

CONFERENCES IN RESEARCH AND PRACTICE IN  
INFORMATION TECHNOLOGY

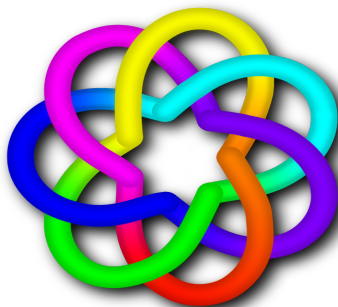
VOLUME 114

# COMPUTING EDUCATION 2011

AUSTRALIAN COMPUTER SCIENCE COMMUNICATIONS, VOLUME 33, NUMBER 2



AUSTRALIAN  
COMPUTER  
SOCIETY



 **CORE**  
*Computing Research & Education*



# COMPUTING EDUCATION 2011

Proceedings of the  
Thirteenth Australasian Computing Education Conference  
(ACE 2011), Perth, Australia,  
17-20 January 2011

John Hamer and Michael de Raadt, Eds.

Volume 114 in the Conferences in Research and Practice in Information Technology Series.  
Published by the Australian Computer Society Inc.



Published in association with the ACM Digital Library.

**Computing Education 2011.** Proceedings of the Thirteenth Australasian Computing Education Conference (ACE 2011), Perth, Australia, 17-20 January 2011

**Conferences in Research and Practice in Information Technology, Volume 114.**

Copyright ©2011, Australian Computer Society. Reproduction for academic, not-for-profit purposes permitted provided the copyright text at the foot of the first page of each paper is included.

Editors:

**John Hamer**

Department of Computer Science  
University of Auckland  
Private Bag 92019, Auckland  
New Zealand  
Email: [j.hamer@auckland.ac.nz](mailto:j.hamer@auckland.ac.nz)

**Michael de Raadt**

Department of Mathematics and Computing  
University of Southern Queensland  
Toowoomba, QLD 4350  
Australia  
Email: [deraadt@usq.edu.au](mailto:deraadt@usq.edu.au)

Series Editors:

Vladimir Estivill-Castro, Griffith University, Queensland  
Simeon J. Simoff, University of Western Sydney, NSW  
Email: [crpit@scm.uws.edu.au](mailto:crpit@scm.uws.edu.au)

Publisher: Australian Computer Society Inc.  
PO Box Q534, QVB Post Office  
Sydney 1230  
New South Wales  
Australia.

Conferences in Research and Practice in Information Technology, Volume 114.  
ISSN 1445-1336.  
ISBN 978-1-920682-94-1.

Printed, January 2011 by University of Western Sydney, on-line proceedings  
Document engineering by CRPIT  
CD Cover Design by Dr Patrick Peursum, Curtin University of Technology  
CD Production by Snap St Georges Terrace, 181 St Georges Terrace, Perth WA 6000,  
<http://www.stgeorges.snap.com.au/>

The *Conferences in Research and Practice in Information Technology* series disseminates the results of peer-reviewed research in all areas of Information Technology. Further details can be found at <http://crpit.com/>.

# Table of Contents

## Proceedings of the Thirteenth Australasian Computing Education Conference (ACE 2011), Perth, Australia, 17-20 January 2011

Preface .....	vii
Programme Committee .....	viii
Organising Committee .....	ix
Welcome from the Organising Committee .....	x
CORE - Computing Research & Education .....	xi
ACSW Conferences and the Australian Computer Science Communications .....	xii
ACSW and ACE 2011 Sponsors .....	xiv

## Contributed Papers

Mobile Messages as a Tool to Stimulate Learning Activities .....	3
<i>Shuk Ying Ho and Kai Wing Ho</i>	
Concrete and Other Neo-Piagetian Forms of Reasoning in the Novice Programmer .....	9
<i>Raymond Lister</i>	
Open Teaching — A Case Study Of Publishing Lecture Videos Publicly .....	19
<i>Richard Buckland</i>	
A Study of Loop Style and Abstraction in Pedagogic Practice .....	29
<i>David J. Barnes and Dermot Shinnners-Kennedy</i>	
Salient Elements in Novice Solutions to Code Writing Problems .....	37
<i>Jacqueline Whalley, Tony Clear, Phil Robbins and Errol Thompson</i>	
Wrong is a Relative Concept: Part Marks for Multiple-choice Questions .....	47
<i>Simon</i>	
How can Software Metrics Help Novice Programmers? .....	55
<i>Rachel Cardell-Oliver</i>	
Computer-supported Collaborative Learning System in Teaching E-commerce .....	63
<i>Eric W.T. Ngai, S.S. Lam and Jandia Kam-ling Poon</i>	
A Method for Analyzing Learning Outcomes in Project Courses .....	73
<i>Mattias Wiggberg and Mats Daniels</i>	
A Virtual Museum of Computing History .....	79
<i>Geoff Berry, Judy Sheard and Marian Quartly</i>	
Evaluating Students Perceptions of Peer Evaluation in IT Project Work .....	87
<i>Jandia K. L. Poon</i>	
Early Relational Reasoning and the Novice Programmer: Swapping as the Hello World of Relational Reasoning .....	95
<i>Malcolm Corney, Raymond Lister and Donna Teague</i>	

My Students Dont Learn the Way I Do .....	105
<i>Michael de Raadt and Simon</i>	
A Unit Testing Approach to Building Novice Programmers Skills and Confidence .....	113
<i>Jacqueline Whalley and Anne Philpott</i>	
Work-Integrated Learning in ICT Degrees .....	119
<i>Chris Pilgrim</i>	
How Active are Students in Online Discussion Forums? .....	125
<i>Dip Nandi, Margaret Hamilton, James Harland and Geoff Warburton</i>	
Helping First Year Novice Programming Students PASS .....	135
<i>Adrian Devey and Angela Carbone</i>	
Assessing Professional Skills in Engineering Education .....	145
<i>Mats Daniels, Åsa Cajander, Roger McDermott and Brian von Kinsky</i>	
MyPyTutor: an Interactive Tutorial System for Python .....	155
<i>Peter Robinson</i>	
A Pedagogically Rich Interactive On-line Learning Platform for Network Technology Students in Thailand .....	161
<i>Woratat Makasiranondh, S. P. Maj and David Veal</i>	
<b>Author Index</b> .....	169

## Preface

Welcome to the Thirteenth Australasian Computing Education Conference (ACE2011). This year, the ACE2011 conference, which is part of the Australasian Computer Science Week, is being held in Perth, Australia from 17-20 January, 2011.

The Chairs would like to thank the program committee for their excellent efforts in the double-blind reviewing process which resulted in the selection of 20 full papers from the 47 papers submitted, giving an acceptance rate of 42%. This is a large increase in submissions from the 30 papers last year, and reflects both the vibrancy of computing education in Australasia and the growing international status of the conference. We again see a strong international presence, with submissions from Australia, Fiji, Finland, Guam, Hong Kong, India, Iran, Japan, Morocco, New Zealand, Pakistan, Sri Lanka, Sweden, Thailand, UAE, United Kingdom, and Queensland.

Unfortunately we were not successful in bidding for an ACE invited keynote speaker to fill an ACSW plenary session this year. We are hopeful that next year we will receive priority. Due to the excellent response to our Call for Papers and the number of accepted papers this year, we have no invited speakers.

A variety of topics are presented in this year's papers, including: programmer education; open education; assessment; collaborative group work; tools; learning styles; work-integrated learning; and online education. Many of the papers have present new innovations, and many demonstrate high quality research.

As with past ACE conferences, we are continuing to hold workshops prior to ACE. Mikko Laakso (University of Turku, Finland) will share ViLLE with members of the Australasian Computing Education community. ViLLE is a freely available collaborative education platform for programming. Angela Carbone (Monash), Judy Sheard (Monash), Donald Chinn (University of Washington, USA) and Mikko Laakso will together run a second workshop aimed at establishing a collaboration to investigate the nature and composition of programming exam papers and to develop taxonomy of question types.

In addition, the conference includes presentations from the Nordic Network in Engineering Education Research, and Australian Council of Deans of Information and Communications Technology on their plans to establish an Academy to benefit learning and teaching of ICT in Australia.

We are grateful to SIGCSE for sponsoring the conference jointly with the ACM. We thank everyone involved in Australasian Computer Science Week for making this conference and proceedings publication possible, and we thank CORE, our hosts Curtin University of Technology, Perth, and the Australasian Computing Education executive for the opportunity to chair this ACE2011 conference.

**John Hamer**  
University of Auckland

**Michael de Raadt**  
University of Southern Queensland

ACE 2011 Programme Chairs  
January 2011

# Programme Committee and Additional Referees

## Chairs

John Hamer, University of Auckland, New Zealand  
Michael de Raadt, University of Southern Queensland, Australia

## Members

Angela Carbone, Monash University, Australia  
Gwyn Claxton, AUT University, New Zealand  
Nell Dale, University of Texas at Austin, USA  
Margaret Hamilton, RMIT University, Australia  
Jacqueline Whalley, AUT University, New Zealand  
Judy Kay, University of Sydney, Australia  
Judy Sheard, Monash University, Australia  
Josh Tenenberg, University of Washington, USA  
Mats Daniels, Uppsala University, Sweden  
Raymond Lister, University of Technology, Sydney, Australia  
Sally Fincher, University of Kent, UK  
Simon, University of Newcastle, Australia  
Tim Bell, University of Canterbury, New Zealand  
Tony Clear, AUT University, New Zealand

## Additional Reviewers

Åsa Cajander  
Adrian Devey  
Anne Philpott  
Helen Purchase  
James Harland  
John Hurst  
Laurence Hellyer  
Lin Wei  
Michael Smith  
Ron Addie  
Sally Firmin  
Theda Thomas

## Conference Webmaster

Michael de Raadt, University of Southern Queensland, Australia



# Organising Committee

## Chair

Assoc. Prof. Mihai Lazarescu

## Co-Chair

Assoc. Prof. Ling Li

## Finance

Mary Simpson  
Mary Mulligan

## Catering and Booklet

Mary Mulligan  
Dr. Patrick Puersum  
Assoc. Prof. Mihai Lazarescu

## Sponsorship and Web

Dr. Patrick Peursum  
Dr. Aneesh Krishna

## Registration

Mary Mulligan  
Dr. Patrick Puersum

## DVD and Signage

Dr. Patrick Peursum  
Mary Mulligan

## Venue

Dr. Mike Robey

## Conference Bag

Dr. Sieteng Soh

# Welcome from the Organising Committee

On behalf of the Australasian Computer Science Week 2011 (ACSW2011) Organising Committee, we welcome you to this year's event hosted by Curtin University. Curtin University's vision is to be an international leader shaping the future through its graduates and world class research. As Western Australia's largest university, Curtin is leading the state in producing high quality ICT graduates. At Curtin Computing, we offer both world class courses and research. Our Computing courses cover three key areas in IT (Computer Science, Software Engineering and Information Technology), are based on the curricula recommendations of IEEE Computer Society and ACM, the largest IT professional associations in the world, and are accredited by the Australian Computer Society. Curtin Computing hosts a top level research institute (IMPCA) and offers world class facilities for large scale surveillance and pattern recognition.

We welcome delegates from over 18 countries, including Australia, New Zealand, USA, U.K., Italy, Japan, China, Canada, Germany, Spain, Pakistan, Austria, Ireland, South Africa, Taiwan and Thailand. We hope you will enjoy the experience of the ACSW 2011 event and get a chance to explore our wonderful city of Perth. Perth City Centre is located on the north bank of the Swan River and offers many fun activities and a wealth of shopping opportunities. For panoramic views of Perth and the river, one can visit Kings Park or enjoy a relaxing picnic in one of the many recreational areas of the park.

The Curtin University campus, the venue for ACSW2011, is located just under 10km from the Perth City Centre and is serviced by several Transperth bus routes that travel directly between Perth and Curtin University Bus Station, as well as several other routes connecting to nearby train services.

ACSW2011 consists of the following conferences:

- Australasian Computer Science Conference (ACSC) (Chaired by Mark Reynolds)
- Australasian Computing Education Conference (ACE) (Chaired by John Hamer and Michael de Raadt)
- Australasian Database Conference (ADC) (Chaired by Heng Tao Shen and Athman Bouguettaya)
- Australasian Information Security Conference (AISC) (Chaired by Colin Boyd and Josef Pieprzyk)
- Australasian User Interface Conference (AUIC) (Chaired by Christof Lutteroth)
- Australasian Symposium on Parallel and Distributed Computing (AusPDC) (Chaired by Jinjun Chen and Rajiv Ranjan)
- Australasian Workshop on Health Informatics and Knowledge Management (HIKM) (Chaired by Ker-ryn Butler-Henderson and Tony Sahama)
- Computing: The Australasian Theory Symposium (CATS) (Chaired by Taso Viglas and Alex Potanin)
- Australasian Computing Doctoral Consortium (ACDC) (Chaired by Rachel Cardell-Oliver and Falk Scholer).

The nature of ACSW requires the co-operation of numerous people. We would like to thank all those who have worked to ensure the success of ACSW2011 including the Organising Committee, the Conference Chairs and Programme Committees, our sponsors, the keynote speakers and the delegates. Many thanks go to Alex Potanin for his extensive advice and assistance and Wayne Kelly (ACSW2010 chair) who provided us with a wealth of information on the running of the conference. ACSW2010 was a wonderful event and we hope we will live up to the expectations this year.

**Assoc. Prof. Mihai Lazarescu and Assoc. Prof. Ling Li**

Department of Computing, Curtin University

ACSW2011 Co-Chairs

January, 2011

# CORE - Computing Research & Education

CORE welcomes all delegates to ACSW2011 in Perth. CORE, the peak body representing academic computer science in Australia and New Zealand, is responsible for the annual ACSW series of meetings, which are a unique opportunity for our community to network and to discuss research and topics of mutual interest. The original component conferences ACSC, ADC, and CATS, which formed the basis of ACSWin the mid 1990s now share this week with six other events - ACE, AISC, AUIC, AusPDC, HIKM, ACDC, which build on the diversity of the Australasian computing community.

In 2011, we have again chosen to feature a small number of plenary speakers from across the discipline: Heng To Shen, Gene Tsudik, and Dexter Kozen. I thank them for their contributions to ACSW2011. I also thank the keynote speakers invited to some of the individual conferences. The efforts of the conference chairs and their program committees have led to strong programs in all the conferences again, thanks. And thanks are particularly due to Mihai Lazarescu and his colleagues for organising what promises to be a strong event.

In Australia, 2009 saw, for the first time in some years, an increase in the number of students choosing to study IT, and a welcome if small number of new academic appointments. Also welcome is the news that university and research funding is set to rise from 2011-12. However, it continues to be the case that per-place funding for computer science students has fallen relative to that of other physical and mathematical sciences, and, while bodies such as the Australian Council of Deans of ICT seek ways to increase student interest in the area, more is needed to ensure the growth of our discipline.

During 2010, CORE continued to negotiate with the ARC on journal and conference rankings. A key aim is now to maintain the rankings, which are widely used overseas as well as in Australia. Management of the rankings is a challenging process that needs to balance competing special interests as well as addressing the interests of the community as a whole.

CORE's existence is due to the support of the member departments in Australia and New Zealand, and I thank them for their ongoing contributions, in commitment and in financial support. Finally, I am grateful to all those who gave their time to CORE in 2010; in particular, I thank Alex Potanin, Jenny Edwards, Alan Fekete, Aditya Ghose, Leon Sterling, and the members of the executive and of the curriculum and ranking committees.

**Tom Gedeon**

President, CORE  
January, 2011

# ACSW Conferences and the Australian Computer Science Communications

The Australasian Computer Science Week of conferences has been running in some form continuously since 1978. This makes it one of the longest running conferences in computer science. The proceedings of the week have been published as the *Australian Computer Science Communications* since 1979 (with the 1978 proceedings often referred to as *Volume 0*). Thus the sequence number of the Australasian Computer Science Conference is always one greater than the volume of the Communications. Below is a list of the conferences, their locations and hosts.

**2012.** Volume 34. Host and Venue - RMIT University, Melbourne, VIC.

**2011. Volume 33. Host and Venue - Curtin University of Technology, Perth, WA.**

**2010.** Volume 32. Host and Venue - Queensland University of Technology, Brisbane, QLD.

**2009.** Volume 31. Host and Venue - Victoria University, Wellington, New Zealand.

**2008.** Volume 30. Host and Venue - University of Wollongong, NSW.

**2007.** Volume 29. Host and Venue - University of Ballarat, VIC. First running of HDKM.

**2006.** Volume 28. Host and Venue - University of Tasmania, TAS.

**2005.** Volume 27. Host - University of Newcastle, NSW. APBC held separately from 2005.

**2004.** Volume 26. Host and Venue - University of Otago, Dunedin, New Zealand. First running of APCCM.

**2003.** Volume 25. Hosts - Flinders University, University of Adelaide and University of South Australia. Venue - Adelaide Convention Centre, Adelaide, SA. First running of APBC. Incorporation of ACE. ACSAC held separately from 2003.

**2002.** Volume 24. Host and Venue - Monash University, Melbourne, VIC.

**2001.** Volume 23. Hosts - Bond University and Griffith University (Gold Coast). Venue - Gold Coast, QLD.

**2000.** Volume 22. Hosts - Australian National University and University of Canberra. Venue - ANU, Canberra, ACT. First running of AUIC.

**1999.** Volume 21. Host and Venue - University of Auckland, New Zealand.

**1998.** Volume 20. Hosts - University of Western Australia, Murdoch University, Edith Cowan University and Curtin University. Venue - Perth, WA.

**1997.** Volume 19. Hosts - Macquarie University and University of Technology, Sydney. Venue - Sydney, NSW. ADC held with DASFAA (rather than ACSW) in 1997.

**1996.** Volume 18. Host - University of Melbourne and RMIT University. Venue - Melbourne, Australia. CATS joins ACSW.

**1995.** Volume 17. Hosts - Flinders University, University of Adelaide and University of South Australia. Venue - Glenelg, SA.

**1994.** Volume 16. Host and Venue - University of Canterbury, Christchurch, New Zealand. CATS run for the first time separately in Sydney.

**1993.** Volume 15. Hosts - Griffith University and Queensland University of Technology. Venue - Nathan, QLD.

**1992.** Volume 14. Host and Venue - University of Tasmania, TAS. (ADC held separately at La Trobe University).

**1991.** Volume 13. Host and Venue - University of New South Wales, NSW.

**1990.** Volume 12. Host and Venue - Monash University, Melbourne, VIC. Joined by Database and Information Systems Conference which in 1992 became ADC (which stayed with ACSW) and ACIS (which now operates independently).

**1989.** Volume 11. Host and Venue - University of Wollongong, NSW.

**1988.** Volume 10. Host and Venue - University of Queensland, QLD.

**1987.** Volume 9. Host and Venue - Deakin University, VIC.

**1986.** Volume 8. Host and Venue - Australian National University, Canberra, ACT.

**1985.** Volume 7. Hosts - University of Melbourne and Monash University. Venue - Melbourne, VIC.

**1984.** Volume 6. Host and Venue - University of Adelaide, SA.

**1983.** Volume 5. Host and Venue - University of Sydney, NSW.

**1982.** Volume 4. Host and Venue - University of Western Australia, WA.

**1981.** Volume 3. Host and Venue - University of Queensland, QLD.

**1980.** Volume 2. Host and Venue - Australian National University, Canberra, ACT.

**1979.** Volume 1. Host and Venue - University of Tasmania, TAS.

**1978.** Volume 0. Host and Venue - University of New South Wales, NSW.

## Conference Acronyms

<b>ACDC</b>	Australasian Computing Doctoral Consortium
<b>ACE</b>	Australasian Computer Education Conference
<b>ACSC</b>	Australasian Computer Science Conference
<b>ACSW</b>	Australasian Computer Science Week
<b>ADC</b>	Australasian Database Conference
<b>AISC</b>	Australasian Information Security Conference
<b>AUIC</b>	Australasian User Interface Conference
<b>APCCM</b>	Asia-Pacific Conference on Conceptual Modelling
<b>AusPDC</b>	Australasian Symposium on Parallel and Distributed Computing (replaces AusGrid)
<b>CATS</b>	Computing: Australasian Theory Symposium
<b>HIKM</b>	Australasian Workshop on Health Informatics and Knowledge Management

Note that various name changes have occurred, which have been indicated in the Conference Acronyms sections in respective CRPIT volumes.

## ACSW and ACE 2011 Sponsors

We wish to thank the following sponsors for their contribution towards this conference.



CORE - Computing Research and Education,  
[www.core.edu.au](http://www.core.edu.au)



Perth Convention Bureau,  
[www.pcb.com.au](http://www.pcb.com.au)



**AUSTRALIAN  
COMPUTER  
SOCIETY**

Australian Computer Society,  
[www.acs.org.au](http://www.acs.org.au)



Curtin University of Technology,  
[www.curtin.edu.au](http://www.curtin.edu.au)



Association for Computing Machinery,  
[www.acm.org](http://www.acm.org)



ACM Special Interest Group on  
Computer Science Education,  
[www.sigcse.org](http://www.sigcse.org)

# CONTRIBUTED PAPERS





# Mobile Messages as a Tool to Stimulate Learning Activities

**Shuk Ying Ho**

School of Accounting and Business Information Systems  
Australian National University  
Canberra 0200, ACT, Australia  
susanna.ho@anu.edu.au

**Kevin K.W. Ho**

School of Business and Public Administration  
University of Guam  
Mangilao, Guam 96923  
kevinkho@ugam.uog.edu

## Abstract

Mobile learning (m-learning) is gaining significance. It has the potential to provide interesting and relevant learning experiences to learners. Firms and education institutes explore the use of mobile devices to facilitate teaching and apprehension. This study proposes to use one of the simple functions of mobile phones, i.e. text messaging, as a tool to stimulate learners to access course materials. This study also examines whether users' personalities influence their reactions to learning via text messaging. Specifically, we used a personality instrument, Myers-Briggs Type Indicator (MBTI), to categorise learners. We conducted an experiment with 78 students, who received text messaging to remind them to access lecture materials for two weeks. We found that extroverts more often followed text messages to participate in discussion learning activities than introverts, and students of a judging type who liked well-defined learning schedules often followed the text messages to work on their learning schedules. A learner's personality does have an effect on the amount of learning they are prepared to participate in, when they respond to mobile text messaging.

**Keywords:** learning, text messaging, learner's personality

## 1 Introduction

Firms increasingly need to offer flexible learning to their employees, so many have begun to deliver training programs via mobile devices such as mobile phones and personal digital assistants. Distance learning institutes send teaching materials to students' mobile devices so that the materials are accessible anywhere, anytime (e.g., travelling on a bus, sitting in a café). Sometimes, the mobile learning process can also be collaborative. That is, students can upload questions and opinions to a mobile site, and share their learning experiences. They can receive immediate feedback and guidance from

instructors who are simultaneously online. Incorporating mobile devices in the learning process results in a paradigm shift in the way individuals learn — from the classroom to the pocket. But so far, scant research has examined the effectiveness of using mobile devices to stimulate students to learn. This study aims at providing empirical evidence to evaluate the effectiveness of incorporating mobile devices in learning. In particular, we focus on a basic function of mobile phones, text messaging, and examine the effectiveness of using text messages to stimulate access to learning materials.

The first objective of this study is to examine the use of mobile text messaging to fully exploit its strength of stimulating students to access course materials. Technology designers sometimes forget the fundamentals in their eagerness to embrace technology. Although the term "m-learning" is now embedded in the university and corporate vernacular, most trainers find themselves doing little more than scattering mobile technologies within training programs at random, hoping that some of them will work. These technologies add little or even no value to the learning process (Roschelle 2003). Obviously, "build it and they will come" is not effortless. This research examines the effects of mobile text messages to stimulate learners to access learning materials.

The second objective is to explore the effects of mobile text messages on stimulating learners to access different types of learning methods (i.e., course discussions, course materials) (Jong and Ferguson-Hessler 1996). In particular, we look at how learners' personality influences their learning preferences. Prior work has developed "personalised" learning applications, but most applications *only* consider the difficulty of teaching materials. Smarter students take harder lectures, and weaker students take easier lectures. Educators generally believe that the psychology of human differences is fundamental to learning (Biggs 2001, Wan, Wang, and Haggerty 2008). Chen, Liu and Chang (2006) developed learning systems which personalised learning schedules for individual students. These studies support a general idea that learning environments will be more effective if they capitalise on the characteristics of both the learning tasks (i.e., participation in a discussion forum, or revising course materials) and the individual (i.e., personality).

In sum, we anticipate that positive learning outcomes will be achieved if the capabilities of m-learning (technology) match the delivery of different types of knowledge (task) for learners with different dispositions (individual). With the above objectives, we formulate a set of research questions:

*RQ1. What are the effects of mobile text messaging on stimulating learners to access various teaching content (e.g., course materials, discussion forums)?*

*RQ2. What is the role of personality in affecting learners' response to text messages?*

The rest of this paper is structured as follows: Section 2 provides a review on learning and learners' personalities. This helps us gain an understanding of personalising m-learning to learners' personalities. Section 3 presents an experiment and its findings. Section 4 discusses the practical implications of the work. Section 5 concludes our work.

## 2 Literature Review

Prior research has shown that web-based learning is effective for disseminating teaching materials online (Alavi and Leidner 2001, Benbunan-Fich 2002, Piccoli, Ahmad and Ives 2001, Narciss, Proske and Koerndle 2007, Shen, Callaghan and Shen 2008, Woo and Reeves 2007, Xu and Wang 2006). For example, Woo and Reeves (2007) studied whether the increase in interactions on web-based education platforms leads to meaningful learning outcomes. Narciss, Proske and Koerndle (2007) presented the result of a large-scale project that developed an authoring tool to support online teaching. Piccoli, Ahmad and Ives (2001) conducted an empirical study to examine the effects of web-based learning, including instructors' and students' performance, self-efficacy, and satisfaction. They also took different learning design, such as learning control and learning materials, into consideration when they developed the web-based learning system. These studies focus on web-based learning. Scant research examines the learning outcomes with mobile devices (Keegan 2002). There is little empirical work and no theoretical framework on mobile learning.

This research focuses on learning activities stimulated by mobile phone text messaging. That is, learners in this study use a web-based learning platform to access teaching materials. Even though the use of text messaging has been proposed for education purpose before (Lim, Hocking, Hellard, and Aiken 2008), our question is focused on whether mobile phone it can be used to stimulate learners to access the web-based learning platform more frequently? We also explore whether learners' personality types influence their responses to their mobile text messages. Most personalisation algorithms are based on individuals' past activities. Our work is a pioneering effort to use individual personality types to generate personalised content. We will use the Myers-Briggs Type Indicator (MBTI) to measure personality types, because it comprises four dichotomies

and these dichotomies cover the key aspects of how individuals perceive and judge information (Table 1). The MBTI has been used in traditional education research (Cooper 1991).

In this preliminary study, we focus on two out of four MBTI dimensions. They are extrovert-introvert, and judging-perceiving. Extrovert-introvert, explains where individuals put their attention and get their energy. Extroverts (E) like to spend time in the outer world of people and things, feel comfortable in groups and like working in them. Introverts (I) attend to their inner world of ideas, feel comfortable being alone and like things they can do on their own. Because extroverts like discussions and enjoy social interactions, we anticipate that extroverts are more likely to respond to mobile text messages that invited them to participate in discussion forums than introverts do.

*H1: After receiving a text message on course discussion participation, extroverts (compared with introverts) are more likely to follow the text message to access course discussion forums.*

	Characteristics	How Does It Relate to M-Learning?
Introversion-Extroversion (I-E)	This dichotomy explains where individuals put their attention and get their energy. Introverts (I) attend to their inner world of ideas, feel comfortable being alone and like things they can do on their own. Extroverts (E) like to spend time in the outer world of people and things, feel comfortable in groups and like working in them.	Discussion and group activities: I-learners prefer to work on individual tasks, whereas E-learners prefer group activities, sharing and discussions.
Sensing-Intuition (S-N)	This dichotomy explains how individuals process information. Sensing (S) type individuals pay attention to facts and physical reality; they start with facts and then form a big picture. Intuition (N) type individuals are interested in new things; they like to see the big picture then uncover the facts.	No covered in this study
Thinking-Feeling (T-F)	This dichotomy explains how individuals make human-related decisions. Thinking (T) type individuals use objective principles and impersonal facts to make decisions. Feeling (F) type individuals put more weight on personal concerns.	No covered in this study
Judging-Perceiving (J-P)	This dichotomy explains how individuals organise their lives. Judging (J) type individuals prefer a more structured and decided lifestyle. Perceiving (P) type individuals prefer a more flexible and adaptable lifestyle.	To personalise a learning schedule: J-learners prefer a well-defined learning timetable. P-learners prefer a checklist from which they can establish their own schedule.

**Table 1. Four Dichotomies in MBTI for Personalised Learning**

Judging-perceiving, explains how individuals organise their lives. Judging (J) type individuals prefer a more structured and decided lifestyle. Perceiving (P) type individuals prefer a more flexible and adaptable lifestyle. Presumably, judging learners prefer to follow the instructor's defined work plan to complete the learning materials. Conversely, perceiving learners like to set up their own plans. Hence, we anticipate that judging learners are more likely to respond to mobile text messages that invite them to revise the course materials than perceiving learners would.

*H2: After receiving a text message on course material revision, judging learners (compared with perceiving learners) are more likely to follow the text messages to revise the required learning materials.*

### 3 Methodology

#### 3.1 Participants

Our target participants were university students from an introductory course on management information systems. We believe that these university students are authenticated participants because they were learners. Generally they accessed their lecture materials on WebCT – a teaching and learning management platform adopted by the university. The experiment spanned two weeks. We recruited 78 volunteers (32 males and 46 females; average age = 20 years) to participate in this preliminary test, which was equivalent to 76% of the students who enrolled in the captioned course. All of them were active mobile phone users. During the experiment period, we sent 468 SMS recommendations to the 78 participants. Each participant received six SMS learning reminders (three reminders per week) to invite them to access the WebCT course materials.

#### 3.2 Procedures

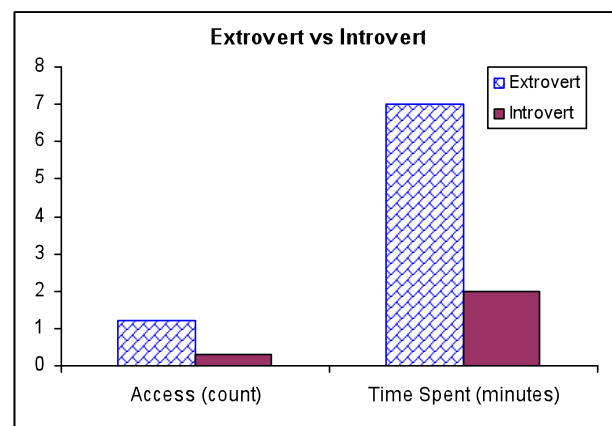
At the beginning of the study, each participant filled in a demographic questionnaire and took an MBTI test. After the registration, the participants received a text messaging reminder on Monday, Wednesday, and Friday. In the first week, they received text messages with content, “content, on XXX topic, is available on WebCT. It is important. Please read.” In the second week, they received text messages with content, “Your lecturer has posted messages in the discussion forum. Enjoy.” We tracked the participants' logon and page access on WebCT.

#### 3.3 Findings

We collected the participants' MBTI scores at the beginning of the study. It took 45 minutes for participants to complete the MBTI questionnaire. The MBTI questionnaire measured four dimensions of an individual's personality: extrovert-introvert, sensing-intuition, thinking-feeling, and judging-perceiving. In this study, we only focused on extrovert-introvert, and judging-perceiving.

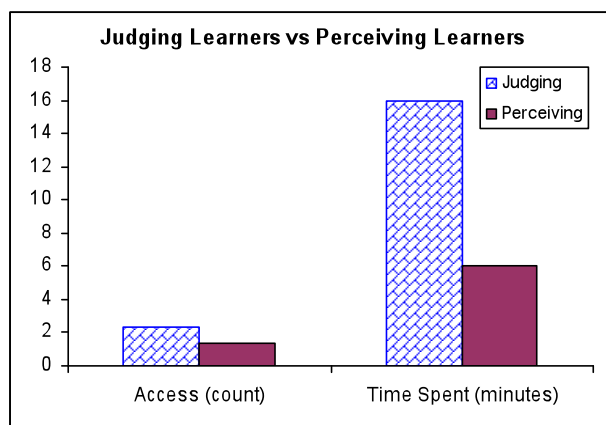
The introvert (0) to extrovert (16) scores also ranged from 0 to 16. The mean of the introvert-extrovert scores for our participants was 9.2. We used a median split to divide our participants into two groups: extrovert learners and

introvert learners. There were 39 introvert learners and 39 extrovert learners. We analysed their access of WebCT discussion forums within one day of sending out the text messages. We collected two data: (1) the number of accesses of discussion forums; and (2) the length of their reading time of discussion forums. Figure 1 presents the descriptive statistics. We conducted two ANOVA tests. The dependent variable of the first test was the number of accesses. On average, there were 1.2 accesses by extrovert learners and 0.3 accesses by introvert learners ( $p < 0.01$ ). The dependent variable of the second test was the amount of time spent on the discussion forums (in minutes). On average, extroverts spent 7 minutes on the discussion forums, and introverts spent 2 minutes ( $p < 0.05$ ). Both dependent variables supported H1, i.e., the results showed that extrovert learners were more likely to follow the reminder to access discussion forums than introvert learners.



**Figure 1. Learning Activities by Extroverts and Introverts**

The perceiving (0) to judging (16) scores also ranged from 0 to 16. The mean of the judging-perceiving scores for our participants was 11.7. We used a median split to divide our participants into two groups: perceiving learners and judging learners. There were 37 perceiving learners and 39 judging learners. Because some learners had equal scores, the two group sizes were not even. We analysed their accesses of WebCT within one day of sending out the text messages. We also collected two data: (1) the number of WebCT logons; and (2) the length of their reading time of the lecture materials. Figure 2 presents the descriptive statistics. We conducted another two ANOVA tests. The dependent variable of the third test was the number of logons. On average, there were 2.3 logons by judging learners and 1.4 logons by perceiving learners ( $p < 0.05$ ). The dependent variable of the fourth test was the length of logon sessions (in minutes). On average, there were 16 minutes by judging learners and 6 minutes by perceiving learners ( $p < 0.01$ ). Both dependent variables supported H2, i.e., judging learners are more likely to follow the reminder to study the required learning materials than perceiving learners.



**Figure 2. Learning Activities by Judging Learners and Perceiving Learners**

## 4 Discussions, Limitations and Future Research

### 4.1 Discussion

This research will provide several vital outcomes for firms, educators and technology designers.

First, our findings will help lecturers understand the teaching opportunities associated with mobile text messaging. It also provides an opportunity for firms to disseminate teaching messages to their employees. Competition within the business world is keen and employee training is a key attraction and retention strategy. In the period 2001 to 2005, the number of employees in Australia completing training courses increased from 4.8 million (42%) to 5.3 million (48%). This statistic indicates how crucial staff training is to firms. At the same time, more and more employees work at home. According to the Australian Bureau of Statistics, in 2006<sup>1</sup> 2.3 million employees (24% of 9.4 million) worked some hours at home; 8% of employees *only* worked at home; and 36% spent time travelling for work. The number of home workers and travel workers is increasing, but how can firms provide training to these employees? In this environment, mobile training is an increasingly important business development strategy. This study provides a first step to understanding how to use mobile devices, even a very simple function (such as text messaging) to stimulate people to learn. Our findings provide instructors with ideas on how learners react with learning text messaging.

Second, this project helps educators (e.g., in high schools, TAFEs and universities) understand how to leverage the potential of the mobile communication channel. Nowadays, teaching is not easy. According to a report by the University of Melbourne, full-time university students spend more time in paid work, and fewer of them study on weekends. The report also finds that it is harder to motivate students to learn. This research explores a new pedagogical practice — shifting

the teaching platform to mobile devices — which may lead to affective learning. M-learning delivers teaching content to an identifiable device; hence, it can be personalised. For instance, teaching materials, training schedules and quizzes can be tailored for individuals and even matched with their personality types. Learners can receive reminders via SMS. This study provides preliminary findings on the effects of personalised m-learning.

Last, this research provides information to technology designers on the effectiveness of an m-learning function: personalisation. Personalisation has been widely adopted in electronic commerce (Tam and Ho 2005, 2006). This study is one of the pioneer efforts to examine the use of personalisation in learning. Our findings confirm that personalised learning should take users' characteristics, i.e., personalities, into consideration. This study examines whether users' MBTI personality types influence their learning preferences. In the future, more research efforts should be invested in exploring the effects of various user characteristics or traits on users' learning preferences.

### 4.2 Limitations and Future Research

There are some limitations for this study. First, as it is a preliminary test, our experiment spanned only for two weeks. It is not sure whether the impacts that we observed in this study would have a long time effect. Hence, as the first possible extension of this research, we plan to conduct a longitudinal study to explore whether they would be a diminishing effects of mobile text messaging on stimulating learners' to access various teaching contents.

Second, we only explored the impacts of personality on learners' response to text messages for two out of four MBTI dimensions. To gain a better understanding of the impacts of different personalities on the learners' response to text messages, we plan to investigate the impacts of the remaining two MBTI dimensions, i.e. Sensing-Intuition and Thinking-Feeling.

## 5 Conclusion

Mobile devices will play a significant role in future education. They have the potential to provide interesting and relevant learning experiences to students. Firms and education institutes explore the use of mobile devices to facilitate teaching and learning. In this study, we examined the effectiveness of text messaging on stimulating students' learning activities. Specifically, we examined whether learners' personalities (extrovert vs. introvert, judging vs. perceiving) influence their reactions to text messaging of learning. With an experiment including 78 students, we confirmed that extrovert students were more interested in text messages on class discussions, and judging students preferred to have text messages to remind them of their learning schedules.

## 6 References

- Alavi, M., and Leidner, D.E. (2001) Technology-Mediated Learning-A Call for Greater Depth and Breadth of Research. *Information Systems Research*, 12(1), pp.1-10.

<sup>1</sup> See:

<http://www.abs.gov.au/AUSSTATS/abs@.nsf/Lookup/6275.0Main+Features1Nov%202005?OpenDocument>  
online available on 1 February 2009.

- Benbunan-Fich, R. (2002) Improving Education and Training with IT. *Communications of the ACM*, 45(6), pp.94-99.
- Biggs, J. (2001) Enhancing Learning: A Matter of Style or Approach? In R.J. Sternberg and L.F. Zhang, editors, *Perspectives on Thinking, Learning and Cognitive Styles*. Lawrence Erlbaum: NJ, pp.73-102.
- Chen, C., Liu, C. and Chang, M. (2006) Personalized Curriculum Sequencing Utilizing Modified Item Response Theory for Web-Based Instruction, *Expert Systems with Applications*, 30(2), pp.378-396.
- Cooper, S.E. (1991) MBTI Learning Style-Teaching Style Discongruencies, *Educational and Psychological Measurement*, 51(3), pp.699-706.
- Jong, T.D., and Ferguson-Hessler, M.G.M. (1996) Types and Qualities of Knowledge, *Journal Educational Psychologist*, 31(2), pp.105-113.
- Keegan, D. (2002) The Future of Learning: From eLearning to mLearning, Online available on 15 Janurary 2009: [http://www.fernuni-hagen.de/ZIFF/ZP\\_119.pdf](http://www.fernuni-hagen.de/ZIFF/ZP_119.pdf).
- Lim, M.S.C., Hocking, J.S., Hellard, M.E., and Aitken, C.K. (2008) SMS STI: A review of the use of mobile text messaging in sexual health, *International Journal of STD & AIDS*, 19(5), pp. 287-290.
- Narciss, S., Proske, A., and Koerndle, H. (2007) Promoting Self-Regulated Learning in Web-Based Learning Environment, *Computers in Human Behavior*, 23(3), pp.1126-1144.
- Piccoli, G., Ahmad, R. and Ives, B. (2001) Web-Based Virtual Learning Environments: A Research Framework and A Preliminary Assessment of Effectiveness in Basic IT Skills Training, *MIS Quarterly*, 25(4), pp.401-426.
- Roschelle, J. (2003) Unlocking the Learning Value of Wireless Mobile Devices, *Journal of Computer Assisted Learning*, 19(3), pp.260-272.
- Shen, L., Callaghan, V., and Shen, R. (2008) Affective E-Learning in Residential and Pervasive Computing Environments, *Information Systems Frontiers*, 10(4), pp.61-472.
- Tam, K.Y., and Ho, S.Y. (2005) Persuasiveness of Web Personalization: An Elaboration Likelihood Model Perspective, *Information Systems Research*, 16(3), pp.271-291.
- Tam, K.Y., and Ho, S.Y. (2006) Understanding the Impact of Web Personalization on User Information Processing and Decision Outcomes, *MIS Quarterly*, 30(4), pp.865-890.
- Wan, Z., Wang, Y., and Haggerty, N. (2008) Why People Benefit from E-Learning Differently: The Effects Of Psychological Processes on E-Learning Outcomes, *Information and Management*, 45(8), pp.513-521.
- Woo, Y. and Reeves, T.C. (2007) Meaningful Interaction in Web-Based Learning: A Social Constructivist Interpretation, *The Internet and Higher Education*, 10(1), pp.15-25.
- Xu, D., and Wang, H. (2006) Intelligent Agent Supported Personalization for Virtual Learning Environments, *Decision Support Systems*, 42(2), pp.825-843.

## 7 Acknowledgement

This project was partially supported by the Australian Research Council (Project ID: DP0879736).



# Concrete and Other Neo-Piagetian Forms of Reasoning in the Novice Programmer

Raymond Lister

Faculty of Engineering and Information Technology  
University of Technology, Sydney  
Sydney,  
NSW, Australia

Raymond.Lister@uts.edu.au

## Abstract

This paper brings together a number of empirical research results on novice programmers, using a neo-Piagetian theoretical framework. While there already exists literature connecting programming with classical Piagetian theory, in this paper we apply neo-Piagetian theory. Using that neo-Piagetian perspective, we offer an explanation as to why attempts to predict ability for programming via classical Piagetian tests have yielded mixed results. We offer a neo-Piagetian explanation of some of the previously puzzling observations about novice programmers, such as why many of them make little use of diagrams, and why they often manifest a non-systematic approach to writing programs. We also develop the relatively unexplored relationship between concrete operational reasoning and programming, by exploring concepts such as conservation and reversibility.

**Keywords:** Neo-Piagetian, Novice Programmer.

## 1 Introduction

After 30 years of teaching computer science and software engineering, at elite institutions like London's Imperial College, Kramer (2007) articulated a common lament of computing academics everywhere:

*Why is it that some software engineers and computer scientists are able to produce clear, elegant designs and programs, while others cannot? Is it possible to improve these skills through education and training?* (p. 37).

Kramer went on to claim that *critical to these questions is the notion of abstraction*. In his Turing Award Lecture, Dijkstra (1992) offered an explanation as to why the ability to abstract is important for programmers:

*... the only mental tool by means of which a very finite piece of reasoning can cover a myriad cases is called 'abstraction'; as a result the effective exploitation of his powers of abstraction must be regarded as one of the most vital activities of a competent programmer.* (p. 864).

Both Kramer and Dijkstra were expressing a widespread belief in the computing community. For example, the ACM's Computing Curricula 1991 (Turner, 1991) includes the statement that "*... the process of abstraction will normally be prominent in all undergraduate curricula*".

In his paper, Kramer drew upon the work of Jean Piaget, who developed a theory about the different levels of abstract reasoning exhibited by people, especially as they mature from child to adult. To perform the abstractions that Kramer and Dijkstra describe, a person must achieve the most abstract of Piaget's levels, which is called formal operational reasoning, but Kramer cites well known research claiming that only 30–35% of adolescents have achieved that stage, and some adults never achieve that stage. Kramer went on to advocate that students who apply to study computing at university should be tested for abstraction ability, and those students who test poorly should be refused admission to computing.

In this paper, we further explore the link between Piagetian theory and programming. Unlike most of the earlier work that has also explored that link, our perspective is not based upon classical Piagetian theory (i.e. as articulated by Piaget), but instead is based upon neo-Piagetian theory. The distinction between classical and neo-Piagetian theory is described in the next subsection of the introduction.

