5	-2.70884	5.46229	1.000	-21.5150	16.0973
		6	5.28845	5.62117	.997	-13.9707	24.5476
		8	11.02371	6.07907	.765	-9.7488	31.7962
		9	-2.23070	4.99539	1.000	-19.6296	15.1682
		10	1.85200	4.87053	1.000	-15.2250	18.9290
		11	7.13705	6.24392	.985	-14.1920	28.4661
	8	1	-6.82381	5.10531	.955	-24.3576	10.7100
		2	.81838	5.47336	1.000	-17.8642	19.5009
		3	-10.35619	4.95992	.593	-27.4513	6.7389
		4	-16.11435	4.92903	.076	-33.1111	.8824
		5	-13.73256	5.28789	.287	-31.8683	4.4032
		6	-5.73526	5.45185	.992	-24.3483	12.8777
		7	-11.02371	6.07907	.765	-31.7962	9.7488
		9	-13.25441	4.80406	.220	-29.8872	3.3784
		10	-9.17171	4.67410	.674	-25.4581	7.1147
		11	-3.88667	6.09193	1.000	-24.6688	16.8954
	9	1	6.43060	3.75006	.819	-6.3726	19.2338
		2	14.07279	4.23751	.064	-.4507	28.5962
		3	2.89822	3.54958	.999	-9.1945	14.9909
		4	-2.85994	3.50630	.999	-14.7809	9.0610
		5	-.47815	3.99507	1.000	-14.2359	13.2796
		6	7.51915	4.20968	.780	-6.9038	21.9421
		7	2.23070	4.99539	1.000	-15.1682	19.6296
		8	13.25441	4.80406	.220	-3.3784	29.8872
		10	4.08270	3.13783	.963	-6.6065	14.7719
		11	9.36774	5.01103	.731	-8.0251	26.7606
	10	1	2.34790	3.58205	1.000	-9.9545	14.6503
		2	9.99009	4.08958	.371	-4.1119	24.0921
		3	-1.18448	3.37161	1.000	-12.7334	10.3645
		4	-6.94264	3.32601	.593	-18.3063	4.4210
		5	-4.56084	3.83780	.979	-17.8732	8.7515
		6	3.43645	4.06074	.999	-10.5610	17.4339
		7	-1.85200	4.87053	1.000	-18.9290	15.2250
		8	9.17171	4.67410	.674	-7.1147	25.4581
		9	-4.08270	3.13783	.963	-14.7719	6.6065
		11	5.28505	4.88657	.989	-11.7819	22.3520

Appendix F: Module group statistics

Dependent Variable		(I) Grp	(J) Grp	Mean Diff (I-J)	Std. Error	Sig.	95% Confidence Interval	
		11	1	-2.93714	5.30052	1.000	-21.1780	15.3037
			2	4.70504	5.65589	.999	-14.6259	24.0360
			3	-6.46952	5.16063	.970	-24.2966	11.3576
			4	-12.22768	5.13096	.407	-29.9625	5.5071
			5	-9.84589	5.47660	.774	-28.6551	8.9633
			6	-1.84859	5.63507	1.000	-21.1134	17.4162
			7	-7.13705	6.24392	.985	-28.4661	14.1920
			8	3.88667	6.09193	1.000	-16.8954	24.6688
			9	-9.36774	5.01103	.731	-26.7606	8.0251
			10	-5.28505	4.88657	.989	-22.3520	11.7819
IndvTest	Games-Howell	1	2	-11.71140	4.83887	.379	-28.1952	4.7724
			3	-1.04762	5.71909	1.000	-20.6835	18.5883
			4	-6.25685	4.51064	.945	-21.6270	9.1133
			5	-5.19465	5.23575	.995	-23.1827	12.7934
			6	4.59163	5.10447	.998	-12.8150	21.9982
			7	.98254	5.01021	1.000	-16.1588	18.1238
			8	2.85714	4.96901	1.000	-14.1062	19.8205
			9	-8.49137	4.71362	.773	-24.5250	7.5423
			10	9.04921	4.73192	.706	-7.1178	25.2162
			11	-6.98413	5.28783	.959	-25.0732	11.1049
		2	1	11.71140	4.83887	.379	-4.7724	28.1952
			3	10.66378	5.87214	.764	-9.4343	30.7619
			4	5.45455	4.70318	.984	-10.5601	21.4692
			5	6.51675	5.40251	.978	-11.9878	25.0213
			6	16.30303	5.27538	.105	-1.6499	34.2560
			7	12.69394	5.18422	.364	-5.0015	30.3894
			8	14.56854	5.14442	.182	-2.9587	32.0958
			9	3.22003	4.89818	1.000	-13.4245	19.8646
			10	20.76061(*)	4.91580	.006	3.9936	37.5276
			11	4.72727	5.45299	.998	-13.8787	23.3332
		3	1	1.04762	5.71909	1.000	-18.5883	20.6835
			2	-10.66378	5.87214	.764	-30.7619	9.4343
			4	-5.20924	5.60475	.997	-24.4876	14.0691
			5	-4.14703	6.20325	1.000	-25.3788	17.0848
			6	5.63925	6.09286	.997	-15.1617	26.4402
			7	2.03016	6.01410	1.000	-18.5547	22.6150
			8	3.90476	5.97983	1.000	-16.5506	24.3601
			9	-7.44375	5.76936	.965	-27.2107	12.3232
			10	10.09683	5.78432	.802	-9.7552	29.9489
			11	-5.93651	6.24727	.996	-27.2630	15.3900
		4	1	6.25685	4.51064	.945	-9.1133	21.6270
			2	-5.45455	4.70318	.984	-21.4692	10.5601
			3	5.20924	5.60475	.997	-14.0691	24.4876
			5	1.06220	5.11061	1.000	-16.5207	18.6451
			6	10.84848	4.97603	.531	-6.1268	27.8238
			7	7.23939	4.87928	.916	-9.4631	23.9419
			8	9.11400	4.83697	.723	-7.4019	25.6299
			9	-2.23452	4.57421	1.000	-17.7774	13.3083
			10	15.30606	4.59307	.061	-.3797	30.9918
			11	-.72727	5.16394	1.000	-18.4112	16.9566

Appendix F: Module group statistics

Dependent Variable	(I) Grp	(J) Grp	Mean Diff (I-J)	Std. Error	Sig.	95% Confidence Interval	
	5	1	5.19465	5.23575	.995	-12.7934	23.1827
		2	-6.51675	5.40251	.978	-25.0213	11.9878
		3	4.14703	6.20325	1.000	-17.0848	25.3788
		4	-1.06220	5.11061	1.000	-18.6451	16.5207
		6	9.78628	5.64163	.809	-9.5012	29.0738
		7	6.17719	5.55649	.988	-12.8761	25.2305
		8	8.05180	5.51937	.924	-10.8542	26.9577
		9	-3.29672	5.29061	1.000	-21.4279	14.8345
		10	14.24386	5.30692	.247	-3.9897	32.4774
		11	-1.78947	5.80805	1.000	-21.6615	18.0826
	6	1	-4.59163	5.10447	.998	-21.9982	12.8150
		2	-16.30303	5.27538	.105	-34.2560	1.6499
		3	-5.63925	6.09286	.997	-26.4402	15.1617
		4	-10.84848	4.97603	.531	-27.8238	6.1268
		5	-9.78628	5.64163	.809	-29.0738	9.5012
		7	-3.60909	5.43297	1.000	-22.1403	14.9221
		8	-1.73449	5.39500	1.000	-20.1106	16.6416
		9	-13.08300	5.16073	.315	-30.6411	4.4751
		10	4.45758	5.17745	.998	-13.2095	22.1246
		11	-11.57576	5.68999	.627	-30.9636	7.8121
	7	1	-.98254	5.01021	1.000	-18.1238	16.1588
		2	-12.69394	5.18422	.364	-30.3894	5.0015
		3	-2.03016	6.01410	1.000	-22.6150	18.5547
		4	-7.23939	4.87928	.916	-23.9419	9.4631
		5	-6.17719	5.55649	.988	-25.2305	12.8761
		6	3.60909	5.43297	1.000	-14.9221	22.1403
		8	1.87460	5.30589	1.000	-16.2514	20.0006
		9	-9.47391	5.06751	.732	-26.7676	7.8198
		10	8.06667	5.08454	.878	-9.3401	25.4734
		11	-7.96667	5.60558	.935	-27.1194	11.1861
	8	1	-2.85714	4.96901	1.000	-19.8205	14.1062
		2	-14.56854	5.14442	.182	-32.0958	2.9587
		3	-3.90476	5.97983	1.000	-24.3601	16.5506
		4	-9.11400	4.83697	.723	-25.6299	7.4019
		5	-8.05180	5.51937	.924	-26.9577	10.8542
		6	1.73449	5.39500	1.000	-16.6416	20.1106
		7	-1.87460	5.30589	1.000	-20.0006	16.2514
		9	-11.34852	5.02678	.480	-28.4669	5.7699
		10	6.19206	5.04395	.975	-11.0417	23.4258
		11	-9.84127	5.56879	.791	-28.8476	9.1651
	9	1	8.49137	4.71362	.773	-7.5423	24.5250
		2	-3.22003	4.89818	1.000	-19.8646	13.4245
		3	7.44375	5.76936	.965	-12.3232	27.2107
		4	2.23452	4.57421	1.000	-13.3083	17.7774
		5	3.29672	5.29061	1.000	-14.8345	21.4279
		6	13.08300	5.16073	.315	-4.4751	30.6411
		7	9.47391	5.06751	.732	-7.8198	26.7676
		8	11.34852	5.02678	.480	-5.7699	28.4669
		10	17.54058(*)	4.79255	.026	1.2110	33.8702
		11	1.50725	5.34215	1.000	-16.7260	19.7405
	10	1	-9.04921	4.73192	.706	-25.2162	7.1178

Dependent Variable	(I) Grp	(J) Grp	Mean Diff (I-J)	Std. Error	Sig.	95% Confidence Interval	
		2	-20.76061(*)	4.91580	.006	-37.5276	-3.9936
		3	-10.09683	5.78432	.802	-29.9489	9.7552
		4	-15.30606	4.59307	.061	-30.9918	.3797
		5	-14.24386	5.30692	.247	-32.4774	3.9897
		6	-4.45758	5.17745	.998	-22.1246	13.2095
		7	-8.06667	5.08454	.878	-25.4734	9.3401
		8	-6.19206	5.04395	.975	-23.4258	11.0417
		9	-17.54058(*)	4.79255	.026	-33.8702	-1.2110
		11	-16.03333	5.35831	.133	-34.3673	2.3007
	11	1	6.98413	5.28783	.959	-11.1049	25.0732
		2	-4.72727	5.45299	.998	-23.3332	13.8787
		3	5.93651	6.24727	.996	-15.3900	27.2630
		4	.72727	5.16394	1.000	-16.9566	18.4112
		5	1.78947	5.80805	1.000	-18.0826	21.6615
		6	11.57576	5.68999	.627	-7.8121	30.9636
		7	7.96667	5.60558	.935	-11.1861	27.1194
		8	9.84127	5.56879	.791	-9.1651	28.8476
		9	-1.50725	5.34215	1.000	-19.7405	16.7260
		10	16.03333	5.35831	.133	-2.3007	34.3673

Table FI.6: Case 2005-S1: Homogeneous Subsets for MODULE SCORE

	Group	N	Subset for alpha = .05
			1
Gabriel(a,b)	10	20	62.35
	8	21	62.81
	6	22	63.45
	1	21	66.48
	7	20	67.25
	3	21	68.62
	11	21	69.95
	2	22	71.77
	5	19	72.16
	4	22	73.18
	9	23	73.65
	Sig.		.280
Hochberg(a,b)	10	20	62.35
	8	21	62.81
	6	22	63.45
	1	21	66.48
	7	20	67.25
	3	21	68.62
	11	21	69.95
	2	22	71.77
	5	19	72.16
	4	22	73.18
	9	23	73.65
	Sig.		.280

Means for groups in homogeneous subsets are displayed.
a Uses Harmonic Mean Sample Size = 21.035.
b The group sizes are unequal. The harmonic mean of the group sizes is used. Type I error levels are not

guaranteed.

Table FI.7: Case 2005-S1: Homogeneous Subsets for PROJECT

	Group	N	Subset for alpha = .05	
			1	2
Gabriel(a,b)	2	22	54.6559	
	8	21	55.4743	
	11	21	59.3610	59.3610
	6	22	61.2095	61.2095
	1	21	62.2981	62.2981
	10	20	64.6460	64.6460
	3	21	65.8305	65.8305
	7	20	66.4980	66.4980
	9	23	68.7287	68.7287
	5	19	69.2068	69.2068
	4	22		71.5886
	Sig.		.120	.425
Hochberg(a,b)	2	22	54.6559	
	8	21	55.4743	
	11	21	59.3610	59.3610
	6	22	61.2095	61.2095
	1	21	62.2981	62.2981
	10	20	64.6460	64.6460
	3	21	65.8305	65.8305
	7	20	66.4980	66.4980
	9	23	68.7287	68.7287
	5	19	69.2068	69.2068
	4	22		71.5886
	Sig.		.120	.425

Means for groups in homogeneous subsets are displayed.
a Uses Harmonic Mean Sample Size = 21.035.
b The group sizes are unequal. The harmonic mean of the group sizes is used. Type I error levels are not guaranteed.

Table FI.8: Case 2005-S1: Homogeneous Subsets for INDIVIDUAL TEST

	Group	N	Subset for alpha = .05	
			1	2
Gabriel(a,b)	10	20	58.6333	
	6	22	63.0909	63.0909
	8	21	64.8254	64.8254
	7	20	66.7000	66.7000
	1	21	67.6825	67.6825
	3	21	68.7302	68.7302
	5	19	72.8772	72.8772
	4	22	73.9394	73.9394
	11	21	74.6667	74.6667
	9	23	76.1739	76.1739
	2	22		79.3939
	Sig.		.058	.120
Hochberg(a,b)	10	20	58.6333	
	6	22	63.0909	63.0909
	8	21	64.8254	64.8254
	7	20	66.7000	66.7000

	1	21	67.6825	67.6825
	3	21	68.7302	68.7302
	5	19	72.8772	72.8772
	4	22	73.9394	73.9394
	11	21	74.6667	74.6667
	9	23	76.1739	76.1739
	2	22		79.3939
	Sig.		.058	.120

Means for groups in homogeneous subsets are displayed.
a Uses Harmonic Mean Sample Size = 21.035.
b The group sizes are unequal. The harmonic mean of the group sizes is used. Type I error levels are not guaranteed.