In section 2, we describe formal operational reasoning, the most abstract type of Piagetian reasoning, and the type most commonly discussed in the literature that connects programming with classical Piagetian theory. We review and re-evaluate that literature from a neo-Piagetian perspective.

In section 3, we describe preoperational reasoning, which is the least abstract type of Piagetian reasoning discussed in this paper. We reinterpret, in preoperational terms, some of the existing literature on novice programmers, which has not previously been linked to Piaget.

In section 4, we look at concrete operational reasoning, the middle level of the three Piagetian levels discussed in this paper, which is more abstract than preoperational reasoning but less abstract than formal operational reasoning. The relationship between programming and concrete operational reasoning has not yet been well explored, especially from a neo-Piagetian perspective, and the primary novel contribution of this paper is to explore that relationship.

## 1.1 Classical Piagetian vs. Neo-Piagetian

Classical Piagetian theory is focussed upon the intellectual development of the child. In Piaget's view, children exhibit increasingly abstract forms of reasoning because of the biological maturation of the brain. Thus, in Piaget's view, a child who exhibits a certain level of abstract reasoning on a given problem will tend to exhibit that same level of abstract reasoning on many other problems. Subsequent work in psychology, however, has shown that children (and adults) exhibit different levels of abstract reasoning on different problems. Some other aspects of classical Piagetian theory have also been cast into doubt (Smith, 1992).

This paper is based upon neo-Piagetian theory. While the types of abstract reasoning are broadly the same in classical and neo-Piagetian theory, the principle difference in neo-Piagetian theory is that people, *regardless of their age*, are thought to progress through increasingly abstract forms of reasoning *as they gain expertise in a specific problem domain*. Thus a person who is a novice in one domain (e.g. chess) will exhibit less abstract forms of reasoning than that same person will exhibit in a domain where he/she is expert (e.g. calculus). For a comprehensive review of neo-Piagetian theories, see Morra *et al.* (2007).

Most neo-Piagetian theorists argue that the increase in abstraction is not a consequence of biological maturity, but instead is due to an increase in the effective capacity of working memory. It is well known that working memory has a very small capacity — seven plus or minus two is the popularly known estimate (Miller, 1956). Despite this severe working memory limitation, people can routinely handle more data because of 'chunking'. That is, if a set of associated data items are already stored in long term memory, they may be retrieved and used in working memory as if they were a single item. For example, remembering a new telephone number for a few seconds will consume working memory capacity, as each digit forms one data item to be stored in working memory. However, once a specific phone number has been committed to long term memory, then it only counts as one data item when it is brought back into working memory. Thus, the well known limitations of working memory do not apply to all data, but just to data that has not yet been learnt.

In classical Piagetian theory, it is customary to talk of a person as being in a particular Piagetian developmental stage of reasoning. Many neo-Piagetians (e.g. Biggs and Collis 1982) prefer to describe a person as exhibiting a particular level of abstract reasoning as he/she works on a specific problem — as we will do in this paper.

## 1.2 The SOLO taxonomy

Our motivation for exploring the implications of neo-Piagetian theory stems from the use of the SOLO taxonomy (Biggs and Collis, 1982) in the BRACElet project. The SOLO taxonomy was inspired by neo-Piagetian theory, so we thought it might be useful to explore the relationship between neo-Piagetian theory and programming.

One of the earliest BRACElet papers to use the SOLO taxonomy (Whalley *et al.*, 2006) reported on the

performance of students in an end-of-first-semester exam. As part of that exam, the students were given a question that began "*In plain English, explain what the following segment of Java code does*". Whalley *et al.* found that some students responded with a correct, line-by-line description of the code while other students responded with a correct summary of the overall computation performed by the code (e.g. "*the code checks to see if the elements in the array are sorted*"). A line-by-line description is a SOLO multistructural response, while a correct summary of the overall computation is a SOLO relational response. A relational response is more abstract than a multistructural response.

In another BRACElet study, Lopez *et al.* (2008) analysed the performance of students in an end-of-semester programming exam. They found that the combination of student scores on tracing tasks and "explain in plain English" tasks accounted for 46% of the variance on a code writing task. Three subsequent studies have reported similar results (Lister, Fidge and Teague, 2009; Venables, Tan and Lister, 2009; and Lister *et al.*, 2010).

## 2 Formal Operational Reasoning

Formal operational reasoning is the most abstract of the Piagetian types of reasoning. We describe this type of reasoning first because (1) it is how expert programmers reason, and (2) in the literature connecting Piaget and programming, formal operational reasoning has received more attention than other forms of Piagetian reasoning.

### 2.1 General Description

A person reasoning at this level exhibits the thinking characteristics traditionally emphasized at university (or at least we academics like to think so). A person thinking formally can reason logically, consistently and systematically. Piaget nominated sixteen combinatorial outcomes of binary propositions (e.g. "p or q", "p implies q", "p is equivalent to q") that he felt were used in formal reasoning, even when the formal reasoning is expressed implicitly in natural language, and not expressed explicitly in propositional logic. Formal operational reasoning also requires a reflective capacity — the ability to think about one's own thinking. Formal operational thinking can involve reasoning about hypothetical situations, or at least reasoning about situations that have never been directly experienced by the thinker. It also involves an awareness of what is known for certain, and what is known with some probability of being true, which in turn allows someone who is thinking formally to perform hypothetico-deductive reasoning — that is, the making of a tentative inference from incomplete data, then actively, systematically seeking further data to confirm or deny the tentative inference. For a more detailed description of formal operational reasoning, see Brainerd (1978) or Flavell (1977).

### 2.2 Formal Reasoning in Programming

Writing programs is frequently referred to as an exercise in problem solving. McCracken *et al.* (2001) defined problem solving as a five step process: (1) abstract the



problem from its description, (2) generate subproblems, (3) transform subproblems into subsolutions, (4) recompose, and (5) evaluate and iterate. Similarly, Fischer (1986) nominated top down design as requiring formal operational reasoning:

*The student must be able to see a problem as a statement summarizing a set of inter-related but unstated components, some of which are given, others probable, and still others are merely possible ... one must be able to conceive of all the possible steps or parts of each module or problem, and be able to determine in what order they must occur.*

### 2.2.1 Predictors of Programming Ability

Almost all the literature that connects Piaget and programming does so in the context of attempting to develop a predictor of programming ability, by using a Piagetian test of reasoning. Various authors have reported mixed results:

- Kurtz (1980) constructed a 15 item test of Piagetian reasoning, where each item was taken from a previously published study. He compared the performance of 23 students on that test with their final grade in an introductory programming course, and reported an  $R^2$  of 0.63, which we regard as very high. However ...
- When Barker and Unger (1983) carried out a similar experiment to Kurtz, using 11 of the items from Kurtz's test, their results for a larger population of 353 students "*failed to produce the spectacular ... correlation*" (p. 156) found by Kurtz.
- Fischer (1986) used a previously published test of Piagetian reasoning and found that 91% of students from an introductory programming course who received a course grade of B+ or higher were classified as "formal operational thinkers".
- Cafolla (1988) performed a linear regression between student performance on a previously published test of formal operational reasoning and their performance in the final exam of an introductory programming course. He reported an  $R^2$  of 0.35, which we regard as low.
- As a binary measure of successful/unsuccessful in learning to program, Hudak and Anderson (1990) used a criterion of a final course grade of 80% or higher in a CS1 course. They were able to correctly classify 72% of successful/unsuccessful students, using a previously published test of formal operational reasoning.
- Bennedsen and Caspersen (2006) used a classic Piagetian experiment – the pendulum test – to estimate the formal operational reasoning capacity of programming students, but they found that this estimate did not correlate well with the students' final CS1 grade.

The mixed results reported in the above works may be due to the classical Piagetian perspective adopted in those works. Recall that the neo-Piagetian perspective is that the level of abstract reasoning an individual manifests varies between problem domains. From that neo-Piagetian perspective, there is no reason to expect that a

person's ability on a non-programming test of abstract reasoning should correlate with that person's ability at programming.

### 2.2.2 Grading: Assignments and Exams

In the above reported mixed results, while several of the Piagetian tests used by those authors had been previously validated, the authors made no attempt to validate (and in some cases, did not even describe) the methods used to produce the student grades. Therefore, one possible explanation of the above mixed results is that the grading approaches used are measuring different programming skills, possibly skills at different levels of Piagetian reasoning. (As an aside, we express surprise at the confidence most academics have in their respective grading schemes).

One grading issue is the relative weighting given to assignments and exams. The code students are required to write in assignments is generally more complicated than the code they write in exams. Therefore, in general, the demands placed upon a novice's problem solving skills are greater for assignments than for exam questions – that is, formal operational reasoning may be tested by assignments more than exams.

On the other hand, sometimes a process that combines quasi-random code changes and copious trial runs (a process which is decidedly not formal operational reasoning) can produce a poor but passing assignment solution, while the student writing code in a paper-based exam does not have that luxury. On the other hand again, exam marking can be generous. Sometimes a student's code is well rewarded even though it bears only a superficial appearance to correct code. For example, Traynor, Bergin, and Gibson (2006) provided an illuminating extract from an interview with a student, where the student described his approach to answering coding questions in an exam, when he didn't really know the answer:

*... you usually get the marks by making the answer look correct. Like, if it's a searching problem, you put down a loop and you have an array and an if statement. That usually gets you the marks ... not all of them, but definitely a pass".*

Dressel (1983) described the grades we give students (in any discipline) as

*"... an inadequate report of an inaccurate judgement by a biased and variable judge of the extent to which a student has attained an undefined level of mastery of an unknown proportion of an indefinite material."*

To summarize this brief discussion on grading, it is difficult to infer that a student has used formal operational reasoning to produce a piece of code, when the only supporting evidence is the code itself – or worse, a non-validated grade. Formal operational reasoning is more about the mental process rather than the final product.

### 2.2.3 Testing and Debugging

The process of programming involves a great deal of testing and debugging. As part of advocating that novice

programmers should be discouraged from a trial and error approach to programming, and instead encouraged to adopt a reflection-in-action approach, Edwards (2004) recommended that novice programmers needed...

*"... practice in hypothesizing about the behavior of their programs and then experimentally verifying (or invalidating) their hypotheses. ... These activities are at the heart of software testing."* (p. 27)

What Edwards was describing is hypothetico-deductive reasoning, a signature skill of formal operational reasoning.

In a book entitled simply *"Debugging"*, Agans (2006) describes a number of approaches to debugging, including *"Make it Fail"* and *"Divide and Conquer"*. A programmer who is thinking in a formal operational manner routinely applies general debugging principles like these. Furthermore, while these approaches are applicable to programming, Agans' book is about debugging anything (e.g. an electrical appliance) and thus his book illustrates another aspect of formal operational thinking – the ability to choose the most appropriate abstract concept to use in a specific case.

### 2.3 A Closing Remark

Formal operational reasoning is how we would like our students to go about writing programs, and how they eventually may go about writing programs, but neo-Piagetian theory tells us that most novices will not immediately begin to think this way about programs. Instead, novices progress toward formal operational reasoning, via less sophisticated forms of reasoning. In the next section, we consider how novices first think about programs.

## 3 Preoperational Reasoning

Preoperational reasoning is the least abstract form of Piagetian reasoning discussed in this paper. From a neo-Piagetian perspective, most novices in any problem domain begin with this type of reasoning. Therefore, even though some novice programmers may progress quickly to more sophisticated forms of reasoning, we should expect to see preoperational reasoning from most novices when they first begin to program.

### 3.1 General Description

A person who is thinking in a preoperational mode tends not to form many abstractions from the objects in the problem environment. Their thinking reflects the direct manipulations they could make to that environment. There is little thinking about the relationships between the objects. To the limited extent that preoperational thinking abstracts beyond the actual objects, the thinking is not systematic. Also, the thinking tends to focus on only one abstract property at any given moment in time, and when more than one abstract thought occurs over time those abstractions are not coordinated, and may be contradictory. For a more detailed description of classical preoperational reasoning, see Brainerd (1978) or Flavell (1977).

From the neo-Piagetian perspective, the low level of abstraction in preoperational thinking, on a particular

problem, is a consequence of the novice's working memory being overwhelmed, since the novice has not yet learnt to chunk knowledge and information in that problem domain.

### 3.2 Preoperational Reasoning in Programming

In this section, we will describe novice programmer behaviours that are the staple diet of conversations among academics who teach programming – often exasperated and incredulous conversations.

An attractive quality of neo-Piagetian theory is that it makes dealing with these behaviours less exasperating for the teacher. Neo-Piagetian theory allows us to see that these behaviours are not the manifestation of cognitive dysfunction by a student, nor are the behaviours due to mental laziness. Instead these behaviours are a normal stage of cognitive development.

#### 3.2.1 Tracing Without Abstracting Meaning

Preoperational students can trace code. That is, they can manually execute a piece of code and determine the values in the variables when the execution is finished. Research suggests that novices need to be able to trace with >50% accuracy before they can begin to understand code (Philpott, Robbins and Whalley, 2007; Lister, Fidge and Teague, 2009; Venables, Tan and Lister, 2009). Students who trace code with less than 50% accuracy are operating at a lower Piagetian level than preoperational, which is called the sensorimotor level, but we do not discuss that level any further in this paper.

A defining characteristic of preoperational reasoning in programming is that, while such a novice can reliably trace code, that novice does not routinely abstract from the code to see a meaningful computation performed by that code. If pressed by their teacher to offer a meaningful computation performed by a piece of code, the novice who is thinking preoperationally may make a reasonable inductive guess, based upon the input/output behaviour they observe from tracing the code, but that novice will not infer the computation deductively, from the code itself. For the novice who is thinking preoperationally, the lines in a piece of code are only weakly related.

The ITiCSE 2004 "Leeds" working group (Lister *et al.*, 2004) collected data from some end-of-first-semester students, using a think-out-loud protocol. Eight of those students could trace code reliably, and answer questions about what values were in the variables after the code had finished executing, but...

*"... While working out their answer, none of these students volunteered any realization of the intent of the code, to count the number of identical elements in the two arrays ..."* (p. 138)

In contrast, when Lister *et al.* (2006) gave the same problem to several expert programmers, those experts tended to avoid tracing. Instead the experts first read the code to deduce what it did – the code, as indicated in the above quote, counted the number of identical elements in two sorted arrays. Having deduced what the code did, the experts then simply counted the number of common elements in the two arrays, and did not hand execute the code.

### 3.2.2 Diagrams

Since novice programmers who reason preoperationally tend not to abstract, and when they do abstract they tend to not be systematic, these novices struggle to make effective use of diagrammatic abstractions of code. Thomas, Ratcliffe, and Thomasson (2004) wrote despairingly of their frustrations at trying to get their novices to make effective use of diagrams:

*... when they might appropriately use [diagrams] themselves, weaker students fail to do so. They are often impatient when the instructor resorts to drawing a diagram, then amazed that the approach works. ... [also] providing [students] with what we considered to be helpful diagrams did not significantly appear to improve their understanding .... This was completely unexpected. We thought that we were 'practically doing the question for them'...*

Lister (2007) reported a related experience. He identified a group of students in his class who were adept at tracing code, but who could not make use of diagrams. Specifically, in the end-of-first-semester exam, he provided the students with code that implemented algorithms that the class had studied during the semester. One or two lines of code were omitted from each piece of code, and the students had to choose the missing lines in multiple choice questions. He provided diagrams of each algorithm tested, but the “middle novice programmers” (as he called them in the paper) struggled to use the diagrams to choose the correct answer.

#### 3.2.2.1 Doodling

The ITiCSE 2004 “Leeds” working group (Lister *et al.*, 2004) collected “doodles” from their end-of-first-semester students. Doodles are, as Lister *et al.* defined the term, the diagrams and other annotations that programmers make as they reason about a piece of code. They looked at the types of doodles generated by their novices on two questions. For one of those questions, one fifth of the students made no doodles at all. For the other question, more than one half of the students made no doodles at all. Lister *et al.* were surprised, but their finding makes sense from a neo-Piagetian perspective, since novices who are reasoning preoperationally lack the mental abstractions to make doodling useful to them.

### 3.2.3 Code Explanation

The neo-Piagetian perspective, and in particular what is known about preoperational reasoning, provides an explanation as to why the BRACElet project has found (as reported in the introduction) that some students struggle with “explain in plain English” tasks. The explanation is based around the neo-Piagetian idea that the working memory of these novices is easily overloaded, as they lack the knowledge structures to “chunk” the code.

First, let us consider a particularly easy piece of code to understand, which increments all elements of an array:

```
for (int i=0 ; i<x.length ; ++i )
    x[i] = x[i] + 1;
```

The preoperational novice need only understand two things about that code: (1) that the variable “i” will take on a set of values which map to all elements of the array “x”, and (2) the single line of code in the body of the loop will increment an element of the array. This code has two critical features that make it an example of the simplest type of iterative/array code that a novice can be called upon to understand, because: (1) each iteration of the loop performs the same process as the other iterations, but each iteration performs that process on a unique element of the array, and (2) no iteration affects what happens on any other iteration. This code is an example of iterative/array code that places low demands on the novice’s working memory.

Second, consider the code below, which sums the elements of an array:

```
int sum = 0;

for (int i=0 ; i<x.length ; ++i )
    sum = sum + x[i];
```

As in the first example, the preoperational novice needs to understand that the variable “i” will take on a set of values which map to all elements of the array “x”. The remaining code in this second example is slightly harder to grasp than the code in the first example. The novice must understand that the single line of code in the body of the array will accumulate values in a single variable, “sum”. However, this second example is still relatively simple code to understand because, like the first example, each iteration of the loop performs the same process. There is an additional burden on the novice in this second example, and that is the line of code that initializes “sum”. In total, however, this second example is only slightly harder for the novice to understand than the first example.

Third, and finally, consider the classic BRACElet “explain in plain English” problem (Whalley *et al.*, 2006; Lister *et al.*, 2006), which is shown below:

```
bool bValid = true;

for (int i = 0 ; i < iMAX-1 ; i++)
{
    if (iNumbers[i] > iNumbers[i+1])
        bValid = false;
}
```

That code checks to see if the array is sorted. As reported in several BRACElet papers, students have some difficulty with explaining this code. There are two aspects of this code that make it more difficult to understand for the novice programmer: (1) the novice needs to reason about two different array elements in each loop iteration, and that (2) once the “bValid” variable changes its value, it cannot get its original value back again.

There is another factor in why pre-operational novices find it hard to explain the above classic BRACElet question – they lack the capacity for *transitive inference*. That type of inference is a form of concrete operational reasoning, which we describe and discuss later in this paper.

### 3.3 A Closing Remark

Programming teachers want their students to develop beyond these preoperational behaviours as quickly as possible, but currently many of our students do not develop beyond these behaviours. While this lack of development may be a failing of some of the students, it may also be due to our existing pedagogical practices. We teachers admonish students for their preoperational behaviours, but we do not offer them learning experiences targeted at moving them beyond these preoperational behaviours. By the end of the next section of the paper, on concrete operational reasoning, we will have identified learning and assessment activities that could be used to encourage the novice to move beyond preoperational reasoning.

## 4 Concrete Operational Reasoning

Concrete operational reasoning is the middle level of the three Piagetian levels discussed in this paper – more abstract than preoperational reasoning but less abstract than formal operational reasoning.

### 4.1 General Description

As is often the case with intermediate levels in many hierarchies, concrete thinking is frequently defined in terms of how it differs from the other two levels of Piagetian thinking. Unlike preoperational thinking, concrete thinking does involve routine reasoning about abstractions from the objects in the environment. However, a defining characteristic of concrete thinking is that the abstract thinking is restricted to familiar, real situations, not hypothetical situations (hence the name “concrete”). Consequently, the hypothetico-deductive reasoning of formal operational reasoning tends not to be manifested in concrete reasoning. For a more detailed description of classical concrete operational reasoning, see Brainerd (1978) or Flavell (1977).

#### 4.1.1 Conservation and Reversing

The archetypal manifestation of concrete thinking is the ability to reason about quantities that are conserved, and processes that are reversible. In Piaget’s own work, the famous illustration of concrete reasoning is the conservation of liquid task. This task involves three glasses, two of which are identical in shape, with both of those glasses initially containing an equal amount of water. When studying children, Piaget would (1) ask the child to agree that the two identical glasses contained the same amount of water; (2) pour the water in one of the glasses into the empty third glass, which had a different width from the other two glasses; and finally (3) Piaget would ask the child if the third glass contained less, more, or the same amount of water as the other glass that contained water. Younger children, who are thinking in a preoperational mode, will give different answers depending on the height of the water in the two glasses. Older children (and adults) who are thinking in a concrete mode will immediately claim that the amount of liquid is the same in both glasses. If pressed to justify that claim, the child (or adult) may argue that, although the liquid is at different heights in the two containers, the effect of the differing container widths compensates for the differing heights. If pressed further, the child (or adult) may argue

that if the liquid was poured back into the glass from which it came, then the liquid will reach the same level as it did initially.

Before dismissing the incorrect preoperational reasoning of a small child on this conservation of liquid task as merely being a manifestation of a biologically immature mind, the reader might consider the more sophisticated but incorrect “commonsense beliefs” of adults. For example, research has demonstrated that many university students who commence study in physics have “commonsense beliefs” about motion and force that are incompatible with Newton (Halloun and Hestenes, 1985). A child’s incorrect thinking on the conservation of liquid task highlights the same sort of incorrect thinking that adults can also make, but on more sophisticated tasks. (Indeed, Piaget saw the development of thinking in children as a recapitulation of the historical development of scientific thinking.) Later in this section of the paper on concrete operational thinking (i.e. in 4.2.3), we will nominate programming tasks that also involve the principles of reversibility and conservation.

The neo-Piagetian explanation for the conservation of liquid task, which generalizes to other conservation tasks, is as follows. A child who reasons preoperationally (and incorrectly) does so because of their limited working memory capacity. That limited capacity only allows such a child to focus on a single measure of the quantity of liquid in a glass – the height of the water. A child (or adult) who reasons concretely (and correctly) first begins to do so when an increase in working memory capacity (due to chunking) allows them to consider two measures of the quantity of liquid in a glass – both the height and the width of the water. When a child is able to simultaneously appreciate the relationship between height and width, the child is then open to learning that the effect of height and width can sometimes cancel each other out, and from there the child can learn the more abstract concept that the quantity of liquid is conserved.

#### 4.1.2 Transitive Inference

Transitive inference is another important characteristic of concrete operational reasoning. It is the type of reasoning where, in general terms, if a certain relationship holds between object A and object B, and if the same relationship holds between object B and object C, then the same relationship also holds between object A and object C. For example, Piaget would sometimes ask a child a question like, “*If Adam is taller than Bob, and Bob is taller than Charlie, who is the tallest?*”

From the neo-Piagetian perspective, the reason why a novice may not be able to perform transitive inference is that the working memory of the novice is overloaded by other information, because the novice has not yet learnt to chunk information in this problem domain. Consequently, the novice cannot hold simultaneously in working memory all the information about the relationships between A, B and C required to perform the transitive inference.

## 4.2 Concrete Reasoning in Programming

There is small amount of classical Piagetian literature on concrete operational reasoning in programming, which describes how young children learn to program (e.g.

Huber, 1985). However, we are not aware of any literature that explicitly connects programming with neo-Piagetian literature on concrete operational reasoning. There is some literature, however, that does describe some of the behaviours of novice programmers that are concrete operational, without making the connection to neo-Piagetian literature. In the next two subsections on concrete reasoning, we describe two examples of that literature.

#### 4.2.1 Ginat: Hasty design, futile patching

Recall from the general description of concrete operational reasoning that any reasoning about abstractions is restricted to familiar, real situations, and that hypothetico-deductive reasoning is not part of concrete operational reasoning. Ginat (2007) observed concrete reasoning, with the absence of hypothetico-deductive reasoning, in novice programmers:

*A hasty design may be based on some simplistic application of a familiar design pattern ... The design pattern's invocation may be relevant. Yet, its utilization may not be based on sufficient task analysis and thorough examination of diverse input cases, but rather on some premature association that seems relevant. Errors are not always discovered, as the test cases on which the program is tested are very limited. The devised program is batched, and "seems correct". Then, an outside source (e.g., a teacher) points out a falsifying input. A patch is offered. Sometimes the patch is sufficient for yielding correctness, but more often than not, the patch is insufficient. An additional patch is offered; and the cycle of batch-&-patch continues.*

The following quotation from Flavell, Miller and Miller (2002) is about concrete reasoning in children. We highlight the similarity between what Flavell *et al.* wrote, and what Ginat wrote, by striking through Flavell *et al.*'s references to the elementary school child and replacing it with references to the novice programmer:

*The [concrete operational novice programmer's] ~~elementary school child's~~ characteristic approach to many conceptual problems is to burrow right into the problem data as quickly as possible.... His is an earthbound, concrete, practical minded sort of problem solving approach, one that persistently fixates on the perceptible and inferable reality right there in front of him. His conceptual approach is definitely not unintelligent and it certainly generates solution attempts that are more rational and task-relevant than the preoperational [novice programmer] ~~child~~ is likely to produce. It does, however, hug the ground of detected empirical reality rather closely, and speculations about other possibilities ... occur only with difficulty and as a last resort. An ivory-tower theorist the [concrete operational novice programmer] ~~elementary school child~~ is not. ... For the concrete operational thinker, the realm of the abstract possibility is seen as an uncertain and only occasional extension of the safer and surer realm of palpable reality (p. 146)*

The purpose in providing the above quote is not to suggest that novice programmers behave like school children. Instead, the purpose is to highlight the behaviour of novices, at any age, in any problem domain, including programming, when their reasoning is concrete operational.

#### 4.2.2 Hazzan: Reducing Abstraction

Although Hazzan (2008) did not describe her work in neo-Piagetian terms, she described how novices simplify programming tasks from formal operational to concrete operational reasoning:

*... students, when facing the need to cope meaningfully with concepts that are too abstract for them, tend to reduce the level of abstraction in order to make these abstract concepts meaningful and mentally accessible ... by dealing with specific examples instead of with a whole set defined in general terms.*

#### 4.2.3 Concrete Operational Tasks

In the remainder of this section on concrete operational reasoning, we propose and explore some examples of formative and/or summative tasks that could be given to students to develop and/or test their concrete operational reasoning.

##### 4.2.3.1 Reversing

As discussed in the general description of concrete operational reasoning, the archetypal manifestation of concrete thinking is the ability to reason about quantities that are conserved, and processes that are reversible. What follows is a task that requires the student to reason about reversing.

The purpose of the following code is to move all elements of the array *x* one place to the **right**, with the **rightmost** element being moved to the **leftmost** position:

```
int temp = x[x.length-1];

for (int i = x.length-2; i>=0; --i)
    x[i+1] = x[i];

x[0] = temp;
```

Write code that undoes the effect of the above code. That is, write code to move all elements of the array *x* one place to the **left**, with the **leftmost** element being moved to the **rightmost** position.

An important feature of a solution written by someone using concrete reasoning is the reversal of the direction of the loop. A novice who reasoned preoperationally might miss that change, at least until that novice had a chance to run their initial solution and discover the error.

Note, however, that we cannot conclude that a novice has performed concrete operational reasoning if the only evidence we have is the novice's final, correct solution, and the novice used a computer. A novice might solve this reversing problem by a process that combines quasi-random code changes and copious trial runs, which is the

behaviour of someone reasoning preoperationally. In contrast, the novice who solves this problem by applying concrete operational reasoning will, after inspecting the given code, produce a correct solution almost immediately (apart, perhaps, from trivial syntax errors). To be confident that the novice performed concrete operational reasoning, we must either: (1) observe the novice as they work on the problem, (2) use software to monitor, or limit, the number of edit–compile–run cycles, or (3) have the student write the solution on paper.

#### 4.2.3.2 Conservation

Cases of conservation in programming systems are more abstract than conservation in physical systems, such as the conservation of the volume of a liquid as it is poured from one container to another. One case in programming is the conservation of a specification when the underlying implementation is changed. Consider the following question:

Below is incomplete code for a method which returns the smallest value in the array “x”. The code scans across the array, using the variable “minsofar” to remember the smallest value seen thus far. There are two ways to implement remembering the smallest value seen thus far: (1) remember the actual value, or (2) remember the value’s position in the array. Each box below contains two lines of code, one for implementation (1), the other for implementation (2). First, make a choice about which implementation you will use (it doesn’t matter which). Then, for each box, draw a circle around the appropriate line of code so that the method will correctly return the smallest value in the array.

```
public int min( int x[] ){
    int minsofar = (a) 0
                    (b) x[0] ;

    for ( int i=1 ; i<x.length ; ++i )
    {
        if ( x[i] < (c) minsofar
                    (d) x[minsofar] )

            minsofar = (e) i
                    (f) x[i] ;
    }

    return (g) minsofar
                    (h) x[minsofar] ;
}
```

An alternative version of this question would provide the code for one of the implementations, and then ask the student to provide the other implementation.

The ability to make simple transformations between implementations, while conserving the specification, is an underappreciated skill in programming pedagogy. The novice who can easily perform simple implementation transformations is then less preoccupied by implementation minutiae, and can then focus upon higher abstractions about a program. Current pedagogical

practice, however, does not ensure that novices have this concrete operational skill before giving them tasks that require formal operational reasoning.

#### 4.2.3.3 Transitive Inference (1)

Consider the following question:

In one sentence, describe the purpose of the following code. Assume that the variables y1, y2 and y3 contain integer values. In each of the three boxes that contain sentences beginning “Swap the values in ...” assume that appropriate code is provided – do NOT write that code.

```
if (y1 < y2)
Swap the values in y1 and y2.

if (y2 < y3)
Swap the values in y2 and y3.

if (y1 < y2)
Swap the values in y1 and y2.
```

A suitable answer to this question would be “*It sorts the three values so that  $y1 \geq y2 \geq y3$* ”. To make such a deduction, the novice must perform transitive inference.

While some readers may at first be sceptical that any novice could have difficulty answering the above question, we report in another paper in these same proceedings (Corney, Lister and Teague, 2011) how we used this question in a class test, in the fifth week of an introductory course, and found that half of our students could not answer the question.

#### 4.2.3.4 Transitive Inference (2)

Here again is the code for the classic BRACElet “explain in plain English” problem (Whalley *et al.*, 2006; Lister *et al.*, 2006):

```
bool bValid = true;

for (int i = 0; i < iMAX-1; i++)
{
    if (iNumbers[i] > iNumbers[i+1])
        bValid = false;
}
```

If a novice is to offer an explanation like “*It checks to see if the array is sorted*”, then the novice must perform transitive inference. That is, the novice must recognize (albeit not necessarily in the mathematical notation used here) that if bValid is still true at a given value of i, then for all p, q such that  $0 \leq p < q \leq i+1 \dots$

$$iNumbers[p] \leq iNumbers[q]$$

### 4.3 A Closing Remark

Today, not only do we expect programming students to move beyond preoperational reasoning quickly, but we also expect them to go directly from preoperational reasoning to formal operational reasoning. However, Piagetian theory indicates that such a transition is

difficult, that instead novices tend to move to formal operational reasoning via the concrete operational level. The primary weakness of today's pedagogy of programming, with its heavy emphasis on writing large amounts of code, on problem solving and on top down design, is that it doesn't provide an opportunity for the novice to develop concrete operational skills, via the types of exercises we have described in this section of the paper. Consequently, the novice who is unable to bridge for themselves the gap between preoperational reasoning and formal operational reasoning remains stuck in preoperational reasoning.

## 5 Conclusion

In the CS1 classroom, our students can exhibit three broad forms of neo-Piagetian reasoning. When we lecture to our class, we tend to talk about programs in terms of formal operational reasoning, the most abstract of the three forms of neo-Piagetian reasoning – it is only natural that we would do so, since we who lecture are accomplished programmers. However, many of our students have not yet reached a point (not in CS1) where they comfortably follow a discussion of a program in formal terms – for those students, we may as well be lecturing in a foreign language, and in a sense we are.

Some of our students tend to reason in a preoperational form, where they can trace the changing values in specific code, but do not reason in terms of abstractions of the code. Some other students tend to reason in a concrete operational form, where they can see some abstractions of specific code, but they can only see those abstractions in the context of that specific code. Most students who tend to reason at the preoperational and concrete operational levels will not spontaneously become more accomplished at the formal operational level simply by being exposed to learning materials couched in formal operational terms.

It is often said CS1 students need to practice more, by writing more programs. Students who tend to reason formally about code, and perhaps also students who tend to reason concretely about code, may indeed benefit from writing more code. However, students who tend to reason preoperationally about code will gain little from being forced to write large quantities of code. Such students can only write code by quasi-random mutation. For students who are predominately reasoning at the preoperational level, and perhaps also for students who are predominately reasoning at the concrete level, we need to develop new types of learning experiences that develop their abstract reasoning without requiring them to write a lot of code.

In computing, there has long been a debate about whether programming skill is innate, or acquired. The neo-Piagetian perspective may afford a means of transcending that dialectic. There may indeed be people who will never learn to reason in a formal operational way about programs, but today's high failure rates in introductory programming courses probably overstates the percentage of people who could never learn to program. Just because a particular individual does not spontaneously manifest formal operational reasoning about programs does not mean that the individual cannot possibly learn to do so, given the right tuition. Perhaps,

with a pedagogical approach informed by neo-Piagetian theory, a pedagogical approach that recognizes preoperational and concrete operational reasoning as legitimate developmental phases, which works at increasing the sophistication of how a student reasons about code, via learning experiences that do not require the student to write copious quantities of code, then perhaps such a student can learn to reason in a formal operational way about programs.

While students who aspire to be professional software engineers must eventually develop formal operational programming skills, perhaps it is unrealistic to expect most of those students to do so in their first semester of programming. Perhaps, in that first semester, we teachers should set our sights on getting the bulk of our students to the point where they can consistently reason at the concrete operational level.

## Acknowledgements

The author's work was partially funded by an Associate Fellowship awarded by the Australian Learning and Teaching Council. The author's thinking about neo-Piagetian theory stems from the empirical results of the BRACElet project. He thanks his many collaborators on that project, especially Tony Clear and Jacqui Whalley.

## References

- Agans, D. (2006) *Debugging* New York: Amacom
- Barker, R. and Unger, E. (1983) *A predictor for success in an introductory programming class based upon abstract reasoning development*. *SIGCSE Bull.* 15, 1 (Feb.), 154-158. DOI=10.1145/952978.801037
- Bennedsen, J. and Caspersen, M. (2006) *Abstraction ability as an indicator of success for learning object-oriented programming?* *SIGCSE Bull.* 38, 2 (June), 39-43. <http://doi.acm.org/10.1145/1138403.1138430>
- Biggs, J. B. and Collis, K. F. (1982): *Evaluating the quality of learning: The SOLO taxonomy (Structure of the Observed Learning Outcome)*. New York: Academic Press.
- Brainerd, C. (1978) *Piaget's theory of intelligence*. Englewood Cliffs, N.J.: Prentice-Hall.
- Cafolla, R. (1988) Piagetian Formal Operations and Other Cognitive Correlates of Achievement in Computer Programming. *Journal of Educational Technology Systems*, vol. 16, no. 1, p45-55.
- Corney, M., Lister, R. and Teague, D. (2011) *Early Relational Reasoning and the Novice Programmer: Swapping as the "Hello World" of Relational Reasoning*. The 13th Australasian Computer Education Conference (ACE 2011), Perth, Australia. *Conferences in Research and Practice in Information Technology (CRPIT)*, Vol. 114. J. Hamer and M. de Raadt, Eds. <http://crpit.com/index.html>
- Dijkstra, E. 1979. *The humble programmer*. In *Classics in Software Engineering*, E. N. Yourdon, Ed. ACM Classic Books Series. Yourdon Press, Upper Saddle River, NJ, 111-125.
- Dressel, P. (1983) *Grades: One More Tilt at the Windmill*, in A. W. Chickering (ed.), *Bulletin*, Memphis State University, Center for the Study of Higher Education, Memphis, December.

- Edwards, S. (2004) Using software testing to move students from trial-and-error to reflection-in-action. *SIGCSE Bull.* 36, 1 (March), 26-30. <http://doi.acm.org/10.1145/1028174.971312>
- Fischer, G. (1986) *Computer Programming: A Formal Operational Task*. 16th Annual Symposium of the Piaget Society, Philadelphia, PA, USA. <http://www.eric.ed.gov/PDFS/ED275316.pdf>
- Flavell, J. (1977) *Cognitive development*. Englewood Cliffs, N.J.: Prentice-Hall
- Flavell, J., Miller, P. and Miller, A. (2002) *Cognitive development*. (4th edition) Upper Saddle River, N.J.: Prentice-Hall.
- Ginat, D. (2007). Hasty design, futile patching and the elaboration of rigor. *SIGCSE Bull.* 39, 3 (June), 161-165. <http://doi.acm.org/10.1145/1269900.1268832>
- Halloun, I. and Hestenes, D. (1985) The initial knowledge state of college physics students, *Am. J. Phys.* 53(11), 1043-1055.
- Hazzan, O. (2008). Reflections on teaching abstraction and other soft ideas. *SIGCSE Bull.* 40, 2 (June), 40-43. <http://doi.acm.org/10.1145/1383602.1383631>
- Huber, L. (1985) Computer Learning Through Piaget's Eyes. *Classroom Computer Learning*, Vol. 6, No. 2 (Oct), pp. 39-43.
- Hudak, M., and Anderson, D. (1990) Formal Operations and Learning Style Predict Success in Statistics and Computer Science Courses. *Teaching of Psychology*, 17(4) 231-234.
- Kramer, J. (2007) Is abstraction the key to computing? *Communications of the ACM*, Vol. 50, 4 (April), 36-42. <http://doi.acm.org/10.1145/1232743.1232745>
- Kurtz, B. (1980) Investigating the relationship between the development of abstract reasoning and performance in an introductory programming class. *SIGCSE Bull.* 12(1), 110-117. <http://doi.acm.org/10.1145/953032.804622>
- Lister R., Adams, E. S., Fitzgerald, S., Fone, W., Hamer, J., Lindholm, M., McCartney, R., Moström, E., Sanders, K., Seppälä, O., Simon, B., and Thomas, L. (2004) A Multi-National Study of Reading and Tracing Skills in Novice Programmers. A multi-national study of reading and tracing skills in novice programmers. *SIGCSE Bull.* 36, 4 (June), 119-150. <http://doi.acm.org/10.1145/1041624.1041673>
- Lister, R., Simon, B., Thompson, E., Whalley, J. L., and Prasad, C. (2006) Not seeing the forest for the trees: novice programmers and the SOLO taxonomy. *SIGCSE Bull.* 38, 3 (June), 118-122. <http://doi.acm.org/10.1145/1140123.1140157>
- Lister, R. (2007): *The Neglected Middle Novice Programmer: Reading and Writing without Abstracting*. 20th Annual Conference of the National Advisory Committee on Computing Qualifications (NACCQ'07), Nelson, New Zealand, Mann, S. and Bridgeman, N., Eds, 133-140. <http://www.naccq.ac.nz/conferences/2007/133.pdf>
- Lister, R., Fidge C. and Teague, D. (2009) Further evidence of a relationship between explaining, tracing and writing skills in introductory programming. *SIGCSE Bull.* 41, 3 (July), 161-165. <http://doi.acm.org/10.1145/1595496.1562930>
- Lister, R., Clear, T., Simon, Bouvier, D. J., Carter, P., Eckerdal, A., Jacková, J., Lopez, M., McCartney, R., Robbins, P., Seppälä, O., and Thompson, E. (2010). Naturally occurring data as research instrument: analyzing examination responses to study the novice programmer. *SIGCSE Bull.* 41, 4 (Jan), pp. 156-173. <http://doi.acm.org/10.1145/1709424.1709460>
- Lopez, M., Whalley, J., Robbins P. and Lister, R. (2008) *Relationships between reading, tracing and writing skills in introductory programming*. Fourth international Workshop on Computing Education Research (ICER). pp. 101-112. <http://doi.acm.org/10.1145/1404520.1404531>
- McCracken, M., Almstrum, V., Diaz, D., Guzdial, M., Hagen, D., Kolikant, Y., Laxer, C., Thomas, L., Utting, I., and Wilusz, T. (2001) A Multi-National, Multi-Institutional Study of Assessment of Programming Skills of First-year CS Students. *SIGCSE Bull.*, 33(4). pp 125-140.
- Miller, G. A. (1956). The magical number seven, plus or minus two: Some limits on our capacity for processing information. *Psychological Review*, 63, 81-97.
- Morra, S., Gobbo, C., Marini, Z. and Sheese, R. (2007) *Cognitive Development: Neo-Piagetian Perspectives*. Psychology Press.
- Philpott, A., Robbins, P., and Whalley, J. (2007) *Accessing the Steps on the Road to Relational Thinking*. 20th Annual Conference of the National Advisory Committee on Computing Qualifications (NACCQ'07), Port Nelson, New Zealand, Mann, S. and Bridgeman, N., Eds, p. 286.
- Smith, L. (Ed.) (1992) *Jean Piaget : critical assessments*. London ; New York : Routledge.
- Thomas, L., Ratcliffe, M., and Thomasson, B. (2004) Scaffolding with object diagrams in first year programming classes: some unexpected results. *SIGCSE Bull.* 36, 1 (March), 250-254. <http://doi.acm.org/10.1145/1028174.971390>
- Traynor, D., Bergin, S., and Gibson, J. P. (2006): *Automated assessment in CSI*. 8th Australian Conference on Computing Education (ACE), Hobart, Australia, ACM International Conference Proceeding Series, 165: 223-228. <http://crpit.com/confpapers/CRPITV52Traynor.pdf>
- Turner, A. (1991) Computing Curricula 1991. *Communications of the ACM*, 34(6), pp. 68-84.
- Venables, A., Tan, G. and Lister, R. (2009) *A Closer Look at Tracing, Explaining and Code Writing Skills in the Novice Programmer*. Fifth International Workshop on Computing Education Research (ICER). pp. 117-128. <http://doi.acm.org/10.1145/1584322.1584336>
- Whalley, J., Lister, R., Thompson, E., Clear, T, Robbins, P., and Prasad, C. (2006): *An Australasian Study of Reading and Comprehension Skills in Novice Programmers, using the Bloom and SOLO Taxonomies*. 8th Australian Conference on Computing Education (ACE), Hobart, Australia, ACM International Conference Proceeding Series, 165: 243-252. <http://crpit.com/abstracts/CRPITV52Whalley.html>



# Open Teaching

## A Case Study on Publishing Lecture Videos Publicly

Richard Buckland

School of Computer Science and Engineering  
The University of New South Wales  
Sydney 2052 Australia  
Email: [richardb@unsw.edu.au](mailto:richardb@unsw.edu.au)

### Abstract

This paper reports on a project to video record lectures for undergraduate computing courses and publish them openly online. It outlines the objectives of the project, the techniques experimented with to record the lectures, the major challenges encountered, and how they were addressed. The paper then highlights a number of the interesting and sometimes unexpected benefits which have followed from publishing the lecture recordings, as well as some disadvantages and warnings. It is my hope that others may be able to benefit from our experiences and feel empowered to record and openly publish their own teaching practice.

**Keywords:** OpenLearning open data online lecture video recording eLearning YouTube

### 1 Introduction

*“The internet is fantastic. I am receiving a university-level education whilst I sit here in my underwear in my mom’s basement, eating nachos and drinking coke. Life is good.”*

YouTube user *caesium* commenting on video ‘Lecture 1: Introduction to Data Structures and Algorithms’ (caesium 2010)

This paper reports on the experiences of a project to video record undergraduate lectures and publish them openly online. In 2008, with the encouragement and enthusiastic support of former students, I started capturing lectures of our first and second year computing courses on video. I delivered the lectures and the former students operated the cameras and live mixing equipment. My original intention was merely to make the recordings available to a handful of off-campus students and the undergraduate teaching academics within our school (CSE, the School of Computer Science and Engineering). However in the spirit of openness in education and after much agonising I also made the recordings publicly and freely available to all via YouTube and iTunes. In the subsequent two years these recorded lectures have been viewed over

a million times by hundreds of thousands of learners of widely varying age, nationality, socio economic background, and prior knowledge and experience.

Since that time there has been a constant demand via email, YouTube comments, and physical mail for further such recordings and for supplementary material to support the current recordings. There has also been a similar volume of messages of thanks and gratitude from learners and fellow teachers worldwide for sharing the lectures. I have been delighted to find that what was originally envisioned as a small project has blossomed into one of the most significant educational activities I have undertaken.

This paper outlines the techniques we experimented with to record the lectures, the major problems we encountered, and how we have addressed them. The paper then highlights a number of the interesting and sometimes unexpected benefits which have followed from publishing the lecture recordings, as well as some disadvantages and warnings. It is my hope that others may be able to benefit from our experiences and feel empowered to record and openly publish their own teaching practice.

### 2 Background

The initial impetus to video record first year CS1 computing lectures in 2008 arose from two problems CSE was facing at that time. Firstly we wished to offer selected secondary school students the chance to take a first year university computing course whilst still at secondary school but recognised they would find it hard to miss school and travel to campus several times each week to attend lectures. Secondly we wished to share teaching practice amongst the pool of teachers for our core first and second year courses to both demonstrate how a new and ambitious syllabus could be implemented, and to ensure that course coverage remained relatively stable from semester to semester.

#### 2.1 For viewing by students

Video recording lectures was a key part of our secondary school outreach program. Secondary school students in the programme were to watch the lecture videos in their own time in advance and then attend university once each week for a face-to-face tutorial and lab. In 2008 we ran a trial with three secondary school students, we expanded this to sixteen in 2009, and forty in 2010.

**Critical requirements:** To be able to have the secondary school students participate in the course in synchrony with our undergraduate students the lecture recordings needed to be available to the participants within a few days of the actual lecture - ideally the next day for the Thursday lectures — since the

---

Links and supplementary material for this paper at <https://wiki.cse.unsw.edu.au/richardb/ACE2011>

Copyright ©2011, Australian Computer Society, Inc. This paper appeared at the Thirteenth Australasian Computing Education Conference (ACE2011), Perth, Australia, January 2011. Conferences in Research and Practice in Information Technology (CRPIT), Vol. 114, John Hamer and Michael de Raadt, Ed. Reproduction for academic, not-for profit purposes permitted provided this text is included.

exercises for week  $n + 1$  week depended on the lecture material from week  $n$ , and many students liked to prepare on the weekend. Furthermore it was imperative that the end-to-end process be reliable and robust - if the lecture recording process ever failed for any reason there was no easy way for the students to recover, that lecture would be missed.

**Desirable requirements:** We wanted the remote secondary school students to feel that they were part of the course community, and that the undergraduate students taking the course were their peers.

Our objective for the lecture recordings was that the viewer would feel that they were sitting in the class. To this end we used a Cinéma Vérité approach to the filming. This precluded using a simple screen-capture plus static camera arrangement such as the proprietary Lectopia system which was then being used to audio record some lectures at UNSW.

Instead we used a human camera operator recorded each lecture from the back row of seats. The operator was instructed to point the camera wherever they felt that they would be looking if they were a student in the class and to zoom on things that they would want to see more closely. We didn't use experienced camera operators, instead we used enthusiastic ex-students of the course. We hoped that this would mean they would have some insight into the mind of the student viewers.

We did not feel that elaborate post production was of key importance, or even that the video quality was critical, so long as it was sufficient that the students were able to see and understand what was going on. In summary we strove to give the recordings a sense of authenticity rather than a sense of professional production.

## 2.2 For viewing by teachers

The second factor motivating our venture into lecture recording was to enable the teachers of our core first and second year courses to share their teaching practice, and to stabilise the content of these courses.

In 2006 CSE's core syllabus was reviewed and substantially restructured. Before the review and restructure first and second year courses had learning objectives predominantly in the lower half of Bloom's taxonomy (Bloom 1956) (*remembering, understanding, applying* in the terminology of Anderson (2001)). The content objectives of each course were almost exclusively low level discipline knowledge and skills. In contrast the new syllabus added a sequence of higher level learning objectives and explicitly included the development of graduate attributes in each course. For example, after the restructure developing effective group-work, leadership, innovation, skepticism, rigour, time management, life long learning skills, and a framework of ethics and professionalism are now objectives of each course rather than simply being addressed at the end of the degree program in a standalone fashion — for example by a single final year course dedicated to group-work and time management.

However in the period immediately following the restructure a number of the pool of teachers who taught these core courses expressed uncertainty as to how this integrated approach could be achieved in their course, or even skepticism that it were possible to do so at all in more than a token way. I felt that in addition to *talking about* ways the new syllabus could be implemented it could be useful and compelling to *directly show* concrete examples of ways of achieving these objectives — and to that end over the past two years I have run, video recorded, and published a reference version of each of these core first and second year courses.

A related challenge which the school wanted to address was the issue of syllabus drift. Each of our large core courses is taught by at least two teachers — one in first semester and one in second semester. Furthermore over time teachers move off courses and new teachers take their place. We have observed that individual teachers have their own understanding and interpretation of the syllabus and that the material covered in a course, and the depth to which it is covered, can change substantially from year to year, even from semester to semester. This can cause problems in subsequent courses which rely on the material which was supposed to be covered. For a personal example: in my Computer Security and Cryptography course in 2008 I had students who had never heard of the RSA cryptosystem, students who had already been taught it once, students who had been taught it twice, and even some outraged students who, because of the particular sequence of semesters in which they had taken the prerequisite courses, had already seen it three times!

The school has instituted a series of measures to limit the problem of syllabus drift. Having a public and published reference version of each course is an important part of our approach.

## 3 Publishing Online: Lofty Goals

The previous section described my initial motivation for video recording lectures. In this section I discuss the decision to then publish those recordings online, free for all to access and view. In the Outcomes section to follow I will posit that this decision to publish openly, mirroring the emerging movement for open data generally (see for example (Uhlir & Schroder 2008), (OECD 2004), and the *data.gov* and *data.gov.uk* sites) was the most important decision of the project. As MIT President Charles M. Vest, whose inspirational OpenCourseWare (OCW) project pioneered open teaching, proclaimed “[*openly sharing lectures*] expresses our belief in the way education can be advanced - by constantly widening access to information and by inspiring others to participate.” (Vest 2001)

The initial goals for undertaking the recording were modest and addressed pragmatic issues at my own institution. In contrast my hopes for publishing the material online were more lofty — to disseminate discipline content and practices to categories of learners traditionally unable to access universities for learning, and to encourage improvements in the professional practice of teaching.

### 3.1 Openness and Autonomy

Openly publishing lectures has the potential for widespread improvement in the practice of teaching by introducing a novel form of openness into the teaching process.

What happens in the classroom has traditionally been a private matter between teacher and students. Referring to Britzman (1986)'s observations on the culture of teachers Rogers & Babinski (2002) posit:

*‘It is this “culture of teachers” which promotes privacy and autonomy, which establishes barriers to genuine dialog among teachers.’ and ‘there are invisible walls constructed by the “culture of teachers” that perpetuate a lack of community’*

Afterwards, in some cases, some aspects of the process may be made open through scholarly reporting and writing on teaching practice, some aspects may

be informally discussed with fellow teachers, and another view of what transpired may be made open via student feedback and ratings. However these are all interpretative events — it is not possible for interested parties to gather primary data on what happened in a particular teaching situation unless they happened to be physically present at the time. In our experience teaching practice is generally unobserved.

Yet openness and genuine collegiality between teachers, open discussion and critical reflection on teaching practices has well established and significant impact on the development of teachers and on the quality of their teaching (see for example Patricka et al. (2010) and Daya (1993)). Indeed open practice and a concomitant culture of review and reflection is the hallmark of a healthy profession.

*When I started my career I really didn't know how to teach maths. I didn't have the training for it. So, in the beginning I had to find out how to teach it. I needed some help. If you're only a beginner, it's difficult to get some help. Also, my colleagues were very busy. They didn't have the time to help me. So I had to solve my problems on my own.*  
(Clement & Vandenberghe 2000)

There are a number of established ways of teachers addressing this issue and sharing their practice. Two of the most widespread are peer review and team teaching. These are both rich methods with being investigated by an active research community and provide the potential for a range of benefits when they are used sensitively in an environment of trust under a developmental rather than judgemental ethos (see for example (Gosling 2005), (Gosling & O'Connor 2009) and (Anderson et al 2009)). Here we simply note that these approaches provide environments where it is acceptable for teachers to observe the teaching of their peers. Openly publishing lectures online provides another such environment, with the potential to share some of the same benefits, and likely subject to similar mitigating factors.

My own teaching practice has been improved significantly over the years by observing other teachers teaching. For the first five or six years of my career as a university lecturer I made a habit of attending and observing the lectures of other teachers at local universities about whom I had heard students speak highly. This had a noticeable effect on the effectiveness of my own teaching (indeed I subsequently received a number of teaching awards). However the observation process was time consuming and awkward to arrange. The current practice of university teaching is a largely private and closed arrangement. Our students know how we teach, but in the large our peers and colleagues do not.

On the other hand it is easy to “visit” the classrooms of those who publish their lectures online. By publishing we open the doors of the classroom and let in the light.

### 3.2 Open Learning

Our second “lofty goal” was open learning — to disseminate our teaching so that it is available to anyone who might wish to learn from it. For example this might include students at our own university, students at other universities, students not yet in university, those unable to attend university for socio economic reasons, or reasons of health or time constraints, and those already in a career looking for ongoing professional development or contemplating a career change.

For me this was what education is about - humanity striving to improve itself, thirsting for knowledge

and to know more. Putting university teaching online is reminiscent of the transformative power of the Children's Television Workshop back in the 1960s and 1970s, where open freely given knowledge and teaching gave new opportunities to inner city children with limited educational options (Lesser 1974).

## 4 Approach

This section sets out in detail the process we use to record and publish the lectures. The specific details of the makes and models of the equipment we used is set out in the Appendix. Our hope is that this will be a useful resource for others interested in dipping their toes into the water.

In general my experience has been that most of the things that I worried about beforehand turned out to be remarkably easy to solve in a satisfactory manner. These included:

1. What hardware to use?
2. What level of quality to aim at?
3. How to publish on the internet?
4. Intellectual property?
5. Will it be too expensive?
6. Will it even work, at all?
7. Fear of looking like an idiot in front of the whole world, irrevocably...

On the other hand the main challenges which arose, and which have still not been solved perfectly, were things I had not anticipated:

1. Correspondence can consume vast amounts of time (and likely contributed to my developing RSI in late 2008)
2. How can we have the recording work every time, without fail?
3. How can we set up the gear and pack away the gear in time?
4. Hateful comments on YouTube can tear your heart

### 4.1 Selection of Lecture Theatre

A critical requirement for the recording process was that it not diminish the experience for the students sitting in the room. At the end of the day they were most important aspect of the lecture and I didn't want them to feel like a studio audience, I didn't want the students in the lecture theatre to feel that the recording was the main thing and they were secondary. We decided to locate all the cameras and recording gear at the back of the room, so the camera and camera operator did not visually intrude during the lecture.

There were only a few lecture theatres which were both large enough for our class (about 250 students) and in which the back row was not too distant from the front of the lecture theatre to record clearly in ambient light. Essentially we needed large, shallow but wide theatres. Furthermore the camera crew had to get themselves and their gear in and out in the brief (10 minute) gap between successive lectures. Luckily we were able to find and get permission to use a wide theatre with a separate back entrance which we could mark off with portable signage and reserve for the camera crew without obstructing safe exit in case of fire or other emergency. The distance from the camera position to the blackboard was about 15 meters.

## 4.2 Keep it simple

We did a number of dress rehearsals before the start of semester and soon discovered that the biggest challenge was set-up and pack-up complexity and speed. Our hard limit was the length of the interval between the previous lecture finishing and the lecture to be recorded starting. This meant we had to be reliably set up and be ready to roll in ten minutes. This became the limiting factor in our recording design. Basically there was no point in having some fantastic but complex piece of equipment if it took so long to set it up that we missed capturing the start of the lecture.

## 4.3 Cameras and Microphones

I wanted to record the output from slides/data projector/blackboard/document camera and also to record the lecturer and to see the backs of the heads of students to give viewers the effect of being in the room rather than the effect of watching a slick Audio-Visual presentation. The decision to adopt a Cinéma Vérité approach had the fortunate consequence of allowing us to use consumer quality recording equipment which was considerably cheaper than professional equipment.

For recording the lecturer we used a high end consumer grade HD digital video camera. It was a common camera so we were easily able to find a spare to swap when the camera went in for its one repair (tip: never plug firewire cables in backwards — you can do it without too much force and it reverses the power supply and blows the camera and the firewire board on the computer. We now use an in-line polarity protecting electronic fuse...). We mounted the camera on a professional tripod with fluid filled head to allow steady recording and smooth panning. For audio the lecturer wears a wireless microphone and transmitting pack, and the receiver is mounted on the camera. The camera we used accepted multiple audio inputs which was very convenient as it permitted us to mix in the microphone audio channel on the camera itself and not have to worry about an extra data stream and extra cables to the computer.

We tried various clever hardware and software approaches to record the images from the blackboard and projected from the data projector/document camera etc. Luckily we were able to try out proposed solutions before having to pay much money as these did not live up to our expectations. The two general types of approaches we initially tried were:

1. Resampling the SVGA signal going into the data projector (quality too low, didn't capture the blackboard, needed slow moving facilities staff to implement a hardware solution for each room we used before we could record in it);
2. Vodcasting software to capture the screen from lecturer laptop (quality too low, didn't capture black board, document camera, or the built-in computer in the lecture theatre, needed a long cable run from front of the theatre to the mixing area at the back)

None of these were reliable and flexible enough for our requirements. So we instead adopted the naive solution of purchasing a second camera (identical to the first) and simply pointing it at the blackboard/screen. That meant we could capture data projected from any source. Furthermore for reliability it meant we were physically isolated from, and so not dependant on, the existing (unreliable) AV equipment in the lecture theatre. For speed of setup we didn't even attach this camera to another tripod — since it was fixed

in position for the duration of the lecture we simply placed it on a sandbag and pointed it in the right direction. At 15 meters with near maximum optical zoom it produced good quality video.

We attached a high grade microphone to the fixed camera as a backup in case the lecturer's radio mike failed, and to capture ambient sound in the room such as questions from students.

## 4.4 Audio Reliability

The camera operator always wore headphones plugged into the camera they were operating so they would know instantly if there was a problem with the sound. We used fresh batteries in the radio mike and receiver each time. This seemed wasteful but after once experiencing losing sound due to flat batteries we never wished to risk that again!

Periodically I offered the resultant half-used AA batteries to students who seemed excited to be offered a gift of such riches. Perhaps they'll recall this and endow the university with similar generosity when they are running Google...

## 4.5 Mixing and Data Logging

Both the technicians in the University's own Audio Visual unit, and the two professional video editors I initially consulted for advice used the same editing software and the same family of hardware so we simply mimicked their setup. No doubt a vast range of other configurations would also have been satisfactory for our purposes but we didn't have the time or tolerance for possible recording problems to permit the luxury of experimentation. Working on the same system as the locally available and friendly experts seemed the most prudent approach to adopt to minimise risk of problems arising which could not be solved rapidly.

We choose to capture on a desktop machine rather than a laptop. It would have been more convenient to capture and mix on a laptop but the laptops we experimented with all had problems processing 3 firewire streams in parallel at high data rates and we wished to have the capability to use 3 cameras if needed (to date we have never needed 3 cameras however). In addition the desktop machine has 2 quad core processors which provided much faster postproduction editing and processing than a laptop.

Mixing and processing audio visual data is astonishingly time consuming. A very minimal edit takes me about five minutes per one minute of footage. This means five hours for a one hour lecture. This was not sustainable in our context as I was already very busy lecturing and running the course and could not find an extra 10-20 hours time to work on video processing each week. Furthermore we needed to release the videos to the off campus students within a day or so. And this fast turnaround would likely be most critical precisely when the course, and consequently I, was most busy (near assignment due dates etc.) After much consultation and brainstorming we decided the most reliable strategy was to mix the footage live *during the lecture* (much like a television studio does) and post the already mixed footage immediately at the end of each lecture.

We used live mixing software which could be preset with common mixes of the two cameras (e.g. camera A only, camera B only, camera B with the output from camera A picture-in-picture, and so on) and fade between the presets at a mouse click. In theory the camera operator probably could have also done the mixing at the same time but for reliable setup, packing up, and dealing with contingencies we had usually

had two operators at the back, one mixing and one operating the camera.

The two cameras were connected to the capture computer via firewire, they were mixed live and the resultant stream was saved to the hard disk in real time. We also kept DV tapes running in each camera in case of catastrophic or in case we wanted to re-mix the streams afterwards. There have been a few times we have needed to remix from these backup tapes after the lecture as occasionally the mixing software freezes or drops a camera, and for example one time the power cord was kicked out from the computer. The remixing in these cases has taken a considerable amount of time and although the editing is marginally slicker when planned rather than performed live the benefit is far outweighed by the extra time cost.

#### 4.6 Hosting and Uploading

There are a range of ways to host and distribute Open Educational Media (OER) such as lecture recordings. They are not mutually exclusive - we used several.

The most commonly used approach currently seems to be to host the media files on servers at the host university, usually in an ad hoc fashion school-by-school or even course-by-course. Advantages of this approach is it is easy to arrange and monitor and other course material can be integrated. The main disadvantage is that, except for local students with whom you can directly communicate, it can be hard for others to find the material. This is largely a *push* model of distribution. The most successful exemplar of this approach is MIT who provide lectures and course material via their dedicated OpenCourseWare portal (MIT-OCW).

To allow others to find the material more easily it can be submitted to well known learning repositories such as Academic Earth, the OpenCourseware Consortium, YouTubeEDU, and iTunesU.

We hosted the video files on our school's local file-server for local students but in order to make the material easily accessible globally we also published the videos on YouTube and on iTunesU. Uploading to YouTube is easy but slow. Some of our recent uploads have taken over 5 hours per lecture.

Two benefits of YouTubeEDU are that Google hosts the files for you at no cost (and with no maintenance effort on your part), and that there is already an active viewer community on YouTube so the videos are easily accessible to an existing and substantial audience. Uploads to YouTube have recently been permitted to exceed the previous 2GB limit per video, and educational users are permitted to upload videos of any duration (normal YouTube uploaders are limited to videos of no more than 10 minutes.) In practice about 2GB is sufficient to provide a high quality HD video of a one hour lecture at 720p resolution. YouTube provides a range of analytical tools to allow posters to see where their viewers are coming from, and also provide audience demographic (age, gender) data when it is known. They also provide clear and detailed data on how your viewers have discovered your videos — which is invaluable in selecting and adjusting keywords and video titles.

An advantage of iTunes is that you host the files yourself and so have full control, a possible disadvantage is that you have to organise and pay for hosting and bandwidth. Academic Earth has the advantage of being able to host course material as well as lecture videos.

Be aware that the YouTube uploading process is quite fragile, accidentally navigating away from the upload page during upload aborts the whole process, as can engaging in other YouTube related activity it seems. YouTube offer a bulk uploader but I have

sometimes found this to be unreliable and have got better results uploading the videos manually one at a time. I upload from a dedicated machine and don't touch the machine again until the upload has complete since accidentally aborting a 5 hour upload after 4 hours is very frustrating. Uploads also fail erratically on completion with terse error messages suggesting that the video file is in the wrong format, trying again with an identical file often works.

Viewers in YouTube can interact with the video, they can add annotations, write comments, rate the video, add it to their own public compilations, and recommend it to their friends. As a teacher I find the feedback data provided by YouTube is useful in improving the effectiveness of the videos, in contrast to the basic hit count data our University provides to those hosting vodcast files on the University's iTunes server. I also like the (admittedly limited) sense of community amongst the YouTube viewers as opposed to the solitary experience offered to my iTunesU viewers.

#### 4.7 IP

Our lectures and course material are released under the Creative Commons Attribution-NonCommercial-ShareAlike 3.0 licence. This means copyright of the lecture itself remains with the lecturer but anyone is permitted to copy, remix and derive further works from the material for non commercial purposes provided they provide attribution and impose the same conditions on the copies and derived works.

### 5 Outcomes

This section first reports on the outcomes of the project with respect to our two initial objectives for recording the lectures. It then reports on the outcomes of openly publishing, whose significance I feel far eclipses the original aims of the project.

#### 5.1 Secondary School Outreach Programme

The secondary school outreach program has eventually proved a substantial success although the 2008 pilot involving three students had a number of problems. That year one student dropped out, and the other two had problems accessing the videos online due to the bandwidth limits of their home internet connection. Furthermore we had problems uploading the videos to YouTube in a timely manner - largely as a result of the problems outlined in the previous section.

In 2009 the program ran again with the already recorded 2008 lectures, this time with 16 students. By this time all of the lectures were already on YouTube but a few students also reported problems with their download quotas. All the files were available for free download over the university wireless network which mitigated this problem to some extent. In 2010 we provided all students with the video files on DVDs so no downloading was required.

The response of the students taking the course has been overwhelming. Although we provide formal transcript qualification at the end of the course suitable for credit at any university three of the four students who have completed secondary school to date have come to study computing at the university — suggesting that they found the experience rewarding. Many of the other students from the course, currently in their final year of secondary schooling, have stated a firm desire to proceed to university and continue to study computing. The level of enthusiasm, excitement, and community amongst the graduates of

the course is quite remarkable. At the 2009 UNSW open day a number of the secondary students turned up at the university of their own initiative and organised their own stall where they demonstrated the project they were currently working on. We observed two of the students speaking excitedly to visiting students about the fun of studying computing at university. Students from the 2009 course have subsequently volunteered to tutor and mentor the 2010 students. This shows considerable dedication from them as at the same time they are studying for their own HSC. Interestingly approximately half of the returning tutor/mentors are female, which contrasts with our typical first year intake of around 20% female students. If this trend persists it will warrant investigation.

On average the academic performance of the 2009 students who watched video lectures and face to face tutorials and labs exceeded the performance of the students who physically attended lectures and had face to face tutorials. Obviously the sample size is small and the group was positively selected so the two groups are not directly comparable but it does not appear that the students who did not have face to face lectures were at any significant educational disadvantage. Indeed the video students themselves reported high levels of satisfaction with the way the course was organised, the average response to the summary question "Overall, I was satisfied with the quality of this course" was 4.8/5.0 (15 responses) for those who watched lecture videos, compared with 4.4/5.0 (184 responses) for the university students who did physically attend lectures. These results are comparable to those experienced in CMU's Open Learning Initiative (Lovett et al 2008).

The high positive response from the video-watching students may well have been a consequence of the resultant shift in focus from lectures to the group based tutorial-lab sessions. These students watched lectures at home privately as preparation for the tut-lab session, which was then seen as the focus of the course each week. Being a tutorial-lab it was highly interactive and personalised perhaps making the overall experience seem much more social and personally relevant. We are currently gathering data from the 2010 course participants to further investigate this hypothesis.

## 5.2 Sharing teaching practice, and Syllabus Stability

We are still working on the project of using the recordings to share ways of teaching higher level content such as group work, skepticism etc. Some teachers outside of the group we were targeting with the lecture recordings have unexpectedly approached us and said they have watched some and in some cases all of the lectures and have found them useful in their own teaching practice. From informal conversations it seems many of the teachers in the school have watched at least one of the lectures but it is extremely unlikely that many have had the time or inclination to watch the full 50 hours for any course. We are currently working on a teacher summary for each course which gives links to small fragments of video demonstrating each of the more novel aspects of the current syllabus. This is a work in progress and is publicly available for any interested teachers (see <https://wiki.cse.unsw.edu.au/openlearning>).

One interesting aspect of having lectures captured as open and persistent objects is that this makes them available as first class objects for academic discourse and analysis. For example in a paper I am currently writing on the effective teaching of skepticism I am able to cite specific moments in various lectures to illustrate points — and then readers are able to see the

teaching at these moments for themselves rather than just reading my second hand reports of my own interpretation of what I recall doing. Further they can roll forward or backwards to any other part of the lecture, or indeed view preceding or subsequent lectures, and investigate to their own satisfaction issues of context and acts of summary and selection I perhaps unknowingly perform when reporting the work. This is consistent with the emerging open data movement discussed in Section 3.0 above, and parallels the modern practice in scientific scholarship of making raw data publicly available in addition to the conclusions drawn from the data. This approach offers many novel and exciting possibilities and suggests that dramatic changes may lie ahead in conducting and disseminating the scholarship of teaching and learning.

## 5.3 Unanticipated Impact

As reported above the lecture videos have been watched over one million times. They have been highly rated by viewers on YouTube all being rated either 5 or 4.5 stars. One of the lecture videos was selected by YouTube as a Featured Video on the YouTube front page world wide. The response from students and teachers worldwide has been somewhat overwhelming and by late 2008 responding to emails and written letters about the lectures was taking hours each week. Many of these interactions were from university students currently studying computing at other institutions but significant impact of the lectures was on *those for whom university was not an option*. I received messages from viewers were from third world countries without local university options, from viewers who were not permitted to attend university, from those who could not afford university, from younger students bored at school but not yet of university age.

*THANKYOU for posting [your lectures]. I am dyslexic with the reading comprehension of a 7yo and the language ability of a 35yo, have pretty much fallen behind at uni within the first 3 weeks of every course (at [another university] not UNSW). Putting the lectures online is a real help as it gives people the chance to learn at their own pace instead of under pressure of the course. It is definitely a worthwhile project, and a great supplement.*

YouTube user *Battlewench* comment on UNSW learning YouTube channel (Battlewench 2009)

Existing mechanisms for course feedback and assessing the effectiveness of my face to face teaching do not translate in any obvious way to assessing the learning arising from the largely anonymous group of remote learners watching the material online. I am currently engaged in research project to investigate possible ways to address this challenge. Nonetheless the flood of qualitative feedback I have received since starting to publish the lecture recordings makes it clear that the project has had an educational impact dwarfing all my previous work. This has been both humbling and inspirational and is incredibly motivational as a teacher.

The online lectures currently average 1,500 viewers per day watching 3,000 videos. It is hard to divide these viewer numbers into students and fellow teachers. An interesting indication is that Google reports 70% of our viewers are over the age of 34. I have received considerable correspondence and YouTube comments from other teachers who have enjoyed watching a different perspective on teaching. As an

indication the most recent teaching comment from a viewer today is quoted below.

I look forward in turn to observing the teaching practice of other teachers who engage in open teaching. Current practitioners of open teaching in computer science include Mehran Sahami, Eric Grimson and John Guttag. I believe that teachers openly sharing our practice in this way will improve the teaching profession and the quality of what we do.

*Wow!! Our high school here in Texas has gone to a 1:1 laptop to student setup. We are in our 3rd year of computers this year. I thought I had exhausted the possibilities! Now I see that I've only begun... I can't wait to try the wiki experience this year with my students. Thank you...!! — (SkyRookie1 2010)*

## 6 Issues

After the mechanical details of recording and publishing lectures are resolved there remain important questions about how to make the process effective for learning. We set out briefly issues which need to be carefully considered.

- **Consent:** Students present in the lectures must not only consent to the recording but the act of recording and perhaps their concerns about being recorded must not interfere with their own learning. Our approach is that the students in the class are our primary concern, the recording is a nice bonus if it works. We consult with students in the first lecture, we strive not to record faces of students, we identify areas they can sit to avoid being recorded, and we ask that if anyone is unhappy with their comments in a lecture being recorded that they advise us as soon as possible after the lecture and we will erase this before posting the footage. Despite providing these avenues for students to exclude themselves from the recording to date no student has raised any concerns. This needs further investigation but perhaps they are simply not as concerned with privacy as say my generation is.
- **Engagement:** watching videos is passive by nature, and one hour lectures are much longer than the 10 minute norm most viewers have become accustomed to on YouTube. To stop viewers falling asleep we use a combination of active camera work, cuts between cameras, and the way the lecture is structured in an effort to keep the students engaged. As mentioned above we strive to make experience for viewers feel like they are present in the class rather than simply watching a passive recording of it. Luckily also the discipline of computing is very interesting in and of itself!
- **Criticism:** Comments on the internet can be made quickly and with little thought. Debate can be robust. Furthermore such comments are made publicly and are available for all to see. I follow a policy of only deleting comments which include offensive language or which are disrespectful to specific students in the audience, and leave other critical comments untouched. Some of the comments and feedback I have received on my teaching has been quite confronting. It seems likely that some teachers may not wish to be exposed to blunt and public criticism of their teaching. Furthermore it is possible that some teachers will not wish to share their teaching even if public commenting is not permitted, engaging in

what Clement & Vandenberghe (2000) refer to as *strategic autonomy* in order to avoid the criticism of colleagues. This may well prove the most significant barrier to widespread adoption of open teaching.

*Like for instance, if you have difficulties with one of the children, the real and problem children, then it's delicate to discuss that with your colleague. You don't feel inclined to tell about it, because the other would doubt your competence. Especially if you're a novice teacher. Then they look at you with Argus eyes, don't they — teacher comment (Clement & Vandenberghe 2000)*

## 7 Conclusion

Publishing videos of my lectures online has proved to involve only a modest amount of work and virtually no cost, despite the university having no useful recording infrastructure in place. This was made possible due to supportive colleagues and university management, enthusiastic student volunteers, and the adoption of a Cinéma Vérité approach and live mixing.

From my point of view the main costs have been the time taken to respond to correspondence from remote learners and the continual guilt I feel for not being able to do this adequately.

The rewards have been unexpected and immense, both emotionally and in the impact I have been able to have with my teaching on students worldwide and on other teachers. I am excited by the potential for video recording to transform the practice of teaching from something that happens largely privately and behind closed doors into an open and communal process.

## 8 Future Work

Further work is needed to develop useful ways of assessing and improving the effectiveness of teaching and course design when learners are largely anonymous; have widely differing and unknown levels of prior knowledge, skills, and learning objectives; and need not follow the course in a linear fashion.

## 9 Acknowledgements

I acknowledge and thank my students from COMP1917 (2008 semester 1), COMP1927 (2009 semester 2), and COMP2911 (2010 semester 1) for being so supportive of our time together being recorded, and for not letting the presence of the cameras dim their enthusiastic and active engagement during lectures. Our courses are based around a strong sense of community. If the classes look fun and interesting, it's because they were, and this arose from the community we formed while each course unfolded - with students contributing at least as much to the mood of the class as I did as lecturer. It was our class, not my class, and it was with great generosity and open heartedness that the students were willing to allow outsiders to share in their experiences. Furthermore I thank my students for their unquestioned trust, which I found quite moving, that I would treat them respectfully in the editing and release of the recordings of our times together.

That this paper was written is due in large part from the encouragement and support I have received

from my many wonderful colleagues with a passion for education. Most directly in the case of this paper Helen Dalton, Jan McLean and Judy Kay who asked great questions, and Matthew Clarke who provided helpful writing advice. Matthew also very kindly read and commented on an earlier version of the paper and this version has been improved by a number of his comments. Likewise the paper has benefited substantially from the critiques and insights contributed by each of the three anonymous reviewers. I thank you all most warmly for the care you put into making very helpful suggestions.

I would like to acknowledge all those who by their encouragement and efforts made the lecture recording possible. At the time we started there were few widely known exemplars of public videos of university courses outside of MIT-OCW (iTunesU and India's *nptelhrd* channel had only just started, in the case of iTunesU most content was audio only) and certainly none locally, and it would have been very easy for someone somewhere to have said "no, it's too hard" or "it won't work." Eliathamby Ambikairajah, the pioneer of lecture recording at UNSW, gave me much useful information about the impact of recording on students in the classroom and watching his recordings provided reassuring evidence that current camera resolutions were sufficient and that it could be done. Patrick Stoddart first suggested the idea of making the recordings public, Mary O'Malley, Tom Cavarovski, Mark Foster, Michael Rampe and Daniel Woo provided wise and practical advice on the mechanics of video recording, Paul Compton and Ashesh Mahidadia first thought of the idea of live mixing. David Collien and Theo Julienne were constant advocates for resuming outreach to show bored secondary school students the joy of computing in the same way they themselves had been inspired by the CSE computing club in the 1990s.

The biggest contributions of any individuals to the project was the heroic and entirely voluntary (they repeatedly refused payment) efforts by Rupert Shuttleworth and Thurston Dang, former students of the courses, who operated the cameras and live-mixed the lectures throughout the project. They set up and packed up the gear, sorted out hardware and software problems, and just generally made everything happen. Their efforts meant I never had to worry about recording when I needed to be worrying about the lecture. Furthermore they are intelligent and effective educators. I deeply valued our times travelling back together to my office after lectures to put away the recording equipment when we could talk about the course together, conducting a postmortem on the lecture that had just happened, and bouncing ideas around for what was still to come. Being open in my teaching practice and the consequent benefits from Thurston and Rupert being able to observe what had happened in lectures and discuss it in a rich manner was a compelling endorsement of open teaching.

## Appendix: Equipment List

**Camera:** 2 x Sony A1 HDV

**Tripod:** Miller + DS-5 head

**Wireless Microphone:** Sony UTX B1 / URX B1

**Computer:** Apple Mac Pro 2x2.4GHz Quad-Core

**Live Mixing:** WireCastHD by Telestream, Inc.

**Postproduction:** Adobe FinalCut Pro

## References

- Anderson, L., Krathwohl, D., Airasian, P., Cruikshank, K., Mayer, R., Pintrich, P., Rath, J. & Wittrock, M. (2001) *A Taxonomy for Learning, Teaching, and Assessing: A Revision of Bloom's Taxonomy of Educational Objectives*, Addison Wesley Longman.
- Anderson T. D., Parker N. & McKenzie J. (2009) *Assessing online laboratories: a peer review of teaching & learning* ATN Assessment Conference 2009, RMIT University.
- Battlewrench (2010), YouTube user, Comment on UNSWlearning channel, comment posted early 2009 (youtube does not show precise date). Accessed at comment page 9 (of 14) on <http://www.youtube.com/user/UNSWlearning> on 4 November 2010.
- Bloom B. S. (1956). *Taxonomy of Educational Objectives, Handbook I: The Cognitive Domain*. David McKay Co Inc., New York.
- Britzman, D. P. (1986), Cultural myths in the making of a teacher: Biography and social structure in teacher education, in *Harvard Educational Review*, 56(4), pp. 442–456.
- caesium (2010), YouTube user, Commenting on video *Lecture 1: Introduction to Data Structures and Algorithms - Richard Buckland*, comment posted January 2010. Accessed at [http://www.YouTube.com/comment\\_servlet?all\\_comments=1&v=RpRRUQFbePU](http://www.YouTube.com/comment_servlet?all_comments=1&v=RpRRUQFbePU) on 5 August 2010.
- Clement, M. & Vandenberghe, R. (2000), Teachers' professional development: a solitary or collegial (ad)venture?, in *Teaching and Teacher Education*, 16, pp. 81–101.
- Daya, C. (1993), Reflection: a necessary but not sufficient condition for professional development, in *British Educational Research Journal*, Volume 19, Issue 1, pp. 83–93.
- Gosling D. (2005), *Peer Observation of Teaching*. SEDA Paper 118: Staff & Educational Development Association Ltd, London.
- Gosling D. & O'Connor K. eds (2009), *Beyond the Peer Observation of Teaching*. SEDA Paper 124: Staff & Educational Development Association Ltd, London.
- Lesser, G. (1974). *Children and Television: Lessons from Sesame Street*. Random House, New York.
- Lovett M., Meyer O. & Thille C. (2008). JIME - The Open Learning Initiative: Measuring the Effectiveness of the OLI Statistics Course in Accelerating Student Learning in *Journal of Interactive Media in Education*, May 2008, 14.
- OECD (2004). *Science, Technology and Innovation for the 21st Century*. Meeting of the OECD Committee for Scientific and Technological Policy at Ministerial Level, 29-30 January 2004 - Final Communiqué.
- Patrick, F., Elliott, D., Hulme, M. & McPhee, A. (2010), The importance of collegiality and reciprocal learning in the professional development of beginning teachers, in *Journal of Education for Teaching*, Volume 36, Issue 3, pp. 277–289.
- Rogers D. L. & Babinski L. M. (2002). *from isolation to conversation: supporting new teachers' development*. SUNY Press, New York.



SkyRookie1 (2010), YouTube user, Commenting on video *Wikis in University Teaching and Learning - Richard Buckland UNSW*, comment posted 15 August 2010. Accessed at <http://www.youtube.com/watch?v=m1-800rBi0o> on 15 August 2010.

Uhlir, P. F. & Schroder, P (2008), Open Data for Global Science, Sydney University Press Law Books 39 in Brian Fitzgerald, Legal Framework for e-Research: Realising the Potential 188.

Vest, C. M. (2001), quoted in *MIT to make nearly all course materials available free on the World Wide Web* MIT news April 4 2001 <http://web.mit.edu/newsoffice/2001/ocw.html> (accessed 4 November 2010).



# A Study of Loop Style and Abstraction in Pedagogic Practice

**David J. Barnes**

School of Computing, The University of Kent,  
Canterbury, Kent. CT2 7NF, UK

d.j.barnes@kent.ac.uk

**Dermot Shinnars-Kennedy**

Department of Computer Science and Information  
Systems, University of Limerick, Limerick, Ireland.

dermot.shinnars-kennedy@ul.ie

## Abstract

This paper describes the results of a study into the use of structure and abstraction in the programming styles of lecturers and teaching assistants involved in teaching programming to students attending university and other third-level institutions. The study was motivated by the hypothesis that the trend towards object-orientation is being matched by pedagogic materials that consistently foster the deployment of abstraction and structure in the solution of programming problems. Unfortunately the evidence does not support the hypothesis. We conclude that the persistent use of abstraction at all levels of implementation is necessary to perfect expertise in its application and secure the benefits of the object-oriented paradigm.

**Keywords:** Abstraction, exit-in-the-middle problems, iteration, object orientation, pedagogy, structured programming.

## 1 Introduction

We have been using programming languages for well over half a century. It has been a period of persistent change during which the evolution of programming practice has had at least three identifiable 'ages': the programming-in-the-absence-of-discipline age; the importance-of-structure-and-abstraction age; and, the adoption-of-object-orientation age. The relative brevity of these ages emphasizes the speed with which we have experienced the transition from one to the next. During this period, the programming community has succeeded in creating artefacts of great variety and immense complexity, albeit not always with the preferred levels of reliability and utility. The history of these ages maps the maturation of the discipline and the emergence and recognition of abstraction as the "key to computing" (Kramer, 2007).

Ostensibly the history of programming pedagogy has proceeded hand-in-glove with developments in programming research and practice. For example, the widespread adoption of object-oriented programming as the paradigm for introductory programming courses in the late 1990s is one indication that the academic community is often anxious to support the changing

industrial landscape, parallel the tool deployment of practitioners and embrace curriculum development to offer students the greatest exposure possible to pivotal concepts like abstraction.

Honouring the research tradition that our role is to question and not to worship we sought to investigate the deployment of abstraction in the pedagogic materials and exemplars of programming teachers. Our motivation derived from the incidence of sample solutions in textbooks and other materials which were either equivocal in the use of abstraction or completely devoid of it. This appeared to lend credence to David Gries' recent summary of programming instruction which he described as a "muddled situation" and noted, "For 50 years, we have been teaching programming ... And yet, teaching programming still seems to be a black art ... In some sense, we are still floundering, just as we were 50 years ago." (Gries 2008)

In the following sections we provide some historical background to characterize three 'ages' of programming which led to the emergence of the centrality of abstraction; present the problem we set for participants in the study; describe the results we observed; and provide some discussion and conclusions.

## 2 Structure and abstraction

It seems reasonable to characterize the era of early machine-level programming, through to the earliest higher-level programming languages, as the absence-of-discipline age. Of greatest concern were the goals of squeezing code into tiny amounts of memory and saving machine cycles. It is customary to see an evolution from this primordial age to an age of the importance-of-structure-and-abstraction as having its origins in the publication of Dijkstra's famous letter (Dijkstra 1968) on the harmfulness of go to statements. One of Dijkstra's main concerns about their use was the resulting increased difficulty in bridging the intellectual gap between the static, textual representation of a program and its dynamic behaviour. In a subsequent work (Dahl, Dijkstra, and Hoare 1972) he observed, "The art of programming is the art of organizing complexity, of mastering multitude and avoiding its bastard chaos as effectively as possible." He identified *abstraction* as the key mental technique for dealing with complexity and "usefully structured" programs as the resultant objects of its application. In addition to facilitating the comprehension and control of complexity, abstraction and useful structure provide the framework for developing programs that are not just solutions to a specific problem but are members of a "family of related programs" that we can think of "as alternative programs for the same task or as similar programs for similar tasks." The fact that these insights

are nearly forty years old does not diminish their significance.

Dijkstra's views sparked an intense debate, which was dominated by arguments regarding the benefits, or otherwise, of writing programs with and without `goto` statements but which was, in essence, about structure and the application of abstraction. Sequential search of a list of values to locate the first instance of a given value was the canonical example used by virtually every contributor to the debate.

In terms of structure, all search algorithms have two identifiable components. The first is the repetitive component which performs the actual search, and the second is the reporting of the outcome. Abstracting the essential features of the search component yields two possible termination scenarios (1) when the search space is exhausted (i.e., the end of the list is reached) or (2) when the search requirements are satisfied (i.e., the value is located).

With respect to the chosen example (i.e., sequential search) much of the debate centred on how a successful search should be handled. One side argued that successful search should be treated as an exception implemented as an early or 'premature' exit from within the repetition using, for example, a `goto` or `break` or their equivalents. Thus Knuth (1974) noted, "in the general case it is a nuisance to avoid the `goto` statements". The counter-argument treated successful search as one possible reason for a 'normal' termination of the repetition using a compound condition in the iteration construct. For example, Wirth (1974) argued, "Often the need for an exit in the middle construct is based on a preconceived notion rather than on a real necessity, and that sometimes an even better solution is found when sticking to the fundamental constructs."

Because of these characteristics, search algorithms are commonly referred to as 'n-and-a-half', 'loop-and-a-half' or 'exit-in-the-middle' problems and might be programmed in Java, for instance, in the two alternative ways shown in Figure 1. (The use of `while` is a matter of taste; a `for` construct could be used with the same effect.) It is worth noting that the `goto`-less version preserves the separation of concerns associated with the search and reporting components. In contrast, the `goto` version offers the option of conflating the detection and reporting of a successful search into a single operation (i.e., `return i` shown as an alternative inside the `while` loop).

From a pedagogic perspective the most influential contribution to the debate on the appropriateness or otherwise of exit-in-the-middle strategies was published by Soloway, Bonar, and Ehrlich (1983). They reported on an empirical study on the cognitive fit between programming language constructs and novice programmers' preferred strategy. Among other things, they concluded that, "Students write programs correctly more often using a construct that permits them to exit from the middle of the loop." This study is frequently cited in discussions concerning pedagogic approaches to looping constructs and their application (for example, see Roberts (1995) and the novice language GRAIL (McIver and Conway 1999) whose only repetition structure is a Soloway-style loop) despite Wirth's (2006) warning that

"[It] defies any regular structure, and makes structured reasoning about programs difficult if not impossible."

```
// goto-less form of loop-and-a-half
public int seqSearch(int[] values, int x)
{
    int i = 0;
    while(i < values.length &&
           values[i] != x) {
        i++;
    }
    return i < values.length ? i : -1;
}

// goto (break) form of loop-and-a-half
public int seqSearch(int[] values, int x)
{
    int i = 0;
    while(i < values.length) {
        if(values[i] == x) {
            break;
            // Alternatively, return i;
        }
        i++;
    }
    return i < values.length ? i : -1;
    // Alternatively, return -1;
}
```

**Figure 1: Typical structure of 'loop-and-a-half' problem solutions**

The transition to the adoption-of-object-orientation age places abstraction and structure at the centre of everything. Partitioning state space into objects and identifying the abstract behavioural aspects of those objects are the central tenets of the object-orientated paradigm. They support encapsulation, inheritance and ultimately generic programming and, thereby, provide the infrastructure to realise programming solutions that are, as Dijkstra urged, not solutions to a specific problem but members of a "family of related programs" that we can think of "as alternative programs for the same task or as similar programs for similar tasks."

### 3 An informal study of personal practice

In an effort to see how influential the earlier ages of the development of programming expertise have been, and to explore the application of structure and abstraction in the programming styles of those involved in teaching programming, we ran a small informal exercise with ten academics from several Universities. With no time limit we asked them to code solutions to a problem specified by Yuen (1994), details of which can be found in the appendix. The solutions, written in a language of their choice, were submitted electronically.