Case 2005-S1: Frequency Tables of Independent variables used in ANCOVA (univariate Analysis of Covariance)

Table FI.9: Gender

TUTGRP			Frequency	Valid Percent	Cumulative Percent
EI0501	Valid	F	9	42.9	42.9
		M	12	57.1	100.0
		Total	21	100.0	
EI0502	Valid	F	12	54.5	54.5
		M	10	45.5	100.0
		Total	22	100.0	
EI0503	Valid	F	6	28.6	28.6
		M	15	71.4	100.0
		Total	21	100.0	
EI0504	Valid	F	9	40.9	40.9
		M	13	59.1	100.0
		Total	22	100.0	
EI0505	Valid	F	8	42.1	42.1
		M	11	57.9	100.0
		Total	19	100.0	
EI0506	Valid	F	8	36.4	36.4
		M	14	63.6	100.0
		Total	22	100.0	
EI0507	Valid	F	1	5.0	5.0
		M	19	95.0	100.0
		Total	20	100.0	
EI0508	Valid	F	5	23.8	23.8
		M	16	76.2	100.0
		Total	21	100.0	
EI0509	Valid	F	15	65.2	65.2
		M	8	34.8	100.0
		Total	23	100.0	
EI0510	Valid	F	11	55.0	55.0
		M	9	45.0	100.0
		Total	20	100.0	
EI0511	Valid	F	6	28.6	28.6
		M	15	71.4	100.0
		Total	21	100.0	

Table FI.10: **OLAGG (entry level aggregate)**

TUTGRP			Frequency	Valid Percent	Cumulative Percent
EI0501	Valid	13	1	4.8	4.8
		18	2	9.5	14.3
		19	1	4.8	19.0
		21	4	19.0	38.1
		22	2	9.5	47.6

Appendix F: Module group statistics

TUTGRP			Frequency	Valid Percent	Cumulative Percent
		23	4	19.0	66.7
		24	4	19.0	85.7
		25	1	4.8	90.5
		26	1	4.8	95.2
		27	1	4.8	100.0
		Total	21	100.0	Mean: 22.00
EI0502	Valid	10	2	9.1	9.1
		20	1	4.5	13.6
		21	3	13.6	27.3
		22	3	13.6	40.9
		23	3	13.6	54.5
		24	4	18.2	72.7
		25	2	9.1	81.8
		26	4	18.2	100.0
		Total	22	100.0	Mean: 22.18
EI0503	Valid	10	2	9.5	9.5
		14	1	4.8	14.3
		17	1	4.8	19.0
		20	2	9.5	28.6
		21	1	4.8	33.3
		22	1	4.8	38.1
		23	5	23.8	61.9
		24	3	14.3	76.2
		25	4	19.0	95.2
		26	1	4.8	100.0
		Total	21	100.0	Mean: 21.29
EI0504	Valid	10	5	22.7	22.7
		18	1	4.5	27.3
		19	1	4.5	31.8
		21	2	9.1	40.9
		22	4	18.2	59.1
		23	2	9.1	68.2
		24	3	13.6	81.8
		25	2	9.1	90.9
		26	1	4.5	95.5
		27	1	4.5	100.0
		Total	22	100.0	Mean: 19.91
EI0505	Valid	10	5	26.3	26.3
		18	1	5.3	31.6
		20	1	5.3	36.8
		23	1	5.3	42.1
		24	3	15.8	57.9
		25	2	10.5	68.4
		26	5	26.3	94.7
		27	1	5.3	100.0
		Total	19	100.0	Mean: 20.53
EI0506	Valid	10	1	4.5	4.5
		16	1	4.5	9.1
		17	1	4.5	13.6
		20	1	4.5	18.2
		22	5	22.7	40.9
		23	2	9.1	50.0
		24	4	18.2	68.2
		25	3	13.6	81.8
		26	2	9.1	90.9
		27	2	9.1	100.0

Appendix F: Module group statistics

TUTGRP			Frequency	Valid Percent	Cumulative Percent
		Total	22	100.0	Mean: 22.55
EI0507	Valid	15	1	5.0	5.0
		17	1	5.0	10.0
		18	3	15.0	25.0
		19	2	10.0	35.0
		21	1	5.0	40.0
		22	3	15.0	55.0
		23	2	10.0	65.0
		24	2	10.0	75.0
		25	1	5.0	80.0
		26	2	10.0	90.0
		27	1	5.0	95.0
		28	1	5.0	100.0
		Total	20	100.0	Mean: 21.85
EI0508	Valid	10	1	4.8	4.8
		18	2	9.5	14.3
		19	2	9.5	23.8
		21	2	9.5	33.3
		22	4	19.0	52.4
		23	6	28.6	81.0
		24	1	4.8	85.7
		25	1	4.8	90.5
		26	1	4.8	95.2
		27	1	4.8	100.0
		Total	21	100.0	Mean: 21.62
EI0509	Valid	10	7	30.4	30.4
		19	1	4.3	34.8
		21	1	4.3	39.1
		22	1	4.3	43.5
		23	4	17.4	60.9
		24	3	13.0	73.9
		25	1	4.3	78.3
		26	2	8.7	87.0
		27	3	13.0	100.0
		Total	23	100.0	Mean: 19.74
EI0510	Valid	10	7	35.0	35.0
		11	1	5.0	40.0
		19	1	5.0	45.0
		22	1	5.0	50.0
		23	5	25.0	75.0
		24	2	10.0	85.0
		25	1	5.0	90.0
		26	2	10.0	100.0
		Total	20	100.0	Mean: 18.10
EI0511	Valid	10	1	4.8	4.8
		16	2	9.5	14.3
		18	1	4.8	19.0
		19	1	4.8	23.8
		20	3	14.3	38.1
		21	1	4.8	42.9
		22	1	4.8	47.6
		23	1	4.8	52.4
		24	3	14.3	66.7
		25	3	14.3	81.0
		26	3	14.3	95.2
		27	1	4.8	100.0
		Total	21	100.0	Mean: 21.76

Appendix F: Module group statistics

TUTGRP	Frequency	Valid Percent	Cumulative Percent
Note: Students with prior computing has an entry level of 10 points			

Table FI.11: **Tests of Between-Subjects Effects – Univariate ANOVA (OLAGG = entry level aggregate)**
Dependent Variable: MODULE_SCORE

Source	Type III Sum of Squares	df	Mean Square	F	Sig.	Partial Eta Sq.	Noncent. Paramtr	Power (a)
Corrected Model	8181.645(b)	11	743.786	4.773	.000	.193	52.502	1.000
Intercept	92265.222	1	92265.222	592.071	.000	.729	592.071	1.000
OLAGG	4443.355	1	4443.355	28.513	.000	.115	28.513	1.000
Group	3782.498	10	378.250	2.427	.009	.099	24.272	.939
Error	34283.661	220	155.835					
Total	1127645.000	232						
Corrected Total	42465.306	231						

a Observed power computed using alpha = .05
b R Squared = .193 (Adjusted R Squared = .152)

Table FI.12: **Parameter Estimates**
Dependent Variable: MODULE_SCORE

Parameter	B	Std. Error	t	Sig.	95% Confidence Interval		Partial Eta Sq.	Noncent. Paramtr	Power (a)
					Lower Bound	Upper Bound			
Intercept	89.141	4.509	19.768	.000	80.254	98.028	.640	19.768	1.000
OLAGG	-.882	.165	-5.340	.000	-1.207	-.556	.115	5.340	1.000
[Group=1]	-3.266	3.853	-.848	.397	-10.859	4.327	.003	.848	.135
[Group=2]	2.191	3.809	.575	.566	-5.316	9.698	.002	.575	.088
[Group=3]	-1.753	3.853	-.455	.650	-9.347	5.841	.001	.455	.074
[Group=4]	1.596	3.821	.418	.677	-5.934	9.126	.001	.418	.070
[Group=5]	1.116	3.958	.282	.778	-6.684	8.916	.000	.282	.059
[Group=6]	-5.807	3.811	-1.524	.129	-13.317	1.703	.010	1.524	.329
[Group=7]	-2.625	3.900	-.673	.502	-10.312	5.062	.002	.673	.103
[Group=8]	-7.269	3.853	-1.887	.061	-14.861	.324	.016	1.887	.468
[Group=9]	1.916	3.783	.507	.613	-5.538	9.371	.001	.507	.080
[Group=10]	-10.831	3.947	-2.744	.007	-18.610	-3.053	.033	2.744	.780
[Group=11]	0(b)	.	.	.	.	.	.	.	.

a Observed power computed using alpha = .05
b This parameter is set to zero because it is redundant.

Table FI.13: Case 2005-S1 Estimated Marginal Means
Dependent Variable: MODULE_SCORE

Group	Mean	Std. Error	95% Confidence Interval	
			Lower Bound	Upper Bound
1	67.309(a)	2.729	61.931	72.686
2	72.765(a)	2.668	67.507	78.023
3	68.822(a)	2.724	63.452	74.191
4	72.171(a)	2.668	66.912	77.429
5	71.691(a)	2.865	66.044	77.338
6	64.768(a)	2.673	59.500	70.035
7	67.950(a)	2.794	62.443	73.457
8	63.306(a)	2.726	57.934	68.678
9	72.491(a)	2.612	67.343	77.639
10	59.744(a)	2.834	54.159	65.328
11	70.575(a)	2.727	65.201	75.948

a Covariates appearing in the model are evaluated at the following values: OLAGG = 21.06.

Table FI.14: **Tests of Between-Subjects Effects – Univariate ANOVA (Gender)**
Dependent Variable: MODULE_SCORE

Source	Type III Sum of Squares	Df	Mean Square	F	Sig.	Partial Eta Sq	Noncent. Paramtr	Power (a)
Corrected Model	4556.044(b)	11	414.186	2.404	.008	.107	26.440	.951
Intercept	66374.507	1	66374.507	385.193	.000	.636	385.193	1.000
Gender	817.753	1	817.753	4.746	.030	.021	4.746	.583
Group	4151.387	10	415.139	2.409	.010	.099	24.092	.937
Error	37909.262	220	172.315					
Total	1127645.000	232						
Corrected Total	42465.306	231						

a Observed power computed using alpha = .05
b R Squared = .107 (Adjusted R Squared = .063)

Table FI.15: **Parameter Estimates**
Dependent Variable: MODULE_SCORE

Parameter	B	Std. Error	t	Sig.	95% Confidence Interval		Partial Eta Sq	Noncent. Paramtr	Power (a)
					Lower Bound	Upper Bound			
Intercept	62.959	4.302	14.633	.000	54.480	71.438	.493	14.633	1.000
Gender	4.080	1.873	2.178	.030	.389	7.770	.021	2.178	.583
[Group=1]	-2.893	4.060	-.713	.477	-10.895	5.108	.002	.713	.109
[Group=2]	2.880	4.034	.714	.476	-5.071	10.831	.002	.714	.110
[Group=3]	-1.333	4.051	-.329	.742	-9.317	6.650	.000	.329	.062
[Group=4]	3.733	4.011	.931	.353	-4.173	11.638	.004	.931	.153
[Group=5]	2.758	4.164	.662	.509	-5.449	10.964	.002	.662	.101
[Group=6]	-6.180	4.007	-1.542	.124	-14.078	1.718	.011	1.542	.336
[Group=7]	-3.664	4.125	-.888	.375	-11.794	4.466	.004	.888	.143
[Group=8]	-7.337	4.052	-1.811	.072	-15.323	.649	.015	1.811	.438
[Group=9]	5.195	4.021	1.292	.198	-2.730	13.119	.008	1.292	.251
[Group=10]	-6.524	4.131	-1.579	.116	-14.666	1.617	.011	1.579	.349
[Group=11]	0(b)	.	.	.	.	.	.	.	.

a Observed power computed using alpha = .05
b This parameter is set to zero because it is redundant.

Table FI.16: Case 2005-S1 Estimated Marginal Means
Dependent Variable: MODULE_SCORE

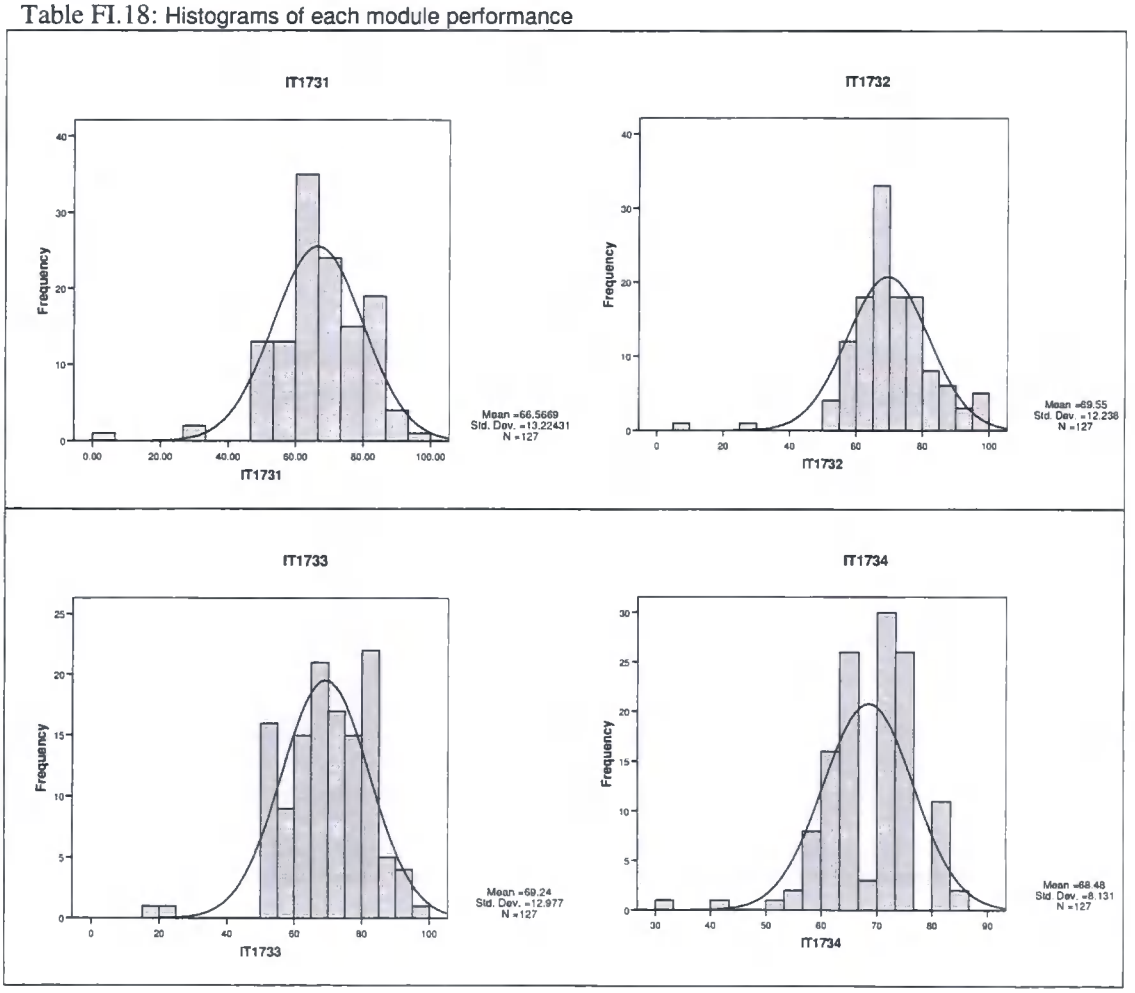
Group	Mean	Std. Error	95% Confidence Interval	
			Lower Bound	Upper Bound
1	66.642(a)	2.866	60.995	72.289
2	72.415(a)	2.814	66.869	77.962
3	68.202(a)	2.871	62.544	73.860
4	73.268(a)	2.799	67.752	78.784
5	72.293(a)	3.012	66.357	78.229
6	63.355(a)	2.799	57.839	68.872
7	65.871(a)	3.003	59.954	71.789
8	62.198(a)	2.878	56.526	67.871
9	74.730(a)	2.782	69.248	80.212
10	63.011(a)	2.951	57.195	68.827
11	69.535(a)	2.871	63.877	75.193

a Covariates appearing in the model are evaluated at the following values: Gender = 1.61 (males)

Table FI.17: Case 2005-S1: Study Programme A (groups 1 to 6)

Case 2005-S1-A	IT1731	IT1732	IT1733	IT1734	IT1735	IT1736
	Computing Maths 1	Electronic Resource Planning	Principles of Computing	Business Information Systems	Creativity	Semestral Project1
N	127	127	127	127	127	127
Mean	66.5669	69.55	69.24	68.48	69.49	Nominal
Std. Error of Mean	1.17347	1.086	1.152	.721	.630	nil
Median	66.0000	68.00	70.00	70.00	68.00	nil
Mode	60.00	66	51	75	68	nil
Std. Deviation	13.22431	12.236	12.977	8.131	7.101	nil
Variance	174.882	149.710	168.404	66.109	50.426	nil
Skewness	-1.016	-.883	-.726	-.910	-.146	nil
Std. Err of Skewness	.215	.215	.215	.215	.215	nil
Kurtosis	4.399	5.028	1.615	2.834	1.497	nil
Std. Err of Kurtosis	.427	.427	.427	.427	.427	nil
Range	95.00	87	77	53	46	nil
Minimum	.00	9	19	32	41	nil
Maximum	95.00	96	96	85	87	nil