The problem involves a slight extension to the standard loop-and-a-half example of determining whether or not a particular value exists in an array. The extension involves finding whether a value occurs up to two times in the array, and reporting the number of occurrences. In addition, if the value occurs twice, it is necessary to report whether the gap between the occurrences is even or odd. The problem has the attraction of being simple enough to use in a short exercise.

In a series of papers, Yuen (1983, 1984, 1994) documented solutions to various problems based on an exit-in-the-middle strategy using go to statements, and provided an extensive rationale for his preference for that strategy over a more ‘abstract’ or ‘structured’ approach.

The temporal aspects of our problem choice are important. Yuen was writing during the period of transition from the ‘programming-in-the-absence-of-discipline’ age to the ‘importance-of-structure-and-abstraction’ age. Since then we have transitioned to the ‘adoption-of-object-orientation’ age. We hypothesized that the evolution of programming expertise would impact on the type of solutions we would receive. We anticipated solutions exhibiting judicious application of the principles of useful structure and abstraction. In short, we expected solutions with a markedly different style to those published originally by Yuen.

### 3.1 The “Anticipated” Solution Style

Figure 2 is illustrative of the type of solution we expected (omitting the output details, which are not germane to the following discussion). It is a modified version of the ‘go-to-less’ sequential search solution shown in Figure 1. The modifications represent a refinement of the abstraction levels required in the original problem. Termination of the while loop is still dependent on an ‘and’ condition. The first part remains a test for the end of the array while the second part has been altered to determine if the search value count has reached two. The loop body has been altered to give effect to the new circumstances. In addition to incrementing the array index it tests for occurrences of the required value and increments a value count accordingly.

```
public void findZeroes(int[] values)
{
    int count = 0;
    int[] pos = new int[2];
    int i = 0;

    while(i < values.length && count < 2) {
        if(values[i] == 0) {
            pos[count] = i;
            count++;
        }
        i++;
    }

    switch(count) {
        case 0 : // report no occurrences
            break;
        case 1 : // report one occurrence
            break;
        case 2 : // report two occurrences
            int gap = pos[1] - pos[0];
            // perform odd/even calc's ...
            break;
    }
}
```

**Figure 2: The anticipated style of solution**

To report the parity of the gap between two occurrences of the search value it is necessary to record their positions. It is possible to achieve this using two

simple variables. However, applying the principles of abstraction to the problem specification we are conscious that the current problem is an instance of a family of problems intended to locate N occurrences of the search value. This explains the introduction of an array for the search value positions.

### 3.2 The Participants' Solutions

Participants were given a free choice over the language they used, and the solutions we received were all coded in either Java or C++. There turned out to be little consistency in any elements of the solutions: the type of control structures; the number of loops; or the variables used to collect the data for reporting purposes. The solution shown in Figure 3 is illustrative of the solutions submitted. Table 1 summarizes the various search methods among these solutions. We have excluded one solution as it coded a separate method for each case (including two almost identical methods for the odd and even gap cases) and employed a different approach for each of the no-, one- and two-zeros cases. In essence, this was an example completely devoid of any structure or abstraction, and we categorize it as an outlier.

```
public void findZeros(int[] values)
{
    int pos1 = -1;
    int pos2 = -1;

    for(int i = 0; i < values.length; i++) {
        if(values[i] == 0) {
            if(pos1 < 0) {
                pos1 = i;
            }
            else {
                pos2 = i;
                break;
            }
        }
    }

    if(pos2 >= 0) {
        // Two zeros ...
    }
    else if(pos1 >= 0) {
        // One zero ...
    }
    else {
        // No zeros ...
    }
}
```

**Figure 3: A typical solution from the informal study**

Search methods	#
Single for-loop, with break in the middle	3/9
for-loop, scanning the whole array but recording only first two zeros	1/9
for-loop, with loop condition testing number of zeros found	2/9
Two while-loops with loop condition testing for a zero	1/9
Two for-loops with break in the middle	1/9
Library search method to find a zero	1/9

**Table 1: Loop structure in the informal study**

Seven solutions used for-loops, one used a pair of while-loops, and one avoided coding the search explicitly by using a library search method. Of the for loop solutions only two used a compound condition testing for the end of the array and whether the zeros had been found yet. The other five either used a break from the middle of the loop or continued to the end of the array even if two zeros had been found (the additional zeros were ignored). Both of the non-for loop solutions used a pair of loops to search for the first and then the second zero.

The use of variables also provides insights into stylistic preferences. Table 2 describes how variables were used, both to manage the flow of control and collect data for the output requirements. Typically, solutions either recorded the positions of the two zeros (*p1p2*; see Figure 3) or a combination of the number of zeros found (*nz*) along with either the position of the first (*p1*) or the distance (*gap*) between them. One of the solutions employing two loops simply toggled a Boolean variable (*even*) for the gap while looking for the second zero.

Id	Purpose	#
nz	number of zeros found	4/9
gap	running count of non-zeros following a zero	2/9
p1	position of first zero	1/9
p1p2	two variables recording positions of the zeros	6/9
even	whether the gap is even or odd	1/9

**Table 2: Use of variables in the informal study**

On the basis of this informal study we made two observations:

- If we ignore the solution that used a library call, half of the solutions were coded using a break from the middle of the loop. The associated break was often 'buried' some way down in the body of the loop (as can be seen in Figure 3).
- All of the solutions solved the *specific* problem that had been set. For instance, all were written to deal with a maximum of two zeros – none sought to use a data structure to record the positions of the zeros or parameterized their methods on the number of zeros to be located. This lack of generalization is particularly noticeable in the solutions that coded separate loops or used two library calls to search for the first and second zeros.

Both observations, and the disparity between the anticipated solution and the actual solutions, caused us a degree of concern. We began to wonder whether these personal practices might also be typical of *pedagogic* practice, so we conducted a second study in order to test the waters in this respect.

#### 4 A study of pedagogic practice

We set the same problem of searching for up to two zeros to groups of academics and postgraduate students at a University that has been teaching object-orientation at introductory level for over ten years. Postgraduates were included in the study because it is common for them to provide support with programming classes, and these students also tend to be recent graduates, reflecting

current academic practice. Staff and postgraduate mailing lists were used to invite voluntary, anonymous participation in the study. All submissions and questions about the exercise were handled anonymously without at any point identifying the participants, other than whether they were a member of staff or a student.

Because we were concerned that the informality of the first study might simply have lead to solutions being offered that actually had no bearing on pedagogic practice, we made an addition to the problem specification. We asked the participants to:

“Bear in mind that [your solution] should be the sort of thing you would be willing to show to introductory programming students as an example of a 'good' solution.”

The idea was to avoid any suggestion that this exercise was just about creating a program that works. It was intended to emphasize that the solution should display some degree of pedagogic value. We indicated that this was part of a research study and that solutions might be used in publications, but did not otherwise give any further motivation. Solutions were received electronically and no time limit was imposed on their production. This allowed subjects to compile and test their solutions before submission, should they wish to do so.

Sixteen submissions were received that were amenable to analysis, written in C++, Java and Python. Eight were from postgraduate students and eight from members of staff. We will not distinguish further between these groups because there was actually little difference to be observed in the styles of their solutions. Table 3 summarises the preferred loop structures in these submissions.

Loop structure	#
for-loop, with break or return in the middle	10/16
for-loop, scanning the whole array but recording only first two zeros	4/16
for-loop, with loop condition testing number of zeros found	1/16
for-loop, recording the positions of <i>all</i> zeros	1/16

**Table 3: Preferred loop structure in the second study**

The preference for using a for-loop over the full range of the array is complete – in contrast to the first study, no solution used an explicit while-loop. In some respects this strong feature of both sets of solutions is not particularly remarkable. While it was common in Pascal to see a clear distinction between the use of a for-loop for a definite (pre-determined) number of iterations and a while-loop for an indefinite number, the C family of languages actually blurs the distinction, and the regular for-loop (as opposed to for-each) is often taught as simply being a syntactic variant of the while-loop. Nevertheless, there is still often quite a strong association in the mind of readers of code between a for-loop and definite iteration.

Of more significance in the second study is the almost complete avoidance of an augmented loop condition to finish the search once two zeros had been identified – just one solution. Ten solutions (60%) used an embedded break, as had four (44%) in the previous study. The four solutions in the second study that used neither an augmented condition nor a break continued scanning after

finding two zeros, which meant that they had to code around losing the position or gap information they had recorded; thus making the loop body more complicated than other solutions. Notable is that one solution used a list to record the positions of all zeros and not just the first two. In C++, Java and Python this solution is particularly easy to code using a dynamic data structure, such as Java's ArrayList, which has the useful additional benefit of keeping track of the number of occurrences found.

Table 4 documents the ways in which variables were used. There is more variety here than in the first study. Two solutions used a list (a 2-element array, in one case) rather than separate variables for the positions. In addition, there were four examples of the style we have labelled *p1-flag* (see Figure 4). This can be seen as a minimalist approach to variable usage: a negative value of the position variable indicates that no zero has been found yet; a non-zero value indicates that a single zero has been found, and this value is then used, along with the current loop variable, to compute the gap when the second zero is found.

Id	Purpose	#
nz	number of zeros found	7/16
seen	whether a zero has been found yet	3/16
gap	running count of non-zeros following a zero	6/16
p1	position of first zero	2/16
p1-flag	position of first zero, with out-of-bounds semantics	4/16
p1p2	two variables recording positions of the zeros	3/16
collection	positions of the zeros	2/16

**Table 4: Use of variables in the second study**

```

public String findZeros(int[] values)
{
    int pos = -1; // position of first zero.

    for(int i = 0; i < values.length; i++) {
        if(values[i] == 0) {
            if(pos < 0) {
                pos = i;
            }
            else {
                if(((i - pos) & 1) == 0) {
                    return ... // Two - odd gap.
                }
                else {
                    return ... // Two - even gap
                }
            }
        }
    }

    if(pos < 0) {
        return ... // No zeros
    }
    return ... // One zero
}
    
```

**Figure 4: A solution from the second study (p1-flag)**

It has to be admitted that the results of this repeat of the original experiment were a considerable surprise to us. We had assumed that the characteristics we had observed in the submissions to the first study were, in effect, 'quick and dirty' solutions to a relatively simple problem. We anticipated that the addition of the 'good solution' rider would result in our seeing quite different solutions – akin to exemplars that would achieve full marks in a student assignment. On the contrary, there was little material difference between the two sets of solutions. We saw just one solution out of sixteen that we felt came close to genuinely being 'a good solution', i.e., using a loop with a single exit point and a collection to store the positions of zeros. While it could be argued that we had not made it clear enough what we were expecting, the risk of being too specific is that the results will not accurately illustrate the participant's normal practice in a teaching situation. In other words, we were not interested in whether they *could* write a good solution but whether they *would*.

## 5 Discussion

Despite the limited nature of this study, it appears to us that several themes emerged strongly and that may well be evidence of widespread pedagogic practice:

- While the problems have an *indefinite* character, there was an overwhelming preference for definite solutions based on using a for-loop that appeared to operate over the full array.
- There was a strong preference for exit-in-the-middle solutions.
- Where an exit-in-the-middle condition was used, it was *always* a physically separate and distinct test from the test for reaching the end of the data structure.
- The solutions we saw were often closely tied to the fact that only two zeros had to be identified; for instance, using either one or two variables to record the zeros' positions, rather than a collection.
- The temptation to conflate aspects of the solution was irresistible for some participants. For example, in the first study one participant toggled a boolean variable to record the parity of the gap whilst searching the list. In the second study, as Figure 4 illustrates, embedding details of the reporting requirements in the search phase was not unusual.

Our view is that these themes are, in fact, part and parcel of a single issue – an approach to program design that focuses on the immediate details of the task at hand rather than on the properties of a more general version of the task. In this study the specific task was to locate the positions of up to two zeros in an array. However, the search aspects are clearly instances of the more general problem of finding the positions of up to  $N$  occurrences of a value in a collection. What we believe we are seeing here is a failure to use abstraction and useful structure in program design at the method level.

Our preferred solution to the problem presented is shown in Figure 5. Employing abstraction, the problem is partitioned into two components or methods. The first is a

customisable search mechanism, with the search value and the required number of occurrences provided as parameters of the search invocation. The features of Java's dynamic List structures are used to store the positions of whatever number of values must or can be recorded. The list is returned as the result of the search (the list could, of course, be converted to an array of int if that would be more convenient.) The second method localises the specific reporting obligations and extracts whatever reporting information is required from the list received as its parameter.

```
/**
 * Find up to max occurrences of x in values.
 * Return the positions of the occurrences.
 */
List<Integer> findValues(int[] values,
                        int x, int max)
{
    List<Integer> pos =
        new ArrayList<Integer>();
    int i = 0;

    while(i < values.length &&
        pos.size() < max) {
        if(values[i] == x) {
            pos.add(i);
        }
        i++;
    }
    return pos;
}

void reportResults(List<Integer> pos)
{
    switch(pos.size()) {
        // Cases for each reporting option.
        ...
    }
}
```

**Figure 5: A solution to all the search-for-n variations**

This is a solution to the “family” of problems defined by the Yuen specification. Its strength is in its simplicity; its power in its malleability; its attraction in its treatment of specifics. The solution could equally well have been coded in any one of several programming languages, including those not categorised as object-oriented. However, this particular solution clearly illustrates a mix of fundamental procedural and object-oriented elements that are appropriate even at a relatively early stage of an introductory programming course. For instance, see the introductory textbook by Barnes and Kölling (2008) where basic iteration and dynamic collections are introduced together.

One important feature of the solutions submitted in the studies is that they all worked – they were correct solutions. This is an important outcome and it should not be overlooked or undervalued. An essential property of any programming solution is that it achieves the desired result. However, as a maturing discipline, software engineering has established and substantially documented the importance of qualities such as, maintainability, portability, adaptability, reusability, flexibility. These are all premised on the application of abstraction. Yet, as the foregoing analysis highlights, the commitment to

abstraction evident in the solutions submitted was equivocal.

For example, the abstract characteristics of all search algorithms centre on the iterative deployment of a search strategy until the search space is exhausted or the required item is found – whichever occurs first. These characteristics were realized in a potpourri of styles but the dominant one separated the detection of the conditions using an exit-in-the-middle mechanism. Only one solution, submitted in response to the second exercise, implemented the iteration in a manner consistent with the abstracted characteristics of a search.

Similarly, recording the positions of the located search values was typically handled using simple variables. Only two participants in the second exercise adopted a collection approach to the recording of the positions.

Significantly, none of the solutions parameterized the search to facilitate variation of the search value or the number of occurrences required. Neither did any of the solutions partition the problem by separating the searching process from the reporting process. In fact, many of the solutions conflated the two processes and implemented them as a single embedded block.

Of course, it is always easy to offer criticism of others' programs and that is not our purpose. Our purpose is to highlight the application of abstraction. Our conclusion is that, despite all of our aspirations and justifications for the adoption of programming tools that provide extensive support for the exploitation of abstraction, we have not, as a community, developed a premeditated disposition for its application. In fact, what we have developed is an approach founded on the principle that when we really need it we will be able to recognize that need and apply it appropriately. We are operating on “just in time abstraction.” That is a precarious perch to position ourselves on.

We suspect that this is (in part, at least) a result of an ‘incomplete birthing’ of the age of object-orientation. A noticeable, and potentially unhelpful side-effect of the arguments into the relative merits of procedural and object-oriented approaches has been that they are sometimes treated as completely distinct animals. This effect is observable, for instance, in a relatively recent catalogue of novice OO misconceptions (Sanders and Thomas 2007) where procedural elements are almost entirely omitted. Yet novices must also grapple with the basic ‘procedural’ elements of methods, such as managing flow of control, where misconceptions are just as likely. Indeed, as Garner, Haden, and Robins (2005) found, it was such procedural elements that often caused more problems than the object-oriented ones. It may well be that in our desire to teach good and genuine object-orientation there has been an unintended neglect of the need for good practice in those ‘procedural’ elements that are an almost inevitable part of any OO program.

## 6 Conclusion

Dijkstra (1989) once observed that, “Breaking out of bad habits, rather than acquiring new ones, is the toughest part of learning.” If we are in the habit of neglecting the deployment of abstraction techniques at the lower levels of design then our ability to deploy them at the higher levels may be compromised. The acquisition of an



habitual tendency for the application of abstraction is a formative process that demands persistent nurturing and pervasive instantiation. We cannot turn it on and off.

The importance of practice is an established mantra of programming teachers. Lack of practice can hinder the development of experience and competence. Indeed, just as an athlete or an artist develops reliable technique through repetition, the act of consistently applying abstraction – whatever the problem – is what ultimately enables a programmer to employ it with greater facility. Gladwell (2008) cites the “10,000 hour rule” which asserts that “ten thousand hours of practice is required to achieve the level of mastery associated with being a world-class expert – in anything.” He also notes that, “Practice isn’t the thing you do once you’re good. It’s the thing you do that makes you good.”

Lamenting a general lack of application of discipline in the field of software engineering, recently, David Parnas laid the blame firmly in the court of teachers, saying that, “We need to: teach [students] what to do and *how* to do it – even in the first course [and] use those methods ourselves in *every* example we present.” (Parnas 2010)

A particular motivation for striving for understanding in problem solving is the need to manage the increasing complexity of software systems that we are now able to build. As computer science graduates leave university and go on to work on real projects with potentially massive impacts on society, those of us who teach have a responsibility to ensure that the mindset they take with them will lead to the creation of comprehensible software artefacts. In fact, this argument is really no different from those made nearly fifty years ago over how to deal with the ‘software crisis’ (Randall 1996) – a phrase we don’t hear so often now, but which is surely just as pertinent now as all that time ago?

We believe that it is essential for teachers of introductory programming to revisit the ideas of useful structure and abstraction in order to ensure that students are taught to apply it from the ground up in designing solutions to software problems.

We conclude with the observation that practice in the current age of object-orientation may have forgotten something of the conclusions of the preceding ages and needs to revisit them in order to adequately equip students with a full set of programming skills. This is not to say that the preceding ages always lived up to the conclusions that were reached – there was no ‘golden age’ – but that OO practitioners need to pay just as much attention to them.

## 7 Acknowledgements

We would like to express our sincere appreciation to the participants in our studies, who allowed us to analyze their solutions.

## 8 Appendix

Yuen’s zeroes problem (Yuen 1994) as we set it in both the informal and the second study:

Inspect an array of  $N$  elements to find which one of the following is true and output a message identifying the

case. Arrays indexed  $1..N$  or  $0..(N-1)$  are equally acceptable:

- a. It contains no zeros.
- b. It contains only one zero.
- c. It has two zeros separated by an even number of non-zeros.
- d. It has two zeros separated by an odd number of non-zeros.

Clarification questions asked by the subjects in the second study included what should be returned if there were more than three zeros. The response was that the gap between the first two was the only item of interest in this case.

## 9 References

- Barnes, David J. and Kölling, Michael (2008): *Objects first with Java - a practical introduction using BlueJ*. Pearson Education Ltd.
- Dahl, O. J., Dijkstra, E. W., and Hoare, C. A., Eds. (1972): *Structured Programming*. Academic Press Ltd.
- Dijkstra, E. W. (1968): Letters to the editor: go to statement considered harmful. *Communication of the ACM* 11(3):147-148.
- Dijkstra, E. (1989): On the Cruelty of Really Teaching Computer Science. *Communications of the ACM* 32(12):1398-1404.
- Garner, S., Haden, P., and Robins, A. (2005): My program is correct but it doesn’t run: a preliminary investigation of novice programmers’ problems. *Proceedings of the 7th Australasian Conference on Computing Education*, Newcastle, New South Wales, Australia, 106:173-180, Australian Computer Society.
- Gladwell, M. (2008): *Outliers – The Story of Success*. Little, Brown and Company: New York.
- Gries, David (2008): Foreword. In *Reflections on the Teaching of Programming*. Bennedsen, J., Caspersen, M.E., and Kölling, M. (eds). Springer, 978-3-540-77933-9.
- Knuth, D. E. (1974): Structured Programming with goto Statements. *ACM Computing Surveys* 6(4):261-301.
- Kramer, J. (2007): Is abstraction the key to computing? *Communications of the ACM* 50(4): 36-42.
- McIver, L. and Conway, D. (1999) GRAIL: A Zeroth programming language. *Proceedings of the International Conference on Computing in Education (ICCE99)*, 43-50.
- Parnas, D. L. (2010): Risks of undisciplined development. *Communications of the ACM* 53(10):25-27
- Randall, B. (1996): The 1968/69 NATO Software Engineering Reports. Dagstuhl-Seminar 9635: History of Software Engineering, Schloss Dagstuhl, Germany, August 26 - 30, 1996.
- Roberts, E. S. (1995): Loop exits and structured programming: reopening the debate. In *Proceedings of the Twenty-Sixth SIGCSE Technical Symposium on Computer Science Education*, Nashville, Tennessee, USA, 26:268-272.

- Sanders, K. and Thomas, L. (2007): Checklists for grading object-oriented CS1 programs: concepts and misconceptions. *SIGCSE Bulletin* **39**(3):166-170.
- Soloway, E., Bonar, J., and Ehrlich, K. (1983): Cognitive strategies and looping constructs: an empirical study. *Communications of the ACM* **26**(11):853-860.
- Wirth, N. (1974): On the Composition of Well-Structured Programs. *ACM Computing Surveys* **6**(4):247-259.
- Wirth, N. (2006): Good Ideas, through the Looking Glass. *Computer* **39**(1):28-39.
- Yuen, C. K. (1983): The programmer as navigator: a discourse on program structure. *ACM SIGPLAN Notices* **18**(9):70-78.
- Yuen, C. K. (1984): Further comments on the premature loop exit problem. *ACM SIGPLAN Notices* **19**(1):93-94.
- Yuen, C. K. (1994): Programming the premature loop exit: from functional to navigational. *ACM SIGPLAN Notices* **29**(3):23-27.

# Salient elements in novice solutions to code writing problems

Jacqueline Whalley<sup>†</sup>, Tony Clear<sup>†</sup>, Phil Robbins<sup>†</sup>, and Errol Thompson<sup>\*</sup>

<sup>†</sup>School of Computing and Mathematical Sciences  
AUT University  
PO Box 92006, Auckland 1142, New Zealand  
{jwhalley, tclear, probbins}@aut.ac.nz

<sup>\*</sup>School of Computer Science  
The University of Birmingham  
Birmingham, B15 2TT, United Kingdom  
kiwiet@acm.org

## Abstract

This paper presents an approach to the evaluation of novice programmers' solutions to code writing problems. The first step was the development a framework comprised of the salient elements, or programming constructs, used in a set of student solutions to three typical code writing assessment problems. This framework was then refined to provide a code quality factor framework that was compared with an analysis using the SOLO taxonomy. We found that combining our framework with the SOLO taxonomy helped to define the SOLO categories and provided an improved approach to applying the principles of SOLO to code writing problems.

**Keywords:** SOLO taxonomy, novice programmers, assessment.

## 1 Introduction

This paper furthers one aspect of the work of the ITiCSE 2009 Paris working group (Lister et al., 2009) in classifying novice student responses to program code writing exercises. The aim of classifying student responses is: 1) to better understand typical patterns of response; 2) to better understand how students typically approach code writing questions; 3) to derive repeatable measures of novice student performance on assessment tasks; 4) to apply a pedagogically accepted theoretical framework to this investigation; 5) to identify areas where students commonly experience difficulty.

As a result of the above insights we would hope eventually to: 1) develop more effective teaching and learning strategies; 2) use this knowledge in order to develop more consistent expectations of novice student performance; and 3) to assist in the design of fair and appropriate assessment instruments in examinations and tests.

## 2 Background

The Paris working group (Lister et al., 2009) applied recognised educational frameworks (Bloom and SOLO) as classification schemes for mapping examination questions and novice student responses. The work of the group extended from code comprehension questions to initial attempts to address code writing questions. These attempts adopted a top down strategy in order to directly

map each student response to a SOLO level. While they achieved a mapping for three very distinct questions and came to a consensus between four raters, the process seemed very context bound and question specific. In subsequent work we have revisited this approach and some of the underlying assumptions.

In this paper by contrast, we begin with the student response as raw data and build from that basis in a grounded manner to empirically derive a framework from the students' own work, before attempting as a subsequent step a SOLO classification of the responses given.

## 3 Methodology

Analysis of program understanding has been defined as “identifying artefacts and understanding their relationships; this process is essentially pattern matching at different abstraction levels” (Tilley 2000). More recently Meerbaum-Salant et al., (2010) have used a form of content analysis (Stemler 2001) of written text in a study which systematically analysed students' written code, and categorised their solutions according to the SOLO and Bloom taxonomies.

The study we report here, while adopting similar methods to *content analysis*, is better defined as a study applying *grounded theory*. Grounded theory (GT) is a method for empirically deriving theory from data, typically through applying an inductive and rigorous process of coding and categorisation. GT was originally conceived as an analytical and conceptual, creative process of constant comparative coding by Glaser and Strauss (1967). They have asserted that “grounded theory allows no speculation...one can be just as systematic with qualitative data as with quantitative data” (p. 200) and in generating theory the key position is that “the theory should fit the data” (p. 201) and not vice versa.

Given our goal of starting with the students' own code to derive the underlying patterns, GT presented an appropriate research method. A bottom up approach to the analysis of student responses to three code writing questions was undertaken. In outlining strategies for GT analysis Glaser and Strauss (1967, pp. 62-63) made the following distinctions between “sampling strategies”:

*“It is important to contrast theoretical sampling based on the saturation of categories, with statistical (random sampling). Theoretical sampling is done in order to discover categories and their properties, and to suggest the interrelationships into a theory.”*

*“The adequate theoretical sample is judged on the basis of how widely and diversely the analyst chose his groups for saturating categories according to the type of theory he wished to develop.”*

To simplify this initial investigation we examined only solutions that were complete and would compile and run. All other solutions were incomplete and showed that students lacked knowledge of basic programming constructs. Therefore SOLO analysis of seriously flawed responses would give inconsistent categorisations. The unit of analysis was the segment of code comprising the student response to a selected question. A subset of solutions was selected for analysis, comprising questions which exercised different programming constructs. We believe that we achieved saturation in the programming constructs from the subset we chose (see Section 4). Student responses were selected from questions that were posed to elicit responses at different SOLO levels. The goal was to establish a set of empirical categories comprised of salient programming elements. The student responses were coded to reflect the primarily syntactic categories of salient programming elements that emerged from the constant comparative analysis of the data.

These salient elements coded at the syntactic level were subsequently condensed through a form of feature extraction into broader concepts or 'code quality' categories. Those categories were used to represent the patterns of code used by novice programmers.

The students' responses were subsequently classified independently, by three researchers, using the SOLO taxonomy. We were interested to see if an existing taxonomy would be sufficient or if the salient elements and quality factors made it easier to measure the level at which a student was answering code writing questions.

The three researchers then met to reconcile the differences in the resultant SOLO codings and to achieve a consensus on their ratings, based upon agreeing common interpretations of the definitions in Table 1. This session was further informed by codings from an additional researcher working remotely. Once a consistent set of understandings had been derived a basis for presenting the findings of this study was achieved.

### 3.1 The SOLO taxonomy

The SOLO taxonomy describes levels of increasingly integrated thinking in a student's understanding of a subject, through five stages. Biggs (1999) describes the types of verbs that apply for each of the levels of the taxonomy (p. 47) and provides an example of ordering outcome items by the taxonomy (pp.176-178). These levels are placed into two phases suggesting that learning passes through various stages from a more quantitative phase (surface) to a more qualitative one (deep, connecting and relating ideas) as learning tasks and their complexity increase. Hattie and Purdie (1998, p. 156) provide a number of examples of the use of the SOLO taxonomy.

More recently SOLO has been used to reliably classify code reading questions and the student responses to those questions (Clear et al., 2008; Sheard et al., 2008). An initial set of guidelines and descriptors for using SOLO to classify student code writing solutions (Table 1) were proposed by Lister et al. (2009). These descriptions are the ones that we initially employed to independently classify the student responses to the questions discussed here.

When the SOLO taxonomy is applied to short segments of code, the learner needs to have more than a working solution; they need to show an understanding of the types of constructs that best implement the solution, to utilise program structures that communicate the intent of their code to others, and produce code that is easy to maintain.

Additionally the phrasing of the problem itself is critical to the possible SOLO level for a response. In analysing code writing tasks, it was found that the nature of the question had an impact on the type of solutions that were possible (Lister et al. 2009). Some questions did not allow for much more than a direct translation into the programming language while others allowed for greater interpretation. This supports the view that a question can be posed to elicit responses at a given SOLO level. The categories defined by Lister et al. (2009) were based on an example for language translation from Hattie and Purdie (1998). Hattie and Purdie (1998) argue that the shift through the SOLO levels shows an increasing understanding of how the phrase should be interpreted rather than just translated. This shift shows an increasing awareness of the relationship between the words and how that relationship communicates meaning.

Phase	SOLO category	Description
Qualitative	Extended Abstract – Extending [EA]	Used constructs and concepts beyond those required in the exercise to provide an improved solution
	Relational - Encompassing [R]	Provides a valid well structured program that removes all redundancy and has a clear logical structure. The specifications have been integrated to form a logical whole.
Quantitative	Multistructural - Refinement [M]	Represents a translation that is close to a direct translation. The code may have been reordered to make a valid solution.
	Unistructural – Direct Translation [U]	Represents a direct translation of the specifications. The code will be in the sequence of the specifications.
	Prestructural [P]	Substantially lacks knowledge of programming constructs or is unrelated to the question.

**Table 1: SOLO categories for code writing solutions**

Although Table 1 represents the SOLO categories as though there were distinct boundaries, it should be recognised that what is represented is actually a continuum (Thompson 2010, Meerbaum-Salant et al. 2010). Supporting this view, a statistical analysis of SOLO levels for code comprehension questions conducted by the Paris working group (Lister et al., 2009), confirmed the ordinality of the SOLO scale.

## 4 Analysis of student code writing solutions

The data consisted of exam questions and students' answers in programming courses offered in New Zealand and Finland. Three exams were used from introductory

programming courses. The answers of nearly 750 students were available for analysis. The programming languages covered were Java, Perl and Python. All of the exams included code tracing questions, most included code-explaining, code writing questions and Parsons questions. As noted in Section 3 above a theoretical sampling strategy was adopted with a proportion of the answers from each examination chosen for further analysis. Three typical CS1 code writing problems were selected from the above corpus as representative of different programming constructs and a progression of technical difficulty. What follows is a detailed discussion of the analysis of each of these three questions applying the methodology described in section 3.

#### 4.1 Discount

The discount problem was taken from a written examination for first semester (CS1) students where the languages used were Python and Finnish. A subsample of valid student responses was analysed. Forty eight responses were analysed and exemplars of typical student responses are given in Table 2 (refer to the appendix for the less common code pattern examples, we have chosen to leave these in Finnish as these constitute the raw data).

*A shop gives reductions of the prices as follows: if the original price of an item is at least 100 Euros but less than 200 Euros, the reduction is 5%. If the original price is at least 200 Euros then the reduction is 10%.*

Pattern	Typical Code Example
3	<pre>def main();     hinta = raw_input("Anna alkuperäinen hinta.");     hinta = float(hinta);     if hinta &gt;= 200:         hinta = 0.90*hinta     elif hinta &gt;= 100:         hinta = 0.95*hinta     print "Hinta Alennettuna:", hinta main()</pre>
5	<pre>def main();     rivi = raw_input("Anna alkuperäinen hinta.");     hinta = float(rivi);     if hinta &gt;= 200:         uusihinta = 0.90*hinta     if hinta &gt;= 100 and hinta &lt;200:         uusihinta = 0.95*hinta     if hinta &lt; 100:         uusihinta = hinta     print "Tuotteen alennettu hinta on:", uusihinta main()</pre>
6	<pre>def main();     rivi = raw_input("Anna alkuperäinen hinta.");     alku_hinta = float(rivi);     if alku_hinta &lt; 100:         print "Hinta on", alku_hinta, "euroa"     elif alku_hinta &lt;200:         hinta = 0.95*alku_hinta         print "Alennettu hinta on", hinta, "euroa"     elif alku_hinta &lt;200:         hinta1 = 0.90*alku_hinta         print "Alennettu hinta on", hinta1, "euroa"     main()</pre>

**Table 2: Example of the most prevalent ‘discount’ solution patterns**

Applying a grounded theoretic strategy a set of empirical codes were derived based primarily upon the syntactic constructs present within the student responses. This resulted in the set of salient elements portrayed in Table 3. The syntactic elements are shown in the second column of the table with one broader grouping based on function or purpose of the code in column one. An abstraction beyond the salient elements is given in column three where key features have been extracted based on the contribution of the salient element to the quality of the end code.

As can be extrapolated from Table 3, focusing on the selection function, solutions that used two selection clauses were preferable and those with three selection clauses had one unnecessary clause. The selection function class was therefore split into two quality factors without or with redundancy respectively. With reference to the printing and calculation functions better solutions had one print or calculation statement outside of the selection statement to remove code repetition. If a discount subroutine was written then the solution to the discount calculation was also judged to be a generalised solution.

Function	Element	Quality Factor
selection	if/else if	no redundancy
	2 x if	
	if/else if/ else	redundancy
	if/else if /else if	
	3 x if	
	& or   used	
discount calculation	in a subroutine	generalised
	in each selection clause	redundancy
	one calculation	no redundancy
printing	in each selection clause	redundancy
	one statement after selection functionality	no redundancy

**Table 3: Salient element framework for the ‘discount’ problem**

In the next step, Table 3 has been condensed to allow us to map student code to the function and quality framework. Using this framework, six patterns of code construction were observed.

Function	Quality Factor	Code Pattern					
		1	2	3	4	5	6
selection	no redundancy	X	X	X			
calculation	generalised	X					
	no redundancy		X		X		
printing	no redundancy	X	X	X	X	X	
Number of solutions		1	1	6	5	24	11
SOLO Classification		M	U	U	U	U	U

**Table 4: Refined quality framework for the ‘discount’ problem**

These code patterns are mapped against their respective functions and quality factors in Table 4. The

concepts extracted in this model then enabled us to conduct a mapping of each response pattern to the SOLO taxonomy.

Despite the large variation in responses, for a relatively simple question, the differences were in the minor detail. Perhaps this is a function of the limited level of complexity of the question which essentially just assesses ability to frame conditional statements.

This question in terms of the SOLO taxonomy is posed at the *multistructural* level. It requires some interpretation to arrive at a suitable solution (cf. Table 1) but a significant part of the specification may be directly translatable into a solution.

The sequence of selection statements in the majority of cases were found to be ‘a direct translation of the specification’ and therefore coded as *unistructural*. However in some cases the sequence had been reordered away from a direct translation of the specification but this reordering provided a less integrated solution than would have been provided by a direct translation response (pattern 3). In other cases a solution had been improved in one ‘quality factor’ aspect, e.g.: removed repetition of the printing statement, but had introduced redundancy by double checking a boundary value.

One student wrote a generalised subroutine to calculate the discount and then used that routine in the main method (pattern 1, Table 4). This response was the only response observed that was not coded as *unistructural* because ‘the code had been reordered to make a valid solution’ (cf. Table 1). Since the code in the subroutine was close to ‘a direct translation of the specification’, although a more sophisticated response, we chose not to code this at the *relational* level. Using the quality factors as a guide we were able to place the student responses along the SOLO continuum (Figure 1).

SOLO level	P	U	M	R
Code Pattern		6 5      4 3 2		1

**Figure 1: Mapping to SOLO for ‘discount’ using quality factors**

Supporting the notion of a continuum the code patterns 4, 3 and 2 lean a little more towards a *multistructural* level response. Code pattern 1 by contrast leans towards a *relational* response because of the use of a subroutine which is a more integrated solution but in this case is still ‘close to a direct translation’ of the specification (cf. Table 1).

## 4.2 Average

This question was taken from a second semester (CS1.5) programming class, with a focus on scripting languages. The question, given below, was part of a practical exam.

*Write a Perl script to allow students to calculate their average exercise mark for a semester. The script should:*

1. *Prompt the student for their name.*
2. *Prompt for the next exercise mark and allow it to be input*
3. *Do this for 5 marks.*
4. *Display the student’s name, and their average mark.*

*Extra credit will be given if your script contains a sensible subroutine that is correctly used.*

Typical code patterns are depicted in Table 5 with a focus on the subroutine function or its absence (Table 6) where the key variations were apparent.

Pattern	Typical Code Example
1	<pre>sub findAve (){     \$_[0] / \$_[1]; } for (my \$i = 0; \$i &lt; 5; \$i++){     print "Enter next mark: \n";     chomp (my \$mark = &lt;STDIN&gt;);     \$total += \$mark; } my \$average = &amp;findAve (\$total,5);</pre>
2,4,5	<pre>sub findAve () { \$_[0] / 5; }</pre>
6	<pre>my \$i = 0; my \$total = 0; for (\$i=0, \$i&lt;5, \$i++){     print "Enter next mark: ";     chomp(\$total += &lt;STDIN&gt;); } my \$average = \$total/5; print "Average : \$average\n"; exit;</pre>
7	<pre>print "Please enter first mark \n"; my \$mark1 = &lt;STDIN&gt;; print "Please enter second mark \n"; my \$mark2 = &lt;STDIN&gt;; print "Please enter third mark \n"; my \$mark3 = &lt;STDIN&gt;; print "Please enter fourth mark \n"; my \$mark4 = &lt;STDIN&gt;; print "Please enter fifth mark \n"; my \$mark5 = &lt;STDIN&gt;; my \$total = 0; my \$total1 = \$total + \$mark1; my \$total2 = \$total1 + \$mark2; my \$total3 = \$total2 + \$mark3; my \$total4 = \$total3 + \$mark4; my \$total5 = \$total4 + \$mark5; my \$average = \$total5/5; print "Total: \$total5 \n"; print "Average: \$average \n";</pre>

**Table 5: Example of the most prevalent ‘average’ solution patterns**

The initial salient element analysis for the averaging problem is provided in Table 6. Solutions that did not use a loop to get the user input had code repetition and were considered to have high redundancy. One student wrote a cohesive, generalised subroutine to get the user input. But

the majority of students wrote a subroutine to calculate the average.

The elements identified with their respective functions and quality factors are depicted in Table 6. We can see in this question while redundancy and generalisation are quality factors in common with the ‘discount’ problem, the requirement for a subroutine has added a new design dimension of cohesion of the subroutine design.

Function	Element	Quality Factor
input	Uses a loop	low redundancy
	No loop	high redundancy
	In a subroutine	generalised
Has subroutine	Sub uses a parameter	generalised
	Also gets input	low cohesion
No subroutine	Sums in a loop	low redundancy
	Sums without loop	high redundancy

**Table 6: Salient element framework for the ‘average’ problem**

Table 7 extends the salient element framework to highlight the varying code patterns relating to the identified function and quality factors.

Function	Quality Factor	Code Pattern						
		1	2	3	4	5	6	7
sub	used	X	X	X	X	X		
	generalised	X		X				
	high cohesion	X			X	X		
number	generalised		X		X			
input	no redundancy	X	X	X	X	X	X	
Number of solutions		20	1	1	1	2	14	1
SOLO classification		M	M	M	M	M	U	U

**Table 7: Refined quality framework for the ‘average’ problem**

As Table 7 shows there was some variation in the way that students wrote their subroutines but with two dominant patterns suggesting a bimodal response pattern based upon a student’s decision to write a subroutine. The primary pattern, pattern 1 (cf. Table 5 for sample code) is an example of a response that used a generalised subroutine. In contrast patterns 2, 4 and 5 did not use a generalised subroutine (also cf. Table 5). Pattern 1 took two parameters, the total and the item count, whereas pattern 3 took one, an array containing the input data. The latter resulted in a solution without cohesion as it consisted of one large subroutine that read input and calculated the average. Only two students used a variable to hold the number of items (patterns 2 and 4) instead of hard coding the ‘5’ into the code. The one example where the solution has redundancy in getting the input did not use a loop. More common was the use of a loop for input but no subroutine at all (pattern 6).

While the patterns of response were divided bimodally at the *unistructural* and *multistructural* SOLO levels the question itself is clearly set at a multistructural level (cf. Tables 1 and 7). Parts of the problem were directly translatable in that students were given the steps of the algorithm required for its solution. The question offered an extra mark for a simple subroutine, without specifying what it had to do, so some interpretation was required for this.

One student wrote a solution that took the line “Do this for 5 marks” to mean repeat the input line 5 times. Most realised that the line was meant to indicate that a loop was required, especially as the accompanying marking scheme clearly stated that a loop was needed.

SOLO level	P	U	M	R
Code Pattern	7	6	5 2 3 1 4	

**Figure 2: Mapping to SOLO for ‘average’ using quality factors**

We classed all solutions without a subroutine as *unistructural* since they were no more than a direct translation of the problem description that opted out of writing a subroutine. Next in the continuum (Figure 2) were those answers where a subroutine was used. We considered the inclusion of a subroutine to indicate a *multistructural* response. We did not consider it to be a *relational* response because it was clearly stated in the question that a subroutine was expected. However reflecting the continuum model we can see from Figure 2 that pattern 7 is close to a *prestructural* response and pattern one approaches the *relational* level of SOLO.

### 4.3 Print a box of asters

This question was taken from the final written exam for a first semester (CS1) java programming class.

The students were asked to write code to print a box of asterisks (\*) with the same number of rows and columns, an example was given of a 5 by 5 box of asters. The students were provided with the method signature which had a single parameter that represented the width and height of the box. The majority of the students did not attempt to answer this question. This appeared to be perceived as a difficult question by the students, although students had prior exposure to writing a method that printed a triangle in their lab class.

Table 8 portrays the typical solution code for both the incorrect and correct solutions. In this question many students wrote functioning code that iterated one too many times (code patterns annotated in Table 8 “<=” with one or more loops) or created a box of fixed width or area.

We can see in this question while redundancy and generalisation are quality factors in common with the ‘discount’ and ‘averaging’ problems, the increased complexity of the problem has introduced further design issues relating to the degree of connectedness of the code. It has also enabled partially correct working solutions to be provided.

Code Patterns	Typical Code Example
1, 7 2, 8 (<=)	<pre>public void printBox(int size){     for(int i = 0; i &lt; size; i++){         for(int j = 0; j &lt; size; j++){             System.out.print("*");         }         System.out.println(" ");     } }</pre>
1, 9 2, 10 (<= 2 <sup>nd</sup> loop)	<pre>public void printBox(int size){     String sStar = "";     for(int i = 0; i &lt; stars; i++){         sStar += " ";     }     for(int j = 0; j &lt; stars; j++){         System.out.println(sStar);     } }</pre>
3 4, 11(<= loop)	<pre>public void printSquare(int size){     for(int j = 0; j &lt; stars; j++){         System.out.println("*****");     } }</pre>
6, 12	<pre>public void printSquare(int size){     for(int j = 0; j &lt; stars; j++){         System.out.println(" ");     } }</pre>
5, 13	<pre>public void printSquare(int size){     System.out.println("*****");     System.out.println("*****");     System.out.println("*****");     System.out.println("*****");     System.out.println("*****"); }</pre>

**Table 8: Example of the most prevalent ‘box of asters’ solution patterns**

In refining from this salient element framework (Table 9) to a refined set of code patterns new choices presented themselves. Depending on the learning goals at this level teachers may choose to emphasise different quality factors in assessing student code. Key distinctions in this instance were between generalisability, connectedness (or level of integration) and potential efficiency.

Function	Element	Quality Factor
iteration	nested	generalised & connected
	2 independent loops	generalised & not connected
	1 loop	not generalised
	No loop	not generalised & redundancy
	Uses parameter	generalised
	Terminates correctly	correct solution
	Loop 1 too many	incorrect solution
	Loop 1 too few	incorrect solution
asters	As a local variable	generalised

**Table 9: Salient element framework for the ‘box of asters’ problem**

In the course context for this question more importance was placed on generalisability and indeed the question itself required a generalised solution. While in our context this focus on generalisability has been adopted as a design principle in other contexts with a

stronger focus on algorithms the efficiency of the solutions maybe a focus. A discussion follows of two refinements of the salient element framework based on two different perspectives of code quality.

The first refinement involved condensing the iteration elements based on the generalisability of the code as a quality factor.

Function	Quality Factor	Code Patterns					
		1	2	3	4	5	6
iteration	generalised	X	X				
	terminates correctly	X		X			X
result	<i>n</i> by <i>n</i> box, correct size	X					
	an <i>n</i> by <i>n</i> box, size incorrect		X				
	a 5 by <i>n</i> box			X	X		
	A 5 by 5 box					X	
	a line of 5 asters						X
	Number of solutions	15	3	4	4	1	4
SOLO classification		R	R	U	U	U	P

**Table 10: Refined quality framework based on generalisability for the ‘box of asters’ problem**

Six unique patterns of code were identified when the code was classified using the quality framework based on generalisability of the print a box of asters method.

If the solutions were able to produce a box of equal width and height of any size (the answer utilised the parameter supplied) then the iteration was considered to be a more generalised solution (see code patterns 1 and 2, Table 10 and code examples Table 8). Those students who chose to answer this question tended to provide a correct generalised solution. The few solutions that were not generalised usually either printed a fixed width box or a single vertical or horizontal line of asters and also gave incorrect output.

Function	Quality Factor	Code Patterns						
		7	8	9	10	11	12	13
iteration	connected	X	X					
	terminates correctly	X		X			X	
result	<i>n</i> by <i>n</i> box, correct size	X		X				
	an <i>n</i> by <i>n</i> box, size incorrect		X		X			
	a 5 by <i>n</i> box					X		
	A 5 by 5 box							X
	a line of 5 asters						X	
	Number of solutions	10	3	8	4	5	6	1
SOLO classification		R	R	R	R	U	P	U

**Table 11: Refined quality framework based on connectedness for the ‘box of asters’ problem**

The second refinement attempt, in Table 11, condensed the categories on the basis of the degree of connectedness (or degree of integration) of the code. In



solutions that use two sequential loops the response had two somewhat disconnected pieces of code. While it could be argued that this solution has some connection between the two loops because the second loop uses the string built by the first loop (cf. Table 8), it is possible to give a more connected answer by nesting the loops.

This problem is considered to be posed at a *relational* level. It provides the learner with a description of the problem to be solved. The description is complete in the sense that it describes fully the requirement from the perspective of the problem domain but provides only minor clues as to how the problem might be programmed.

Since the algorithm was not provided there is no method of clearly defining what a direct translation solution (*unistructural*) would look like. For this reason it is not possible for students to provide a correct solution that can be considered to be *unistructural* or *multistructural*.

Across all 13 patterns identified in Tables 10 and 11 for the quality factor of correctness, there were three code patterns observed that provided a correct solution (cf. Table 11, patterns 7 and 9 and Table 10, pattern 1). The other answers failed to output a generalised solution and did not provide a correct solution (typically by incorrectly terminating the loop). For the purpose of this SOLO analysis we ignored the loop termination bug as it is not significant when considering the SOLO class of the answer (which assesses the level of integration of a response).

The first correct method (Table 8, patterns 1 and 7) used a nested loop which provided a connected and generalised solution. The nested loop solution is less efficient ( $O(n^2)$ ) than the second generalised solution that uses somewhat disconnected sequential loops ( $O(n)$ ) (Table 8, patterns 1 and 9). These solutions are considered to be *relational*.

Code patterns 5 and 13, 3, 4 and 11 provide a direct translation of the exemplar given in the question and can only ever produce a 5 by 5 or a 5 by  $n$  box of asters. Some of these code patterns use a loop rather than five sequential print statements and are therefore considered to be slightly better solutions. This subtle difference is illustrated in Figure 3 along a SOLO continuum. However all student solutions following these patterns fail to provide the required generalised solution and were classified as a *unistructural* response.

SOLO level	P			U			M			R		
	6	12		5	3	4	13	11		9	1	7
Code Pattern										10	2	8

Figure 3: Mapping to SOLO for 'box of asters' using quality factors

Solutions that printed a row or column of asters, patterns 6 and 12, show that the students recognise that a loop is required to solve the problem but do not really grasp how that loop functions and their solutions were therefore coded as *prestructural*.

## 5 Conclusion

In previous research, mapping from student code to the SOLO taxonomy has proven difficult (Clear et al., 2008, Lister et al., 2009), since the mapping process seems very context bound and question specific. Therefore achieving consistent ratings is challenging, especially for code writing problems. In this study a grounded approach has been adopted to work from student responses bottom up through a two layer coding and concept mapping process. Based upon the resulting refined quality framework we have been able to identify critical elements, which can contribute to more consistent and supportable SOLO categorisations of novice programming students' responses to writing questions. A depiction of the empirically grounded mapping process is given in Figure 4.

We believe that this salient element and quality framework helps to define the SOLO categories and provides a novel way of matching the principles of SOLO for code writing problems. The process of identifying salient elements at the syntactic level should be readily reproducible for chosen problems by knowledgeable CSED researchers. In the feature extraction stage there are some basic features that are replicable and discernable across different code writing questions given to CS1 students. These features encompass the degree of redundancy, efficiency, generalisability and integration observed in the solution code. In some cases degree of coupling and cohesion also play a role. The features represent abstractions from the code itself, based upon qualitative judgments, which can be adapted depending upon the design goal for the code writing exercise.

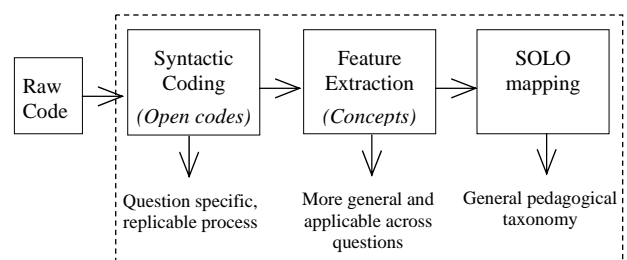


Figure 4: The mapping process

Further study needs to be undertaken that allows us to establish the reliability of the suggested coding practice. Potentially multi-rater studies could be conducted (Clear et al., 2008).

The study has caused us to suggest refinements to prior work. In coding the averaging problem (Section 4.2) we observed students responses that highlight an issue with the description provided by Lister et al. (2009) for *multistructural* responses to code writing questions. Therefore a revised version of Table 1 is provided below to include this refined *multistructural* response definition.

Phase	SOLO category	Description
Qualitative	Extended Abstract – Extending [EA]	Used constructs and concepts beyond those required in the exercise to provide an improved solution
	Relational - Encompassing [R]	Provides a valid well structured program that removes all redundancy and has a clear logical structure. The specifications have been integrated to form a logical whole.
Quantitative	Multistructural - Refinement [M]	Represents a translation that is close to a direct translation. The code may have been reordered to make a <i>more integrated and/or valid solution</i> .
	Unistructural – Direct Translation [U]	Represents a direct translation of the specifications. The code will be in the sequence of the specifications.
	Prestructural [P]	Substantially lacks knowledge of programming constructs or is unrelated to the question.

**Table 12: Refined SOLO categories for code writing solutions**

The previous definition stated that a *multistructural* response represented a translation that is close to a direct translation. The code may have been reordered to make a 'valid' solution. Yet in the averaging problem we saw responses that were a slightly reordered translation and that were correct valid solutions, but the reordering led to a less integrated solution. Such translations, while often still at the *multistructural* response level, tended towards the *unistructural*.

In our experience, educators need to be careful when applying SOLO to classify student responses. It is easy to lose sight of the intention of SOLO. *Unistructural* means focusing on a single concept or salient element. *Multistructural* focuses on multiple concepts or salient features but not integrating them all. *Relational* requires seeing all the salient elements or features and utilising them (i.e. seeing the relationships between the concepts and features). In this study these terms have been interpreted through Table 12 above, and the process of assigning a SOLO level to a student response has involved an assessment of the degree of distance between the answer and the specification. Thus the judgment for the code writing process relates to the level of translation of the specification demanded to implement the code. This reflects the level of abstraction or integration of thought required by the student.

The insights from this study may further serve to explain the contradictory findings of Meerbaum-Salant et al., (2010), who noted when applying a combined BLOOM and SOLO taxonomy, that students performed less well on a lower level *Multistructural Creating* task than on a deemed higher level *Relational Applying* task. Lopez et al., (2008) would also support that view in suggesting that code writing is a higher order skill than tracing – the *Relational Applying* task in the Meerbaum-Salant et al., (2010) study. But perhaps the code

writing/reading distinction is more subtle than a simple hierarchy. In the latter study we surmise that the authors have not taken into account the degree of translation demanded by the code writing task, but posited a SOLO level for the question 'in the abstract' based upon it requiring the combination of an assumed sequence of steps. Such tasks may indeed involve more integration of thought than implied by a *multistructural* classification, which requires some degree of re-ordering and integration of thought (cf. Table 12), perhaps placing it closer to the *Relational* level on the SOLO continuum.

To conclude we believe the salient element framework presented in this paper should serve as an aid to CSED Researchers and CS Educators seeking to analyse novice programmer responses to code writing questions, and to map them consistently to a SOLO level. By doing so we intend that our understanding of the programming process exercised by novice programmers may be deepened, and that we may build a greater awareness of what reasonable expectations may be set for novice performance on code writing tasks. The intended impacts from this deeper, research derived, understanding are: more consistent and equitable designs for code writing questions, an improved learning experience for the novice and an overall increase in the quality of teaching and assessment of novice programmers.

## 6 Acknowledgements

We wish to express our thanks to members of the ITiCSE 2009 Paris working group and especially Otto Seppälä who was a member of the subgroup who provided data for one of the questions and helped with assigning SOLO categories to the responses.

## 7 References

- Biggs, J. B. (1999): *Teaching for quality learning at University*, Buckingham. Open University Press.
- Clear, T., Whalley, J., Lister, R., Carbone, A., Hu, M., Sheard, J., et al. (2008): Reliably Classifying Novice Programmer Exam Results using the SOLO Taxonomy. S. Mann and M. Lopez (eds.), *Proc. of the 21st Annual NACCQ Conference*, 1: 23-30. Auckland, New Zealand: NACCQ.
- Glaser, B. and Strauss, A. (1967): *The Discovery of Grounded Theory*. Mill Valley, CA: Sociology Press.
- Hattie, J. and Purdie, N. (1998): 'The SOLO model: Addressing fundamental measurement issues', in Dart, B. and Boulton-Lewis, G. (eds.), *Teaching and Learning in Higher Education*. 145–176. ACER Press.
- Lister, R., Clear, T., Simon, Bouvier, D. J., Carter, P., Eckerdal, A., et al. (2009): Naturally occurring data as research instrument: analyzing examination responses to study the novice programmer. *SIGCSE Bull.* **41**(4): 156-173.
- Lopez, M., Whalley, J., Robbins, P. et al., (2008): Relationships between reading, tracing and writing skills in introductory programming. M. Caspersen, R. Lister and M. Clancy (eds.), *Proc. of the Fourth International Computing Education Research*

*Workshop (ICER 2008)*. 101-112. Sydney, Australia: ACM.

Meerbaum-Salant, O., Armoni, M. and Ben-Ari, M. (2010): Learning Computer Science Concepts with Scratch. K. Sanders, M. Caspersen and M. Clancy (eds.), *Proc. of the Sixth International Computing Education Research Workshop (ICER 2010)*. 69-76. Aarhus, Denmark: ACM.

Sheard, J., Carbone, A., Lister, R., Simon, B., Thompson, E. and Whalley, J. L. (2008): Going SOLO to assess novice programmers. *SIGCSE Bull.*, **40** (3): 209-213.

Stemler, S. (2001): An Overview of Content Analysis *Practical Assessment, Research & Evaluation*, **7**(17). Retrieved August 16, 2010 from <http://PAREonline.net/getvn.asp?v=7&n=17>

Thompson, E. (2010): Using the Principles of Variation to Create Code Writing Problem Sets. To appear in *Proc of the 11th Annual Conference of the Subject Centre for Information and Computer Sciences*, Durham, UK, <http://www.ics.heacademy.ac.uk/>

Tilley, S. (2000): The canonical activities of reverse engineering. *Annals of Software Engineering*, **9**: 249-271.

## Appendix

This appendix provides example code for the less frequent code patterns observed for the discount pattern.

Pattern	Typical Code Example
1	<pre>def main();     print "Losken tuotteen alennettu hinnan."     rivi = raw_input("Anna tuotteen hinta.\n")     hinta = float(rivi)     aleenushinta = laske_alennelta_hinta(hinta);     print "Tuotteen alennettu hinta on:", uusihinta, "euroa" main()  def laske_alennelta_hinta(hinta):     bonus = 0.0     if 100 &lt;= hinta &lt; 200:         bonus =0.05     if hinta &gt;= 200:         bonus =0.10     return hinta * (100-bonus)</pre>
2	<pre>def main();     rivi = raw_input("Anna tuotteen alkuperainen hinta.")     hinta = float(rivi)     alennus = 1.0     if hinta &gt;= 200:         alennus = 0.9     if hinta &gt;= 100 and hinta &lt;= 200:         alennus = 0.95     alennettu_hinta = alennus * hinta     print "Tuotteen alennettu hinta on:", alennettu_hinta main()</pre>
4	<pre>def main();     alkup_raw = raw_input("Anna alkuperainen hinta.\n");     alkup = float(alkup_raw);     if alkup &gt;= 200:         ale = 0.90     elif alkup &gt;= 100:         ale = 0.95     else:         ale = 1.0     alehinta = alkup*ale     print " Tuotteen alennettu hinta on.", alehinta main()</pre>



# Wrong is a relative concept: part marks for multiple-choice questions

Simon

University of Newcastle

simon@newcastle.edu.au

## Abstract

Most of the literature concerning multiple-choice exam questions assumes that they are marked on a simple scale of 1 for a right answer and 0 for a wrong one. In this paper I present a case for multiple-choice questions that are worth more than one mark, for parity with other questions on the exam. I propose that with such questions, it is appropriate to award part marks for some of the wrong answers. I present some multiple-choice questions of this sort that have been used in the final exam for an introductory programming course, and analyse the students' answers to confirm that the questions are valid assessment items.

**Keywords:** assessment, multiple choice, computing education, introductory programming

## 1 Introduction

Multiple-choice questions (MCQs) are used often, and probably increasingly, in computing education, typically because they save a great deal of time and effort in marking (Woodford & Bancroft 2005) – although much of the saving can be lost to the time and effort of constructing a good multiple-choice paper.

It appears to be the general belief that MCQs are flawed in that they can only test simple factual recall, but many authors (Lister 2000, Nicol 2007, Roberts 2006) argue that they can be used for higher purposes, with Lister, for example, suggesting that MCQs can test the first four levels of the Bloom Taxonomy (Bloom et al 1956).

In the final examination for an introductory programming course I have taken a small number of open-ended written questions and replaced them with multiple-choice questions. In two breaks with tradition, each of these questions is worth more than one mark, and some of the wrong answers are allocated part marks.

The rest of this paper explains why it was decided to replace the open questions with multiple choice and why it was decided to allocate multiple marks to each question with the possibility of part marks; examines the questions and their construction; analyses the questions in the light of the students' responses; and concludes that the changes appear to have been reasonable ones.

## 2 Multiple-Choice Questions

Multiple-choice questions have been discussed at great length in the literature, and have their own terminology, which is summarised by authors such as Isaacs (1994) and Woodford and Bancroft (2005).

A question is called an *item*. The leading part of the question, preceding the multiple answers, is called the *stem*. The multiple answers are called *options*, the correct answer is called the *key*, and the incorrect answers are called *distractors* (or sometimes *distracters*).

There are many guidelines for writing good multiple-choice questions and for writing good multiple-choice tests (Gronlund 1982, Hansen & Dexter 1997, Isaacs 1994, Woodford & Bancroft 2005).

### 2.1 The traditional view of MCQs

Bloom's Taxonomy of cognitive learning objectives (Bloom et al 1956) proposes a quasi-hierarchy of knowledge, comprehension, application, analysis, synthesis, and evaluation. There has been much debate about whether the categories do in fact form a hierarchy, what order they should take if they do (Anderson et al 2001), and whether in their current single form they are applicable to all areas of learning – Bloom himself apparently thought not (Anderson et al 2001). Even so, many educators take them as given, and frame their discussions accordingly.

The traditional view of MCQs is that they offer broad coverage of the material in a course, but at the expense of depth of coverage. Essentially, they can be used to test remembering (knowledge) and perhaps understanding (comprehension), but no more than those two levels of Bloom's taxonomy.

Within these perceived limitations, MCQs can be used to test mastery of a topic, in what is known as criterion referenced assessment (have students grasped what they need to grasp in order to pass the course?). They can be used to rank students, in what is known as norm referenced assessment (how does each student score relative to the rest of the class?). And they can be used to find out what students do not know, in what is known as diagnostic assessment, with the intention either of remediating the students or of improving the course (Isaacs 1994).

There appears to be universal agreement that marking of multiple-choice examinations is easy. Even when the marking is not automated, it is probably easier than marking of open-ended questions. Further, analysis tools have been developed for distractors, for items, and for whole tests, so that once a test has been created and undertaken it is not difficult to validate it.

It appears to be widely acknowledged that while the use of MCQs reduces the time and effort spent marking, creation of a good MCQ test takes a great deal more time and effort than creation of a good open-ended test (Isaacs 1994, Lister 2000). For example, effort is required to ensure that each of the distractors is plausible to students who do not know the right answer, so that these students will not be able to guess the right answer simply by eliminating the implausible answers (Isaacs 1994, Hansen & Dexter 1997). One useful approach to choosing strong distractors is to base them on known student misunderstandings (Isaacs 1994).

## 2.2 Alternative views of MCQs

Notwithstanding the traditional views summarised in the preceding section, MCQs do not have to be quick and easy to answer, and are not limited to the assessment of recall and perhaps understanding.

One way to increase the cognitive load of answering a multiple-choice question is to make the stem longer and more involved. Questions with longer stems can take more time to read and absorb, but this can be offset by first describing a single involved scenario (called *stimulus material*), then asking several MCQs based on that same scenario (Isaacs 1994).

Even a short stem does not necessarily imply a speedy answer. In a mathematics exam, a stem such as ‘What is the value of the function  $5x^2 + x \sin(x)$  when  $x$  is 0?’ should not require a great deal of time from a well-prepared student, while ‘What is the integral with respect to  $x$  of the function  $5x^2 + x \sin(x)$ ?’ would probably require significantly more time.

The previous example should also make it clear that MCQs can assess at higher Bloom levels than knowledge and comprehension. The second question is clearly one that requires application – unless, of course, the students have been shown exactly that example in class and are merely required to remember the right answer rather than to calculate it.

The questions presented above are incomplete, in that they are provided without their key and distractors. A full set of extremely implausible distractors could, of course, considerably reduce the cognitive load of the question. What makes a multiple-choice question work is the choice of suitable distractors, distractors that will indeed distract the less knowledgeable student by being plausible to that student.

Lister (2000, 2005) offers examples of non-trivial MCQs for a first programming course, with the understanding that each question will require substantial time from the students. These questions, he suggests, can assess at the lower four levels of Bloom’s Taxonomy.

Another variation from the simple memory-recall MCQ is the *non-restricted item*, a question with more than one correct answer, in which students are expected to select all of the correct options and none of the incorrect ones. Kolstad and Kolstad (1994) discuss the circumstances in which these questions might apply, and the various approaches to marking these questions, which can be quite intricate and clearly not a simple 0/1.

All of these variations offer ways of making MCQs more complex than the simple tests of recall that they are

often taken to be, and suggest that the field is by no means yet fully explored.

## 3 A Different Perspective on MCQs

In a recent examination I have taken a small number of what were once open-ended questions and recast them as multiple-choice questions. In considering how to go about this, I made a number of decisions for which I found no clear precedent in the literature.

### 3.1 An MCQ can be worth more than 1 mark

In a test consisting entirely of MCQs, unless otherwise reported one assumes that the questions are worth 1 mark each, regardless of whether the questions are traditional recall questions or more involved ones such as those described by Lister (2000, 2005). In the latter case, it still appears that the questions are of comparable time or difficulty across the whole test, so it is reasonable to allocate the same mark to each of them. Lister (2000) does discuss the ‘brutality’ of the all-or-nothing approach of a complex MCQ, and concludes that this is akin to the brutality of a compiler either accepting or rejecting one’s code. This conclusion perhaps overlooks the fact that a compiler generally offers some assistance with removing errors, rather than considering them as final and penalising them accordingly.

In a test that mixes MCQs and open-ended written questions, the 0/1 marking scheme for the MCQs might no longer be suitable. With open-ended written questions, the mark generally reflects to some extent the time and effort required to answer the question. It would seem reasonable, then, to apply the same principle to the MCQs. If a written question requiring about 5 minutes of the student’s time is worth, say, 10 marks, surely a multiple-choice question that requires the same time should earn the same number of marks.

It follows that in some circumstances it might be contingent on the examiner to allocate more than one mark for certain MCQs.

### 3.2 Distractors can be allocated marks

In his essay *The relativity of wrong* Isaac Asimov (1988) observes that wrong is not a simple true/false concept. The Earth was once considered flat. Various scientists from several thousand to several hundred years ago postulated that it was actually spherical, and the postulate eventually became accepted fact. In recent decades, observations from space have led to the conclusion that it is in fact more of an oblate spheroid. But technically it is none of these, because of the perturbations of its mountain ranges, ocean trenches, and other smaller deviations from smoothness.

It is therefore wrong to say that the Earth is flat. It is also wrong to say that the Earth is round, but in some sense this would generally be considered less wrong than saying it is flat. In other words, wrong is a relative concept, and some wrong answers or ideas are more wrong than others.

Consider the following multiple-choice question:

- The shape of the Earth is
- (a) a plane
  - (b) an oblate spheroid

- (c) a sphere
- (d) a torus.

Assuming that that we are prepared to ignore mountain ranges, ocean trenches, and other topographical anomalies, and that we remain blissfully unaware that the oblateness is marginally but measurably greater south of the equator than north of it, option b would be correct.

But if we chose to give some reward to a student who describes the Earth as a sphere in preference to a torus or a plane, we might decide to award half a mark to option c. This would be our recognition that not all wrong answers are wrong to the same degree.

Part of the reasoning for this notion stems from the observation (Isaacs 1994, Tew & Guzdial 2010) that student misconceptions make a good source of distractors in MCQs. If the same Earth-shape question were asked as an open-ended rather than a multiple choice question, a part mark would almost certainly be awarded to students who answered that it was spherical.

With open-ended written answers it is more or less obligatory to award part marks to answers that are partly but not fully correct: if the marks awarded are always all or nothing, it might be concluded that the marking scheme is not particularly appropriate.

Why, therefore, should it be unacceptable to award part marks for wrong answers simply because the form of the question has changed from open-ended to multiple-choice?

With these considerations in mind, I have used MCQs as part of a written exam, have allocated them a total mark commensurate with the time and effort required to answer them, and have allocated part marks for choosing some of the distractors.

It should be emphasised that this was not an experiment in setting a multiple-choice exam applying the new ideas of multiple marks and part marks. The experiment was to replace three highly specific written-answer questions in an exam with equivalent MCQs, for reasons that will be explained in the next section. However, the lessons learnt in this exercise can certainly be applied to exams that are fully multiple-choice.

### 3.3 Context

The course used as a context for this analysis is an introductory programming course offered at several campuses in Australia and overseas. Two offerings of the course have been analysed, one at an Australian campus and one at an overseas campus. There were 150 students in the two offerings, split fairly evenly between the campuses.

The programming language taught in this course is Visual Basic. Readers accustomed to other C-based languages should not be alarmed at the lack of semicolons in the code provided in the questions.

## 4 The Questions

The BRACElet project (Whalley & Lister 2009) suggests that examinations include questions testing students' ability to trace code, to read and explain code, and to write code. The code-explaining questions present students with a small piece of code and ask them to explain what the code does, not line by line but as an

overall purpose. Many students do not score well on these questions.

When analysing the code-explaining questions, the project pays attention to the SOLO level (Lister et al 2006, Whalley et al 2006, Sheard et al 2008) at which students answer. While the preferred response is *relational*, an explanation of the overall purpose of the code, many students give a *multistructural* answer, explaining the code line by line. Some BRACElet papers (Sheard et al 2008) discuss the possibility that this is because despite ample coaching, students do not understand what kind of answer is being sought. Other BRACElet papers (Simon 2009) wonder whether students' problems with code-explaining questions are due simply to their difficulties with written English expression. This might particularly be a problem for students whose first language is not English, whether at an overseas or at a local campus.

To address these possible confounds I decided to replace the open-ended code-explaining questions with comparable multiple-choice code-explaining questions. All of the multiple-choice options are expressed in English, and all are in the SOLO relational form, so when students cannot answer these questions correctly it is not because they cannot express their understanding correctly and not because they fail to appreciate what type of explanation is being sought; it is unequivocally because they do not understand the code.

While I initially imagined that it would be difficult to find plausible distractors for questions of this sort, when I turned back to earlier exams that had included similar questions in the open-ended written form I found a number of recurring wrong answers. For example, students forgetting that arrays indexes begin at zero tended to think that an iteration through an array was omitting the last element. I used these common wrong answers or their equivalent as the multiple-choice distractors, conforming with the recommendation (Isaacs 1994) that to be plausible, distractors should correspond to common student misunderstandings. However, because I really wanted to know what students thought was the purpose of each code segment, I added an open-ended option that permitted students to write what they thought the purpose was if they felt unable to choose any of the other options.

In previous exams for this course the three code-explaining questions were each worth five marks, as a reasonable reflection of the time and effort it would take to answer them. Recasting the questions in multiple-choice format makes no appreciable difference to that time and effort, so when creating the exam I decided that the questions would be worth the same five marks, although at the time I imagined awarding full marks for the right answer and no marks for any of the wrong ones.

When marking began I was reminded of the brutality (Lister 2000) of all-or-nothing multiple-choice marking. In particular, I realised that while in previous offerings students would have been awarded some of the five marks for an answer that was in some way close to the right answer, students were now being awarded no marks at all for choosing that same answer from a list.

I therefore decided to award part marks for these questions. Specifically, a wrong distractor would be

awarded the same mark that the same answer would have earned in response to a comparable open-ended question. This mark was easy to determine, because I had in previous years set and marked comparable open-ended questions.

This approach has an obvious consequence for students who choose the 'none of the above' option and write their own explanation of the purpose of the code: such answers would be marked in exactly the same way as the equivalent open-ended question – except that a completely correct answer would not earn the entire maximum mark, because the student had failed to recognise the correct answer among the options.

The three questions are progressively more complex, as suggested in the BRACElet 2009.1 (Wellington) specification (Whalley & Lister 2009). The first question involves non-iterative code; the second, iterative code without control logic within the loop; and the third, iterative code with control logic within the loop.

The set of three questions was prefaced with the following instruction:

The next three questions are multiple choice, but please answer them in the 12-page answer booklet. If your answer to a question is one of a-d, you need only write the question number and the letter of the answer. If your answer is e, please write the question number and the letter e, then your own explanation.

#### 4.1 Question 24 and marks

The first of the three questions tests only the application of assignment statements and simple arithmetic. However, the outcome of the code is unlikely to be self-evident to most novice programmers, who would therefore need to desk-check (hand-execute) the code to be sure of the answer. This observation contrasts with the expressed preference of Lister and Whalley (2009) that students determine the purpose of the code just by reading it, not by tracing it.

The number to the left of each option is the mark that was awarded to students choosing that option.

24. What is the purpose or outcome of the following piece of code?

```
b = a + b
a = b - a
b = b - a
```

- 1 (a) to find the sum of the values of a and b
- 1 (b) to find the difference of the values of b and a
- 2 (c) to find the sum and the difference of the values of a and b
- 5 (d) to swap the values of a and b
- ? (e) something else – please write the purpose

#### 4.2 Question 25 and marks

The second question tests the application of a simple counting loop that iterates through an array executing a simple arithmetic assignment statement.

25. What is the purpose or outcome of the following piece of code?

```
For i = 0 To dPayment.Length - 1
    dBalance = dBalance + dPayment(i)
Next
```

- 1 (a) to add a payment to a balance
- 0 (b) to count the payments
- 3 (c) to add all payments except the last to the balance
- 5 (d) to add all payments to the balance
- ? (e) something else – please write the purpose

#### 4.3 Question 26 and marks

The third question tests the application of a simple counting loop, again iterating through an array, but this time involving a boolean decision.

26. What is the purpose or outcome of the following piece of code?

```
sOne = sTitle(0)
For i = 1 To iTitleLastIndex
    If sTitle(i).Length > sOne.Length Then
        sOne = sTitle(i)
    End If
Next
```

- 5 (a) to find the longest title in the array of titles
- 2 (b) to find the length of the longest title in the array of titles
- 2 (c) to move the longest title in the array of titles to the first place in the array
- 1 (d) to sort the array of titles according to title length
- ? (e) something else – please write the purpose

### 5 Analysis of the Questions

There are standard techniques for analysing MCQs by examining the students' answers. In this section I describe three such techniques, apply each technique to these questions and, where appropriate, propose a modified form of the analysis to suit the novel aspects of these questions.

#### 5.1 Item facility

The facility of an item or question is a measure of how easy that question is for students to answer, defined as the mean mark of all students for the question (Isaacs 1994). With traditional multiple-choice questions, this is simply the proportion of students who answer the question correctly, because correct answers score one mark and incorrect answers score zero.

Isaacs (1994) suggests that if the purpose of a test is to rank students, a facility of 40% to 60% is generally considered desirable, whereas if the purpose is to determine students' mastery of the subject matter, the aim should be a facility of more than 80%. However, he



**Table 1: Facility of each question, both in 0/1 sense (binary) and in actual mark sense (continuous)**

	Q24	Q25	Q26
binary facility	51%	53%	46%
continuous facility	61%	75%	62%

acknowledges that the true aim of a test is usually a hybrid of the two purposes.

I have measured two different facilities for each of these questions, as shown in Table 1. The ‘binary facility’ is the simple proportion of students who answered the question completely correctly, ie who chose the key as their answer; the ‘continuous facility’ is the average mark over all students for the question, and thus acknowledges the part marks that were awarded for some distractors including option e, the open-ended answer.

We see from Table 1 that none of the questions has a facility in the range suggested for mastery, but that the adoption of part marks for some wrong distractors does noticeably raise the facility of each question (in that the continuous facility of each question is higher than its binary facility). This is an inevitable consequence of the fact that some students who did not choose the key were nevertheless awarded some marks for the options they chose.

## 5.2 Item discrimination

The discrimination of an item is a measure of how well it distinguishes the good students from the poor students. From that definition alone it is clear that discrimination is of more interest in a test whose purpose is to rank students than in a test whose purpose is to assess students’ mastery of the subject matter. There are several approaches to the calculation of discrimination, varying, for example, on what proportion of students make up the ‘good’ and the ‘poor’ students. Gronlund (1982) divides the class into thirds according to their result on the whole test, and defines the discrimination of a question as the number in the upper third who answered the question correctly, minus the number in the lower third who answered it correctly, divided by the number in each third.

The greatest possible discrimination is 100%, achieved when all of the good students and none of the poor students answer the item correctly. The worst discrimination is –100%, achieved when none of the good students and all of the poor students answer it correctly.

As with item facility, we are interested in the ‘binary discrimination’ of each question, based on whether students chose the single correct option or one of the distractors, but also in a ‘continuous discrimination’, based on the average marks of students in the upper and lower thirds. We calculate the continuous discrimination by subtracting the average mark of the lower third of the class from the average mark of the upper third, and dividing the result by the maximum mark for the question.

The discrimination measures, shown in Table 2, indicate that all of the questions have positive discriminations, which is good. It also shows that for each question the continuous discrimination is lower than the

**Table 2: Discrimination of each question in both binary and continuous senses**

	Q24	Q25	Q26
binary discrimination	52%	46%	66%
continuous discrimination	37%	33%	51%

binary discrimination. This is to be expected, as the award of part marks blurs the lines between those who can answer the question correctly and those who can’t.

## 5.3 Distractor usefulness

Mitkov and Ha (2003) estimate the usefulness of a distractor (also known as its effectiveness) by comparing the numbers of students in the upper and lower groups who selected that distractor. They suggest that a distractor can be *good*, attracting more students from the lower group than from the upper group; *poor*, attracting more students from the upper group than from the lower group; or *not useful*, attracting no students.

Table 3 shows, for each distractor from each question, the number of students from the upper and lower thirds who selected that distractor. It follows from these figures that all of the distractors are good: all of them distracted some students, and all distracted more students from the lower third of the class than from the upper third.

## 6 Discussion

In a final exam that also incorporates 20 traditional multiple-choice questions and a number of open-ended written questions, I have included three multiple-choice questions intended to discern whether students can correctly determine the purpose of a small segment of program code.

While the traditional multiple-choice questions were allocated one mark each (and thus 1% of the maximum possible mark for the exam), the three non-traditional questions were judged to require significantly more time and effort, and so were allocated more marks – in this case, five marks per question.

While the multiple-choice options included the correct answer and a number of plausible distractors, I believed that some students might not find any of the answers to their satisfaction, so I included a ‘none of the above’ option, and asked students choosing this option to furnish their own answers.

In keeping with the way that the same questions would have been marked if they were open-ended rather than

**Table 3: Usefulness of each distractor from each question: numbers of students from the upper and lower thirds who selected it**

	Q24	Q25	Q26
fixed option (a)	0 : 5	0 : 4	–
fixed option (b)	1 : 2	0 : 2	0 : 3
fixed option (c)	1 : 9	9 : 19	3 : 9
fixed option (d)	–	–	1 : 12
open-ended option (e)	5 : 12	1 : 4	1 : 7

multiple choice, part marks were allocated for some of the distractors.

### 6.1 Traditional question analysis

A number of the measures traditionally used to analyse multiple-choice questions have been applied to these three questions. With regard to facility (Table 1), each of the questions has a facility within the range suggested appropriate for ranking tests, while none of them has a facility within the range suggested appropriate for mastery tests. As the exam is a hybrid of these two, the facility of the questions appears appropriate.

With regard to discrimination (Table 2), each of the questions discriminates reasonably between students in the top third of the class overall and students in the bottom third.

With regard to the usefulness of distractors (Table 3), every one of the distractors was classified as good: none of them was rated poor or not useful. This is particularly gratifying as, when I set out to write these questions in multiple-choice form, it was by no means obvious that I would be able to devise any meaningful distractors.

### 6.2 Novel question analysis

The measures of item facility and item discrimination are usually predicated on the expectation that each question is worth a single mark, and each answer will be awarded either the full mark or zero. As these questions were worth multiple marks, and parts marks were possible, I have supplemented these traditional measures (which I now call binary facility and binary discrimination) with what I call continuous facility and continuous discrimination.

The continuous facility shows that these questions are easier than if they were marked all-or-nothing; that is, their continuous facility is higher than their binary facility. This is the expected consequence of allocating part marks to the questions, which will of course lead to a higher average mark.

The questions discriminate less clearly between higher- and lower-performing students with the continuous discrimination than with the traditional binary discrimination. Again this is in accord with expectations, as it replaces a stark black and white marking scheme with one that recognises shades of grey.

### 6.3 Conclusions

As some other authors have asserted, and in contrast with the traditional view, multiple-choice questions are not restricted to the testing of pure recall. They can also be used to test aspects of learning that are considered as higher-level, for example according to Bloom's taxonomy.

It is possible and practical to allocate multiple marks to a single multiple-choice question. Where multiple-choice questions are mixed with open-ended questions in a single test, the allocation of multiple marks permits the mark for each question to reflect the time and effort required to answer that question.

In recognition that wrong is a relative concept, it is possible and practical to award part marks for selection of a wrong answer to a multiple choice question. Where the

distractors clearly display differing degrees of wrongness, this is recommended as a way to distinguish students who are more or less on the right track from those who have no idea.

The decision to allocate part marks for certain distractors would normally go hand in hand with the decision to allocate multiple marks to multiple-choice questions. However, it could equally well be applied to single-mark questions, with fractional marks allocated and tallied by a spreadsheet or other software.

### 6.4 Implications and consequences

What are the consequences for educators of the ideas developed in this paper?

First, the ideas might lead some educators to give deeper consideration to the use of multiple-choice testing. Those who have always thought of a multiple-choice exam as a large number of one-mark questions testing factual recall might now appreciate that there is scope for using multiple-choice questions to examine other levels of knowledge, and that those questions need not be 'short' in the sense that they can be answered almost as quickly as they can be read. Programming educators who are newly considering the possibility of longer multiple-choice questions are strongly advised to read Raymond Lister's ACE2005 paper (Lister 2005).

Does this different approach to MCQs defy the reasons for using MCQs in the first place? Not at all. It seems that the main reasons for using MCQs are consistency and ease of marking, and an MCQ with multiple marks and part marks for wrong answers can be marked automatically just as a traditional 0/1 MCQ can. Whether the answers are entered directly into a computer by students, scanned from a mark-sense answer sheet, or laboriously read by human markers, some software such as a spreadsheet is then used to allocate marks to each student. It is a trivial matter to apply in a spreadsheet the ideas presented in this paper.

One idea from this paper that does not lend itself to automated marking is the option e in each question: if you don't believe any of the other options is right, write your own answer. But this option is by no means integral to the marking approach, and can be readily dispensed with. Readers are reminded that the option was provided in this exam solely because I wanted to know what students really thought the answer was – not exactly a standard goal in assessment!

Will the ideas presented here lead to lower failure rates? I hope not. I hope that whatever form of assessment is used in computing education, it is used well and wisely, and applied in such a way that the most capable students do well, the barely capable students pass, and the students who have not (yet) achieved capability fail. This should be no different when using multi-mark part-marked MCQs. One difference that might become evident, though, is a smoother gradation of marks across the range.

One consequence of the multiple-mark notion is that a fully multiple-choice exam need no longer be constrained to have all its questions of comparable 'length', that is, requiring comparable time and effort to answer them. It would be perfectly reasonable to construct an exam in which, say, the first 20 questions are traditional short

MCQs worth one mark each, the next eight questions are mid-length MCQs worth five marks each, and the final four are seriously long MCQs worth ten marks each – always bearing in mind that much of the effort saved in marking will almost certainly be spent instead on the creation of the questions and their options.

## 7 Future Work

If it proves feasible, I intend to conduct a quantitative comparison of students' results under the written-answer approach and under the multiple-choice approach.

In line with the SOLO analysis carried out within the BRACElet project, it would be interesting to examine the answers written by students who chose option e. Is the proportion of multistructural answers reduced? If so, might it be possible to attribute this to the provision of several plausible answers, all in the relational form?

While my own exam for this introductory programming course has long consisted of a number of traditional 0/1 MCQs and a number of longer open-ended questions, consideration will now be given to replacing more of the longer questions with MCQs, and perhaps ultimately to making the exam fully multiple-choice.

## 8 Acknowledgements

I am grateful to the BRACElet project for the suggestion that programming examinations should include code-explaining questions, and for the copious analysis that its members have carried out on the worth of such questions for both assessment and research. I am also grateful to the anonymous reviewers of this paper: their suggestions have unquestionably improved the paper.

## 9 References

- Anderson, Lorin W & David R Krathwohl (eds) (2001). *A taxonomy for learning, teaching, and assessing – a revision of Bloom's taxonomy of educational objectives*. Addison Wesley Longman, Inc.
- Asimov, Isaac (1988). *The relativity of wrong: essays on the solar system and beyond*. Doubleday Inc, New York.
- Bloom, Benjamin, M. D. Englehart, E. J. Furst, W. H. Hill, and David Krathwohl (1956). *The taxonomy of educational objectives: the classification of educational goals, handbook I: cognitive domain*. David McKay Co Inc.
- Gronlund, Norman (1982). *Constructing achievement tests*. New York: Prentice-Hall Inc.
- Hansen, James D and Lee Dexter (1997). Quality multiple-choice test questions: item-writing guidelines and an analysis of auditing testbanks. *Journal of Education for Business* **73**(2): 94-97.
- Isaacs, Geoff (1994). *Multiple choice testing*. HERDSA Green Guide No 16. Higher Education Research and Development Society of Australasia Inc, Campbelltown, Australia.
- Kolstad, Rosemarie K & Robert A Kolstad (1994). Applications of conventional and non-restricted multiple-choice examination items. *Clearing House* **67**(6): 317-319.
- Lister, Raymond (2000). On Blooming first year programming, and its Blooming assessment. *Fourth Australasian Computing Education Conference (ACE2000)*, 158-162.
- Lister, Raymond (2005). One small step toward a culture of peer review and multi-institutional sharing of educational resources: a multiple-choice exam for first semester programming students. *Seventh Australasian Computing Education Conference (ACE2005)*, 155-164.
- Lister, Raymond, Beth Simon, Errol Thompson, Jacqueline L Whalley, & Christine Prasad (2006). Not seeing the forest for the trees: novice programmers and the SOLO taxonomy. *SIGCSE Bulletin* **38**(3):118-122.
- Mitkov, Ruslan & Le An Ha (2003). Computer-aided generation of multiple-choice tests. *HLT-NAACL 03 workshop on Building educational applications using natural language processing – Volume 2*: 17-22.
- Nicol, David (2007). E-assessment by design: using multiple-choice tests to good effect. *Journal of Further and Higher Education* **31**(1):53-64.
- Roberts, Tim (2006). The use of multiple choice tests for formative and summative assessment. *Eighth Australasian Computing Education Conference (ACE2006)*, 175-180.
- Sheard, Judy, Angela Carbone, Raymond Lister, Beth Simon, Errol Thompson, & Jacqueline L Whalley (2008). Going SOLO to Assess Novice Programmers. *SIGCSE Bulletin* **40**(3):209-213.
- Simon (2009). A note on code-explaining examination questions. *Ninth International Conference on Computing Education Research – Koli Calling 2009*, Koli, Finland, November 2009.
- Tew, Allison Elliott and Mark Guzdial (2010). Developing a validated assessment of fundamental CS1 concepts. *41st ACM Technical Symposium on Computer Science Education (SIGCSE 2010)*, 97-101.
- Whalley, Jacqueline L, Raymond Lister, Errol Thompson, Tony Clear, Phil Robbins, PJ Ajith Kumar, & Christine Prasad (2006). An Australasian Study of Reading and Comprehension Skills in Novice Programmers, using the Bloom and SOLO Taxonomies. *Eighth Australasian Computing Education Conference (ACE2006)*, Hobart, Australia, 243-252.
- Whalley, Jacqueline L and Raymond Lister (2009). The BRACElet 2009.1 (Wellington) specification. *Eleventh Australasian Computing Education Conference (ACE2009)*, pp-pp.
- Woodford, Karyn & Peter Bancroft (2005). Multiple choice questions not considered harmful. *Seventh Australasian Computing Education Conference (ACE2005)*, 109-115.



# How can software metrics help novice programmers?

Rachel Cardell-Oliver

School of Computer Science and Software Engineering  
The University of Western Australia,  
M002, 35 Stirling Highway, Crawley, WA, 6009  
Email: [rachel.cardell-oliver@uwa.edu.au](mailto:rachel.cardell-oliver@uwa.edu.au)

## Abstract

Many computing education studies have reported poor learning outcomes in programming courses for novices. Yet methods for measuring students' ability to generate computer programs remains an open research problem. In this paper we review some limitations of existing approaches for assessing student programs. We then propose a set of valid and reliable metrics for the direct measurement of novices' program code. Distributions of each metric are given for student populations. The metrics can be utilised for both diagnostic and formative assessment. Examples of formative assessments are given and a new diagnostic metric is presented for Perkins' "stoppers" and "movers" learning styles. This metric captures the multi-dimensional distance of a student's program from a target solution. The metric can be used by instructors to triage a large set of submissions and to tailor formative feedback to individuals.

## 1 Introduction

Programming courses for novices have two main learning outcomes: the ability to comprehend (read) program code and the ability to generate (write) program code. Although many studies have reported poor learning outcomes, the research problem of how to *measure* those outcomes has received surprisingly little attention. In particular, while criterion-based analysis of learning outcomes related to program comprehension has been the subject of several major studies (Lister & Leane 2003, Lister et al. 2004, Decker 2007), the problem of criterion-based measurement of students' code generation skills remains an open problem.

How can learning outcomes related to novices' ability to generate program code be measured? The first part of this paper (sections 2 and 3) reviews some limitations of existing approaches for measuring student programs and in particular highlights problems with the use of students' final grade as a metric to validate claims in empirical studies. In the second part of the paper (sections 4 to 6) we propose a suite of software metrics to measure the quality of students' programs, and so to inform strategies for improving programming skills. The focus of this paper is on software metrics that provide useful feedback to students (formative assessment) and to their lecturers (diagnostic assessment) (Crisp 2008). Our metrics

can also be used for grading assignments (summative assessment) but that is not our primary goal here.