Table FI.18: Histograms of each module performance



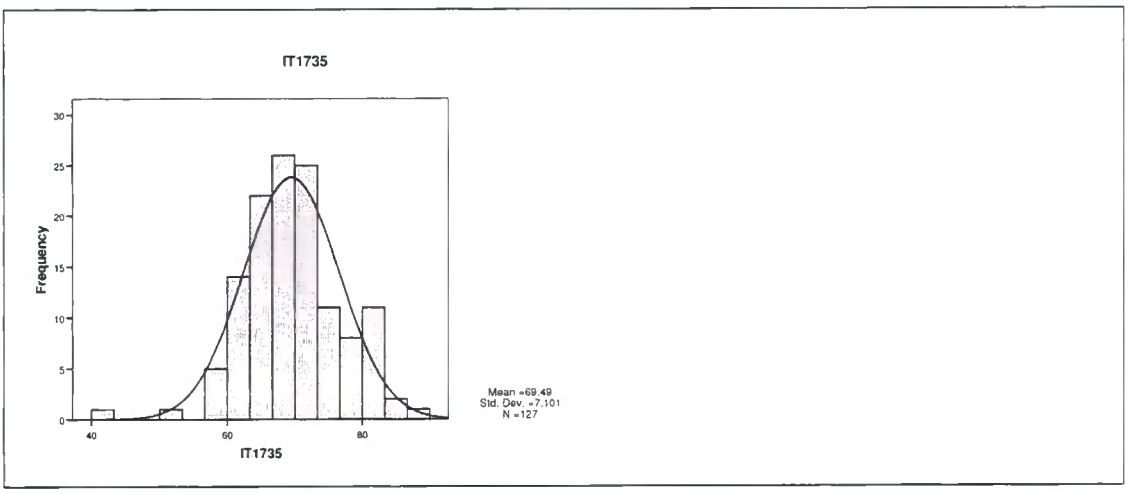


Table FI.19: Case 2005-S1-A Detailed module performance for groups: 01, 02, 03, 04, 05, 06

TUTGRP		IT1731	IT1732	IT1733	IT1734	IT1735
EI0501	N	21	21	21	21	21
	Mean	71.0000	76.14	66.48	68.38	71.95
	Std. Error of Mean	2.95603	3.141	2.550	2.322	2.356
	Median	73.0000	76.00	67.00	70.00	71.00
	Mode	70.00	78	63(a)	70(a)	68(a)
	Std. Deviation	13.54622	14.392	11.686	10.642	10.796
	Variance	183.500	207.129	136.562	113.248	116.548
	Skewness	-1.223	-2.076	-2.879	-2.058	-.948
	Std. Error of Skewness	.501	.501	.501	.501	.501
	Kurtosis	2.171	7.947	10.962	6.303	1.914
	Std. Error of Kurtosis	.972	.972	.972	.972	.972
	Range	59.00	71	58	49	46
	Minimum	32.00	25	22	32	41
	Maximum	91.00	96	80	81	87
EI0502	N	22	22	22	22	22
	Mean	63.7273	70.77	71.77	72.00	69.18
	Std. Error of Mean	3.74465	2.607	2.648	1.802	1.046
	Median	68.0000	68.00	72.00	74.50	71.00
	Mode	53.00	68	51(a)	75	71
	Std. Deviation	17.56398	12.228	12.421	8.452	4.905
	Variance	308.494	149.517	154.279	71.429	24.061
	Skewness	-2.364	.389	-.030	-.763	-.845
	Std. Error of Skewness	.491	.491	.491	.491	.491
	Kurtosis	7.774	-1.065	-.319	.305	.085
	Std. Error of Kurtosis	.953	.953	.953	.953	.953
	Range	80.00	43	45	33	18
	Minimum	.00	52	51	52	57
	Maximum	80.00	95	96	85	75
EI0503	N	21	21	21	21	21
	Mean	62.8095	67.71	68.62	67.24	65.86
	Std. Error of Mean	2.00312	1.329	2.853	1.627	1.003
	Median	60.0000	67.00	70.00	66.00	67.00
	Mode	60.00	66	51	61(a)	68(a)
	Std. Deviation	9.17943	6.092	13.075	7.456	4.597
	Variance	84.262	37.114	170.948	55.590	21.129

Appendix F: Module group statistics

TUTGRP		IT1731	IT1732	IT1733	IT1734	IT1735
	Skewness	.754	-.673	-.122	-.056	-1.527
	Std. Error of Skewness	.501	.501	.501	.501	.501
	Kurtosis	-.406	.966	-1.370	-1.357	3.104
	Std. Error of Kurtosis	.972	.972	.972	.972	.972
	Range	31.00	26	40	25	19
	Minimum	50.00	52	51	55	52
	Maximum	81.00	78	91	80	71
EI0504	N	22	22	22	22	22
	Mean	70.7273	66.91	73.18	69.18	70.91
	Std. Error of Mean	2.21680	1.419	2.384	1.465	1.471
	Median	69.0000	66.50	75.00	70.00	71.50
	Mode	56.00(a)	64	61(a)	70	63(a)
	Std. Deviation	10.39772	6.654	11.181	6.870	6.900
	Variance	108.113	44.277	125.013	47.203	47.610
	Skewness	.715	.886	-.109	.507	-.131
	Std. Error of Skewness	.491	.491	.491	.491	.491
	Kurtosis	.112	1.616	-1.276	.152	-.578
	Std. Error of Kurtosis	.953	.953	.953	.953	.953
	Range	39.00	30	36	27	26
	Minimum	56.00	55	55	58	57
	Maximum	95.00	85	91	85	83
	N 05	19	19	19	19	19
	Mean	63.7368	64.05	72.16	67.89	70.63
	Std. Error of Mean	2.73042	2.524	3.011	1.622	1.556
	Median	61.0000	65.00	76.00	66.00	68.00
	Mode	50.00	65	60(a)	75	68
	Std. Deviation	11.90164	11.002	13.124	7.070	6.784
	Variance	141.649	121.053	172.251	49.988	46.023
	Skewness	.963	1.456	-.424	.040	-.158
	Std. Error of Skewness	.524	.524	.524	.524	.524
	Kurtosis	.600	2.599	-1.205	-1.312	-.683
	Std. Error of Kurtosis	1.014	1.014	1.014	1.014	1.014
	Range	43.00	45	40	23	24
	Minimum	50.00	50	51	57	57
	Maximum	93.00	95	91	80	81
EI0506	N	22	22	22	22	22
	Mean	67.0455	71.18	63.45	66.05	68.50
	Std. Error of Mean	2.92657	3.582	3.139	1.549	1.291
	Median	66.0000	71.50	64.50	65.50	67.00
	Mode	63.00	77	68	65	67
	Std. Deviation	13.72685	16.800	14.725	7.267	6.053
	Variance	188.426	282.251	216.831	52.807	36.643
	Skewness	-.654	-2.354	-1.011	-1.904	1.162
	Std. Error of Skewness	.491	.491	.491	.491	.491
	Kurtosis	1.605	8.900	2.843	6.163	1.203
	Std. Error of Kurtosis	.953	.953	.953	.953	.953
	Range	61.00	86	67	34	24
	Minimum	29.00	9	19	41	59
	Maximum	90.00	95	86	75	83

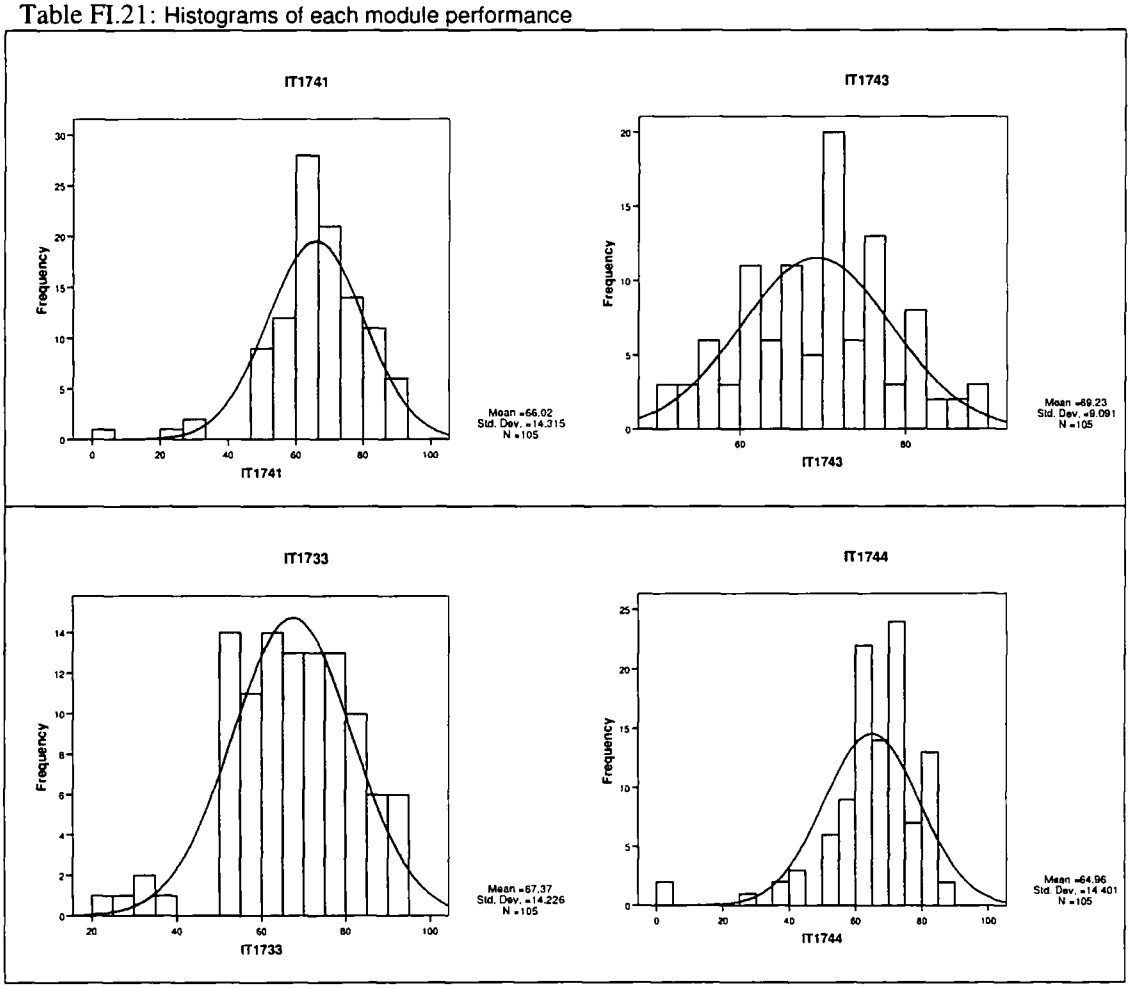
a Multiple modes exist. The smallest value is shown

Table FI.20: Case 2005-S1: Study Programme B (groups 7 to 11)

Case 2005-S1-B	IT1741	IT1743	IT1733	IT1744	IT1745	IT1746
	Computing Maths 1	Manu-facturing	Principles of Computing	Internet Computing	Communic ation Skills	Semestral Project2
N	105	105	105	105	105	105
Mean	66.02	69.23	67.37	64.96	68.72	Nominal
Std. Error of Mean	1.397	.887	1.388	1.405	.934	nil
Median	66.00	70.00	68.00	67.00	71.00	nil
Mode	60	71	51(a)	70	65(a)	nil
Std. Deviation	14.315	9.091	14.226	14.401	9.567	nil
Variance	204.923	82.640	202.370	207.383	91.529	nil
Skewness	-1.176	.008	-.493	-2.030	-3.587	nil
Std. Err. of Skewness	.236	.236	.236	.236	.236	nil
Kurtosis	4.013	-.539	.602	6.853	24.688	nil
Std. Error of Kurtosis	.467	.467	.467	.467	.467	nil
Range	93	39	73	89	85	nil
Minimum	0	51	22	0	0	nil
Maximum	93	90	95	89	85	nil

a Multiple modes exist. The smallest value is shown

Table FI.21: Histograms of each module performance



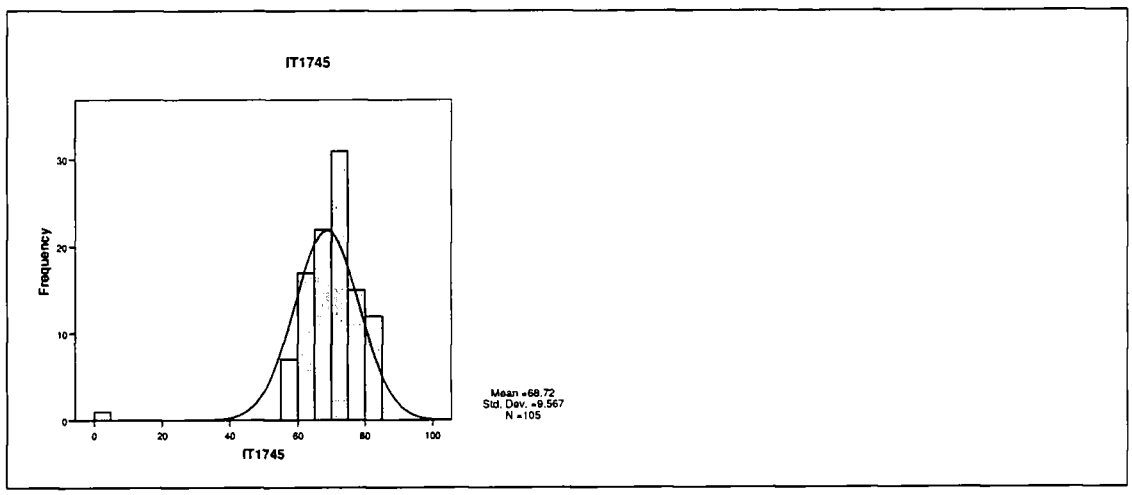


Table FI.22: Case 2005-S1-B Detailed module performance for groups: 07, 08, 09, 10, 11

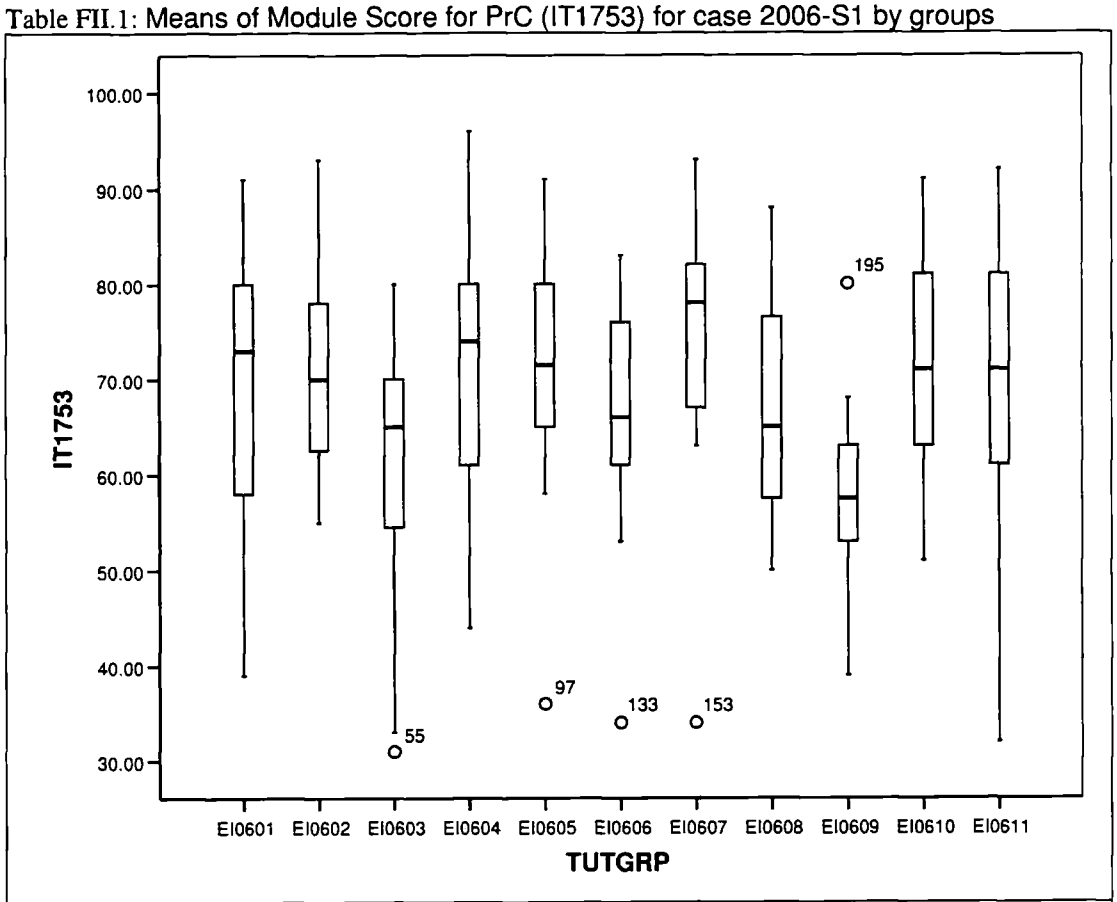
TUTGRP		IT1741	IT1743	IT1733	IT1744	IT1745
EI0507	N	20	20	20	20	20
	Mean	66.30	70.30	67.25	64.60	71.30
	Std. Error of Mean	4.844	2.025	3.441	4.184	1.348
	Median	69.50	71.00	64.50	67.50	73.50
	Mode	60	70(a)	57(a)	72(a)	75
	Std. Deviation	21.663	9.056	15.389	18.709	6.027
	Variance	469.274	82.011	236.829	350.042	36.326
	Skewness	-1.656	-.402	-.340	-2.337	-.493
	Std. Error of Skewness	.512	.512	.512	.512	.512
	Kurtosis	3.893	.956	1.024	7.185	-.984
	Std. Error of Kurtosis	.992	.992	.992	.992	.992
	Range	93	36	65	82	19
	Minimum	0	52	28	0	61
	Maximum	93	88	93	82	80
EI0508	N	21	21	21	21	21
	Mean	62.71	68.10	62.81	62.05	69.62
	Std. Error of Mean	3.406	2.255	3.073	2.731	1.538
	Median	62.00	70.00	67.00	63.00	71.00
	Mode	52(a)	60(a)	58(a)	35(a)	65
	Std. Deviation	15.608	10.334	14.081	12.516	7.046
	Variance	243.614	106.790	198.262	156.648	49.648
	Skewness	-.832	.114	-1.601	-1.016	.124
	Std. Error of Skewness	.501	.501	.501	.501	.501
	Kurtosis	1.097	-.651	2.822	.402	.080
	Std. Error of Kurtosis	.972	.972	.972	.972	.972
	Range	59	38	57	43	30
	Minimum	26	52	22	35	55
	Maximum	85	90	79	78	85
EI0509	N	23	23	23	23	23
	Mean	68.22	71.61	73.65	67.91	70.83
	Std. Error of Mean	2.318	1.418	2.492	1.682	1.481
	Median	71.00	71.00	76.00	68.00	71.00
	Mode	60	66(a)	63(a)	70	80
	Std. Deviation	11.115	6.801	11.949	8.067	7.101

Appendix F: Module group statistics

TUTGRP		IT1741	IT1743	IT1733	IT1744	IT1745
	Variance	123.542	46.249	142.783	65.083	50.423
	Skewness	.058	-.185	-1.248	-.060	-.189
	Std. Error of Skewness	.481	.481	.481	.481	.481
	Kurtosis	-.815	-.906	2.671	.138	-.623
	Std. Error of Kurtosis	.935	.935	.935	.935	.935
	Range	40	24	55	33	25
	Minimum	50	58	37	50	55
	Maximum	90	82	92	83	80
EI0510	N	20	20	20	20	20
	Mean	66.55	64.30	62.35	63.85	63.25
	Std. Error of Mean	2.588	1.433	2.332	3.863	3.616
	Median	66.50	64.00	61.50	66.00	65.50
	Mode	50(a)	61(a)	51	58(a)	66
	Std. Deviation	11.573	6.408	10.429	17.276	16.173
	Variance	133.945	41.063	108.766	298.450	261.566
	Skewness	.168	.332	.785	-2.770	-3.368
	Std. Error of Skewness	.512	.512	.512	.512	.512
	Kurtosis	-.712	-.645	-.217	10.249	13.506
	Std. Error of Kurtosis	.992	.992	.992	.992	.992
	Range	38	22	35	85	80
	Minimum	50	55	51	0	0
	Maximum	88	77	86	85	80
EI0511	N	21	21	21	21	21
	Mean	66.14	71.43	69.95	66.05	68.29
	Std. Error of Mean	2.134	2.358	3.581	3.181	1.466
	Median	66.00	72.00	72.00	66.00	70.00
	Mode	60	60(a)	53	66(a)	66(a)
	Std. Deviation	9.779	10.805	16.409	14.579	6.717
	Variance	95.629	116.757	269.248	212.548	45.114
	Skewness	.334	-.172	-.381	-.875	-.696
	Std. Error of Skewness	.501	.501	.501	.501	.501
	Kurtosis	-1.104	-1.030	-.382	1.166	.234
	Std. Error of Kurtosis	.972	.972	.972	.972	.972
	Range	32	37	62	62	25
	Minimum	52	51	33	27	55
	Maximum	84	88	95	89	80

a Multiple modes exist. The smallest value is shown

Table FII.1: Means of Module Score for PrC (IT1753) for case 2006-S1 by groups



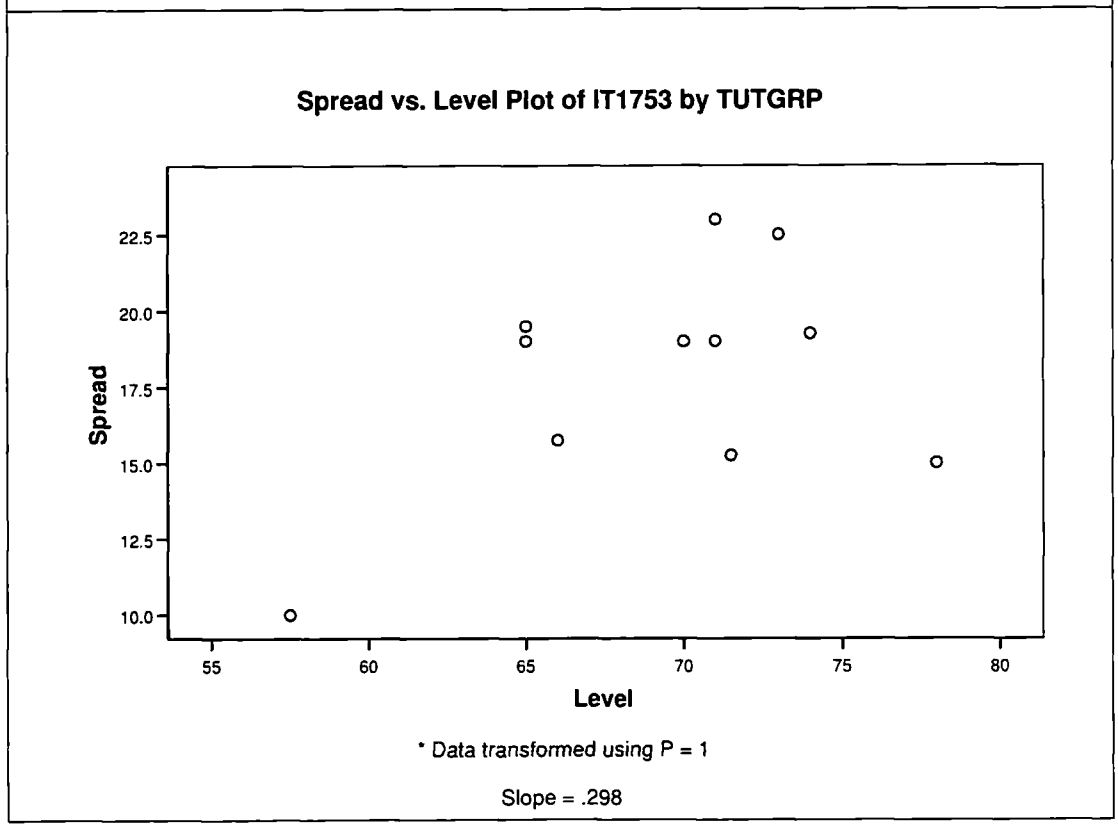


Table FII.2: Case 2006-S1 Descriptives

	Group	N	Mean	Std. Deviation	Std. Error	95% Confidence Interval for Mean		Minimum	Maximum
						Lower Bound	Upper Bound		
MODULE SCORE	1	21	68.33	13.566	2.960	62.16	74.51	39	91
	2	23	71.26	11.522	2.402	66.28	76.24	55	93
	3	23	61.70	12.517	2.610	56.28	67.11	31	80
	4	22	71.77	13.543	2.887	65.77	77.78	44	96
	5	22	71.45	11.883	2.534	66.19	76.72	36	91
	6	22	66.00	10.614	2.263	61.29	70.71	34	83
	7	24	75.38	12.328	2.516	70.17	80.58	34	93
	8	24	66.92	12.007	2.451	61.85	71.99	50	88
	9	22	58.45	8.081	1.723	54.87	62.04	39	80
	10	23	71.70	10.927	2.278	66.97	76.42	51	91
	11	21	69.67	14.867	3.244	62.90	76.43	32	92
	Total	247	68.46	12.743	.811	66.86	70.05	31	96
Project	1	21	67.4762	15.77219	3.44177	60.2968	74.6556	40.00	98.00
	2	23	70.6087	13.68057	2.85260	64.6928	76.5246	50.00	98.50
	3	23	66.6957	7.94547	1.65674	63.2598	70.1315	50.00	80.00
	4	22	72.2500	17.21693	3.67066	64.6164	79.8836	24.00	95.00
	5	22	65.2727	13.97594	2.97968	59.0761	71.4693	40.00	90.00
	6	22	59.5682	11.64688	2.48312	54.4042	64.7321	43.00	84.00
	7	24	74.0208	10.84624	2.21398	69.4409	78.6008	50.00	95.50
	8	24	66.4792	9.88574	2.01792	62.3048	70.6535	54.00	82.00
	9	22	56.4545	7.61407	1.62332	53.0787	59.8304	40.00	82.00
	10	23	69.7826	11.46618	2.39086	64.8243	74.7410	52.00	92.50
	11	21	68.5476	16.53172	3.60752	61.0225	76.0728	45.00	97.50
	Total	247	67.0830	13.41073	.85330	65.4023	68.7637	24.00	98.50
IndvTest	1	21	68.8364	13.87376	3.02750	62.5211	75.1517	38.59	88.50
	2	23	71.6673	11.07066	2.30839	66.8800	76.4546	54.11	93.64
	3	23	59.6641	14.90008	3.10688	53.2208	66.1074	23.27	80.18
	4	22	71.6300	12.53869	2.67326	66.0707	77.1893	52.36	95.86
	5	22	74.0711	12.38319	2.64010	68.5807	79.5615	32.93	90.93
	6	22	68.8257	11.05438	2.35680	63.9245	73.7270	30.43	82.52
	7	24	76.0280	13.60103	2.77630	70.2848	81.7712	27.20	92.59
	8	24	67.1206	13.16120	2.68652	61.5631	72.6781	47.51	90.37
	9	22	59.3495	9.27675	1.97781	55.2365	63.4626	38.89	78.91
	10	23	72.6153	11.30617	2.35750	67.7262	77.5045	47.64	89.73
	11	21	70.1614	15.31943	3.34297	63.1880	77.1347	25.61	89.21
	Total	247	69.1116	13.46518	.85677	67.4241	70.7992	23.27	95.86

Table FII.3: **Test of Homogeneity of Variances**

	Levene Statistic	df1	df2	Sig.
MODULE_SCORE	1.227	10	236	.274
Project	3.660	10	236	.000
IndvTest	1.099	10	236	.363

Table FII.4: ANOVA case 2006-S1 for module Score

				Sum of Squares	df	Mean Square	F	Sig.
MODULE SCORE	Between Groups	(Combined)		5483.565	10	548.357	3.755	.000
		Linear Term	Unweighted	10.091	1	10.091	.069	.793
			Weighted	7.348	1	7.348	.050	.823
			Deviation	5476.218	9	608.469	4.167	.000
		Quadratic Term	Unweighted	.918	1	.918	.006	.937
			Weighted	2.439	1	2.439	.017	.897
			Deviation	5473.779	8	684.222	4.686	.000
	Within Groups			34461.738	236	146.024		
	Total			39945.304	246			
Project	Between Groups	(Combined)		6056.277	10	605.628	3.743	.000
		Linear Term	Unweighted	197.671	1	197.671	1.222	.270
			Weighted	195.660	1	195.660	1.209	.273
			Deviation	5860.617	9	651.180	4.024	.000
		Quadratic Term	Unweighted	153.899	1	153.899	.951	.330
			Weighted	135.734	1	135.734	.839	.361
			Deviation	5724.882	8	715.610	4.423	.000
	Within Groups			38186.271	236	161.806		
	Total			44242.549	246			
IndvTest	Between Groups	(Combined)		6532.421	10	653.242	4.050	.000
		Linear Term	Unweighted	1.161	1	1.161	.007	.932
			Weighted	2.944	1	2.944	.018	.893
			Deviation	6529.477	9	725.497	4.497	.000
		Quadratic Term	Unweighted	40.632	1	40.632	.252	.616
			Weighted	47.715	1	47.715	.296	.587
			Deviation	6481.762	8	810.220	5.023	.000
	Within Groups			38070.075	236	161.314		
	Total			44602.496	246			

Table FII.5: Post Hoc Procedure: Multiple comparisons case 2006-S1

Dependent Variable		(I) Group	(J) Group	Mean Diff (I-J)	Std. Error	Sig.	95% Confidence Interval	
							Lower Bound	Upper Bound
MODULE SCORE	Games-Howell	1	2	-2.928	3.812	.999	-15.94	10.09
			3	6.638	3.947	.836	-6.81	20.08
			4	-3.439	4.135	.999	-17.53	10.65
			5	-3.121	3.896	.999	-16.42	10.17
			6	2.333	3.726	1.000	-10.42	15.08
			7	-7.042	3.885	.766	-20.28	6.20
			8	1.417	3.843	1.000	-11.69	14.52
			9	9.879	3.425	.173	-1.97	21.72
			10	-3.362	3.736	.998	-16.13	9.41