This paper addresses two research questions:

1. which software metrics are appropriate (or not) for measuring novices' ability to generate program code? and
2. how can measurement with software metrics contribute to student learning?

In Sections 2 and 3 we answer the first question by reviewing some representative studies that evaluate novice students' ability to write computer programs. Several common pitfalls in the use of metrics in such experiments are identified. For example, we show that final grade is neither a valid nor reliable metric for drawing conclusions about students' code generation ability. In section 4 we propose a list of significant attributes for novices' programs together with metrics for each attribute. These commonly used metrics are both valid and reliable for the attributes they measure (Fenton & Pfleeger 1998). Sample distributions of the metrics are shown for large student populations for a selection of different Java programming exercises.

Sections 5 and 6 address the second research question by demonstrating two ways in which metrics can be used in programming courses for formative and diagnostic assessment. Section 4 presents examples of the formative feedback generated by the measurement tools used by students during their laboratory sessions. Section 6 considers the role of metrics for diagnostic assessment, that is feedback to lecturers about problems encountered by a cohort of students. We show how students can be classified according to Perkins' "stoppers", "movers" and "tinkerers" learning styles (Perkins et al. 1989) using a distance measure between the metrics vectors of a canonical solution program and the code submitted by each student.

## 2 Assessment of Code Generation Skills

This section focuses on related work that proposes specific methods for measuring the programs generated by novices. These approaches can be evaluated in terms of the validity, reliability and cost of the metrics they propose using the validation framework of Kitchenham *et al* (Kitchenham et al. 1995). *Validity* is the extent to which a measurement reflects the property of interest, in our case novices' ability to generate correct, efficient and readable computer programs. Establishing the validity of a metric requires examination of detailed and explicit criteria and their measurement instruments (Kitchenham et al. 1995). *Reliability* is whether the measurement is repeatable and is in agreement with other measures for the same property. Poorly documented grading schemes may be interpreted differently by different markers, and so they have low reliability. *Cost* of a metric is the

effort required to make the measurement. Metrics that can be evaluated automatically by a computer have lower measurement cost than those that require detailed inspection by a human marker.

In 2001 an international study was undertaken to measure code generation skills (McCracken et al. 2001). 217 students from different institutions took a laboratory examination in which they wrote programs for a set of mathematical problems about reverse-polish and infix calculators. The students' programs were evaluated against a set of well documented general evaluation criteria that included executable tests for correctness and expert inspection of program style. For programs that did not work a "degree of closeness" metric was also used as a subjective evaluation of how close a student's source code was to a correct solution. The performance of students in all the test groups against all these criteria was much lower than their instructors expected with an average general evaluation score of 22.9 out of 110 and an average degree of closeness score of 2.3 out of 5.

A number of flaws with the study were identified by the authors and others (McCracken et al. 2001, Decker 2007). The test had limited validity for its purpose because results were affected by a number of external factors. The mathematical focus of the problem set, the difficulty of the required data structures and problems with presentation and instructions for the test all created cognitive overload for the test that many students were unable to overcome. Detailed rubrics for assessing the exercise meant that grades were reliable for the given exercise, but the reliability of the measures have not been tested on any other programming tasks. The results presented using averages and standard deviations are inappropriate summary measures for these skewed populations. Instead, the median and quartile ranges should be presented. Overall, the metrics used proved neither valid nor reliable for general use and the cost of collection was high.

Autograder is a system for automatically grading student programs (Morris 2003). Laboratory exercises and automatic marking scripts are developed using a 10 step process starting with a canonical instructor-written solution and test suite, to detailed specifications for the students and finally post-facto feedback on the quality of the grading process. This assessment has criterion-referenced validity in that tests are provided for each of the requirements given to students in the specification. However, the system appears to mark only functional correctness, and does not consider planning, style, efficiency or other non-functional quality attributes of programs. Furthermore, the problem of degree of closeness of non-working solutions is addressed by providing work-arounds for "minor" errors such as naming and output format errors using regular expression matching, Java reflection, and overriding of directly or indirectly dependent methods. Although these work-arounds meet the goal of satisfying students that the marking is "fair" in recognising their effort, they lower the validity of the metric for the overall goal of measuring program generation ability. In our view a better approach is to set a target standard for assessment, but then allow students to refactor their code and resubmit it rather than for the instructor to create artificial work arounds. For these reasons the system has limited validity for the purpose of measuring students' overall program generation skills. Autograder is fully automatic once the programming exercise and its marking script have been developed, so it has low measurement cost even for large cohorts of students.

Lister and Leany present a criterion-based framework for assessing both code generation and code comprehension skills (Lister & Leaney 2003). The fo-

cus of that paper is on how to *specify* criteria for outcomes rather than how to *measure* outcomes as in this paper. Explicit and clear criteria are communicated to the students for the grades of fail, pass, credit, distinction and high distinction. For example, the criteria to be satisfied by a credit or distinction student include "to have demonstrated, within a small well-defined program context, that you are able to: (a) Apply the basic OO programming concepts of classes, instances, events, and methods. (b) Apply the basic program control constructs of sequence, selection and iterations. (c) Take an informal problem description and translate it into a readable, working program. (d) Apply the basics of testing and debugging." Programming skills were assessed in an assignment with several parts: 5 exercises adding small amounts of code (10 to 15 lines) to an existing system, and the final exercise to add an entire new class. Details of the measurement protocol and marking criteria are not given in the paper but the approach is certainly amenable to the specification of detailed metrics and measurement protocols for each criteria as introduced in this paper, in which case the validity, reliability and cost of the tests could be assessed.

There are many other published studies that use software metrics for assessment. Their goals range from detecting plagiarism to teaching software quality. In this paper we can only discuss a representative sample of such studies. More broadly, this paper draws on prior research on how students learn to program (Robins et al. 2003), on measuring learning outcomes in CS1 programming courses (Lister et al. 2004, Decker 2007, Robins 2010), on the automatic assessment of programming (Ala-Mutka 2005), and the theory of software metrics (Fenton & Pfleeger 1998, Kitchenham et al. 1995).

### 3 Evaluation of Final Score as a Metric

Many studies of student learning in CS1 draw conclusions based on students' final letter grades or percentage scores. In this section we identify some difficulties with this approach and caution against using either final grades or final scores as the measurement baseline for empirical studies.

Final grades in University courses are assigned as a weighted sum of assessed components. They may be adjusted to meet institutional requirements for pass rates and grade distribution. This is called norm-referenced grading (Lister & Leaney 2003). In CS1 courses assessed components include programming exercises, programming projects, tests and examinations. Final score is most often not a valid metric and studies based on final grades can be difficult to interpret because "when using overall course grade as the success marker, one should know if there was a curve placed on the grades, or even the basic breakdown of what is considered A work". (Decker 2007).

Furthermore, final score or grade is not a reliable metric because it is highly sensitive to minor changes. Figure 1 shows typical distributions of student scores for different types of assessment taken from a particular cohort of 180 students, being those students who sat the final exam out of 200 who enrolled initially. The scores shown are for 4 of the 8 components used for the final mark. Each component has been scaled to a mark out of 20 for easier comparison. The top figures (a) and (b) are practical work scores and the bottom figures (c) and (d) are examination scores. Three of the components (a, b and c) assessed code generation skills while exercise (d) assessed code comprehension skills.

It is interesting to note that practical work components (a) and (b) exhibit the typical bimodal distribu-

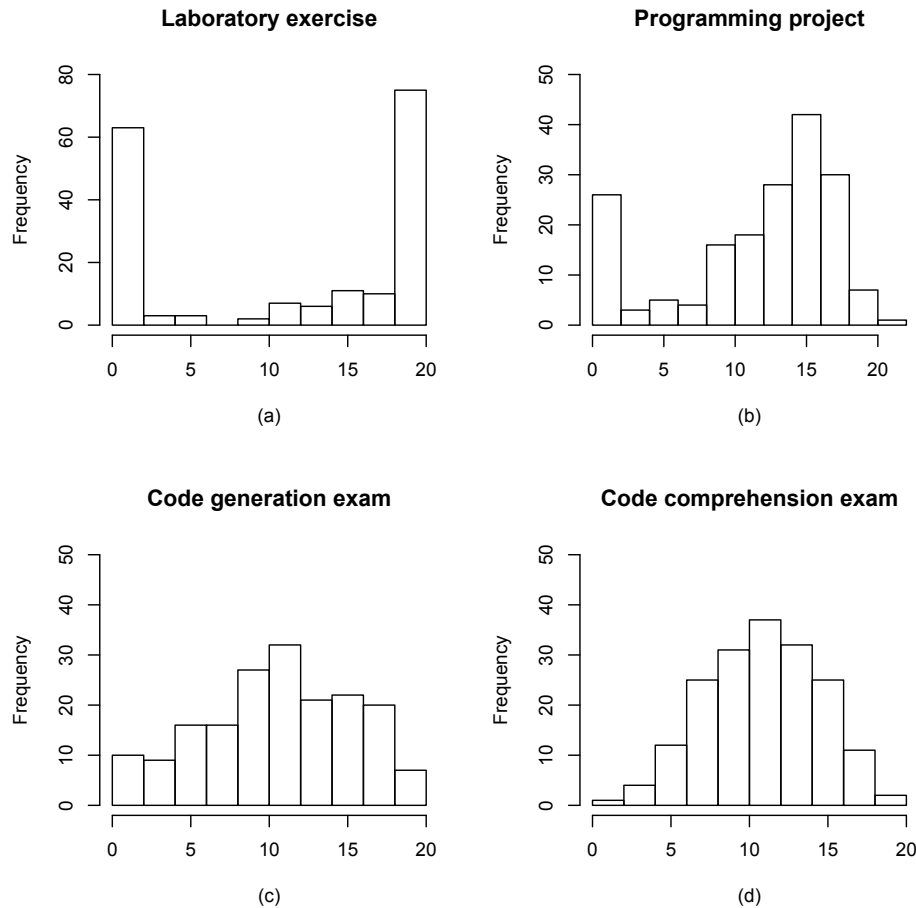


Figure 1: Distribution of Scores for Practical (a,b) and Written (c,d) Components assessing Code Generation (a,b,c) and Code Comprehension (d)

tion cited in many studies (Robins 2010). However, written components (c) and (d) are much closer to a normal distribution. Low scores in practical work occur when students either fail to submit, or submit almost nothing. At the other end of the marking scale, students who submit a “mostly working” program typically score well in the practical work, since in most cases lab exercises are designed for formative (rather than summative) learning. To this end, students are provided with test cases and style checking software so that most should be able to complete the exercise correctly. It is dangerous to make assumptions about the students who do *not* to submit, that is the lower group of the bimodal distribution. For example, students’ behaviour in first year courses is shaped by the way they cope with the new environment of university study and the need for many to adopt more independent learning strategies than they used at school. On the other hand, we have observed that bimodal distributions occur in coursework components for courses at all levels of our degree and in both programming and non-programming subjects. The common factor seems to be that in courses that have a high number of assessed components, and where the work required to obtain a good mark for a component is not reflected in the assessment weight, that students make strategic decisions about whether to submit that work or not. A typical student will complete some but not all of the low-weighted lab exercises.

Finally, we mention some unwanted side effects of the standard practice of calculating a final score in computer programming courses as a weighted sum of

assessed practical and written components. Two different types of distribution have been observed for assessed tasks in CS1: normal and bimodal. The choice of weights for the summed components leads to either the bimodal distribution observed in some institutions, or the normal distribution observed in others. That is, final score is not a valid nor a reliable metric. For these reasons extreme caution must be used about any conclusions drawn from correlations with the distribution of final marks. This includes claims about failure rates (high or low) since the failure rate is highly sensitive to the weighting of assessment components.

It is outside the scope of this paper to present a detailed alternative to the norm-referenced schemes used for summative assessment in most existing courses. A better approach is to use criterion-referenced (rather than norm-referenced) assessment for final grades (Lister & Leaney 2003) and certainly for empirical studies that aim to measure the effectiveness of new pedagogies. The main problem with the final grade metric is that it combines different aspects of a multi-dimensional attribute (students’ ability to read and to generate code) into a single measurement. This approach almost always leads to problems with validity and reliability (Kitchenham et al. 1995). From the measurement theory point of view the correct approach is to use a *vector* of measurements rather than a single value to capture multi-dimensional properties such as the quality of program code. Full details can be found in textbooks on software metrics (Fenton & Pfleeger 1998). In the next sections of this paper we demonstrate ways in which

such vectors of measurements can be used to help novice programmers.

#### 4 Software Metrics for Novice Programmers

Software metrics are used to measure the quality of the software produced by professional software engineers. The rationale for software measurement in industry is to improve the quality of the software production process and the quality of the software end product. Can software metrics offer the same benefits for novices?

In software metrics practice, first the specific goals of measurement are identified and then attributes that contribute to each goal are chosen together with measures for each attribute (Fenton & Pfleeger 1998). This practice is essentially the same as defining a criterion-referenced grading scheme (the goals) together with measures and measurement instrument is provided for each criteria (the attributes and their measures).

In education there are three main goals for measurement: diagnostic, formative and summative (Crisp 2008). *Diagnostic assessment* is measurement of the performance of a cohort of students in order to identify any individual learning difficulties, common problems, and areas of strength. *Formative assessment* provides feedback directly to students with the goal of helping them to improve their understanding of the subject. *Summative assessment* measures how well students have achieved learning objectives. This paper focuses on measurement for diagnostic assessment and for formative feedback to students.

Five areas for measurement and a collection of established metrics for each have been identified for our overall goal of measuring students' ability to generate program code. Each metric is valid and reliable for the attribute it measures and most of the metrics can be collected automatically using open-source software engineering tools. A measurement vector can thus be produced automatically for each Java class submitted by a student.

**Program Size** Non-comment lines of code, lines of code, number of methods, number of fields, and subsets of these based on Java modifiers. Measured using the Java Reflection library and a lines of code counter application.

**Functional correctness** Vector of individual test case results (pass, fail or error), number of test cases run, number of tests passed, code coverage of a test suite, input coverage of a test suite, coverage of other properties such as algorithmic complexity. Measured using JUnit with instructor-written or student-written test cases and the Emma code coverage tool.

**Efficiency** Execution time for a given test set. Measured using JUnit.

**Program Style** Number of code violations for naming conventions, hiding, complexity, coding conventions, magic numbers. Measured using Checkstyle and PMD.

**Client Validation** A client acceptance test of a sequence of user inputs and defined responses, assessed against a checklist of items. This metric is not fully automatic but requires some expert inspection. It can be used for assessing graphical programs.

Figure 2 shows the population distributions of metrics for the attributes of program size, functional

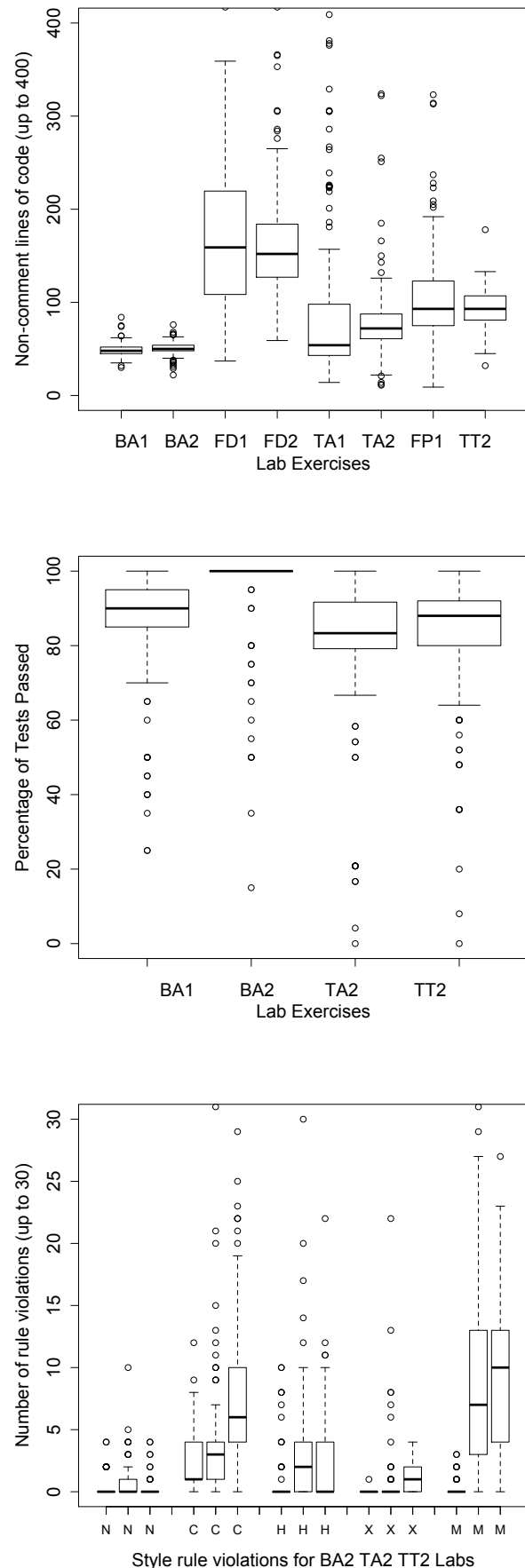


Figure 2: Metric Population Distributions for Program Size (top), Functional Correctness (middle) and Program Style (bottom) for categories N=naming, C=coding, H=hiding, X=exceptions, M=magic numbers



correctness and program style. Programming exercises are identified by three letter identifiers, each a single Java class. The exercises are presented in the order they appear during the course: that is, from easiest to hardest. The number of submissions for each of these exercises ranges from 132 to 200. The central line of each box is the median, and the box shows upper and lower quartiles of the population. Limits of the main population (99% range) are shown by the whisker dotted lines and outlier values are indicated by dots.

In a typical laboratory exercise the signatures of all public methods are given and the students provide implementation code for each method. Given these constraints on student classes we might expect the distributions for each metric to have small variation. However, Figure 2 shows that the range and variation for each metric is large and that most populations have a long tails. These examples of student populations for selected software metrics show how much student programs differ from one another. Section 6 explores some possible reasons for this variation.

## 5 Formative Feedback

Our software metrics are simply a number or vector for some attribute of a Java class. The metrics are therefore not much use on their own to improve student learning. Fortunately, the measurement tools we use can be configured to generate informative messages that alert students to the defects in their code and provide some advice about how to fix them.

During the laboratory session students use JUnit, Checkstyle and PMD to check their code. JUnit tests are written by the instructor. If the student's code fails a test case then an error message is provided to explain the problem and help them to identify what needs to be done. For example, the following messages are generated when a student mistakenly initiates a character count variable to 1 instead of 0.

```
Initial character count should be 0
  expected:<0> but was:<1>
There are no alpha chars in the string " *** 42 !!"
  expected:<0> but was:<1>
Error code ? expected for mostFrequent in empty string
  expected:<?> but was: <a>
```

PMD identifies style violations and returns feedback messages that refer to a line of code and the problem identified. For example, the following error messages are a sample of those generated for code that passed all the functional tests, but was highly complex and inefficient.

```
32 Avoid unnecessary comparisons in boolean expressions
180 Avoid really long methods.
180 Method names should not start with capital letters
293 The method 'isChar' has Cyclomatic Complexity of 54.
```

After the student has completed a lab exercise they submit their source code for assessment. The program is assessed using the same tools as used in the lab and a summary of the results is emailed to the student. The following example shows the type of formative feedback provided in these emails:

```
Highly inefficient code:
Your class executed in 10.207 seconds and the cohort
median execution time was 0.049 seconds.
If your execution time is much higher than the
median then you can improve efficiency.
Ask a lab demonstrator for help with this.
```

```
Your class has 00255 non-comment lines of code
(NCLOC). The expected NCLOC was around 70.
Warning (only, no marks): If your code is very long
```

```
(say, over 100 lines) then it may be using some
Java types such as arrays incorrectly.
Ask a lab demonstrator for help with this.
```

## 6 Metrics for Diagnostic Assessment

This section demonstrates how metrics vectors of attributes of student code can provide diagnostic information about the difficulties students are having. This in turn can be used to modify course delivery, providing ways for students to catch up and focussing help where it is needed to address problems in student learning.

Many CS1 courses offer a sequence of laboratory programming exercises for formative assessment. In an ideal world, students should correct problems in their code as soon as possible, rather than waiting for a submit, mark and return assessment cycle. For this reason, in our lab classes students are provided with JUnit and instructor-written test cases, as well as the style checkers PMD or Checkstyle with a selection of coding rules specifically configured for novice programmers. That is, all the metrics discussed in the last section are available to students from when they first start working on their assignments. However, as shown in Figure 2 this does not mean that when the exercises are submitted that there are no defects.

Our approach to diagnostic assessment is illustrated using Perkins' classification of novice programmers' learning styles as "stoppers" and "movers".

"When novice programmers see fairly quickly how to proceed, naturally they do so. When, however, a clear course of action does not present itself, the young programmer faces a crucial branch point: what to do next? ... Some students quite consistently adopt the simplest expedient and just stop. They appear to abandon all hope of solving the problem on their own." (Perkins et al. 1989), page 265

Stoppers are students who tend to stop and give up when they can not immediately see how to proceed with a problem. Non-starters are stoppers who may make some progress in lab classes but choose not to submit their programs for assessment. Movers, on the other hand, will try different approaches when faced with a problem. Movers can be further divided into two types. Extreme movers are called "tinkerers". When faced with a problem they make changes but more or less at random:

"Students often program by means of an approach we call tinkering - they try to solve a programming problem by writing some code and then making small changes in the hopes of getting it to work." (Perkins et al. 1989), page 272

The ability to diagnose a students' learning style is useful because different types of learners require different types of feedback. Movers are the most successful learners. They are adept at solving problems on their own and are likely to thrive whatever teaching and learning environment they are in. Tinkerers and stoppers both have learning strategies that interfere with their progress in solving programming problems. We have observed in our own classes that stoppers can be discouraged by receiving a detailed list of defects in their work, and this can easily turn stoppers into non-starters. On the other hand, movers and tinkerers can usually make good use of detailed feedback and will re-evaluate and improve their programs. To gain maximum benefit from explicit feedback all

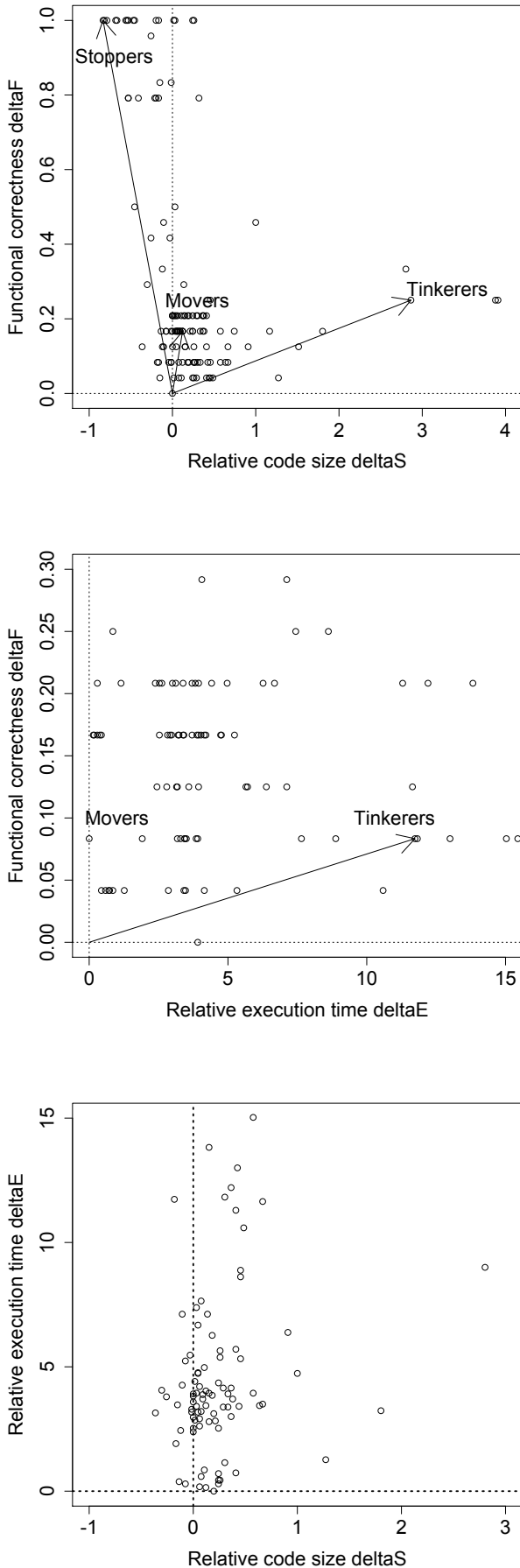


Figure 3: Relative code size vs Functional correctness vs Efficiency distances. Canonical solution shown by dotted lines. Distance from 0,0 is stopper, mover or tinkerer strength. Extreme outliers are not shown.

students should be given the opportunity to refactor their programs and re-submit. In many cases we conjecture that incomplete submissions are the result of a lack of organisation or lack of time to refactor rather than serious difficulties with the material. However, if these problems are not addressed, then this can lead to learning edge momentum problems, that is, inability to deal with new concepts when related ones have not been mastered (Robins 2010).

We now show how to identify a students' learning approach using measurements of the programs they generate and submit. In order to classify students we considered different attributes of their programs: size, functional correctness, efficiency and style. We then set out to characterise learning style in terms of *distance* from a target solution for each attribute. These classifications were validated by hand inspection of students' code noting whether the submission was not completed, completed but by tinkering, or completed to a good standard. Three of the attributes (size, correctness and efficiency) provided meaningful information about students progress. Each was normalised by taking its ratio to the same metric for a canonical solution. The attribute of number of style violations was not used since code inspections suggested that style violations were typically the result of careless mistakes rather than a lack of understanding of programming tasks.

The size distance  $\Delta_S$  of a student submission  $S$  from a canonical solution  $C$  is defined as

$$\Delta_S = ((size(S) - size(C))/size(C))$$

where the unit of measurement for size is non-comment lines of code which has a ratio scale. By construction  $\Delta_S$  has a median of 0 (best) and ranges from  $-1$  (no code submitted) to a positive maximum  $m > 1$  where the size of  $m$  depends on the particular exercise. For example, the maximum is 6.167 for the TA2 exercise and 0.798 for TT2.

The target for the number of test cases passed by a student submission  $S$  is  $T$  the number of test cases in the instructor-provided test class. The proportion of tests passed is not necessarily a ratio scale for any test suite, but it can be made so where certain conditions are satisfied construction of the test suite. The details of such a construction will be discussed in another paper. The functional correctness metric

$$\Delta_F = (T - passes(S))/T$$

defines the distance of a submission from its functional correctness target.  $\Delta_F$  ranges from 0 (best where every test is passed) to 1 (no tests are passed). Classes for which the tester could not be run are recorded as 0 tests passed giving  $\Delta_F = 1$ .

The target for efficiency is the number of milliseconds taken to execute a canonical solution class  $C$  using an instructor provided JUnit test suite designed for performance evaluation. The class with the lowest execution time amongst the functionally correct programs (hopefully the instructors' class but not always) is chosen as canonical solution class for the execution time target. Efficiency distance is defined by

$$\Delta_E = (time(S) - time(C))/time(C)$$

where the unit for measuring execution time is milliseconds and it has a ratio scale. Since some student submissions do not terminate, we use an upper bound timeout  $U$  for the efficiency test.  $\Delta_E$  is only meaningful for programs that implement the full functionality, and so non-submissions or submissions that do not pass most tests are not considered.  $\Delta_E$  ranges from the target of 0 for the most efficient class to a maximum threshold of  $(U - time(C))/time(C)$ .

We can now describe Perkins' learner types in terms of a triple of  $\Delta_F$ ,  $\Delta_E$  and  $\Delta_S$  measures. The symbol  $\approx$  means close to the target value while  $\ll$  and  $\gg$  mean far from the target value.

**Non-starter** No submission or one that can not be compiled or run:  $\Delta_S = -1 \vee \Delta_F = 1$

**Stopper** Incomplete submission that fails most tests and is smaller than required:  $\Delta_S \ll 0 \wedge \Delta_F \gg 0$

**Tinkerer** Writes verbose and inefficient code:  $\Delta_S \gg 0 \wedge \Delta_E \gg 0$ . Tinkerers, however, often have a good functional correctness score, because they will continue to add to their classes until the most obvious measurable objectives (tests passed) are met.

**Mover** Submits code that meets all the criteria of functional correctness, efficiency and readability and good design:  $\Delta_S \approx 0 \wedge \Delta_F \approx 0 \wedge \Delta_E \approx 0$

Figures 3 show the pair-wise relations between each of the three metrics. Each dot on the graphs represents an individual submission. The target for each metric is indicated by a dotted line. The x,y distance from the target gives an indication of the students' distance from the expected solution. Students to the right of the vertical line in the top and middle plots of Figure 3 are movers or tinkerers and to the left are stoppers. The bottom plot of Figure 3 shows that the defects of large code size and large execution times are largely independent of one another. It can thus be seen that all three measures are necessary for diagnostic assessment since each provides new information for distinguishing between student submissions.

The classification of submissions provided by this metric triple can be used to create a triage ranking of the submissions of a large cohort students. That is, a list of possible tinkerers is generated automatically. The instructor then examines the submissions of each student by hand, and can offer one-to-one help for students with specific difficulties. Stoppers may need more tutorial time or simply the opportunity to resubmit. Tinkerers may need further training on particular programming techniques such as debugging or planning. Movers are already making good use of the tools that are available in the lab. The range of performance across these three metrics suggests that marking schemes used in many institutions (including our own) that are based only on functional correctness can provide misleading feedback to students about their progress.

## 7 Conclusion and Discussion

We have posed the question: how can software metrics help novice programmers? This paper contributes several answers to that question. Limitations in the validity or reliability of metrics proposed in previous studies of students' code generation skills are identified. Problems with norm-referenced metrics such as final score or final grade that have been widely used in previous studies are detailed. We argue that existing grading schemes can be improved by introducing measurable targets based on software metrics. Meaningful targets can be set by studying distributions of these metrics for populations of novice programmers and metrics for sample solutions to lab exercises. Finally, we have shown that using software metrics for assessment should not be seen simply as a boon to busy academics by automating their marking duties, but that software metrics have a valuable role to play in formative and diagnostic assessment.

This study has suggested several ways in which teaching code generation skills to novice programmers could be improved.

1. Instructors should provide clear and measurable criteria on what students are expected to be able to do.
2. Assessment of programming tasks should be based on several different attributes, not just functional correctness.
3. Care should be taken to reduce the cognitive overload in program generation tasks as much as possible.
4. Students should be (strongly) encouraged to refactor and improve their code, rather than submit and forget.
5. Refactoring and quality improvement should be taught with reference to the professional context of software quality and software metrics.
6. Programming courses should allow for students who learn at different speeds, and allow for students with different levels of programming skill.

Our recommendations echo those of other studies including (Lister & Leaney 2003) and (Robins 2010). The contribution of this paper is to provide further evidence that these approaches are necessary and to offer a framework in which the effectiveness of such strategies can be scientifically evaluated based on the theory and practice of software metrics.

## Acknowledgements

The author would like to thank the students of CITS1200 at The University of Western Australia and research students and colleagues Lesley Zhang, Adam Khalid, Terry Woodings, Nick Spaddacini and the anonymous referees for their valuable feedback and suggestions.

## References

- Ala-Mutka, K. (2005), 'A survey of automated assessment approaches for programming assignments', *Journal of Computer Science Education* **15**(2), 83–102.
- Crisp, G. (2008), 'Raising the profile of diagnostic, formative and summative e-assessments. providing e-assessment design principles and disciplinary examples for higher education academic staff.', online. Retrieved February 2010.  
**URL:** <http://www.altc.edu.au/resource-raising-profile-eassessments-crisp-adelaide-2008>
- Decker, A. (2007), 'How students measure up: An assessment instrument for introductory computer science'. PhD thesis, The State University of New York at Buffalo, USA.
- Fenton, N. E. & Pfleeger, S. L. (1998), *Software Metrics: A Rigorous and Practical Approach*, PWS Publishing Co., Boston, MA, USA.
- Kitchenham, B., Pfleeger, S. & Fenton, N. (1995), 'Towards a framework for software measurement validation', *IEEE Transactions on Software Engineering* **21**(12), 929–944.
- Lister, R., Adams, E. S., Fitzgerald, S., Fone, W., Hamer, J., Lindholm, M., McCartney, R., Moström, J. E., Sanders, K., Seppälä, O., Simon, B. & Thomas, L. (2004), 'A multi-national study of reading and tracing skills in novice programmers', *SIGCSE Bull.* **36**(4), 119–150.

- Lister, R. & Leaney, J. (2003), First year programming: Let all the flowers bloom, *in* T. Greening & R. Lister, eds, 'Fifth Australasian Computing Education Conference (ACE2003)', Vol. 20 of *CRPIT*, ACS, Adelaide, Australia, pp. 221–230.
- McCracken, M., Almstrum, V., Diaz, D., Guzdial, M., Hagan, D., Kolikant, Y. B.-D., Laxer, C., Thomas, L., Utting, I. & Wilusz, T. (2001), 'A multi-national, multi-institutional study of assessment of programming skills of first-year cs students', *SIGCSE Bull.* **33**(4), 125–180.
- Morris, D. (2003), Automatic grading of student's programming assignments: an interactive process and suite of programs, *in* '33rd Annual Frontiers in Education', Vol. 3, pp. S3F – 1–6 vol.3.
- Perkins, D., Hancock, C., Hobbs, R., Martin, F. & Simmons, R. (1989), Conditions of learning in novice programming, *in* 'Studying the Novice Programmer', Lawrence Erlbaum Associates, New Jersey, pp. 261–279.
- Robins, A. (2010), 'Learning edge momentum: a new account of outcomes in CS1', *Computer Science Education* **20**(1), 37–71.
- Robins, A., Rountree, J. & Rountree, N. (2003), 'Learning and teaching programming: A review and discussion', *Journal of Computer Science Education* **13**(2), 137–172.

# Computer-supported Collaborative Learning System in Teaching E-commerce

E. W. T. Ngai<sup>1</sup>, S. S. Lam<sup>2</sup>, J. K. L. Poon<sup>3</sup>

<sup>1</sup>Department of Management & Marketing, The Hong Kong Polytechnic University,  
Hung Hom, Kowloon, Hong Kong, PR China

mswtngai@polyu.edu.hk

<sup>2</sup>Lee Shau Kee School of Business & Administration, The Open University of Hong Kong,  
Homantin, Hong Kong, PR China

sslam@ouhk.edu.hk

<sup>3</sup>Hong Kong Community College, The Hong Kong Polytechnic University,  
Hung Hom, Kowloon, Hong Kong, PR China

ccjandia@hkcc-polyu.edu.hk

## Abstract

This paper describes on a successful experience of using a computer-support collaborative learning system in teaching e-commerce. We created a teaching and learning environment for 39 local secondary schools to introduce the subject of e-commerce using the computer-support collaborative learning system which enable students to be more knowledgeable and better skilled in the subject of e-commerce. We focus here on the practical implications of the project-based learning approach for the teaching and learning of introductory e-commerce from the perspective of the business context. Evaluation results indicate that this approach to teaching is very promising. Both our own experience and student evaluations indicate that students like it and are interested in the approach of learning by doing. Finally, the paper puts forward four propositions that can guide hypothesis generation for future research.

**Keywords:** E-commerce education, Collaborative learning, Project-based teamwork game.

## 1 Introduction

As e-commerce technologies mature, more and more business transactions will be conducted and completed on the Internet. Moreover, the development of E-commerce depends on the familiarity and trust of users (Gefen, 2000; Gefen & Straub, 2000). It is suggested that the E-commerce education should be started as early as possible so that student can broaden their horizons and explore the E-world. E-commerce education should therefore be further publicised and extended to secondary schools. However, a survey conducted on business education at secondary and postsecondary levels indicated that the infusion of e-commerce topics into existing curricula is insufficient preparation for roles in companies where e-commerce is an integral part of everyday

operations (Morrison & Oladunjoye, 2000). The situation is even worse in Hong Kong secondary schools. Although their computer literacy courses now often include IT skills as part of the curriculum, there is no specific coverage of e-commerce.

E-commerce is business, as well as technologically oriented with coverage of multiple disciplines required, most of which are discussed by university-level education rather than at school. The subject of e-commerce is new not only to students but also teachers in secondary schools, making its teaching in that context difficult and challenging. Traditional face-to-face approaches may not be very efficient, and more importantly, may not be able to stimulate the student's interest. Instead of relying on secondary school teachers to be capable of handling all the tasks involved, the teaching process can be deconstructed into a series of separate processes that can be provided or supported by tertiary institution teachers and IT professionals. To promote and accelerate the understanding of e-commerce in secondary schools, the Hong Kong Special Administrative Region (HKSAR) government has funded teaching projects that focus on teaching e-commerce in secondary schools using non-traditional approaches. In consequence, a number of website design competitions for secondary students have been run which have focused either on the value of the content or the attractiveness of the web page, but have overlooked the importance of practicability and applicability. As a result, a computer-supported collaborative learning (CSCL) system that allows interactions and collaboration can then be developed to facilitate the teaching and learning environments of e-commerce.

To teach e-commerce students, Dhamija, Heller and Hoffman (1999) have provided a Web-based market system by which participants may experience buying and selling goods and services electronically. In this study, we have tackled the problem by adopting a game-based teaching approach, for the following reasons. Firstly, it is essentially an open learning setting. "In the open learning settings the learners are able and encouraged to work at their own rate. It allows the learning environment to be freed of some of the more traditional constraints of

education and makes learning more learner-centred.” (Forsyth et al., 1999). Secondly, it provides flexibility for students from different backgrounds and with different previous exposure to e-commerce to collaborate and learn at their own pace. Thirdly, it is a very important factor for motivating secondary school learners, game-based learning is more enjoyable than traditional learning (Ebner & Holzinger, 2007). The setting of the game is similar to a project, where students have to work as a member of a group as well as being self-directed. Game-based learning is a similar way of constructing and teaching courses to problem-based learning (PBL), wherein specific problem scenarios are placed within a context as stimulus and focus for student activity (Boud & Feletti, 1991; Pearson, 2006). PBL requires that “students work co-operatively in a small group, usually with the assistance of a tutor and with access to other resources, to: (a) clarify the problem; (b) identify learning needs to address the problem; (c) undertake individual reading/study; and, (d) apply newly acquired insights and understandings to re-address the problem” (Pearson, 2006).

In the following sections, we will set out the rationale and the detailed design of our project-based learning approach. We will start our discussion by briefly describing the CSCL in teaching and learning of e-commerce.

## 2 Computer-support Collaborative Learning System in Teaching and Learning of E-commerce

CSCL is an emerging field of research and practice in learning, aiming at enhancing the quality of learning. It uses synchronous and asynchronous software, text-based, audio-based or video-based communication tools, as well as shared workspaces to improve learning and instruction in various areas of education (Dillenbourg & Fischer, 2007).

In the past, although there were a number of studies investigate the acceptance, efficiency and effectiveness of CSCL in education (Collazos et al., 2007; El-Bishouty, Ogata & Yano, 2007; Kreijnsa, Kirschner & Jochems, 2003; Teoa et al., 2003; Monahan, McArdle & Bertolotto, 2008; Zaiane & Luo, 2001), the studies did not focus on specific groups such as secondary school. Many of them found a positive effect of CSCL on the learning environment.

CSCL environments are a learning medium. They can be described as a context where the computer facilitates interactions between learners for the acquisition of knowledge, skills and attitudes (Dillenbourg, 1999; Kaye, 1992). CSCL allows computers to enhance the interactions between learners (Dillenbourg, 1999). In addition, in order to enhance students’ learning processes, it supports collaboration between them (Kerijns et al., 2003). With the shared goal of the positive construction of knowledge, learners work together in this environment to accomplish a task (Dewiyanti et al., 2007). Because CSCL combines collaborative learning with the use of information and communications technology, a variety of educational, social and motivational benefits have been suggested. We believe that CSCL is an effective platform to facilitate

school teachers to introduce the subject of e-commerce to the students.

With collaborative learning, learning by the students is emphasized and student-centrism becomes the dominant characteristic of the environment in our developed CSCL system. Students, as active participants in their own learning processes, learn to solve problems and work collaboratively with their peers. Where peers play a fundamental and crucial role in encouraging learning, such learning takes place in a constructive, authentic context and is a social, collaborative activity. In order to create knowledge and meaning together with peers, listening to others and considering their viewpoints, as well as articulating one’s own perspective, is required. Student-centric learning can be cultivated in such an environment, as collaborative activities enhance students’ engagement and involvement with their team or other peers, as well as with the instructor in teaching and learning of e-commerce. Based on these arguments, the following proposition is posited:

Proposition 1: The CSCL can operationalize a constructive e-commerce teaching and learning environment

### 2.1 Group Project-based Learning

Facing increasing demands in the world of business, a profound approach to learning also exemplifies imaginative and adaptive skills, encompassing a wide sphere of interests. Student should be able to focus their attention on the overall meaning or message of a given learning context. In project-based learning, they are provided with numerous opportunities to relate ideas to each other and construct their own meanings (Ma, 1994). This learning environment will stimulate students to reveal their ideas and arguments without any fear of the negative consequences, such as being penalized or ridiculed, that may result in real life (Dewiyanti et al., 2007). In a CSCL environment, students have the opportunity to take a degree of control of their own learning (that is, when moving from the teacher-led to the student-led model) and to be active learners who are not only picking up new information but also connecting and integrating it with their previous knowledge to obtain a higher level of understanding.

Group-oriented learning processes are involved through a positive motivational and effective cognitive aspect. Teamwork in learning extends the centre of “metacognitive activity” by initiating a level of cognitive expectation beyond the individual (Alavi, 1994). Jones (1994) also proposes the framework of the “constructivist learning environment” using group practices as activity contexts. Groups may lend themselves well to computer mediated communication and the designs will intrinsically support collaboration and mutual communication about learning. Therefore, we assume group-based learning among team members provides an environment to help the students to learn the subject of e-commerce. The following sub-proposition is posited.

Proposition 1a: The CSCL can operationalize a constructive e-commerce teaching and learning environment by inducing Group Project-based Learning

## 2.2 Student Involvement/Participation

The degree and quality of participation in CSCL may be generally higher than in the traditional classroom (Weinberger & Fischer, 2006). Text-based CSCL such as e-mail, forum and chat (Kreijns, Kirchner & Jochems, 2003), using parallel discussion among peers and facilitators, may support participation. Learners can elaborate upon and express their ideas and contributions without disruptions from “co-present peers”, which may result in longer and more elaborate or sophisticated outcomes (Kern, 1995). A few recent studies also suggested that the adoption of information communication technology would improve the result of learning second language (Blake, 2000; González-Bueno, 1998; Warschauer, Turbee & Roberts, 1996).

The constructivists generally assert that knowledge is actively constructed by individuals, and that social interactions with others also play an important role in the construction process (Perkins, 1999; Wen et al., 2004). Researchers, in the field of educational technology, have also applied the constructivist theory to Internet-based or Web-based instruction (e.g., Tsai, 2001a; Yakimovicz & Murphy, 1995). CSCLs, incorporating active exploration within a virtual (pre-designed) environment, are extensively used by constructivists for two reasons (Rieber, 1992). Firstly, they provide an authentic context in which learners can explore and experiment, allowing them to construct their own mental models, in this case, e-commerce models. Secondly, the interactivity inherent in this context allows learners to see instant results and feedback as they create models or test their theories about the modelled concepts (Rieber, 1992) which is important the context of in e-business. Therefore, the following proposition is posited.

Proposition 1b: The CSCL can operationalize a constructive e-commerce teaching and learning environment by inducing student involvement or participation.

## 2.3 The Development of the Teamwork Game

We believe that the following issues must be properly addressed in the design of such a project-based teamwork game: (1) Which e-commerce concepts should be covered, and to what extent? (2) What IT knowledge and skills should the game enable students to acquire? (3) What are the most effective media for teaching students about each aspect?

Firstly, the e-commerce concepts covered in this game included Internet marketing concepts, website design, development and evaluation and security. The students were also expected to have a basic understanding of the typical components and operations of an e-commerce website.