Appendix F: Module group statistics

Dependent Variable	(I) Group	(J) Group	Mean Diff (I-J)	Std. Error	Sig.	95% Confidence Interval	
		11	-1.333	4.392	1.000	-16.32	13.65
	2	1	2.928	3.812	.999	-10.09	15.94
		3	9.565	3.547	.235	-2.48	21.61
		4	-.512	3.756	1.000	-13.30	12.28
		5	-.194	3.491	1.000	-12.06	11.67
		6	5.261	3.300	.876	-5.95	16.47
		7	-4.114	3.479	.981	-15.91	7.68
		8	4.344	3.432	.970	-7.29	15.98
		9	12.806(*)	2.956	.004	2.72	22.90
		10	-.435	3.311	1.000	-11.67	10.80
		11	1.594	4.037	1.000	-12.22	15.41
	3	1	-6.638	3.947	.836	-20.08	6.81
		2	-9.565	3.547	.235	-21.61	2.48
		4	-10.077	3.892	.286	-23.31	3.16
		5	-9.759	3.637	.241	-22.12	2.60
		6	-4.304	3.454	.973	-16.05	7.44
		7	-13.679(*)	3.626	.018	-25.97	-1.39
		8	-5.221	3.580	.926	-17.36	6.92
		9	3.241	3.127	.993	-7.46	13.94
		10	-10.000	3.465	.161	-21.77	1.77
		11	-7.971	4.164	.705	-22.19	6.25
	4	1	3.439	4.135	.999	-10.65	17.53
		2	.512	3.756	1.000	-12.28	13.30
		3	10.077	3.892	.286	-3.16	23.31
		5	.318	3.841	1.000	-12.76	13.40
		6	5.773	3.669	.884	-6.75	18.29
		7	-3.602	3.830	.997	-16.62	9.42
		8	4.856	3.787	.967	-8.02	17.74
		9	13.318(*)	3.362	.014	1.74	24.90
		10	.077	3.678	1.000	-12.46	12.62
		11	2.106	4.343	1.000	-12.70	16.92
	5	1	3.121	3.896	.999	-10.17	16.42
		2	.194	3.491	1.000	-11.67	12.06
		3	9.759	3.637	.241	-2.60	22.12
		4	-.318	3.841	1.000	-13.40	12.76
		6	5.455	3.397	.871	-6.11	17.02
		7	-3.920	3.571	.989	-16.04	8.20
		8	4.538	3.525	.966	-7.43	16.50
		9	13.000(*)	3.064	.006	2.50	23.50
		10	-.241	3.407	1.000	-11.83	11.34
		11	1.788	4.116	1.000	-12.29	15.86
	6	1	-2.333	3.726	1.000	-15.08	10.42
		2	-5.261	3.300	.876	-16.47	5.95
		3	4.304	3.454	.973	-7.44	16.05
		4	-5.773	3.669	.884	-18.29	6.75
		5	-5.455	3.397	.871	-17.02	6.11
		7	-9.375	3.384	.204	-20.86	2.11
		8	-.917	3.336	1.000	-12.24	10.40
		9	7.545	2.844	.258	-2.17	17.26
		10	-5.696	3.211	.788	-16.61	5.21
		11	-3.667	3.956	.997	-17.24	9.91

Appendix F: Module group statistics

Dependent Variable	(I) Group	(J) Group	Mean Diff (I-J)	Std. Error	Sig.	95% Confidence Interval	
	7	1	7.042	3.885	.766	-6.20	20.28
		2	4.114	3.479	.981	-7.68	15.91
		3	13.679(*)	3.626	.018	1.39	25.97
		4	3.602	3.830	.997	-9.42	16.62
		5	3.920	3.571	.989	-8.20	16.04
		6	9.375	3.384	.204	-2.11	20.86
		8	8.458	3.513	.384	-3.43	20.35
		9	16.920(*)	3.050	.000	6.52	27.32
		10	3.679	3.395	.990	-7.83	15.19
		11	5.708	4.106	.944	-8.32	19.73
	8	1	-1.417	3.843	1.000	-14.52	11.69
		2	-4.344	3.432	.970	-15.98	7.29
		3	5.221	3.580	.926	-6.92	17.36
		4	-4.856	3.787	.967	-17.74	8.02
		5	-4.538	3.525	.966	-16.50	7.43
		6	.917	3.336	1.000	-10.40	12.24
		7	-8.458	3.513	.384	-20.35	3.43
		9	8.462	2.996	.185	-1.75	18.67
		10	-4.779	3.346	.934	-16.12	6.56
		11	-2.750	4.066	1.000	-16.65	11.15
	9	1	-9.879	3.425	.173	-21.72	1.97
		2	-12.806(*)	2.956	.004	-22.90	-2.72
		3	-3.241	3.127	.993	-13.94	7.46
		4	-13.318(*)	3.362	.014	-24.90	-1.74
		5	-13.000(*)	3.064	.006	-23.50	-2.50
		6	-7.545	2.844	.258	-17.26	2.17
		7	-16.920(*)	3.050	.000	-27.32	-6.52
		8	-8.462	2.996	.185	-18.67	1.75
		10	-13.241(*)	2.857	.002	-22.98	-3.50
		11	-11.212	3.673	.125	-23.97	1.54
	10	1	3.362	3.736	.998	-9.41	16.13
		2	.435	3.311	1.000	-10.80	11.67
		3	10.000	3.465	.161	-1.77	21.77
		4	-.077	3.678	1.000	-12.62	12.46
		5	.241	3.407	1.000	-11.34	11.83
		6	5.696	3.211	.788	-5.21	16.61
		7	-3.679	3.395	.990	-15.19	7.83
		8	4.779	3.346	.934	-6.56	16.12
		9	13.241(*)	2.857	.002	3.50	22.98
		11	2.029	3.964	1.000	-11.57	15.63
	11	1	1.333	4.392	1.000	-13.65	16.32
		2	-1.594	4.037	1.000	-15.41	12.22
		3	7.971	4.164	.705	-6.25	22.19
		4	-2.106	4.343	1.000	-16.92	12.70
		5	-1.788	4.116	1.000	-15.86	12.29
		6	3.667	3.956	.997	-9.91	17.24
		7	-5.708	4.106	.944	-19.73	8.32
		8	2.750	4.066	1.000	-11.15	16.65
		9	11.212	3.673	.125	-1.54	23.97
		10	-2.029	3.964	1.000	-15.63	11.57
Project	Games- 1	2	-3.13251	4.47025	1.000	-18.3835	12.1185

Appendix F: Module group statistics

Dependent Variable	(I) Group	(J) Group	Mean Diff (I-J)	Std. Error	Sig.	95% Confidence Interval	
Howell							
		3	.78054	3.81977	1.000	-12.5380	14.0990
		4	-4.77381	5.03185	.996	-21.9140	12.3664
		5	2.20346	4.55239	1.000	-13.3267	17.7336
		6	7.90801	4.24402	.736	-6.6414	22.4574
		7	-6.54464	4.09237	.872	-20.6238	7.5345
		8	.99702	3.98971	1.000	-12.7858	14.7798
		9	11.02165	3.80539	.174	-2.2610	24.3043
		10	-2.30642	4.19071	1.000	-16.6849	12.0721
		11	-1.07143	4.98598	1.000	-18.0799	15.9370
	2	1	3.13251	4.47025	1.000	-12.1185	18.3835
		3	3.91304	3.29880	.980	-7.4247	15.2508
		4	-1.64130	4.64877	1.000	-17.4958	14.2132
		5	5.33597	4.12502	.965	-8.6806	19.3526
		6	11.04051	3.78196	.151	-1.8164	23.8974
		7	-3.41214	3.61096	.997	-15.6958	8.8715
		8	4.12953	3.49418	.981	-7.7893	16.0483
		9	14.15415(*)	3.28215	.005	2.8613	25.4470
		10	.82609	3.72203	1.000	-11.8230	13.4752
		11	2.06108	4.59907	1.000	-13.6493	17.7715
	3	1	-.78054	3.81977	1.000	-14.0990	12.5380
		2	-3.91304	3.29880	.980	-15.2508	7.4247
		4	-5.55435	4.02723	.944	-19.5833	8.4746
		5	1.42292	3.40930	1.000	-10.3489	13.1948
		6	7.12747	2.98508	.401	-3.1038	17.3588
		7	-7.32518	2.76523	.257	-16.7292	2.0788
		8	.21649	2.61090	1.000	-8.6455	9.0784
		9	10.24111(*)	2.31948	.003	2.3615	18.1207
		10	-3.08696	2.90879	.991	-13.0205	6.8465
		11	-1.85197	3.96976	1.000	-15.7225	12.0185
	4	1	4.77381	5.03185	.996	-12.3664	21.9140
		2	1.64130	4.64877	1.000	-14.2132	17.4958
		3	5.55435	4.02723	.944	-8.4746	19.5833
		5	6.97727	4.72782	.919	-9.1416	23.0962
		6	12.68182	4.43167	.175	-2.5074	27.8710
		7	-1.77083	4.28666	1.000	-16.5169	12.9753
		8	5.77083	4.18876	.945	-8.6959	20.2375
		9	15.79545(*)	4.01359	.017	1.8004	29.7905
		10	2.46739	4.38064	1.000	-12.5608	17.4956
		11	3.70238	5.14664	1.000	-13.8273	21.2321
	5	1	-2.20346	4.55239	1.000	-17.7336	13.3267
		2	-5.33597	4.12502	.965	-19.3526	8.6806
		3	-1.42292	3.40930	1.000	-13.1948	10.3489
		4	-6.97727	4.72782	.919	-23.0962	9.1416
		6	5.70455	3.87871	.921	-7.5124	18.9215
		7	-8.74811	3.71217	.418	-21.4175	3.9213
		8	-1.20644	3.59868	1.000	-13.5285	11.1156
		9	8.81818	3.39318	.290	-2.9113	20.5476
		10	-4.50988	3.82030	.981	-17.5281	8.5084
		11	-3.27489	4.67896	1.000	-19.2524	12.7027

Appendix F: Module group statistics

Dependent Variable	(I) Group	(J) Group	Mean Diff (I-J)	Std. Error	Sig.	95% Confidence Interval	
	6	1	-7.90801	4.24402	.736	-22.4574	6.6414
		2	-11.04051	3.78196	.151	-23.8974	1.8164
		3	-7.12747	2.98508	.401	-17.3588	3.1038
		4	-12.68182	4.43167	.175	-27.8710	2.5074
		5	-5.70455	3.87871	.921	-18.9215	7.5124
		7	-14.4526(*)	3.32680	.004	-25.7556	-3.1497
		8	-6.91098	3.19967	.544	-17.8031	3.9812
		9	3.11364	2.96666	.992	-7.0669	13.2942
		10	-10.21443	3.44705	.138	-21.9269	1.4980
		11	-8.97944	4.37951	.617	-24.0187	6.0599
	7	1	6.54464	4.09237	.872	-7.5345	20.6238
		2	3.41214	3.61096	.997	-8.8715	15.6958
		3	7.32518	2.76523	.257	-2.0788	16.7292
		4	1.77083	4.28666	1.000	-12.9753	16.5169
		5	8.74811	3.71217	.418	-3.9213	21.4175
		6	14.45265(*)	3.32680	.004	3.1497	25.7556
		8	7.54167	2.99561	.322	-2.6036	17.6869
		9	17.56629(*)	2.74534	.000	8.2194	26.9131
		10	4.23822	3.25852	.964	-6.8106	15.2870
		11	5.47321	4.23271	.964	-9.1189	20.0653
	8	1	-.99702	3.98971	1.000	-14.7798	12.7858
		2	-4.12953	3.49418	.981	-16.0483	7.7893
		3	-.21649	2.61090	1.000	-9.0784	8.6455
		4	-5.77083	4.18876	.945	-20.2375	8.6959
		5	1.20644	3.59868	1.000	-11.1156	13.5285
		6	6.91098	3.19967	.544	-3.9812	17.8031
		7	-7.54167	2.99561	.322	-17.6869	2.6036
		9	10.02462(*)	2.58982	.014	1.2242	18.8250
		10	-3.30344	3.12861	.992	-13.9261	7.3193
		11	-2.06845	4.13354	1.000	-16.3791	12.2422
	9	1	-11.02165	3.80539	.174	-24.3043	2.2610
		2	-14.15415(*)	3.28215	.005	-25.4470	-2.8613
		3	-10.24111(*)	2.31948	.003	-18.1207	-2.3615
		4	-15.79545(*)	4.01359	.017	-29.7905	-1.8004
		5	-8.81818	3.39318	.290	-20.5476	2.9113
		6	-3.11364	2.96666	.992	-13.2942	7.0669
		7	-17.56629(*)	2.74534	.000	-26.9131	-8.2194
		8	-10.02462(*)	2.58982	.014	-18.8250	-1.2242
		10	-13.32806(*)	2.88988	.002	-23.2085	-3.4476
		11	-12.09307	3.95593	.128	-25.9296	1.7434
	10	1	2.30642	4.19071	1.000	-12.0721	16.6849
		2	-.82609	3.72203	1.000	-13.4752	11.8230
		3	3.08696	2.90879	.991	-6.8465	13.0205
		4	-2.46739	4.38064	1.000	-17.4956	12.5608
		5	4.50988	3.82030	.981	-8.5084	17.5281
		6	10.21443	3.44705	.138	-1.4980	21.9269
		7	-4.23822	3.25852	.964	-15.2870	6.8106

Appendix F: Module group statistics

Dependent Variable		(I) Group	(J) Group	Mean Diff (I-J)	Std. Error	Sig.	95% Confidence Interval	
			8	3.30344	3.12861	.992	-7.3193	13.9261
			9	13.32806(*)	2.88988	.002	3.4476	23.2085
			11	1.23499	4.32786	1.000	-13.6417	16.1117
		11	1	1.07143	4.98598	1.000	-15.9370	18.0799
			2	-2.06108	4.59907	1.000	-17.7715	13.6493
			3	1.85197	3.96976	1.000	-12.0185	15.7225
			4	-3.70238	5.14664	1.000	-21.2321	13.8273
			5	3.27489	4.67896	1.000	-12.7027	19.2524
			6	8.97944	4.37951	.617	-6.0599	24.0187
			7	-5.47321	4.23271	.964	-20.0653	9.1189
			8	2.06845	4.13354	1.000	-12.2422	16.3791
			9	12.09307	3.95593	.128	-1.7434	25.9296
			10	-1.23499	4.32786	1.000	-16.1117	13.6417
IndvTest	Games-Howell	1	2	-2.83093	3.80716	1.000	-15.8506	10.1887
			3	9.17230	4.33803	.574	-5.5838	23.9284
			4	-2.79361	4.03882	1.000	-16.5670	10.9798
			5	-5.23471	4.01695	.963	-18.9364	8.4670
			6	.01068	3.83670	1.000	-13.1109	13.1322
			7	-7.19164	4.10775	.801	-21.1646	6.7813
			8	1.71580	4.04761	1.000	-12.0609	15.4925
			9	9.48685	3.61628	.276	-2.9568	21.9305
			10	-3.77895	3.83713	.995	-16.8925	9.3346
			11	-1.32497	4.51013	1.000	-16.7169	14.0670
		2	1	2.83093	3.80716	1.000	-10.1887	15.8506
			3	12.00323	3.87058	.103	-1.1871	25.1936
			4	.03733	3.53199	1.000	-11.9797	12.0544
			5	-2.40377	3.50697	1.000	-14.3331	9.5256
			6	2.84161	3.29897	.998	-8.3666	14.0498
			7	-4.36071	3.61061	.978	-16.6133	7.8919
			8	4.54673	3.54204	.967	-7.4675	16.5610
			9	12.31778(*)	3.03980	.009	1.9818	22.6538
			10	-.94801	3.29947	1.000	-12.1433	10.2473
			11	1.50597	4.06253	1.000	-12.4359	15.4479
		3	1	-9.17230	4.33803	.574	-23.9284	5.5838
			2	-12.00323	3.87058	.103	-25.1936	1.1871
			4	-11.96590	4.09866	.151	-25.9015	1.9697
			5	-14.407(*)	4.07711	.036	-28.2717	-.5423
			6	-9.16161	3.89964	.422	-22.4524	4.1292
			7	-16.3639(*)	4.16660	.012	-30.4978	-2.2301
			8	-7.45650	4.10732	.765	-21.3962	6.4832
			9	.31455	3.68299	1.000	-12.3049	12.9340
			10	-12.95124	3.90007	.062	-26.2345	.3320
			11	-10.49726	4.56379	.453	-26.0333	5.0388
		4	1	2.79361	4.03882	1.000	-10.9798	16.5670
			2	-.03733	3.53199	1.000	-12.0544	11.9797
			3	11.96590	4.09866	.151	-1.9697	25.9015
			5	-2.44110	3.75719	1.000	-15.2212	10.3390
			6	2.80429	3.56382	.999	-9.3284	14.9370
			7	-4.39804	3.85411	.986	-17.4750	8.6790

Appendix F: Module group statistics

Dependent Variable	(I) Group	(J) Group	Mean Diff (I-J)	Std. Error	Sig.	95% Confidence Interval	
		8	4.50940	3.78995	.980	-8.3510	17.3698
		9	12.28045(*)	3.32536	.025	.9161	23.6448
		10	-.98534	3.56428	1.000	-13.1080	11.1373
		11	1.46864	4.28039	1.000	-13.1596	16.0969
	5	1	5.23471	4.01695	.963	-8.4670	18.9364
		2	2.40377	3.50697	1.000	-9.5256	14.3331
		3	14.40700(*)	4.07711	.036	.5423	28.2717
		4	2.44110	3.75719	1.000	-10.3390	15.2212
		6	5.24539	3.53902	.918	-6.8010	17.2917
		7	-1.95693	3.83119	1.000	-14.9561	11.0423
		8	6.95051	3.76663	.747	-5.8302	19.7312
		9	14.72156(*)	3.29877	.003	3.4521	25.9910
		10	1.45576	3.53949	1.000	-10.5803	13.4918
		11	3.90974	4.25977	.997	-10.6528	18.4723
	6	1	-.01068	3.83670	1.000	-13.1322	13.1109
		2	-2.84161	3.29897	.998	-14.0498	8.3666
		3	9.16161	3.89964	.422	-4.1292	22.4524
		4	-2.80429	3.56382	.999	-14.9370	9.3284
		5	-5.24539	3.53902	.918	-17.2917	6.8010
		7	-7.20232	3.64175	.664	-19.5676	5.1630
		8	1.70512	3.57378	1.000	-10.4253	13.8355
		9	9.47617	3.07673	.108	-1.0066	19.9589
		10	-3.78963	3.33351	.986	-15.1143	7.5351
		11	-1.33565	4.09023	1.000	-15.3692	12.6980
	7	1	7.19164	4.10775	.801	-6.7813	21.1646
		2	4.36071	3.61061	.978	-7.8919	16.6133
		3	16.36394(*)	4.16660	.012	2.2301	30.4978
		4	4.39804	3.85411	.986	-8.6790	17.4750
		5	1.95693	3.83119	1.000	-11.0423	14.9561
		6	7.20232	3.64175	.664	-5.1630	19.5676
		8	8.90744	3.86332	.447	-4.1716	21.9865
		9	16.67849(*)	3.40875	.001	5.0645	28.2925
		10	3.41269	3.64220	.997	-8.9433	15.7687
		11	5.86668	4.34549	.953	-8.9467	20.6801
	8	1	-1.71580	4.04761	1.000	-15.4925	12.0609
		2	-4.54673	3.54204	.967	-16.5610	7.4675
		3	7.45650	4.10732	.765	-6.4832	21.3962
		4	-4.50940	3.78995	.980	-17.3698	8.3510
		5	-6.95051	3.76663	.747	-19.7312	5.8302
		6	-1.70512	3.57378	1.000	-13.8355	10.4253
		7	-8.90744	3.86332	.447	-21.9865	4.1716
		9	7.77105	3.33603	.434	-3.5858	19.1279
		10	-5.49475	3.57424	.899	-17.6152	6.6257
		11	-3.04077	4.28869	1.000	-17.6739	11.5924
	9	1	-9.48685	3.61628	.276	-21.9305	2.9568
		2	-12.3178(*)	3.03980	.009	-22.6538	-1.9818
		3	-.31455	3.68299	1.000	-12.9340	12.3049
		4	-12.2805(*)	3.32536	.025	-23.6448	-.9161
		5	-14.7216(*)	3.29877	.003	-25.9910	-3.4521
		6	-9.47617	3.07673	.108	-19.9589	1.0066

Dependent Variable	(I) Group	(J) Group	Mean Diff (I-J)	Std. Error	Sig.	95% Confidence Interval	
		7	-16.6785(*)	3.40875	.001	-28.2925	-5.0645
		8	-7.77105	3.33603	.434	-19.1279	3.5858
		10	-13.2658(*)	3.07726	.004	-23.7324	-2.7992
		11	-10.81182	3.88422	.209	-24.2333	2.6096
	10	1	3.77895	3.83713	.995	-9.3346	16.8925
		2	.94801	3.29947	1.000	-10.2473	12.1433
		3	12.95124	3.90007	.062	-.3320	26.2345
		4	.98534	3.56428	1.000	-11.1373	13.1080
		5	-1.45576	3.53949	1.000	-13.4918	10.5803
		6	3.78963	3.33351	.986	-7.5351	15.1143
		7	-3.41269	3.64220	.997	-15.7687	8.9433
		8	5.49475	3.57424	.899	-6.6257	17.6152
		9	13.26580(*)	3.07726	.004	2.7992	23.7324
		11	2.45398	4.09063	1.000	-11.5731	16.4811
	11	1	1.32497	4.51013	1.000	-14.0670	16.7169
		2	-1.50597	4.06253	1.000	-15.4479	12.4359
		3	10.49726	4.56379	.453	-5.0388	26.0333
		4	-1.46864	4.28039	1.000	-16.0969	13.1596
		5	-3.90974	4.25977	.997	-18.4723	10.6528
		6	1.33565	4.09023	1.000	-12.6980	15.3692
		7	-5.86668	4.34549	.953	-20.6801	8.9467
		8	3.04077	4.28869	1.000	-11.5924	17.6739
		9	10.81182	3.88422	.209	-2.6096	24.2333
		10	-2.45398	4.09063	1.000	-16.4811	11.5731

Table FII.6: Case 2006-S1: Homogeneous Subsets MODULE SCORE

	Group	N	Subset for alpha = .05		
			1	2	3
Gabriel(a,b)	9	22	58.45		
	3	23	61.70	61.70	
	6	22	66.00	66.00	66.00
	8	24	66.92	66.92	66.92
	1	21	68.33	68.33	68.33
	11	21	69.67	69.67	69.67
	2	23		71.26	71.26
	5	22		71.45	71.45
	10	23		71.70	71.70
	4	22		71.77	71.77
	7	24			75.38
	Sig.		.109	.263	.414
Hochberg(a,b)	9	22	58.45		
	3	23	61.70	61.70	
	6	22	66.00	66.00	66.00
	8	24	66.92	66.92	66.92
	1	21	68.33	68.33	68.33
	11	21	69.67	69.67	69.67
	2	23		71.26	71.26
	5	22		71.45	71.45
	10	23		71.70	71.70

	4	22		71.77	71.77
	7	24			75.38
	Sig.		.109	.263	.414

Means for groups in homogeneous subsets are displayed.
a Uses Harmonic Mean Sample Size = 22.411.
b The group sizes are unequal. The harmonic mean of the group sizes is used. Type I error levels are not guaranteed.

Table FII.7: Case 2006-S1: Homogeneous Subsets PROJECT

	Group	N	Subset for alpha = .05		
			1	2	3
Gabriel(a,b)	9	22	56.4545		
	6	22	59.5682	59.5682	
	5	22	65.2727	65.2727	65.2727
	8	24	66.4792	66.4792	66.4792
	3	23	66.6957	66.6957	66.6957
	1	21	67.4762	67.4762	67.4762
	11	21	68.5476	68.5476	68.5476
	10	23		69.7826	69.7826
	2	23		70.6087	70.6087
	4	22		72.2500	72.2500
	7	24			74.0208
	Sig.		.086	.052	.692
Hochberg(a,b)	9	22	56.4545		
	6	22	59.5682	59.5682	
	5	22	65.2727	65.2727	65.2727
	8	24	66.4792	66.4792	66.4792
	3	23	66.6957	66.6957	66.6957
	1	21	67.4762	67.4762	67.4762
	11	21	68.5476	68.5476	68.5476
	10	23		69.7826	69.7826
	2	23		70.6087	70.6087
	4	22		72.2500	72.2500
	7	24			74.0208
	Sig.		.086	.052	.692

Means for groups in homogeneous subsets are displayed.
a Uses Harmonic Mean Sample Size = 22.411.
b The group sizes are unequal. The harmonic mean of the group sizes is used. Type I error levels are not guaranteed.

Table FII.8: Case 2006-S1: Homogeneous Subsets INDIVIDUAL TEST

	Group	N	Subset for alpha = .05	
			1	2
Gabriel(a,b)	9	22	59.3495	
	3	23	59.6641	
	8	24	67.1206	67.1206
	6	22	68.8257	68.8257
	1	21	68.8364	68.8364
	11	21	70.1614	70.1614
	4	22	71.6300	71.6300
	2	23	71.6673	71.6673

	10	23		72.6153
	5	22		74.0711
	7	24		76.0280
	Sig.		.070	.650
	9	22	59.3495	
	3	23	59.6641	
	8	24	67.1206	67.1206
Hochberg(a,b)	6	22	68.8257	68.8257
	1	21	68.8364	68.8364
	11	21	70.1614	70.1614
	4	22	71.6300	71.6300
	2	23	71.6673	71.6673
	10	23		72.6153
	5	22		74.0711
	7	24		76.0280
	Sig.		.070	.650

Means for groups in homogeneous subsets are displayed.
a Uses Harmonic Mean Sample Size = 22.411.
b The group sizes are unequal. The harmonic mean of the group sizes is used. Type I error levels are not guaranteed.

Case 2005-S1: Frequency Tables of Independent variables used in ANCOVA (univariate Analysis of Covariance)

Table FII.9: Gender

TUTGRP			Frequency	Valid Percent	Cumulative Percent
EI0601	Valid	F	9	42.9	42.9
		M	12	57.1	100.0
		Total	21	100.0	
EI0602	Valid	F	13	56.5	56.5
		M	10	43.5	100.0
		Total	23	100.0	
EI0603	Valid	F	7	30.4	30.4
		M	16	69.6	100.0
		Total	23	100.0	
EI0604	Valid	F	7	31.8	31.8
		M	15	68.2	100.0
		Total	22	100.0	
EI0605	Valid	F	9	40.9	40.9
		M	13	59.1	100.0
		Total	22	100.0	
EI0606	Valid	F	9	40.9	40.9
		M	13	59.1	100.0
		Total	22	100.0	
EI0607	Valid	F	16	66.7	66.7
		M	8	33.3	100.0
		Total	24	100.0	
EI0608	Valid	F	8	33.3	33.3
		M	16	66.7	100.0
		Total	24	100.0	
EI0609	Valid	F	9	40.9	40.9
		M	13	59.1	100.0
		Total	22	100.0	
EI0610	Valid	F	10	43.5	43.5
		M	13	56.5	100.0
		Total	23	100.0	
EI0611	Valid	F	11	52.4	52.4
		M	10	47.6	100.0
		Total	21	100.0	

Table FII.10: OLAGG (entry level aggregate)

TUTGRP			Frequency	Valid Percent	Cumulative Percent
EI0601	Valid	10	3	14.3	14.3
		14	1	4.8	19.0
		21	4	19.0	38.1
		22	2	9.5	47.6
		23	3	14.3	61.9
		24	3	14.3	76.2
		25	1	4.8	81.0
		26	4	19.0	100.0
		Total	21	100.0	Mean: 21.05
EI0602	Valid	10	3	13.0	13.0
		16	1	4.3	17.4
		20	1	4.3	21.7
		22	2	8.7	30.4
		23	1	4.3	34.8
		24	10	43.5	78.3
		25	4	17.4	95.7
		26	1	4.3	100.0
		Total	23	100.0	Mean: 21.70
EI0603	Valid	10	1	4.3	4.3
		15	1	4.3	8.7
		18	1	4.3	13.0
		20	2	8.7	21.7
		22	2	8.7	30.4
		23	1	4.3	34.8
		24	10	43.5	78.3
		25	3	13.0	91.3
		26	2	8.7	100.0
		Total	23	100.0	Mean: 22.48
EI0604	Valid	10	3	13.6	13.6
		19	1	4.5	18.2
		20	1	4.5	22.7
		21	1	4.5	27.3
		23	6	27.3	54.5
		24	2	9.1	63.6
		25	3	13.6	77.3
		26	3	13.6	90.9
		27	2	9.1	100.0
		Total	22	100.0	Mean: 21.95
EI0605	Valid	16	1	4.5	4.5
		18	1	4.5	9.1
		19	1	4.5	13.6
		20	2	9.1	22.7
		21	1	4.5	27.3
		22	3	13.6	40.9
		23	2	9.1	50.0
		24	4	18.2	68.2
		25	3	13.6	81.8
		26	3	13.6	95.5
		27	1	4.5	100.0
		Total	22	100.0	Mean: 22.82
EI0606	Valid	19	1	4.5	4.5
		21	2	9.1	13.6
		22	1	4.5	18.2
		23	4	18.2	36.4
		24	7	31.8	68.2

		25	5	22.7	90.9
		26	2	9.1	100.0
		Total	22	100.0	Mean: 23.64
EI0607	Valid	10	1	4.2	4.2
		17	1	4.2	8.3
		20	3	12.5	20.8
		23	2	8.3	29.2
		24	8	33.3	62.5
		25	6	25.0	87.5
		26	3	12.5	100.0
		Total	24	100.0	Mean: 23.04
EI0608	Valid	10	2	8.3	8.3
		18	1	4.2	12.5
		20	4	16.7	29.2
		21	2	8.3	37.5
		22	1	4.2	41.7
		23	2	8.3	50.0
		24	5	20.8	70.8
		25	5	20.8	91.7
		26	2	8.3	100.0
		Total	24	100.0	Mean: 21.88
EI0609	Valid	10	1	4.5	4.5
		17	1	4.5	9.1
		19	1	4.5	13.6
		20	2	9.1	22.7
		21	2	9.1	31.8
		22	1	4.5	36.4
		23	2	9.1	45.5
		24	5	22.7	68.2
		25	3	13.6	81.8
		26	2	9.1	90.9
		27	2	9.1	100.0
		Total	22	100.0	Mean: 22.59
EI0610	Valid	10	5	21.7	21.7
		20	1	4.3	26.1
		22	2	8.7	34.8
		23	8	34.8	69.6
		24	3	13.0	82.6
		25	3	13.0	95.7
		26	1	4.3	100.0
		Total	23	100.0	Mean: 20.48
EI0611	Valid	10	10	47.6	47.6
		17	1	4.8	52.4
		22	3	14.3	66.7
		23	1	4.8	71.4
		24	5	23.8	95.2
		26	1	4.8	100.0
		Total	21	100.0	Mean: 16.76

Note: Students with prior computing has an entry level of 10 points

Table FII.11: **Tests of Between-Subjects Effects – Univariate ANOVA (OLAGG = entry level aggregate)**
Dependent Variable: MODULE_SCORE

Source	Type III Sum of Squares	df	Mean Square	F	Sig.	Partial Eta Sq	Noncent. Paramtr	Power (a)
Corrected Model	10876.785(b)	11	988.799	7.994	.000	.272	87.932	1.000
Intercept	83724.737	1	83724.737	676.860	.000	.742	676.860	1.000
OLAGG	5393.220	1	5393.220	43.601	.000	.156	43.601	1.000
Group	5572.330	10	557.233	4.505	.000	.161	45.049	.999
Error	29068.518	235	123.696					
Total	1197493.000	247						
Corrected Total	39945.304	246						

a Observed power computed using alpha = .05
b R Squared = .272 (Adjusted R Squared = .238)

Table FII.12: **Parameter Estimates**
Dependent Variable: MODULE_SCORE

Parameter	B	Std. Error	t	Sig.	95% Confidence Interval		Partial Eta Sq	Noncent. Paramtr	Power (a)
					Lower Bound	Upper Bound			
Intercept	87.171	3.594	24.254	.000	80.090	94.252	.715	24.254	1.000
OLAGG	-1.044	.158	-6.603	.000	-1.356	-.733	.156	6.603	1.000
[Group=1]	3.142	3.499	.898	.370	-3.750	10.035	.003	.898	.145
[Group=2]	6.747	3.446	1.958	.051	-.043	13.536	.016	1.958	.496
[Group=3]	-2.001	3.476	-.576	.565	-8.850	4.848	.001	.576	.088
[Group=4]	7.529	3.491	2.157	.032	.651	14.406	.019	2.157	.575
[Group=5]	8.112	3.526	2.301	.022	1.167	15.058	.022	2.301	.630
[Group=6]	3.512	3.563	.986	.325	-3.507	10.532	.004	.986	.166
[Group=7]	12.266	3.469	3.536	.000	5.433	19.100	.051	3.536	.941
[Group=8]	2.590	3.420	.757	.450	-4.149	9.328	.002	.757	.117
[Group=9]	-5.125	3.516	-1.458	.146	-12.052	1.802	.009	1.458	.306
[Group=10]	5.910	3.408	1.734	.084	-.804	12.624	.013	1.734	.408
[Group=11]	0(b)	.	.	.	.	.	.	.	.

a Observed power computed using alpha = .05
b This parameter is set to zero because it is redundant.