Secondly, the necessary IT knowledge and skills included preparing a simple HTML file and creating a simple web graphic. More in-depth discussion of IT technologies was considered too difficult for secondary students and is not required to learn the e-commerce concepts.

Finally, the Internet is a new channel for the delivery of non-printed teaching materials that can be accessed by students at any time. They can learn from the online

courseware at their own pace; they can also practice their IT skills and perform transactions in the controlled e-shop environment. Students could also obtain additional support from their team-members and teacher, and from our specially arranged consultation session.

In summary, the teamwork game approach in this project was a mix of multiple teaching paradigms including:

- “Self-directed learning” – through the courseware, mainly used in order to introduce students to fundamental e-commerce concepts.
- “Coaching and collaboration” – the team was coached by a teacher and students have to collaborate to complete the game.
- “Learning support” – an additional consultation session was provided to help students solve technical problems.
- “Practical exercise” – students had to write a proposal, build up the e-shop, evaluate the outcome and review the business as well as the learning experience. A virtual shopping mall was developed within which students were to set up their e-shop. This environment narrowed down the IT skills required so that students could focus on learning the key knowledge and skills.
- “Personal involvement” – students were given virtual money so that they could use the e-shop. This allowed them to experience buying on the Internet and also learn from others’ experience of doing so.
- “Peer and professional evaluation” – students were asked to perform peer evaluation according to set guidelines. They also learned how to evaluate e-commerce websites through the exercise. Professional judges came from industrial and academic backgrounds and provided objective evaluation and feedback.

These teaching devices were logically integrated in the game in successive stages. Looking into the details of the game in the following sections can reveal them.

## 2.4 Collaboration and Cooperation in the E-commerce Project-based Teamwork Game

We believe that most secondary school teachers and students have little knowledge or concept of e-commerce. This means that students would require additional support to complete our project-based teamwork game. Louvieris and Lockwood (2002) suggest that the traditional role of the teacher should be deconstructed into a series of separate individual capabilities supported by appropriate IT infrastructure, with the different tasks within the teaching and learning process handled by “best in class” experts. The teamwork game enabled secondary school teachers to collaborate with tertiary institution teachers and IT professionals to teach students on new subject areas in which they may not themselves have been trained. Each contributed to different elements of the teaching and learning process of the game. Figure 1 illustrates how the teaching and learning processes of the teamwork game were supported by area experts.

The learning outcomes were established by us (that is to say, tertiary institution teachers). We also provided the

content, and the Web-based teaching and game playing platform was developed by IT professionals from the university. To improve business students' skill in e-commerce, either the amount of cooperative learning must be increased or some formal instruction must be introduced (Susser & Ariga, 2006). In addition to face-to-face teaching and learning, the Internet provides an alternative channel for students to learn as well as to collaborate and cooperate with their classmates and teachers, anytime and anywhere. Different teaching and learning activities could be conducted either face-to-face, on the Internet, or both. The web-based teamwork game utilized both channels to improve the collaboration and cooperation of students and teachers. Figure 2 summarizes the teamwork activities involved and how the Internet can mix with face-to-face learning for each teaching paradigm. For the CSCL system to achieve successful in teaching and learning performance of e-commerce, it must be willing to collaborate and cooperate in teamwork. Thus the following proposition is posited:

Proposition 2: The CSCL can foster teaching and learning performance by facilitating collaboration and cooperation in team work.

## 2.5 Details of the Project

"Project-based Teamwork Game in e-commerce" was proposed by the Department of Management and Marketing of The Hong Kong Polytechnic University, and funded by the Quality Education Fund (QEF), HKSAR, China. The project aimed to promote the concept of "learning by doing" via a project-based e-commerce game and to encourage schools to make use of this type of learning to nurture various abilities, knowledge, skills and learning attitudes in students. The project enabled students to enjoy learning via the project-based game, to enhance their effectiveness in communication, to develop a spirit of teamwork and to develop their creativity in the PBL environment. Students were required to develop a simplified, yet realistic, e-shop that would address the e-commerce educational needs of small and medium sized business proprietors. The idea of this exercise was to provide an integrated learning experience, including the e-business proposal, e-shop development and writing of the final debriefing report.

The educational goals of this project were as follows:

- To promote e-commerce education in secondary schools;
- To encourage non-textbook learning of e-commerce concepts through action learning and game playing;
- To design and develop a Web-based courseware product in order to facilitate schoolteachers and students to work through the game on the Internet. This included e-business planning, building a virtual e-shop and e-business evaluation.

To achieve these goals, the game must be well designed and developed and should be fully thought out. Initially, schoolteachers and students in nine pilot secondary schools showed interest in the project. Eventually, the targets were extended to all Hong Kong senior students and teachers with an interest. Before starting the

competition, a meeting was held involving eight pilot school teachers and project team members at which we demonstrated the interface and the functions of the competition website; discussed the schedule of the competition period; and talked through the rules of the game.

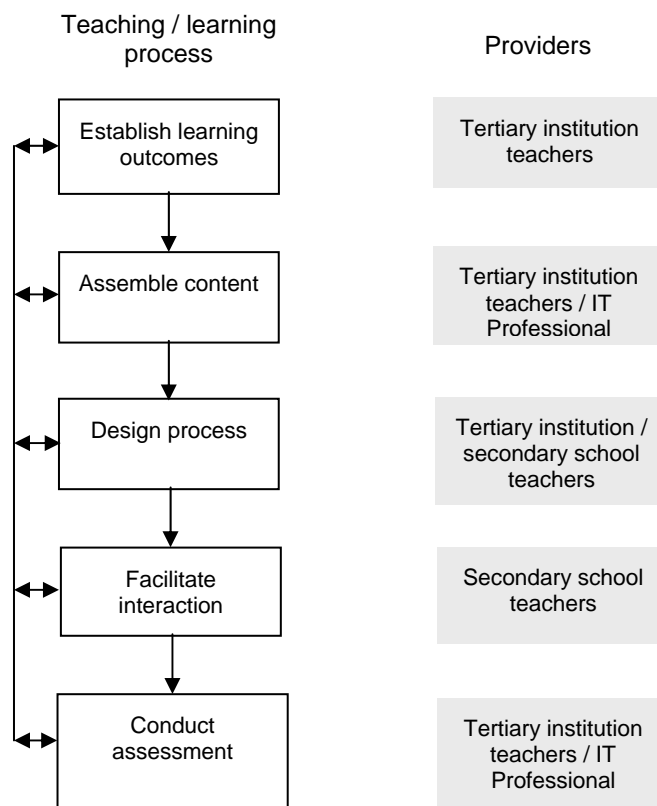


Figure 1: Providers of teaching and learning processes in the game

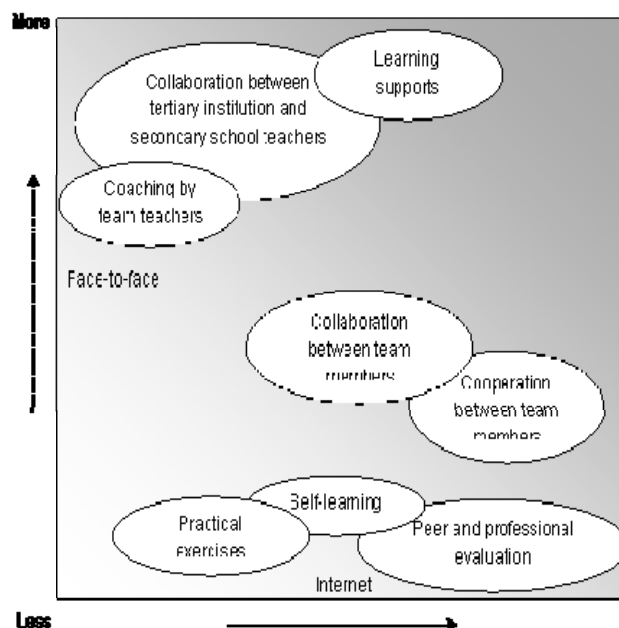


Figure 2: Web-based teamwork teaching and learning environment



As well as this, we provided a workshop introducing the details of the competition. Invitation letters were sent to all secondary schools in Hong Kong. The workshop was divided into four sessions. Session one introduced the details of the game; the second session presented the e-commerce concept and the procedure of the e-shop development; the third guided users through the systems on the competition Web-site; and the final segment was a question and answer session. A free flow of ideas and creativity was encouraged during both the meeting and workshop. Also, the latest technology and IT news was introduced to all teachers and members.

Interactive online tutorials, including the history and development of e-commerce and the use of Flash and Dreamweaver, were provided to all participants to enable them to acquire some basic knowledge of e-commerce and the skills required for the e-shop development. In addition, we provided a helpdesk service to team members. Through an online booking system, students could make an appointment to consult our tutor. The tutor would answer technical questions, such as how to use Flash and Dreamweaver to design and develop the e-shop, how to use the e-shop platform, e-commerce and e-marketing knowledge, and so on.

Furthermore, a series of meetings with teachers and team members were conducted during the e-shop development stage. In these meetings, we demonstrated the teaching materials to teachers and discussed student feedback.

Since the peer evaluation was conducted using an inbuilt online evaluation system (OES), judges and participants were able to leave comments on the system. After the peer evaluation, team members could access the OES to review others' comments on their work. The judges indicated the weakest and strongest performers in terms of the e-shop content and debriefing report.

In open learning, courseware is the key device by which to provide a flexible learning environment for students. Essentially, we developed some interactive, self-learning courseware, which may be briefly described as follows:

- Introduction to e-commerce;
- E-commerce Development;
- Online E-commerce Software Development Tutorial.

### 2.5.1 Introduction to E-commerce

Firstly, we introduced the basic concept of e-commerce to the students, beginning with the topics of "What is e-commerce?", the characteristics of the Internet and the World Wide Web, and finishing with a discussion of the advantages and disadvantages of e-commerce.

### 2.5.2 E-commerce Development

This section aimed to introduce the development life cycle of e-commerce and can be outlined in Figure 3.

### 2.5.3 Online E-commerce Software Development Tutorial

We provided an online tutorial using Macromedia Dreamweaver and Macromedia Flash. These were chosen as the tools for e-commerce development because of their

ease of use and suitability for secondary students' learning.



**Figure 3: Online e-commerce software development tutorial**

## 2.6 System Overview

We designed and developed a virtual shopping mall as a platform for the students to build their e-shops. Figure 4 depicts the basic architecture of the competition website. The shopping mall consisted of several floors:

The main (first) page of the website displayed the ground floor of the shopping mall (see Figure 5). An elevator was located at the left hand screen of the screen, which allowed the user to visit different floors. In addition, a counter recorded the number of visitors accessing the competition website. There was a menu at the right hand side, which provided general information about the competition, courseware, helpdesk services and final evaluation.

All e-shops were located between the first and highest floors. The senior and junior groups' e-shops were divided across different floors, with the latter located at lower and the former at higher floors. A floor plan was designed and placed within each floor. To enter the e-shop, users could either select from the right hand side menu or click the target shop number on the floor plan (see Figure 6).

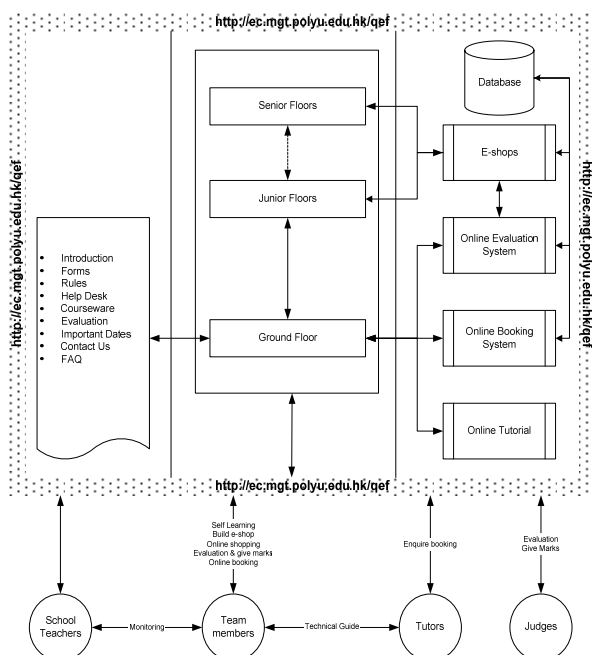
## 2.7 The Competition Process

The competition of the game was divided into the following five phases.

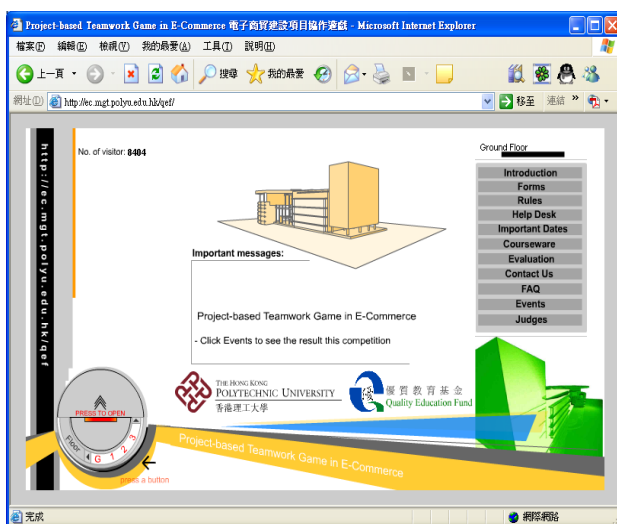
### 2.7.1 Phase I - Registration

Participants could join the competition only as a team. Each team consisted of at least two and no more than six students from secondary schools in Hong Kong. No more than two teams from each school in each section (that is, junior and senior) were permitted. Each team member had to be from the same school and each team was required to have one leader who helped form the team and acted as a contact point between its members and the organizer.

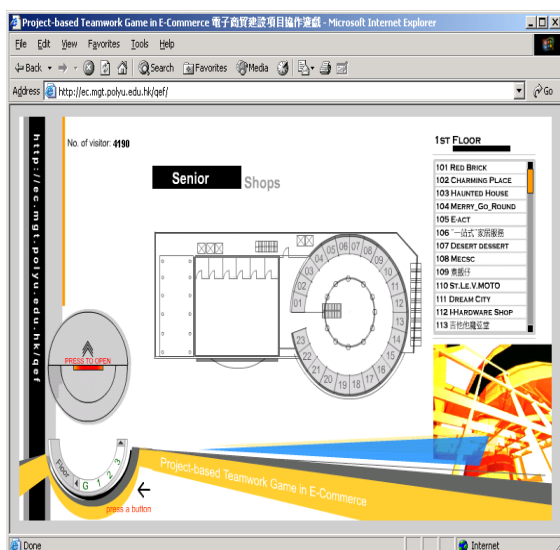
As well as this, students were required to find at least one advisor (that is, a schoolteacher) to be allowed to join in. The role of the advisors, as coaches, was to guide their team through the competition. Most teams chose teachers



**Figure 4: Project-based teamwork game - e-commerce website architecture**



**Figure 5: Ground floor of the virtual shopping mall**



**Figure 6: First floor of virtual shopping mall**

of either Computer Studies or Business and Economics as their guides. In the early stages, these teachers provided valuable suggestions to the students as to how they should sell in their e-shops and how to write a business plan.

Each team was asked to submit a registration form declaring their intent to participate. Once their registration had been accepted, they were allocated a set of user IDs and passwords to access the competition server. A workshop was conducted to introduce the competition and the concept of e-commerce to the participants, and to demonstrate the competition website.

### 2.7.2 Phase II - Simple Business Plan

Each team was required to propose a business plan outlining the product line, targets and operation of the business; and to forecast their profitability. A template was provided so that students would know what should be included in a business proposal. Students had to collaborate and apply what they had learned through self-learning in order to do this successfully.

### 2.7.3 Phase III (E-shop Content)

Each team then built their e-shops using a template provided on the competition server. There was no need to build the shopping cart or checkout process and so on. Participants were only required to upload three main items to the competition server, namely (a) images of their products and services with a description; (b) a Flash logo to represent their shop; and (c) a Flash banner to promote it. This allowed students to practise their IT skills.

Learning support was provided. During this period, if participants encountered any problems related to building the e-shop, a face-to-face helpdesk service was available. Participants were able to make an appointment with the tutors through an online booking system.

#### 2.7.4 Phase IV - Online Shopping

Each team member, acting as a customer, was encouraged to visit other e-shops and buy their products/services, so as to obtain some first-hand experience of Internet purchasing. This is not something a secondary school student would normally have done before in real life.

### 2.7.5 Phase V - Report Submission and e-shop Evaluation

Students were asked to follow a set of guidelines to write a debriefing report as a conclusion to the competition. At the same time, each team was asked to evaluate other e-shops and give them marks that would form part of their final results, using the OES. As well as this, the debriefing report and the e-shops were evaluated by professional judges (either industrial or academic experts). This provided further valuable information for assessing the students' levels of achievement.

## 2.8 Assessment Criteria

The judging process was the major vehicle for assessing students' ability to apply their subject-specific knowledge and understanding. Two parties (participants and judges) were involved in the assessment, both using the online evaluation system (OES). The OES had been designed and developed to enable students and judges to give marks

based on listed assessment criteria by clicking on one of the number in the scale (0 = “no evidence provided”, 1 = “inadequate”, 2 = “barely adequate”, 3 = “satisfactory”, 4 = “good” and 5= “excellent”). Refer to Figure 7 for the screen layout for part of the OES.

Assessment Criteria	0. No Evidence provided	1. Inadequate	2. Barely adequate	3. Satisfactory	4. Good	5. Excellent	Weight
<b>Business Viability 生意的可行性</b>							
1. Identification of clear business objectives and sound business strategies 訂立清晰的生意目標和穩健的公司策略							7.0%
2. The selected product/service for sell is suitable for B2C 所售賣的產品或服務符合顧客的要求							7.0%
<b>Content 內容方面</b>							
3. Presentation style and language accuracy 表達的風格和語言的準確度							15.0%
4. The information of the product/service is clear and complete which assist the consumer in making a buying decision. 提供清晰的產品或服務的資料，例如定價、貨品描述等							15.0%

Figure 7: Online Evaluation System

Participants were also invited to assign a score for other team's e-shops content to form part of the final competition results. A total of seven judges from different areas (including a legislative councillor, university professor, chairman of the Hong Kong Association for Computer Education, IT director and the managing director of a private company) were involved, all of whom had a special interest in, and strong knowledge of, IT or e-commerce. At least two judges evaluated each team's e-shop content and debriefing report.

There were four major assessment criteria for evaluating the content of the e-shops:

- Business viability: Identification of clear business objectives and sound business strategies. The selected product/service was suitable for B2C.
- Content: Presentation style and language accuracy. The information on the product/service was clear and complete which would assist the consumer in making a buying decision.
- Creativity: Demonstration of originality and creativity. Innovation and level of differentiation from competitors.
- Design: The product/service categories were classified logically. The product/service description, images, banner, and logo were visually attractive. Reasonable download times for client.

Hard copies of the debriefing reports were mailed to the judges but they could also view them online via the OES. There were two major assessment criteria for evaluating the debriefing reports:

- Report on e-shop setup: Content should cover the review of marketing strategy (4Ps), presentation of sales reports and suggestions for e-shop

enhancement. Overall, the report should be presented in a clearly logical and sequential form.

- Report on learning experience/enhancement of knowledge: Students should specify what they have learnt, compare the e-mall with other e-malls on the Internet and suggest further development of the game. Again, this content should be well organized.

### 3 Feedback of Project-based Teamwork Game on Secondary Schools

Secondary school teachers and students were strongly appreciative of the online multimedia courseware. These teaching materials included the concept of, and how to develop, e-commerce together with an animated, step by step beginner's learning guide to e-commerce development tools such as Flash and Dreamweaver. The e-shop platform allowed students to participate without having to have strong programming skills. Teachers could also make use of this platform to teach e-commerce concepts using a teamwork game. We will now discuss the impact of the project on students, school teachers and participating schools.

#### 3.1 Learning of Students

Feedback was collected and analysed from students at the end of the competition. The direct impact of the project on students' learning included:

- Broadening students' horizons. As most of the students did not have a concept of e-commerce at the beginning, the opportunity to learn about this using non-textbook means – that is to say, learning by doing via the project-based e-commerce game – could be a very effective pedagogy for broadening students' horizons in non-school curriculum subjects.

“We strongly appreciated the workshop because it's not just about the competition; it also provides us with a chance to learn more about the fast growing world of e-commerce which will help broaden our horizons”.  
Quoted from e-shop 206

- Increasing students' sense of achievement. Most of the participating teams successfully completed the development of their e-shops. Students were extremely happy to see what had been sold from their own e-shop.

“This competition helps us to develop analytical skills since we have to make a financial analysis to interpret of the firm's performance. Moreover, this game allows us to develop our creativity, which is essential in the fast-changing world nowadays”. “The game helps us to equip ourselves for the future commercial world”. Quoted from e-shop 213

- Fostering students' development of their potential and specific abilities. Students demonstrated their ability to use website development tools such as MS FrontPage and Flash in their e-shop development.

“We have learnt many business concepts and knowledge and gained many IT skills during the competition”. “The competition is not just about IT skills, but also contains a lot of reading, learning, thinking, and English writing. It is very different from other competitions that we had joined before, but somehow, we enjoyed doing this project very much and have gained a deep insight to the real business world”. Quoted from e-shop 302

“Before joining this project, we were not interested in knowing how to use the software like Flash, because it is quite difficult for us. After doing this project, we know that we are capable of mastering it. Our technical skills have greatly improved after this project. We have also learnt to understand what marketing is and how to write amazing and persuasive descriptions for products to attract buyers. How to co-ordinate with the group members so as to optimize the results”. Quoted from e-shop 205

- Equipping students with a variety of learning approaches. We provided a workshop to introduce the details of the competition and the fundamental concept of e-commerce. We also developed courseware including online tutorials for the development of the e-commerce site and the use of Flash and Dreamweaver to support the e-shop development. A peer evaluation was adopted as a learning approach.
- Training students to better meet social demands. One of the our reasons for choosing the software packages MS FrontPage and Flash was that no prior programming experience was necessary to use them and it would be easy for participants to pick them up, developing skills which would subsequently be marketable in the information system/e-commerce workplace. Apart from sharpening their computer skills, other business concepts like economics and marketing were introduced to students through their submission of the business plan. For example, students needed to decide what to sell and analyse the demand and supply of the products/services provided, as well as developing a pricing strategy, incoming analysis and budget analysis.
- Cultivating students' team spirit. As students were required to join the competition as a team, a strong sense of commitment, cooperation and responsibility was built up among members.

#### 4 Conclusions

This paper has described a collaborative project between school and university educators. We successfully

introduced e-commerce as a subject to improve teaching and learning within the participating secondary schools, focusing on key aspects of introducing the concept and the underpinning skills and knowledge, by developing and running an e-business via the e-shop simulation. We adopted a project-based learning approach by working on tasks and integrated projects that simulated a business environment, thereby enabling students to learn how to solve real-life e-commerce problems. Throughout the project, students not only had the opportunity to reaffirm the skills, knowledge, concepts and attitudes they had already acquired but also to develop a strong teamwork spirit and develop their creativity. The project requires students to learn some IT technical skills (such as HTML, Flash computing skills). We encountered difficulty to let some junior groups (Forms 1 -3) of school students to learn how to create objects moving in more sophisticated ways using Flash. Another difficulty encountered was that we cannot provide frequent updates in the self-learning IT teaching materials due to limited resources. This project has been characterized by schools volunteering to supervise the students to participate. We believe that this type of project-based teamwork game enables secondary students to apply their learning, enhance their effectiveness in communication, develop their creativity and stimulate their interest in entrepreneurship. It also motivates students to improve their IT skills, gain in confidence and stimulate their desire to learn and achieve.

#### 5 Acknowledgements

We would like to acknowledge Department of Management & Marketing, The Hong Kong Polytechnic University to partly sponsor this project. This project was supported in part by Quality Education Fund (QEF), HKSAR, China.

#### 6 References

- Alavi, M. (1994). Computer-mediated collaborative learning: an empirical evaluation. *MIS Quarterly*, 18(2), 159-174.
- Blake, B. (2000). Computer mediated communication: A window on L2 Spanish interlanguage. *Language Learning & Technolog.* 4(1), 120-136.
- Boud, D., & Feletti, G. I. (1991). *Introduction*. In D. Boud, & G. Feletti (Eds), *The Challenge of Problem-based Learning*, 13-20. London: Kogan Page.
- Collazos, C. A., Guerrero, L. A., Pino, J. A., Renzi, S., Klobas, J., Ortega, M., Redondo, M. A., & Bravo, C. (2007). Evaluating collaborative learning processes using system-based measurement. *Educational Technology & Society*. 10 (3), 257-274.
- Dewiyanti, S., Brand-Gruwel, S., Jochems, W. & Broers, N. J. (2007). Students' experiences with collaborative learning in asynchronous computer-supported collaborative learning environments. *Computers in Human Behaviour*, 23, 496-514.
- Dhamija, R, Heller, R., & Hoffman, L. J. (1999). Teaching e-commerce to a multidisciplinary class. *Communications of the ACM*, 42(9), 50-55.
- Dillenbourg, P. (1999). *Introduction: what do you mean by "collaborative learning"?* In P. Dillenbourg (Ed),

- Collaborative learning: Cognitive and computational approaches. Amsterdam: Pergamin Press.
- Dillenbourg, P. & Fischer, F. (2007). Basics of Computer-Supported Collaborative Learning. *Zeitschrift für Berufs- und Wirtschaftspädagogik*, 21, 111-130.
- Ebner, M., & Holzinger, A. (2007). Successful implementation of user-centered game based learning in higher education: an example from civil engineering. *Computers & Education*, 49(3), 873-890.
- El-Bishouty, M.M., Ogata, H. & Yano, Y. (2007). PERKAM: Personalized knowledge awareness map for computer supported ubiquitous learning. *Educational Technology & Society*, 10 (3), 122-134.
- Forster, M., & Masters, G. (1996). Projects. The Australian Council for Educational Research Ltd.
- Gefen, D. (2000). E-commerce: the role of familiarity and trust. *Omega*, 28, 725-737
- Gefen, D. & Straub, D. (2000). The relative importance of perceived ease of use in IS adoption: A study of E-commerce adoption. *Journal of the Association for Information Systems*, 1 (8), 1-28.
- González-Bueno, M. (1998). The effects of electronic mail on Spanish L2 discourse. *Language Learning & Technology*, 1(2), 55-70.
- Jones, A (1994). Constructivism and Other Approaches to Teacher Education. *Teacher Education Quarterly*, 21(3). 28-38.
- Kaye, A (1992). Learning together apart. In A. Kaye (Ed). *Collaborative learning through computer conferencing: The najaden papers*. Berlin: Springer.
- Kern, R. G. (1995). Restructuring classroom interaction with networked computer: Effects on quantity and characteristics of language production. *The Modern Language Journal*, 79, 457-476.
- Kreijns, K., Kirschner, P.A. & Jochems, W. (2003). Identifying the pitfalls for social interaction in computer-supported collaborative learning environments: a review of the research. *Computers in Human Behavior*, 19(3), 335-353.
- Louvieris, P., & Lockwood, A. (2002). IT induced business transformation in higher education: and analysis of the UniCafé experience and its implications. *Computers & Education*, 38, 103-115.
- Ma, J. (1994). Problem-based learning with database systems. *Computers and Education*, 22(3), 257-263.
- Monahan, T, McArdle, G. & Bertolotto, M. (2008). Virtual reality for collaborative e-learning. *Computers & Education*, 50, 1339-1353.
- Patterson, M. L. (1996) Social Behavior and Social Cognition. A Parallel Process Approach, In J. Nye & A. M. Brower (Eds). *What's Social about Social Cognition: Research on Socially Shared Cognition in Small Groups*, 87-105. Newbury Park, C. A.: Sage Publications.
- Pearson, J. (2006). Investigating ICT using problem-based learning in face-to-face and online learning environments. *Computers & Education*, 47, 56-73.
- Perkins, D. N. (1999). The many faces of constructivism. *Educational Leadership*, 57(3), 6–11.
- Rieber L. P. (1992) Computer-based microworld: A Bridge between Constructivism and Direct Instruction. *Educational Technology Research and Development*, 40(1), 93-106.
- Tsai, C.-C. (2001a). The interpretation construction design model for teaching science and its applications to Internetbased instruction in Taiwan. *International Journal of Educational Development*, 21, 401–415.
- Tsai, C.-C. (2001b). A review and discussion of epistemological commitments, metacognition, and critical thinking with suggestions on their enhancement in internet-assisted Chemistry classrooms. *Journal of Chemical Education*, 78 (7), 970-974.
- Teoa, H.H., Chana, H., Weib, K. & Zhang, Y. (2003). Evaluating information accessibility and community adaptivity features for sustaining virtual learning communities. *Int. J. Human-Computer Studies*, 59, 671-697.
- Warschauer, M., Turbee, L. & Roberts, B. (1996). Computer learning networks and student empowerment. *System*, 24(1), 1-14.
- Weinberger, A. & Fischer, F. (2006). A framework to analyze argumentative knowledge construction in computer-supported collaborative learning. *Computers and Education*, 46, 71-95.
- Wen, M. L, Tsai, C-C., Lin, H-M, Chuang, S-C (2004). Cognitive-metacognitive and content-technical aspects of constructivist Internet-based learning environments: a LISREL analysis, *Computers & Education*, 43, 237 – 248.
- Yakimovicz, A. D., & Murphy, K. L. (1995). Constructivism and collaboration on the Internet: Case study of a graduate class experience. *Computers & Education*, 24, 203–209.
- Zaiane, O.R. & Luo, J. (2001). Towards evaluating learners' behaviour in a Web-based distance learning environment. *Proceedings of IEEE International Conference on Advanced Learning Technologies*, 357-360.





# A method for analyzing learning outcomes in project courses

Mattias Wiggberg and Mats Daniels

Department of Information technology

Uppsala University

PO Box 337, 751 05 Uppsala, Sweden

mattias.wiggberg@it.uu.se and matsd@it.uu.se

## Abstract

Including projects in courses is widely recognized as important in preparing students for their profession. Typical examples are capstone courses, where the students are supposed to use their previous knowledge and skills to show mastery of their craft. What this actually amounts to in terms of learning objectives is however often quite fuzzy, resulting in contradictory ideas about how to actually run them in order to reach these learning objectives. This paper presents a method for analyzing learning in project courses, based on the combination of the theory of communities of practice and an identification of key features in project work. The method can be used to gain an understanding of project courses in order to set up a learning environment suitable for its learning objectives. The focus is on the students' experiences and how they are mapped on desirable learning outcomes. The results are stories related to key features of project work expressed in the theory of communities of practice that capture the strengths and weaknesses of the studied project course.

**Keywords:** Communities of practice, computing education research, learning outcomes, and project courses.

## 1 Introduction

Group, or project, work is today seen as an integral and important aspect of computer science education, e.g. as expressed in the ACM/IEEE Computing Curricula (2005). The folk pedagogy (Bruner 1996, Lister 2008) of computer science teachers support this by pointing out that project work give the students added challenges and increases their preparation for work life, especially if known development methods from industry is used to emphasize the reality aspect. It is not uncommon to involve industry partners as mock clients in order to increase the feeling of reality in student projects.

There is no shortage of interesting and innovative ideas on how to run project courses, but the educational value is seldom demonstrated. The complexity of evaluating the learning outcome of student projects is daunting (Wait et al. 2004 and Barker 2005) and thus mostly not covered in the literature. Examples of efforts to investigate the effect of student projects in terms of learning outcomes are given

by Baker and Garvin-Doxas (2004), Berglund (2005), Daniels et al. (2010), Kinnunen and Malmi (2004), and Wiggberg (2010). Much remains however to be done before the body of knowledge among the computer science teachers related to learning outcomes of various implementations of student projects has reached an acceptable level in comparison to the number of study hours the students spend in such projects. With this in mind it is vital to make sure that the learning experiences are of high quality. The guiding question in this paper is thus:

*How can we design and set up computer science student projects in order to make them contribute to students' development and be of a high educational quality, based on a firm research ground?*

A method for analyzing learning outcomes from student projects is presented in order to address this question. The method is built on four key features for learning in student projects, identified by Wiggberg (2008), and is used in combination with the theory of communities of practice (Wenger 1998). One common and important assumption with regard to learning in project courses is that they will prepare the students for their profession, in this case as IT-workers. Investigating this assumption can be approached by using the theory of communities of practice (Wenger 1998) as base, since it provides a platform for discussing work related experiences as well as learning outcomes in terms of how the students enter the community of practice of IT-workers.

The paper first present the theory of communities of practice with special consideration to computer science student project courses, followed by a fuller description of the method. The paper concludes by presenting an illustrative story based on one of key features: *the way work is allocated*, as illustration of the result and a reflection on how this result can be used in designing computer science project courses.

## 2 Communities of Practice

In the theory communities of practice, learning is thought of as a result of social participation (Wenger 1998, p. 4). Wenger suggest that learning is a natural and inevitable part of life. Learning in communities of practice opposes the assumptions that learning is an individual process where a one- or bi-directional communication between the learner and the teacher is seen as an effective way of transferring knowledge. Communities of practice instead place the participation in a social process, the practice of a community in a certain domain, as the way to learn.

Wenger puts it:

*Learning is not refined as an extraneous goal or as a special category for learning something else. Engagement in practice - in its unfolding, multi-dimensional complexity - is both the stage and the object, the road and the destination.* (Wenger 1998, p. 95)

Viewing communities of practice as a concept, it can be thought of as a group of human beings sharing a common interest, or a set of problems, in a topic. The group members increase their expertise in the topic by interacting with each other through certain practices in an ongoing process (Wenger et al. 2002). Although the perspective on learning assumes it is an ongoing activity, it does not make it trivial by saying that everything is learning in the communities of practice-sense. Learning has to do with the practices and the learners' ability to negotiate meaning (Wenger 1998, pp. 95–97). Connecting learning with practicing is a central assumption in this work.

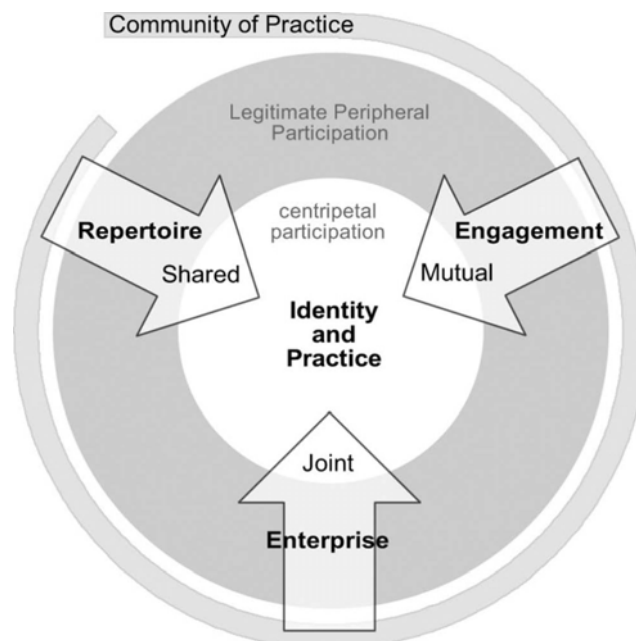
Communities of practice exist in places where we share our everyday lives, and often we do not recognize them as communities of practice (Wenger 1998). That is, communities in this context are where we do things together with other people, in a shared domain of interest and with certain practices. A community of practice is a special case of a community and there is a need to clarify what makes it thus.

Wenger put crucial requirements on the community in order to call it a community of practice. These requirements are captured by these three characteristics:

1. **Domain** A shared domain of interest exists among the members. Competence in the domain subject distinguishes a member from a non-member. This implies a commitment to the domain by the members.
2. **Community** Relationships that enables learning from each other. Members of the community share information and help each other. They are also actively engaged in joint activities and discussions.
3. **Practice** Community members are practitioners who work actively in the domain. They develop resources, such as tools, experiences, stories, and ways of addressing recurrent problems in order to enhance their common work. The process of developing resources takes time and requires interaction between the community members.

According to Wenger, a community of practice is recognized by the presence of these three characteristics in a combination. The cultivation of the community of practice is based on development of the three characteristics. Important aspects of communities of practice thus involve shared repertoire, domain, mutual engagement and joint enterprise practice, see figure 1.

A very central mechanism in communities of practice is the concept *legitimate peripheral participation* and the transition to central participation and the reverse process. Lave and Wenger (1991) formulates legitimate peripheral participation as a theoretical description of how newcomers, people who are new to the community, enters a community of practice can become more experienced members of the community of practice. Starting with less



**Figure 1: An overview of Communities of Practice (Henderson & Bradey, 2008).**

prestigious low-risk tasks a participant can get acquainted with the community's tasks, cultural expressions, skills and other expressions of the communities shared repertoire. As time goes by the member becomes more senior in the community and gains access to the more central functioning (Lave and Wenger 1991).

An important observation from the theory of legitimate peripheral participation is that newcomers who gain access to experts and can study their practice, understands their own activities within the community. In contrast, newcomers with less access to the more central members of the community have a more flat learning curve than members with more access (Lave and Wenger 1991). Legitimate peripheral participation hence can be used to reflect how members of the community through practices can become more experienced members of the community. Doing that, they will also be more engaged in different practices in the community.

## 2.1 The Project Group as a Community of Practice

Courses based on computer science student projects are the focus of this paper and the student group in such a course is seen as a community of practice. That community consists of the project group itself and their closest collaborators such as teachers and industry contacts. The idea is that this is a part of the IT-worker community of practice in a constellation of practices as defined in (Wenger 1998).

The group formed by the project participants is not a blank slate. It contains people who have knowledge in different areas in what is thought to be valuable to the project. That means that when the project group is starting to form their community, some members are already more knowledgeable and have a greater opportunity to take a more central role in the community. This is a prerequisite for the group to become a functional community of



practice. The process of legitimate peripheral participation and the movement from there to more central parts of the activities in the community needs, or at least benefits from, having a varied level of knowledge among the members in the initial phase of the community.

## 2.2 Communities of Practice as Yardstick

Communities of practice and especially the central concept of legitimate peripheral participation provide the stage for analyzing computer science student projects as learning environments.

The assumption here is that a communities of practice is created in computer science student projects and that there are enough ties to the community of practice of IT-workers refer to these as constellations of practices. By using empirical data to analyze how participants contribute - or not - to constitute a community in the project course, it is possible to unwind actions and behaviors that facilitate - or prevent - the possibility to become a central member of that community, and hence also belong to the community of IT-workers.

The learning outcomes of computer science project courses are defined by the formal course descriptions and interpreted by the teachers giving the course. The learning outcomes are in this work “translated” to practices under the assumption that the courses works as communities of practices. In the study presented in this paper, the following set of practices was used, derived as described in Wiggberg (2010):

1. Working efficiently and constructively in a large project team of developers;
2. Planning and follow-up of a complex project task, and taking care of unexpected things that might occur.
3. Getting experiences of applying previous knowledge; and learning skilful use of tools used in the IT-community.
4. Integrating smaller tasks into a larger task.

## 3 Method

The general idea is to explore which effects variations of these features can have on the learning outcomes in the student cohort. The method to analyze learning outcomes is based on a matrix with the key features on one axis and a set of practices deemed important in the community of practice in question on the other axis. The steps in using the matrix are:

- 1) semi structured interviews give information about the features in the current project;
- 2) identifying intersections where experiences of key features influenced important practices or vice versa;
- 3) stories are woven to illustrate the selected intersections with the aid of the interviews.

The stories are the result of the method and reveal the students’ experiences in relation to the learning outcomes in projects.

The starting point of the method can be seen as the identification of key features in the setting under study and to identify important practices in the intended community. Previous work (Wiggberg 2008) has identified four such

features that will be used in this paper. The next step is to get information about these features from the student group in question, including selecting students to conduct semi-structured interviews with. The conceptual framework of key features is used to categorize interview data and to identify experiences connected to the key features, and establishing relations between key features and identified important practices.

A matrix is then created, where key features are columns and identified important practices constitute the rows as depicted in table 1. The excerpts from the semi-structured interviews are then analyzed in order to find experiences of key features influencing important practices or vice versa that can be mapped onto intersections in the matrix. In the final step, for each column in the matrix (representing one key feature) the excerpts from intersections in the column that are marked as interesting are traced back to their original interview. Interviews showing a high presence of findings in a column are selected as candidates for being used as a foundation for the creation of a story<sup>2</sup>.

	Mechanism for Work Allocation	Connection to External Stakeholders	Focus on Result or Process	Level of Freedom in Task
Practice 1	Observed connection	Observed connection		
Practice 2				Observed connection
Practice . . .		Observed connection		
Practice n			Observed connection	

**Table 1: Matrix combining key features and identified important practices.<sup>3</sup>**

In the following section an illustration of the outcome of the method is presented.

## 4 An illustrative story: A Matter of Motivation and Work

### 4.1 Background

Data was collected from 11 students participating in a computer science project course (Wiggberg 2010). The

<sup>2</sup> Excerpts from different interviews with the same student can inform the same story.

<sup>3</sup> Real data are not present in this general illustration. Cells are filled by interesting observations from interviews.

aim with the project was to develop rescue robots, and the project involved building upon software and hardware from earlier instances of the same project course. To fulfill the main task of the project the students had to build rescue robots that autonomously could locate victims in hazardous environment. This included using advanced sensor systems as well as managing advanced communication between the robots. The project team consisted of 22 students (Abbasi et al. 2009).

The practices identified were discussed in relation to legitimate peripheral participation in the theory of communities of practice. The key reason for relating to legitimate peripheral participation is that newcomers who gain access to experts and can study their practice, understands their own activities within the community (Lave and Wenger 1991). Legitimate peripheral participation is thus a natural concept for reflecting on how a student becomes a more experienced (central) member of the community. A natural observation is that being engaged in different practices in the community is an important aspect in the transformation from being a peripheral to becoming a central member of the community. The aim is to capture aspects of the project under study that promote – or hinder - the transformation towards becoming a central member and to express these experiences of the students in stories. One such story is given below.

David<sup>4</sup> was an Information Technology engineering student that wanted to work in the IT business. David participated in the project HUGE (Abbasi et al. 2009)). During the course, David and the other participants were expected to experience practices identified as important for the movement into the larger community of practice of IT-workers. Recalling the practices identified as capturing this experience is described by:

1. Working efficient and constructive in a large project team of developers.
2. Planning and follow-up of a complex project task, and taking care of unexpected things that might occur.
3. Get experiences of using obtained knowledge, and learn how to use certain tools used in the IT-community.
4. Integration of smaller tasks into a larger task.

Students participating in the course had these practices highlighted by the team of teachers and, to some extent, had also been reading about them in the formal course description.

## 4.2 Story

David had been reasonably motivated to participate in the student project. He had confidence in his skills in relation to the project, and felt confident in making himself and his needs listened to. During the project, David thought a lot about how the process of work allocation was implicitly and explicitly handled in the project group.

Initially David reflected on the amount of people in the

project. He said that his possibility to get an overview of different subtasks and work performed was low due to the complexity of the main task. The number of teammates in the project had made that situation even harder. David was worried that this would lead to problems with becoming involved in discussions with and learning from his fellow students. During the project David got his expectations of peer learning partly confirmed in that the team had to split both the task and the work in smaller pieces. He found the discussions on technical challenges in these smaller groups rewarding. David had the feeling that they had to split tasks and work to a larger extent than they initially wished.

**Interviewer:** *How has it worked out in the project, have you worked together... it is a pretty hectic time period with....*

**David:** *[...] well it could be a disadvantage that we are as many as we are. There is a focus on ones own piece at the price of not getting an overall image of it all as one surely would have if there had been fewer in the project, one would have had to know about all the things then.*

Lack of time, the complexity of the task and the stress those two caused were suggested as reasons for less communication among the teammates. The stress led to more solitary work and less interaction between the teammates.

**Interviewer:** *How do you collaborate, like do people work with different things, or do two or more collaborate on the same thing?*

**David:** *We have said earlier, or rather from the start, that we would try to sit down [together] more when doing ordinary programming. But unfortunately this didn't happen as much as we wanted due to lack of time.*

Continuing on the thread of lack of time David described this situation:

**David:** *We needed to divide ourselves a bit in order to manage to do everything. That is a bit unfortunate since one should makes more mistakes when one work alone and don't have anyone to discuss with except when one is really lost and knows one really has to talk with someone. It thus doesn't get to the same cumbersome walkthrough and ends up with the same feedback as on everything else one does.*

These thoughts on stress as a factor for limiting discussion and peer learning were mixed with David's impression on how the members of the project chose tasks. What students did in the project was based on their personal interests.