Table FII.13: Case 2006-S1 Estimated Marginal Means
Dependent Variable: MODULE_SCORE

Group	Mean	Std. Error	95% Confidence Interval	
			Lower Bound	Upper Bound
1	67.647(a)	2.429	62.862	72.433
2	71.252(a)	2.319	66.683	75.820
3	62.504(a)	2.322	57.929	67.079
4	72.034(a)	2.372	67.362	76.706
5	72.618(a)	2.378	67.933	77.302
6	68.017(a)	2.391	63.307	72.728
7	76.771(a)	2.280	72.279	81.263
8	67.095(a)	2.270	62.622	71.568
9	59.380(a)	2.375	54.701	64.060
10	70.415(a)	2.327	65.830	75.000
11	64.505(a)	2.550	59.482	69.528

a Covariates appearing in the model are evaluated at the following values: OLAGG = 21.70.

Table FII.14: **Tests of Between-Subjects Effects – Univariate ANOVA (Gender)**
Dependent Variable: MODULE_SCORE

Source	Type III Sum of Squares	df	Mean Square	F	Sig.	Partial Eta Sq	Noncent. Paramtr	Power (a)
Corrected Model	6322.396(b)	11	574.763	4.017	.000	.158	44.189	.999
Intercept	84407.297	1	84407.297	589.946	.000	.715	589.946	1.000
Gender	838.830	1	838.830	5.863	.016	.024	5.863	.674
Group	5950.171	10	595.017	4.159	.000	.150	41.587	.998
Error	33622.908	235	143.076					
Total	1197493.000	247						
Corrected Total	39945.304	246						

a Observed power computed using alpha = .05
b R Squared = .158 (Adjusted R Squared = .119)

Table FII.15: **Parameter Estimates**
Dependent Variable: MODULE_SCORE

Parameter	B	Std. Error	t	Sig.	95% Confidence Interval		Partial Eta Sq	Noncent. Paramtr	Power (a)
					Lower Bound	Upper Bound			
Intercept	64.050	3.492	18.342	.000	57.171	70.930	.589	18.342	1.000
Gender	3.805	1.571	2.421	.016	.709	6.900	.024	2.421	.674
[Group=1]	-1.696	3.694	-.459	.647	-8.974	5.583	.001	.459	.074
[Group=2]	1.752	3.611	.485	.628	-5.362	8.865	.001	.485	.077
[Group=3]	-8.806	3.627	-2.428	.016	-15.951	-1.661	.024	2.428	.677
[Group=4]	1.324	3.663	.361	.718	-5.894	8.541	.001	.361	.065
[Group=5]	1.351	3.654	.370	.712	-5.847	8.549	.001	.370	.066
[Group=6]	-4.103	3.654	-1.123	.263	-11.301	3.095	.005	1.123	.201
[Group=7]	6.252	3.581	1.746	.082	-.804	13.307	.013	1.746	.413
[Group=8]	-3.475	3.587	-.969	.334	-10.541	3.591	.004	.969	.162
[Group=9]	-11.649	3.654	-3.188	.002	-18.847	-4.451	.041	3.188	.888
[Group=10]	1.690	3.613	.468	.640	-5.428	8.808	.001	.468	.075
[Group=11]	0(b)	.	.	.	.	.	.	.	.

a Observed power computed using alpha = .05
b This parameter is set to zero because it is redundant.

Table FII.16: Case 2006-S1 Estimated Marginal Means
Dependent Variable: MODULE_SCORE

Group	Mean	Std. Error	95% Confidence Interval	
			Lower Bound	Upper Bound
1	68.300(a)	2.610	63.158	73.443
2	71.748(a)	2.502	66.818	76.677
3	61.190(a)	2.503	56.259	66.121
4	71.320(a)	2.557	66.282	76.357
5	71.347(a)	2.551	66.323	76.372
6	65.893(a)	2.551	60.868	70.918
7	76.248(a)	2.468	71.385	81.110
8	66.521(a)	2.447	61.700	71.342
9	58.347(a)	2.551	53.323	63.372
10	71.686(a)	2.494	66.773	76.600
11	69.996(a)	2.614	64.847	75.145

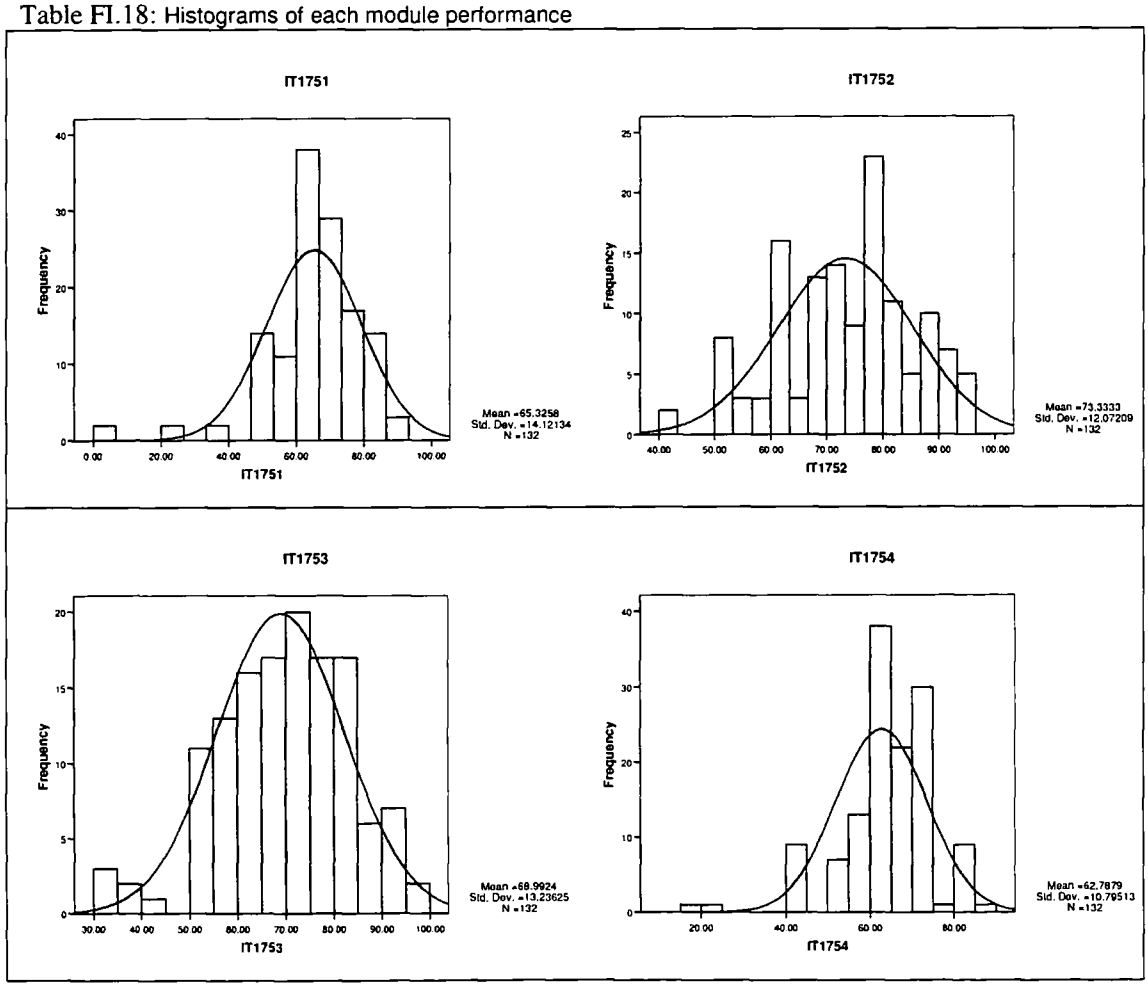
a Covariates appearing in the model are evaluated at the following values: Gender = 1.56 (males)

Table FII.17: Case 2006-S1: Study Programme A (groups 1 to 5 and 11)

Case 2006-S1-A	IT1751	IT1752	IT1753	IT1754	IT1755	IT1756
	Computing Maths 1	Digital Electronics	Principles of Computing	Business Information Systems	Internet Computing	Web Design & Multimedia
N	132	132	132	132	132	132
Mean	65.3258	73.3333	68.9924	62.7879	65.4773	68.8258
Std. Error of Mean	1.22910	1.05074	1.15207	.93960	.93072	.66535
Median	66.0000	75.0000	70.5000	63.0000	66.0000	68.0000
Mode	70.00	78.00	63.00(a)	70.00	70.00	67.00(a)
Std. Deviation	14.12134	12.07209	13.23625	10.79513	10.69313	7.64428
Variance	199.412	145.735	175.198	116.535	114.343	58.435
Skewness	-1.736	-.331	-.433	-1.170	-.310	-.461
Std. Err. of Skewness	.211	.211	.211	.211	.211	.211
Kurtosis	6.208	-.361	.326	3.592	.445	1.164
Std. Error of Kurtosis	.419	.419	.419	.419	.419	.419
Range	92.00	54.00	65.00	71.00	59.00	49.00
Minimum	.00	41.00	31.00	15.00	31.00	41.00
Maximum	92.00	95.00	96.00	86.00	90.00	90.00

a Multiple modes exist. The smallest value is shown

Table FI.18: Histograms of each module performance



Appendix F: Module group statistics

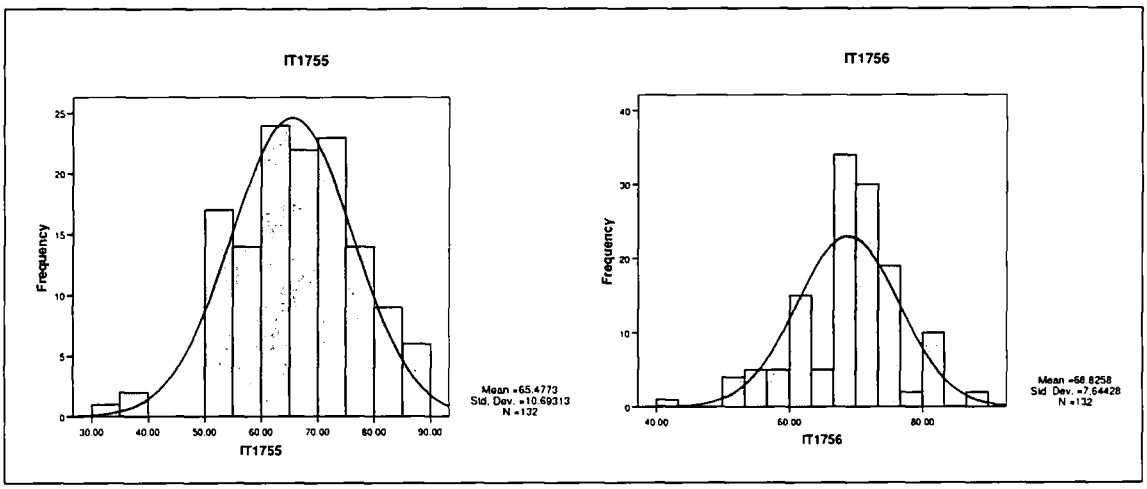


Table FI.19:Case 2006-S1-A Detailed module performance for groups: 01, 02, 03, 04, 05, 11

TUTGRP		IT1751	IT1752	IT1753	IT1754	IT1755	IT1756
EI0601	N	21	21	21	21	21	21
	Mean	63.0952	67.0952	68.3333	61.9524	64.6190	67.7143
	Std. Error of Mean	3.90232	2.81400	2.96032	2.36792	2.68814	2.01085
	Median	65.0000	65.0000	73.0000	61.0000	65.0000	67.0000
	Mode	70.00	50.00	80.00	51.00(a)	55.00(a)	63.00
	Std. Deviation	17.88269	12.89537	13.56589	10.85116	12.31859	9.21489
	Variance	319.790	166.290	184.033	117.748	151.748	84.914
	Skewness	-2.060	.555	-.328	.738	-.615	.386
	Std. Error of Skewness	.501	.501	.501	.501	.501	.501
	Kurtosis	7.675	-.386	-.707	.558	1.759	.680
	Std. Error of Kurtosis	.972	.972	.972	.972	.972	.972
	Range	92.00	43.00	52.00	43.00	56.00	39.00
	Minimum	.00	50.00	39.00	43.00	31.00	51.00
	Maximum	92.00	93.00	91.00	86.00	87.00	90.00
EI0602	N	23	23	23	23	23	23
	Mean	74.2174	78.7826	71.2609	67.2609	69.3478	71.5652
	Std. Error of Mean	1.76366	1.37015	2.40242	1.90300	1.89195	1.34521
	Median	75.0000	78.0000	70.0000	68.0000	66.0000	71.0000
	Mode	77.00(a)	78.00(a)	58.00	70.00	61.00(a)	71.00
	Std. Deviation	8.45822	6.57099	11.52159	9.12647	9.07348	6.45140
	Variance	71.542	43.178	132.747	83.292	82.328	41.621
	Skewness	-.101	-.397	.615	-.682	.881	-1.512
	Std. Error of Skewness	.481	.481	.481	.481	.481	.481
	Kurtosis	-.936	.623	-.559	2.036	-.206	4.768
	Std. Error of Kurtosis	.935	.935	.935	.935	.935	.935
	Range	30.00	28.00	38.00	42.00	32.00	31.00
	Minimum	60.00	63.00	55.00	41.00	58.00	50.00
	Maximum	90.00	91.00	93.00	83.00	90.00	81.00
EI0603	N	23	23	23	23	23	23
	Mean	62.2174	68.8261	61.6957	61.5217	62.1304	67.7826

Appendix F: Module group statistics

TUTGRP		IT1751	IT1752	IT1753	IT1754	IT1755	IT1756
	Std. Error of Mean	3.88519	1.96577	2.60998	2.47906	1.63618	1.86072
	Median	63.0000	69.0000	65.0000	63.0000	62.0000	68.0000
	Mode	70.00	78.00	66.00	70.00	60.00	67.00(a)
	Std. Deviation	18.63271	9.42748	12.51702	11.88915	7.84685	8.92370
	Variance	347.178	88.877	156.676	141.352	61.573	79.632
	Skewness	-1.739	-1.074	-1.039	-1.649	.312	-1.319
	Std. Error of Skewness	.481	.481	.481	.481	.481	.481
	Kurtosis	4.889	2.222	1.049	4.981	-.455	2.797
	Std. Error of Kurtosis	.935	.935	.935	.935	.935	.935
	Range	91.00	44.00	49.00	61.00	30.00	40.00
	Minimum	.00	41.00	31.00	22.00	50.00	41.00
	Maximum	91.00	85.00	80.00	83.00	80.00	81.00
EI0604	N	22	22	22	22	22	22
	Mean	66.1364	76.5455	71.7727	61.4545	67.3636	71.2727
	Std. Error of Mean	2.65905	2.73088	2.88745	2.88505	2.27403	1.20637
	Median	70.0000	78.0000	74.0000	65.0000	66.5000	72.0000
	Mode	70.00(a)	78.00	53.00(a)	70.00	58.00(a)	67.00(a)
	Std. Deviation	12.47205	12.80895	13.54334	13.53207	10.66613	5.65838
	Variance	155.552	164.069	183.422	183.117	113.766	32.017
	Skewness	-1.786	-.884	-.287	-2.315	.177	-.442
	Std. Error of Skewness	.491	.491	.491	.491	.491	.491
	Kurtosis	4.877	.940	-.243	6.032	-.834	-.411
	Std. Error of Kurtosis	.953	.953	.953	.953	.953	.953
	Range	60.00	52.00	52.00	57.00	38.00	20.00
	Minimum	25.00	42.00	44.00	15.00	50.00	60.00
	Maximum	85.00	94.00	96.00	72.00	88.00	80.00
EI0605	N	22	22	22	22	22	22
	Mean	60.7273	70.5909	71.4545	64.9091	62.2727	67.7727
	Std. Error of Mean	2.64590	2.48342	2.53352	1.55270	2.43725	.89475
	Median	61.0000	72.0000	71.5000	64.0000	63.5000	68.0000
	Mode	65.00	78.00	66.00	60.00	50.00	68.00
	Std. Deviation	12.41037	11.64825	11.88327	7.28279	11.43171	4.19673
	Variance	154.017	135.682	141.212	53.039	130.684	17.613
	Skewness	-.621	.082	-1.040	.700	-.530	-.828
	Std. Error of Skewness	.491	.491	.491	.491	.491	.491
	Kurtosis	1.695	-.565	2.495	-.196	-.284	1.166
	Std. Error of Kurtosis	.953	.953	.953	.953	.953	.953
	Range	57.00	44.00	55.00	25.00	45.00	18.00
	Minimum	26.00	50.00	36.00	55.00	35.00	57.00
	Maximum	83.00	94.00	91.00	80.00	80.00	75.00
EI0611	N	21	21	21	21	21	21
	Mean	65.1905	78.0476	69.6667	59.2857	67.1429	66.6190
	Std. Error of Mean	1.81871	2.98367	3.24429	2.23728	2.51877	2.04994
	Median	66.0000	78.0000	71.0000	61.0000	70.0000	67.0000
	Mode	72.00	89.00	70.00(a)	61.00	75.00	56.00(a)

Appendix F: Module group statistics

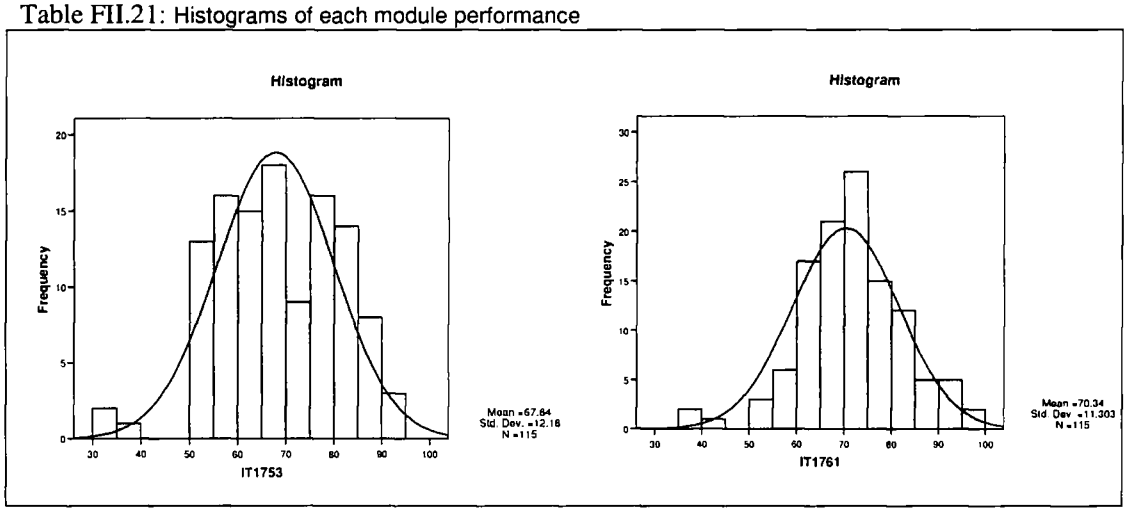
TUTGRP		IT1751	IT1752	IT1753	IT1754	IT1755	IT1756
	Std. Deviation	8.33438	13.67288	14.86719	10.25253	11.54247	9.39402
	Variance	69.462	186.948	221.033	105.114	133.229	88.248
	Skewness	-.497	-.689	-.637	-.733	-1.199	.578
	Std. Error of Skewness	.501	.501	.501	.501	.501	.501
	Kurtosis	-.388	-.489	.563	-.546	1.105	-.359
	Std. Error of Kurtosis	.972	.972	.972	.972	.972	.972
	Range	30.00	44.00	60.00	33.00	44.00	33.00
	Minimum	50.00	51.00	32.00	40.00	36.00	55.00
	Maximum	80.00	95.00	92.00	73.00	80.00	88.00

Table FI.20: Case 2006-S1: Study Programme B (groups 6 to 10)

	IT1761	IT1763	IT1753	IT1764	IT1765	IT1766
	Computing Maths 2	Manufactu -ring Processes	Principles of Computing	Fundamen-tals of Networking	Communi-cation Skills 1	Innovation Project
N	115	115	115	115	115	115
Mean	70.3391	68.2435	67.8435	62.4957	68.2000	71.3130
Std. Error of Mean	1.05399	.90322	1.13582	1.20951	.77806	.63647
Median	71.0000	66.0000	67.0000	63.0000	66.0000	72.0000
Mode	65.00	66.00(a)	58.00	70.00	65.00	76.00
Std. Deviation	11.30276	9.68594	12.18036	12.97053	8.34371	6.82535
Variance	127.752	93.817	148.361	168.235	69.618	46.585
Skewness	-.211	.480	-.170	-1.428	-.716	-.319
Std. Err. of Skewness	.226	.226	.226	.226	.226	.226
Kurtosis	.942	-.175	-.208	6.205	2.880	.040
Std. Error of Kurtosis	.447	.447	.447	.447	.447	.447
Range	63.00	44.00	59.00	91.00	57.00	35.00
Minimum	35.00	50.00	34.00	.00	30.00	52.00
Maximum	98.00	94.00	93.00	91.00	87.00	87.00

a Multiple modes exist. The smallest value is shown

Table FII.21: Histograms of each module performance



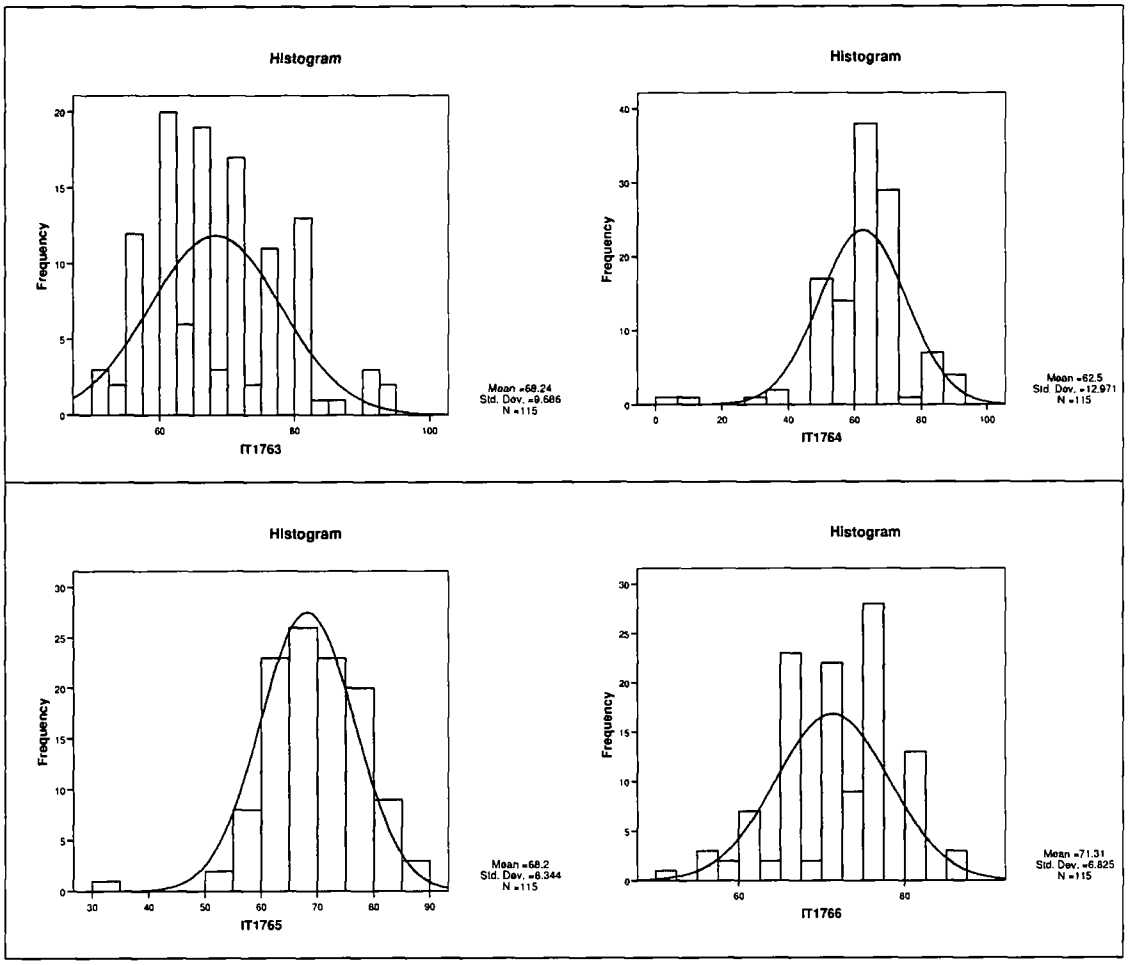


Table FI.22: Case 2006-S1 Detailed module performance for groups: 06, 07, 08, 09, 10

TUTGRP		IT1761	IT1763	IT1753	IT1764	IT1765	IT1766
EI0606	N	22	22	22	22	22	22
	Mean	71.0000	62.0455	66.0000	59.6818	65.3636	73.3182
	Std.Err. of Mean	2.63263	1.53386	2.26301	2.11712	1.48228	1.08987
	Median	70.5000	61.5000	66.0000	61.5000	65.0000	74.0000
	Mode	66.00	60.00(a)	77.00	51.00	65.00	66.00(a)
	Std. Deviation	12.34812	7.19442	10.61446	9.93017	6.95253	5.11195
	Variance	152.476	51.760	112.667	98.608	48.338	26.132
	Skewness	-.855	1.054	-1.121	-1.487	-.643	-.189
	Std. Error of Skewness	.491	.491	.491	.491	.491	.491
	Kurtosis	2.376	1.379	2.726	3.687	.250	-1.250
	Std. Error of Kurtosis	.953	.953	.953	.953	.953	.953
	Range	57.00	30.00	49.00	44.00	26.00	15.00
	Minimum	35.00	51.00	34.00	28.00	50.00	66.00
	Maximum	92.00	81.00	83.00	72.00	76.00	81.00
EI0607	N	24	24	24	24	24	24
	Mean	71.0833	68.0417	75.3750	63.1667	66.5417	74.7917
	Std. Error of Mean	2.59872	2.11660	2.51648	2.96090	2.17194	1.02678
	Median	71.0000	66.0000	78.0000	63.0000	66.0000	75.5000
	Mode	71.00	66.00	78.00(a)	60.00(a)	65.00	76.00
	Std. Deviation	12.73105	10.36918	12.32817	14.50537	10.6402	5.03016

Appendix F: Module group statistics

TUTGRP		IT1761	IT1763	IT1753	IT1764	IT1765	IT1766
	Variance	162.080	107.520	151.984	210.406	113.216 8	25.303
	Skewness	-.320	.456	-1.563	-1.900	-1.774	.152
	Std. Error of Skewness	.472	.472	.472	.472	.472	.472
	Kurtosis	2.237	.062	4.425	7.768	5.053	-.037
	Std. Error of Kurtosis	.918	.918	.918	.918	.918	.918
	Range	63.00	43.00	59.00	81.00	50.00	20.00
	Minimum	35.00	50.00	34.00	10.00	30.00	66.00
	Maximum	98.00	93.00	93.00	91.00	80.00	86.00
EI0608	N	24	24	24	24	24	24
	Mean	72.8750	74.1250	66.9167	68.2500	72.4167	67.0417
	Std. Error of Mean	2.25488	1.57000	2.45091	2.06001	1.71303	1.27045
	Median	72.0000	72.0000	65.0000	70.5000	75.0000	67.0000
	Mode	72.00	81.00	58.00	63.00(a)	76.00	66.00
	Std. Deviation	11.04659	7.69140	12.00694	10.09197	8.39211	6.22393
	Variance	122.027	59.158	144.167	101.848	70.428	38.737
	Skewness	-.094	.529	.217	-.050	-.244	-.371
	Std. Error of Skewness	.472	.472	.472	.472	.472	.472
	Kurtosis	-.384	-.496	-1.361	-.672	-.677	-.665
	Std. Error of Kurtosis	.918	.918	.918	.918	.918	.918
	Range	42.00	27.00	38.00	38.00	32.00	21.00
	Minimum	51.00	63.00	50.00	50.00	55.00	56.00
	Maximum	93.00	90.00	88.00	88.00	87.00	77.00
EI0609	N	22	22	22	22	22	22
	Mean	67.3636	67.0455	58.4545	57.5909	65.9545	67.5455
	Std. Error of Mean	1.43026	1.92104	1.72294	1.63784	1.13480	1.20376
	Median	68.5000	66.0000	57.5000	58.5000	65.0000	68.5000
	Mode	60.00	56.00(a)	53.00	50.00(a)	63.00(a)	71.00
	Std. Deviation	6.70853	9.01046	8.08130	7.68213	5.32270	5.64613
	Variance	45.004	81.188	65.307	59.015	28.331	31.879
	Skewness	-.252	.420	.367	-.736	-.010	-.238
	Std. Error of Skewness	.491	.491	.491	.491	.491	.491
	Kurtosis	-1.166	-.587	2.288	1.801	-.339	-1.220
	Std. Error of Kurtosis	.953	.953	.953	.953	.953	.953
	Range	22.00	32.00	41.00	35.00	19.00	18.00
	Minimum	55.00	53.00	39.00	36.00	56.00	58.00
	Maximum	77.00	85.00	80.00	71.00	75.00	76.00
EI0610	N	23	23	23	23	23	23
	Mean	69.1304	69.3913	71.6957	63.1739	70.3913	73.8261
	Std. Error of Mean	2.62122	2.14655	2.27847	3.76359	1.54733	1.63055
	Median	66.0000	67.0000	71.0000	65.0000	70.0000	75.0000
	Mode	65.00	60.00(a)	63.00	65.00	64.00(a)	80.00
	Std. Deviation	12.57091	10.29448	10.92718	18.04956	7.42073	7.81986
	Variance	158.028	105.976	119.403	325.787	55.067	61.150
	Skewness	-.001	.724	-.013	-1.881	.197	-.795
	Std. Error of	.481	.481	.481	.481	.481	.481

Appendix F: Module group statistics

TUTGRP		IT1761	IT1763	IT1753	IT1764	IT1765	IT1766
	Skewness						
	Kurtosis	-.214	.607	-.923	6.362	-.673	1.529
	Std. Error of Kurtosis	.935	.935	.935	.935	.935	.935
	Range	52.00	44.00	40.00	91.00	29.00	35.00
	Minimum	43.00	50.00	51.00	.00	56.00	52.00
	Maximum	95.00	94.00	91.00	91.00	85.00	87.00