**Interviewer:** *What lies behind where one ends up? I have understood that you have divided the tasks.*

**David:** *Well, at the start it was by interest. Partly what one might have seen as fun or what one feels one should sort of try out.*

David thought that the level of competence was approximately equal since all students had similar backgrounds. Starting from an even level, David felt that participants quickly became specialized in different areas. Discussions became less frequent after some time since fewer people have the needed level of information and

---

<sup>4</sup> Names have been changed to preserve the anonymity of those involved.

knowledge.

In David's opinion, each student had the responsibility for his own learning during the project. He said clearly that it couldn't be the teachers' responsibility. The belief that teachers did not have the responsibility for learning led to shortcomings with respect to the desired learning, since some students didn't take their responsibility for learning. Those project members had, according to David, an impact on the rest of the students in the project. Their lack of seriousness affected the learning opportunities in the project, especially for David and others who wanted to learn much.

Ownership was something that David found to vary in a similar way as the taking of responsibility for learning. Those who saw the project just as any another course took less ownership. David expressed that it felt like these didn't belong to the project in the same way.

**Interviewer:** *Are there some in the project that belong to it more than others?*

**David:** *That is my definite impression. Yeah, those that feel that they, well, knows more are hugely engaged and stay on in the room and feel that they, that they really want it to function, yes it is clear that there are such persons. There are also those that feel like this is just a course they happened to take.*

This also led to clustering of ambitious students around more important parts of the project. David had a hard time defining what parts were seen as important, apart from identifying parts that were crucial for the development of the physical deliverable as important.

## 5 Discussion

This story is based on the column in table 1 captured with the key feature *Mechanisms for work allocation* and the general aspect *motivation* is highlighted. David's descriptions and impressions on how work was allocated and the consequences provide insights from a student's point of view into most of the practices identified as important. The story blends these insights together and a dissection of the story with respect to the identified practices one by one is given below as an aid to interpret the story.

One of the practices mentioned, *cooperation among students in order to make them learn how to work with colleagues and learn how to combine different tasks into a main project*, is in focus in David's story. The desired practice *working efficiently and constructively in a large project team of developers* was in parts well implemented in the project. Both the project, in terms of the complexity of the task, and the number of developers was large. But, according to the collected experiences, the efficiency and constructiveness could have been improved. David pointed out how problematic this practice is to get experience with in a project.

The practice *getting experiences of applying previous knowledge and learning skills of tools used in the IT-community* was affected by the divide-and-conquer behavior opted for by the students. When students divide such a complex task as the one in the project, and at the same time feel stress about fulfilling the task, it tend to lead to students working within a small field. While it might be fair to assume they will be experts, or at least

more skilled, in that particular field, it will also restrict them from learning in other areas. Since much time is spent on a specific task it will become difficult to switch to other tasks. This switch is further restricted by the perceived stress. Knowing that the project consumes 20 weeks of study for each student indicate that the experience of applying previous knowledge could extend to more fields. Overall, there is a risk for conflict between deep learning in one subject and enriching knowledge in a broader sense.

Perceived stress in a project seem to lead to opting for a divide-and-conquer division of work, affecting the practice *integrating smaller tasks into a larger task*. The divide-and-conquer approach to the project tasks not only separates the tasks, but further separates the students, thus making it more difficult for them to share goals, feel that they are doing this together, and therefore sharing the experience. This problem gets worse and worse the more they divide. The experience described does not mention positive effects of this, it rather points at the limiting effect it has on discussions. Valuable discussions are lost and hence the number of peer learning situations is reduced. Especially it is interesting to note that the more tasks were divided up, the harder it became for the students to help each other. Here, the amount of people involved in the projects could be problematic since it makes it harder to get an overview of the project, and hence the opportunities for sharing knowledge are becoming fewer. In terms of legitimate peripheral participation a development towards less interaction is not fruitful since members of the project/community will not be able to share and thus become more central in the community.

Practicing *integrating smaller task into a larger one* was problematic according to David's experience. David pointed at the general complexity as being too high and there being too many people involved in order to get the overview needed to perform both a nice integration and an effective environment for peer learning. David noted time pressure leading to stress as a factor that influenced the manner in which the students used their peers in discussions. When the students had to divide the work, they also lost some of the valuable discussions that created learning opportunities.

In summary, it was clear that David felt profoundly influenced by how work was allocated in the project and also that he at times only had a fuzzy understanding of how it was done. Bringing up David's thoughts on learning, it is clear that he views it as all students have their own responsibility for learning during the project. This belief was somewhat in contradiction to what one of the course responsible teachers' said when he stated that the reason for the complexity of the task is to function as a catalyst to forces the students to collaborate and thus securing the goal of collaboration among the students.

David's comment that the responsibility for learning was the individual student and not the teachers' is interesting. The intention relative student learning with the course, and its design to meet the intention, seemed to have been ignored by David. Blaming fellow students for negative impact on learning was hence the result of David's analysis. The intention with the course, increased learning through the project, seemed to have been forgotten, which might imply that this intention was not on

top of David's head. The observation is that the design of the course didn't lead to the students getting access to the practices leading them to increased learning. While it is up to the student to engage and therefore their responsibility, the course should also be designed in a way that accounts for differences in personality, drive, etc.

Another very interesting part of the story is when David states that it is the student's own responsibility to learn in the project. The students that didn't take that responsibility placed themselves apart from the tasks that mattered in the project. Such a behavior does not help the process of legitimate peripheral participation, nor give the students enhanced learning. It is also possible that the highly motivated students dominate the important tasks and basically prevent less dominant students from being fully involved.

It is interesting to note that the way David experienced this had both negative and positive implications on how it affected the set of important practices, i.e. how it influenced his role as legitimate peripheral participant or central participant. It is for instance doubtful that he got enough training with regard to working constructively in a large project team, but had ample practice in integrating smaller tasks into a larger task. This should not be seen as criticizing the teacher, but rather as an indication of the high complexity of the goal, i.e. to aid the students in becoming members of the community of practice of IT-workers.

## 6 Conclusion

The importance of project courses and the amount of time students, and not least teachers, spend in them warrants careful consideration of how to set up and run them. The described method is a step towards addressing this issue in a scientific manner.

## Acknowledgment

We would like to thank the students who participated in the interviews. We also would like to thank the reviewers pointing out improvements to make.

## References

- Abbasi, Z. A., Backvall, P., Bergman, J., Blommé, C., Brohäll, J., Edlund, J., et al. (2009): *Project huge - final report* (Tech. Rep.). Department of Information Technology. Uppsala University, Sweden.
- Barker, L. J., & Garvin-Doxas, K. (2004): Making Visible the Behaviors that Influence Learning Environment: A Qualitative Exploration of 125 Computer Science Classrooms. *Computer Science Education*, 14, pp. 119-145.
- Barker, L. (2005) When do group projects widen the student experience gap? *Proceedings of the 10th annual SIGCSE conference on innovation and technology in computer science education*, New York, NY, USA: ACM Press, pp. 276-280.
- Berglund, A. (2005): *Learning computer systems in a distributed project course: The what, why, how and where*, ACTA Universitatis Upsaliensis, Uppsala, Sweden.
- Bruner, J. (1996): *The Culture of Education*. London, England. Harvard University Press.
- Computing Curricula, (2005): The Joint Task Force for Computing Curricula, 2005.  
[http://www.acm.org/education/curric\\_vols/CC2005-March06Final.pdf](http://www.acm.org/education/curric_vols/CC2005-March06Final.pdf).
- Daniels, M., Cajander, Å., Pears, A. and Clear, T. (2010): Engineering education research in practice: Evolving use of open ended group projects as a pedagogical strategy for developing skills in global collaboration, *International Journal of Engineering Education*, vol 26, no 4, pp 795-806.
- Kinnunen, P., & Malmi, L (2004): Do students work efficiently in a group? -problem-based learning groups in basic programming course. *Proceedings of the fourth Finnish/Baltic sea conference of computer science education*, Helsinki, Finland: Helsinki University of Technology, pp. 57-66.
- Lave, J., & Wenger, E. (1991): Situated learning: Legitimate peripheral participation (learning in doing: Social, cognitive and computational perspectives) (1st ed.). *Cambridge University Press*. Paperback.
- Lister, R. (2008): After the Gold Rush: Toward Sustainable Scholarship in Computing. In Simon & M. Hamilton (Eds.), *Proceedings of tenth Australasian computing education conference*, Vol. 78, Wollongong, Australia.
- Waite, W. M., Jackson, M. H., Diwan, A., & Leonardi, P. M. (2004): Student culture vs group work in computer science. *Proceedings of the 35th SIGCSE technical symposium on computer science education*, New York, NY, USA: ACM Press, pp. 12-16
- Wenger, E. (1998): *Communities of practice: Learning, meaning, and identity*, Cambridge University Press
- Wenger, E., McDermott, R., & Snyder, W. M. (2002): Cultivating communities of practice: A guide to managing knowledge. *Harvard Business School Press*.
- Wiggberg, M. (2008): *Unwinding processes in computer science student projects*, Uppsala University, Department of Information Technology, No. 2008-001.
- Wiggberg, M. (2010): *Computer science project courses – Contrasting students' experiences with teachers expectations*, ACTA Universitatis Upsaliensis, Uppsala, Sweden.

# A Virtual Museum of Computing History: an educational resource bringing the relationship between people and computers to life

**Geoff Berry**

Faculty of Arts  
Monash University  
Melbourne, Australia

Geoff.Berry@monash.edu

**Judy Sheard**

Faculty of Information Technology  
Monash University  
Melbourne, Australia

Judy.Sheard@monash.edu

**Marian Quartly**

Faculty of Arts  
Monash University  
Melbourne, Australia

Marian.Quartly@monash.edu

## Abstract

Teaching computing history is widely accepted as an important way of helping IT students to better understand their field, especially in its context as an integral part of modern culture. While physical museums have proven constructive in this, the Monash Museum of Computing History (MMoCH) is developing a Virtual Museum Project (VMP), bringing its physical exhibits to life with computer-generated animation. The viewer's human relationship with computers is placed in the context of current developments by way of a short journey into space exploration. The 'virtual tour' takes in the Ferranti Sirius, the PDP-9, and the hand-held HP-65, before concluding back on earth, in the very computer screen before the viewers, who are thus encouraged to recognise their relationship with computing in the dynamic (ongoing) history of IT.

**Keywords:** history of computing, computer science education, animation, computing museum.

## 1 Introduction

The idea that students of computer sciences should have some understanding of the cultural as well as technical aspects of their field has been recognized as vital to the future of academic studies in the discipline (Giangrandi and Mirolo, 2006; Lee, 1998; Medina, 2004). Physical museums of computing history go a long way towards introducing students to these societal contexts (Williams, 2003). Some museums also include online or virtual tours to complement the physical exhibition. These are usually static representations of the physical museum, which are easily accessible resources that illustrate the collections

but do not bring them to life. An especially successful example can be found at the Computer History Museum website (Computer History Museum, 2010). Others, such as the virtual museum described by Giangrandi and Mirolo (2006), employ an animated guide to help visitors negotiate their way through a virtual environment. This is meant by design to be an abstract exercise, with the idea of a museum being used as a metaphor to help students become immersed in their engagement with the materials. Another example of a virtual museum is the Virtual Museum of Computing, which is, in effect, a collection of links to information about historical computing artifacts and events (Virtual Museum of Computing, 2010).

The Virtual Museum Project (VMP) responds to the challenge of creating and implementing a virtual museum of computing history that goes beyond these approaches. It is designed to bring items from a physical exhibition to life, in a computer-generated animated sequence that is related to real world history and the immediate cultural context of the viewer. The first stage of the VMP constitutes an animation featuring three of the major items on exhibition in Monash University's Museum of Computing History (MMoCH), which is housed on Monash's Caulfield campus in Melbourne, Australia. These items are the Ferranti Sirius (1962), the PDP-9 (1969), and the hand-held HP-65 (1973). The VMP follows the MMoCH's chronological, narrative approach to the presentation of computing history, focusing on the speedy evolution of the computer, in just over a decade, from a bulky, room-sized behemoth to a hand-held device (Ainsworth, Sheard, and Avram, 2008a).

The VMP animation traces this evolution with a journey that begins with the installation of the Ferranti Sirius in the physical museum, and then leaves earth to explore space, before finally returning the viewers to their own computer screen. Thus they are invited to follow a particularly exciting period of computing history, from the early '60s to the early '70s, that brings to life not only the speed of computing evolution but the way these past developments relate to the viewers' own experiences with computers today.

The VMP animation thus highlights the way the relationship between people and computers has changed our experiences of space. Modern technological evolution has resulted in computers becoming exponentially more

Copyright © 2011, Australian Computer Society, Inc. This paper appeared at the 13th Australasian Computer Education Conference (ACE 2011), Perth, Australia. Conferences in Research and Practice in Information Technology (CRPIT), Vol. 114. J. Hamer and M. de Raadt, Eds. Reproduction for academic, not-for profit purposes permitted provided this text is included.

powerful and accessible, even as they have reduced in size; meanwhile, international communication has become commonplace, such that the world itself seems smaller. The items chosen for display in the VMP animation reveal the rapidity and extent of this evolution, while the animated narrative places today's student of computer science in its ongoing context.

The VMP design allows for future development of other dimensions of the museum experience, for example animations which link exhibits across museums. This first stage concentrates on a brief but exhilarating animated journey outside of the architecture of the physical museum. This paper outlines the development of this exciting educational resource, gives a brief description of its pedagogical use to date, and presents ideas for its future evolution.

## 2 Background and Motivation

The interface between the individual and the computing technology now available to them continues to become more accessible and immediate with every passing year. This proximity – experienced with the desktop PC, portable laptop, reading tablet and even more ubiquitous applications – raises important questions about the relationship between people and computers.

The MMoCH was established in 2001 at the Caulfield campus of Monash University in Melbourne, as a way of exploring these themes in an educational context. The physical museum is unique in that it preserves an evolutionary trajectory of computing both in the wider community and specifically in terms of Monash University's own history of information processing. In exhibiting major artifacts of the technologies involved in this history it makes a significant contribution to their conservation, while acting as an educational resource for school and university students, as well as the wider community (Ainsworth, Avram, and Sheard, 2010).

A chronological organization was chosen for the MMoCH exhibition because this helps to fill in the gaps often perceived between students' appreciation of the technological power of computing and their personal relationship with computers. The MMoCH sought to develop a narrative that would explore the rapid ascent of the 'information revolution,' such that a bridge could be formed between the original technological objects and their cultural context (Ainsworth et al., 2008a; Ainsworth, Sheard, and Avram, 2008b). This approach is supported by Giangrandi and Mirolo's (2006) recent study, which reveals that students of computer science have difficulty placing the history of the technological artifacts (about which they are curious) in a historical perspective. As these authors conclude, such educational efforts can help to change students' attitudes toward knowledge, "so that they will not miss opportunities to broaden their cultural horizons" (p.305). (Giangrandi and Mirolo, 2006)

A thematic motivation for the VMP was provided by the stated aim of the MMoCH to explore the relationship between people and computers. Computing has had a significant impact on the way we do business, run government, supply education and live our everyday lives because it has changed the way we communicate, handle, use and disseminate information. Educational resources such as the VMP can help students, amongst others, to

find perspective amongst the social upheavals and dramatic changes in employment patterns and workplace cultures that have accompanied adoptions of this technology (Katz, 1995). Such study helps promote understanding of the societal, economic, and political contexts for the development of computing technology (Katz, 1995).

With judicious choice of exhibitions and a creative narrative approach, then, the relationship between computing power and human culture can be explored. In this way, students learn from IT history as well as from its applications and new developments, as proposed by John Impagliazzo and John A N Lee, consistent advocates for the teaching of computing history (Impagliazzo and Lee, 2004; Lee, 1996a; Lee, 1996b; Lee, 1998). While the MMoCH achieves this as a physical museum, its development into the virtual world allows for a significant expansion in its explanatory and pedagogical powers.

The motivation behind the VMP animation is inspired first and foremost by the Museum's role as a teaching facility. School and university students today typically respond well to computer-animated displays, interactive graphics, and any foray into the world of virtual reality. In this they form part of an ongoing history of people for whom computers became a matter of personal, as well as institutional, use (Ensmenger, 2004). With this in mind it seemed logical that a museum dedicated to education about computing history should utilise the kinds of graphic displays preferred by its intended audience of school students and young adults.

Furthermore, the VMP is an enhancement to the physical MMoCH. The teaching of computing history in its sociocultural context gains from having the technologies discussed available for students to observe or handle. It is useful to have concrete objects to teach abstract concepts, as evidenced when the study of logarithms is aided if students can perform calculations on a slide rule or Napier's bones (Williams, 2000). An example of this approach is the use of an operational PDP-11/10 computer to reinforce the teaching of computing topics such number systems and machine languages (Harms and Berque, 2001). The VMP puts viewers in direct relation to the concepts behind IT evolution by placing the concrete use of computers in the context of the history of computing.

The VMP is designed, then, with two aims firmly in mind: to help students understand the history behind the very activity they are engaged in when viewing the animation, and to increase accessibility to the physical resource of a computing history museum, thereby making it more widely available as an educational tool. The VMP allows computer science students to broaden their knowledge of their own field in an interesting way. While the very nature of computer technology is concentrated on rapid evolution, this tendency to look forwards to the future can too easily discard lessons from the past. The VMP, in its animation, concentrates on the way computers have been used to change our world in the past, and how this leads directly to the way things are today. In this sense it fosters understanding of the rationale behind IT development, as well as its technological features (Zhang and Howland, 2005). The

animation can be used in different computing classes such as programming, interface design and computer technology to challenge the students' understanding of the development of computing and reflect upon its relationship to education, work and society. It is left without spoken narration so that lecturers can adapt their own commentary, depending on the specific purpose of their class, and students can make their own interpretation of the history displayed.

### 3 The Exhibits

While the MMoCH tracks computing history from the earliest computing devices such as the abacus, the VMP begins in more recent history, choosing three signal moments from the early '60s to early '70s and linking them in an animated narrative. Following are the three machines chosen from the MMoCH that appear in the VMP. The images shown are still shots from the animation

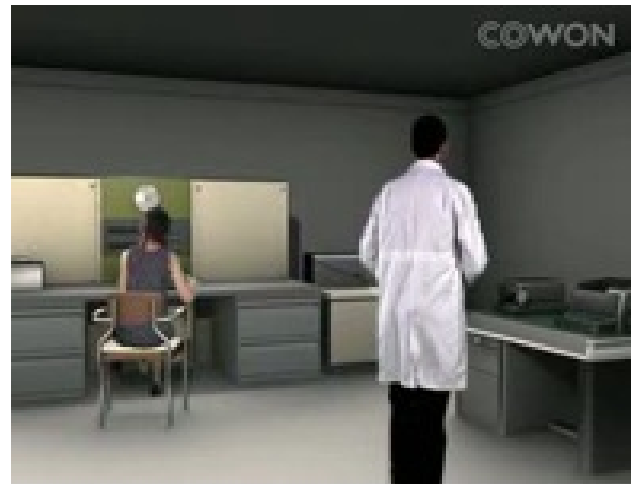
#### 3.1 The Ferranti Sirius

The Ferranti Sirius was the first computer installed at Monash University. The model currently on display at the MMoCH is the original one pulled from the crate upon its delivery in 1962 (Figure 1). This momentous occasion was marked with the kind of transport and loading equipment we expect today for a baby elephant or small power substation. Trucks that are now considered classic models, a crane such as we witness in silhouette against the city skyline, and working men directing the delicate operation in their fashions of the day all combine to give a picture of the time. These images form part of the Ferranti Sirius display in the MMoCH, as photographs on backboards and as information panels accompanying the actual computer, and are incorporated from the original materials into the VMP animation.



**Figure 1 The Ferranti Sirius computer being prepared for transportation**

The Sirius in operation certainly required some hands-on attention. Input was via paper tape which had to be hand-fed into the Creed Paper tape reader, after being punched in the patented Ferranti Westrex paper tape punch. This tape punch, along with a beautifully styled clock (now known as 'retro') and the working innards of the computer, a mass of wires and transistors looped together to carry the binary signals from input to result, are amongst the parts on display in the museum and used in the animation (Figure 2).



**Figure 2 The Ferranti Sirius computer in operation**

The technicians would arrive each morning, allowing a couple of hours to calibrate the acoustic delay line storage by hand (which certainly makes the time it takes for our desktops to boot up pale into insignificance by comparison!). The power of the Ferranti Sirius, in all its magnificent bulk, equaled around 1/100,000th of the computing power offered by the modern personal computer.

#### 3.2 The PDP-9

The second major artifact chosen for the VMP is the PDP-9, a stylish burnt orange cabinet model made by the Digital Equipment Corporation (Figure 3). The PDP-9 Programmed Data Processor (PDP) was the name of a series of minicomputers, some of them ground-breaking and very influential. The name 'PDP' intentionally avoided the use of the term 'computer' because at the time of the first PDPs, computers had a reputation of being large, complicated, and expensive machines, and the word 'minicomputer' was not invented yet.

The PDP-9 helped ensure the safe lunar landing of 1969 and the MMoCH displays the very computer featured in the Australian/American film *The Dish*, which commemorates the vital part played in this historic moment by Australian engineers.





**Figure 3 Operating a PDP-9 computer**

### 3.3 The HP-65

The HP-65 was the first computer in space. When held in an astronaut's hand in 1973, it could never have been imagined the extent to which the 'personal computer' (this was the first time such an idea had been introduced) would revolutionize everyday life across the world. With its red flashing numbers, the HP-65 packed enormous punch (for its time!) in terms of the processing it carried out (Figure 4).

Users could write programs up to 100 lines in length and record them on blank cards, or they could buy pre-programmed cards. These tiny cards had magnetic media on one side and a writable surface on the other. After passing one through the calculator, the card could be slid into a slot just above the top row of keys, where the writable side of the card served as key labels for the calculator's five program-defined keys.



**Figure 4 The HP-65 in space**

## 4 The Animated Narrative Approach

Studying the meteoric history of modern computing should not only be mandatory for today's 'net savvy youth – it should be available in the language to which they have become accustomed. To this end the VMP design team created a storyboard for its animators that would reflect the information-rich cultural world that students of today inherit. While seemingly traditional in its chronological approach, the animated narrative format explores important shifts in computing history in a way that makes it both accessible and meaningful to the student.

The three artifacts chosen to represent a short period of computing history in the VMP animation are represented from two related perspectives: the context contemporary to their use (and thus their relevance as information processing inventions in their own right), and their position in relation to the current state of technological evolution. In this way the animated narrative engages the interest and intelligence of the museum 'visitor' by providing frameworks of understanding both historical and immediate in import.

The narrative chosen as a storyboard for this animation concerns the way computing history can be considered in light of the current global phenomena of enormously increased access to computing technology. It begins with the installation of the gigantic Ferranti Sirius at Monash University in 1962, the physical dimensions of which are revealed in visuals of computer scientists feeding in and retrieving the paper tape it used to input and output data.

The animation then passes a bank of cupboard-sized information processors. These whirring machines, with their circular tape reels rotating against a backdrop of box-like compartments, represent an almost iconic image of computing days now a thing of the past. The visual journey then passes into a room where the PDP-9 features, depicted in its role in helping to facilitate the Apollo space exploration.

After lingering on this seminal piece of computing history, the viewer's attention is drawn towards a window, through which we see a scene of the night sky. We zoom into the window frame and then out into the night sky, where a satellite crosses the sky and approaches the moon. This trajectory suggests the famous Apollo moon-landings and the viewer's focus now turns to the image of an astronaut holding the HP-65 in his hand.

The astronaut then proceeds to punch some information into the hand-held HP-65 with their finger, before the 'shot' zooms again through the space vehicle's window frame towards the earth, which is seen hanging in space. Utilising NASA photographs of the planet at night, we now see a series of twinkling cities against the dark background of earth's continents (Figure 5). This image introduces the final phase of the storyboard.





**Figure 5 The earth at night as viewed from space**

This final phase of the narrative is dedicated, beyond the history of the three computer artifacts represented, to the contemporary diffusion of computing power, its increased capacities and decreased size, and people's everyday access to it. In order to link the historical exploration of space to the immediate situation of the VMP viewer in front of their screen, a storyboard was created that returns to earth while pointing out the way computer users are interconnected by a network that links people with each other via their computing technologies (Figure 6).



**Figure 6 Networked computers**

A brief note on the history of technological evolution helps explain this part of the animated narrative. One relevant aspect of technological development is the way it has expanded our physical capacities. This can be seen, for example, from the simplest tools of early agricultural settlements to the more recent microscope and telescope. In the twentieth century our vision followed explorations into space, thereby, paradoxically, drawing us closer together as we recognised the planet as a unity.

The viewer of the VMP animation is treated to a compact version of this narrative of technological evolution. After escaping the earth's bounds with a brief flight into space, the viewer is returned to earth and its glittering cities courtesy of a landing in one of those bright centres of activity in Australia. Following a trajectory down to Melbourne, the animated narrative shows how each city is in fact connected to the others by pulses of light, such that they can be visualised as clusters joined by shafts of information. When the landing is complete, and the viewer is reminded that they are looking now at their own computer terminal, the connection is completed: the miniaturisation and proliferation of computers brings all users together.

The animated narrative is open to other interpretations which can be explored in an educational context, making it a valuable teaching tool for computer science educators seeking to give their students greater comprehension of the history of their subject. For example, the animation can be used to show the changing relationship of people with computing technology in terms of accessibility, affordability and the skills needed to use the technology.

## 5 Design and Implementation

The VMP design team created the storyboard for the animated narrative, before presenting it to members of the Multimedia Department of Monash University. The animators took the artefacts chosen for the VMP and created three-dimensional visual images of them. Once drawn, these figures could be manipulated, as if from any camera angle, so that the viewer's perspective of them seems to be manoeuvred around the 'space' of the screen. Animated sequences, forming part of the narrative, were then produced using these figures. These were then combined with still images and movie segments to complete the storyline.

In consultation with the animators, it was decided that high quality graphics should take precedence over the provision of interactivity at this stage. Students expect high quality graphics as a minimum standard, so this should be the starting point for any teaching tool seeking successful engagement with this audience. These graphics can then be utilized in later developments towards interactivity. The VMP's narrative format could further be adapted to interactive games, education applications, as well as being linked to artifacts and educational aids from other museums. Endless manipulations are available as further developments of the narrative storyboard chosen for the VMP.

For now, however, the VMP concentrates on the pedagogical aim at its genesis: increasing accessibility to a tool for teaching computing history in a way that is exciting and relevant to today's students of computer science.

## 6 Educational Applications

The VMP animation has been used in computing classes at institutions in Australia and Finland. The animated narrative has been explored from a number of different aspects relevant to a broad range of computer science courses. For example:

- Interface Design – the film was used to show early computer interfaces. This was used as a lead in to a discussion about the concept of an interface and how interfaces have evolved with the changing emphasis from the computer to the human
- Introduction to Computing – the film was used to show students examples of early computing applications and demonstrate the fast evolution of the technology
- Computer Technology – the film was used to illustrate aspects of the evolution of technology, showing the dramatic decrease in size of the computers, the widening contexts in which computers could be used in as well as the different input/output media associated with it
- Programming – the film was shown to demonstrate how the tasks involved in loading and executing a program have changed

Initial student and staff responses to the animated narrative have been positive in regards to its potential to bring to life the recent history of computing in an educational context. Further evaluation of the educational potential will be conducted before the next stage of the VMP development.

## 7 Conclusions and Further Work

The main collaborators in the original, physical MMoCH project concluded that computers “are present in all aspects of life in the 21st century. It is the Museum’s role to record this technological development, particularly as it relates to the Australian context, and the impact of this development on our society.” (Ainsworth et al., 2008a) The VMP presents an innovative and highly accessible way to not only fulfil this mandate but to extend it in accord with the MMoCH’s focus on the relationship between people (especially students!) and their computing machines.

Its high quality graphic animation makes the VMP attractive to secondary and tertiary students of computer science in a way that opens up new opportunities for them to understand the history of the technical tools they access on a daily basis. It does this with reference to the physical museum of computing history from which it has evolved, showing that the aim of teaching the history of computer science can be grounded in real artifacts at the same time as it moves outside of the limits of the physical museum. The benefits of this to the field of teaching computing history to students are immediate but also lend themselves to further development.

The VMP has the potential to further evolve this style of teaching computer science and history, for instance, by expanding its focus from high quality animation to an interactive or virtual reality application. Another possible development linking the VMP to the physical MMoCH could be the provision of hand-held, on-site devices that enable interactivity and access to further information on each computing machine. This application of location-based services (LBS) has proven to be effective in the development of innovative applications such as individual tours, educational games, and even virtual tour guides for

visitors (see (Hsi et al., 2004) and (Brown et al., 2003) for examples of these).

This article forms part of the overall project for explaining the animation to teachers of computing history and can be used as a valuable aid when read alongside it. The animated narrative that is the result of the VMP is available for viewing at:

<http://www.youtube.com/watch?v=N0f0MU-vfGg>

It is also linked to the MMoCH site at:

<http://www.infotech.monash.edu.au/museum/>

## 8 Acknowledgements

We wish to thank Daniel Simmonds and Tom Chandler for their animation work on this project and Barbara Ainsworth and Chris Avram from the Monash Museum of Computing History team.

## References

- Ainsworth, A. B., Avram, C. and Sheard, J. (2010) The Monash University Museum of Computing History: Ten Years On. In A. Tatnall (Ed.), *History of Computing. Learning from the Past* (Vol. 325, pp. 216-227): Springer Boston.
- Ainsworth, A. B., Sheard, J. and Avram, C. (2008a) The Monash Museum of Computing History: Part 1. *ACM SIGCSE Bulletin inroads* 40(2), 31-34.
- Ainsworth, A. B., Sheard, J. and Avram, C. (2008b) The Monash Museum of Computing History: Part 2. *ACM SIGCSE Bulletin inroads* 40(4), 31-34.
- Brown, B., MacColl, I., Chalmers, M., Galani, A., Randell, C. and Steed, A. (2003) *Lessons from the lighthouse: Collaboration in a shared mixed reality system*. In proceedings of the Conference on Human Factors in Computing Systems, Ft. Lauderdale, Florida, USA.
- Computer History Museum, California, USA. [www.computerhistorymuseum.org](http://www.computerhistorymuseum.org) [accessed 14 August 2010]
- Ensmenger, N. (2004) Power to the people: Toward a social history of computing. *IEEE Annals of the History of Computing*.
- Giangrandi, P. and Mirolo, C. (2006) "Numeri e Macchine" - A virtual museum to learn the history of computing. In proceedings of the ITiCSE 2006, Bologna, Italy.
- Harms, D. and Berque, D. (2001) *Using a PDP-11/10 to teach content and history in courses*. In proceedings of the ACM SIGCSE 2001, Charlotte, North Carolina, USA.
- Hsi, S., Semper, R., Brunette, W., Rea, A. and Borriello, G. (2004) "eXspot: A Wireless RFID Transceiver for Recording and Extending Museum Visits". In proceedings of the Proceedings of UbiComp.
- Impagliazzo, J. and Lee, J. A. N. (2004) Using computing history to enhance teaching. In J. Impagliazzo & J. A. N. Lee (Eds.), *History of Computing and Education* (pp. 165-175). Boston: Kluwer Academic Publishers.

- Katz, K. (1995) The present state of historical content in computer science texts: A concern. *ACM SIGCSE Bulletin* 27(4), 43-50.
- Lee, J. A. N. (1996a) History in the computer science curriculum. *ACM SIGCSE Bulletin* 28(2), 15-20.
- Lee, J. A. N. (1996b) "Those who forget the lessons of history are doomed to repeat it" or, why I study the history of computing. *IEEE Annals of the History of Computing* 18(2), 54-61.
- Lee, J. A. N. (1998) History in the computer science curriculum: Part II. *ACM SIGCSE Bulletin* 30(2), 11-13.
- Medina, E. M. (2004) Beyond the ballot box: Computer science education and social responsibility. *ACM SIGCSE Bulletin inroads* 36(4), 7-10.
- The Virtual Museum of Computing (VMoC) [accessed 14 August 2010]
- Williams, M. R. (2000) Do we teach computer science as religion or as history? *ACM SIGCSE Bulletin inroads* 32(4), 4-5.
- Williams, M. R. (2003) The computer history museum. *ACM SIGCSE Bulletin inroads* 35(4), 12-13.
- Zhang, C. and Howland, J. E. (2005) Brief and yet bountiful: The history of computing, why students need you? *JCSC* 20(4), 308-314.



# Students' Perceptions of Peer Evaluation in Project Work

J. K. L. Poon

Hong Kong Community College, The Hong Kong Polytechnic University,  
Hung Hom Bay, Kowloon, Hong Kong, PR China

ccjandia@hkcc-polyu.edu.hk

## Abstract

To capitalise on the benefits of group work and at the same time alleviate the potential negative effect of benefiting from collective effort without contributing equally (also referred to as free-riding) and internal conflict, this project was conducted for the following purposes: design an approach to peer assessment of individual contributions to group work, evaluate student perceptions on the proposed approach, and refine the approach. Under the proposed peer evaluation approach, the final project grade of an individual student is the sum of two components: the group project mark awarded by the lecturer and the contribution of the individual student to the team as assessed by other team members. Fifty-eight full-time computing students taking the module 'Project Work' at a post-secondary institution in Hong Kong were involved in this study. Data were collected from a questionnaire survey and a focus group interview. Findings revealed that participants strongly agreed that the proposed peer evaluation method was fair and could motivate them to work harder. This project provides a tested and workable peer evaluation approach for group projects at the post-secondary level, as well as maximises the learning values that group projects can bring to students.

**Keywords:** peer assessment, group project work, higher education.

## 1 Introduction

Group project work is a prevailing student-centred learning approach in information and communication technology (ICT) education (Bushell, 2006; Clark et al., 2005; Clear, 2007; Herbert, 2007; Joyce, 2002; Richards, 2009). Given that a project requires collaborative investigation and the production of a series of project artefacts, it can provide opportunities to include tasks and activities more directly related to the business world and professional practice, to apply conceptual skills and theoretical knowledge, to experience and learn about group dynamics, to familiarise with different perspectives and problem-solving approaches, to develop and extend interpersonal and social skills such as collaboration and networking, and to increase student motivation and engagement (e.g., Chen et al., 2009; Riordan et al., 2008; Sobral, 1997; Topping, 2009; Tseng and Tsai, 2007).

Although the potential of group project work as an instructional tool is rarely disputed, its use often brings

about problems that limit and even negate its possible benefits. Specifically, the difficulties associated with accurately and fairly assessing individual performance, conflict within groups, and free-riding of individual members are frequently cited as group project-related problems (e.g., Barfield, 2003; Bower and Richards, 2006; Bushell, 2006; Dyrud, 2001; Herbert 2007; Joyce, 2002; Richards, 2009; Tenenber, 2007; Topping, 2009).

The implementation of peer evaluation in assessing project work, as an alternative approach for entirely instructor-assessed, has gradually received attention from diverse disciplines in the higher education process (Fellenz, 2006; Herbert, 2007; Sung et al., 2005; Topping, 2003). Peer evaluation should be understood as an educational arrangement in which students comment on the quality of the work of their fellow students, for formative or summative purposes (Topping, 2003). It is considered an unbiased assessment approach because group members have been working together for a period of time and they themselves 'would seem in a natural position to provide reliable, valid evaluations of each other' (Cederblom & Lounsbury, 1980, p.568). However, student responses to using peer evaluation in project work are varied. Studies revealed that the reported reactions on peer evaluation range from being viewed as 'fair and equitable' and 'qualified endorsements' to 'traditional peer assessment is relatively ineffective in addressing free-rider problems' (Cheng and Warren, 1997; Conway et al., 1993; Falchikov and Goldfinch, 2000; Goldfish and Raeside, 1990; Herbert, 2007; Lejk and Wyvil, 2001; Sadler and Good, 2006).

Despite the potential advantages of peer evaluation, its success is not unconditional. The peer evaluation approach must be designed, organised, and implemented systematically and adequately in order to be accepted by students (Tseng & Tsai, 2007). The literature does not provide adequate data that can help explain this variety in reactions to specific or similar peer evaluation approach, particularly in Hong Kong sub-degree level. To capitalise on the stipulated benefits of group work and to ensure equal marks among students, we plan to implement peer evaluation in a computing subject entitled 'project work'. Thus, the present study aims to (i) design a fair peer evaluation method that can maximise student learning from a group project, (ii) explore students' perceptions on the proposed peer evaluation method, and (iii) refine the proposed peer evaluation method to be implemented in the following academic year.

## 2 Design of the Proposed Peer Evaluation Method

Peer evaluation has been used by educators for a considerable amount of time and for many different purposes. Most of these purposes are served using peer

evaluation on learning outcomes such as reports, presentations, or classroom contributions. However, the focus of this paper is on the use of peer evaluation of individual contributions to group work that concentrates on complex group processes that lecturers cannot easily observe and assess.

There are many different schemes for assessing individual contribution to group work. Goldfinch and Raeside (1990) adopted a two-part approach in group awards. The first part determines the performance of the tasks of the group members, whereas the second part measures the contribution to group dynamics. In the study by Gatfield (1999), the peer evaluation component constitutes 50 percent of the final grade of the students. The 50/50 split is considered a safe margin to provide an incentive for students to take the issue seriously and a safeguard against some students in the group wanting to sabotage an individual (Clark, 2005). The outcomes for implementing peer evaluation were acceptable, and the occurrence of dysfunctional groups became less. Subsequently, Lijk and Wyvill (2001) conducted a study and compared the results derived from using two different approaches to assess peer contributions: holistic and category-based approach. The holistic approach was concluded to produce a wider spread of grades than the category-based approach and therefore was considered a better relative indication of individual contribution to the project. The study also stated that the holistic approach supports the purposes of summative peer evaluation method, whereas the category-based approach is useful for formative assessment.

## 2.1 Approach to Determine Individual Final Mark

The proposed approach for determining the final mark of a student in the group project was inspired by the works of Goldfinch and Raeside (1990) and Gatfield (1999). The final marks of individual members within the same group are varied and depend on two elements: group project and individual contribution to the group. The group project is worth 50 percent of the total marks and is assessed by the lecturer according to the submitted group report. Members within the group should receive equal marks. Individual contribution is also worth 50 percent of the total marks and is evaluated by other members within the same group. Students who contribute more or less will receive more or less marks in the half portion of the group marks. This method is equivalent to a 'zero-sum' game where an increase in marks for one student must be accompanied by a decrease in marks for another. This fixed-pie evaluation system precludes the possibility of grade inflation. The 50/50 split has been used and evidenced a desirable ratio in the study of Gatfield (1999) and Clark (2005) respectively.

## 2.2 Design of the Proposed Peer Evaluation Form

The proposed peer evaluation form (Table 1) was designed to collect feedback on the contribution and performance of other members of the group throughout the group project process. It consists of two parts, A and B. Part A is used to grade peer performance on the areas of choosing a topic, planning, designing, testing, documentation, report write

up, and reliability in meetings and completing tasks. The ratings ranging from A (excellent in both performance and contribution) to F (poor performance and no contribution) provide formative evaluation on the performance of students in each categorised area and do not contribute to the student's final grade. Descriptions of each rating are also provided for ease and consistency in student marking. These dimensions are consistent with the assessment criteria because each submitted project will be evaluated according to the following project factors:

- Overall design resulting in specifications from which the solution can be developed;
- Design and developing of solution;
- Documentation, including user guide; and
- Design and implementation of all appropriate testing

Part B is used to measure the overall contribution of an individual member to the project work. Each student is required to allocate a single percentage to all other group members according to their overall contribution to the group work. The holistic approach supports the purpose of summative peer evaluation method and is a better indication of individual contribution (Lijk & Wyvill, 2001). After collecting all the completed peer evaluation forms, the lecturer consolidates the comments in Part A for the compilation of the feedback report on the individual students. The percentages in Part B are used for the calculation of individual marks according to the approach discussed in Section 2.1. Subsequently, individual students will receive a report that contains the final project grade and anonymous descriptive feedbacks provided by their group members.

## 3 Research method

A cross-sectional field study was conducted. The unit of analysis was individual. As one of the research objectives is to evaluate student perceptions and experiences of using the proposed peer evaluation method, both quantitative and qualitative approaches were used.

### 3.1 Participants

Fifty-eight students enrolled in the module 'Project Work' participated in this study. 'Project Work' is a core subject for the programme 'Foundation Diploma – Computing Stream' offered by a post-secondary institution in Hong Kong. The aim of this subject is to provide students with practical experience in the application of the subjects studied, such as Programming Concepts, Database Development, Internet Applications, and Information Technology Applications. The subject lasts for one semester (13 weeks) and involves group collaboration in relation to a 'realistic' IT project. Students are required to form groups of five to six members and produce a written report. Assessment of the project is based on its design, testing, implementation, documentation, and overall presentation.

The participants formed 10 project groups. They had prior experiences in conducting group projects from either other subjects or secondary schools, but none of them had taken part in peer evaluation prior to this study.

PEER ASSESSMENT FORM						
Project Title						
Names of Group Members						
For each member of your group, but <b>excluding yourself</b> , award grades and percentage of contribution by completing Parts A & B respectively.						
<b>PART A:</b> Feedback on group members' performance (Give a grade between A-F)						
Tasks to be assessed		Group Members' Full Names				
1. Choosing the topic of the IT project						
2. Design and development of the IT solution						
3. Programming, testing and debugging						
4. Documentation including the user guide						
5. Reliability in completing tasks						
6. Reliability in attending meetings						
Grades:	A. Excellent - exceptionally active and constructive, <b>excellent</b> ideas and organization skills B. Good - active and constructive C. Fair – cooperative, attended and participated in most meetings D. Marginal - sometimes absent and did not participated E. Unsatisfactory - did not contribute or was unreliable F. Poor - rarely attended meetings, was disruptive for team work					
<b>PART B:</b> Allocation of percentage of contribution to all your group members.						
Group members' full names (exclude yourself)						Total %
Please allocate the % of contribution (should add up to 100%)						100%
Student's Name (in full) _____ Signature _____						
<u>Remarks:</u>						
1. The grades and percentage of contribution that you award will not be disclosed to any of the group members.						
2. The lecturer holds the final right to override the group's decision if unfair discrimination is experienced. In this case, equal marks are awarded						

Table 1: Proposed Peer Evaluation Form

### 3.2 Instruments

The instruments used in this study were a questionnaire and a focus group interview. The main quantitative method was an end of semester survey comprised of nine statements. As the purpose of the questionnaire is to explore the perceptions of the proposed peer evaluation approach, the questionnaire items were derived from the issues raised in previous studies on peer evaluation. The first six questionnaire items were related to the most reported issues such as the appropriateness, fairness, and acceptance of using peer evaluation and were linked to a five-Likert scale. The range descriptors were from

1.	Students should take part in assessing their group members' contribution to a group project.
2.	I am able to assess fairly my group members' contribution to our group project.
3.	It is fair to take the peer evaluation marks into account when allocating marks to each student for the project.
4.	I have understood the project evaluation procedures.
5.	The peer evaluation form is appropriate for assessing the group members' contribution to the project.
6.	Peer evaluation should be used earlier in this subject.
7.	Do you think the 50% allocated to peer evaluation is the right amount? <input type="checkbox"/> Yes. <input type="checkbox"/> No. Please specify, _____ %
8.	What would you like/dislike about this proposed peer evaluation approach? Like: _____ Dislike: _____
9.	Any further comments/suggestions?

Table 2: Questionnaire Statements

'strongly agree' to 'strongly disagree'. The last three survey statements were open-ended questions provided with space for writing down the reason for the answer. The survey questions are summarized in Table 2.

The focus group interview was employed to collect qualitative data because although survey data can identify a problem, they generally fail to probe its causes. Conducting an interview after the survey aimed to clarify the questions asked in the questionnaire and to explore further the opinions, views, and suggestions of the students on the peer evaluation process. Focus group was used instead of individual interview because feedback from multiple students could be obtained in a shorter time and the exchange among interviewees raises ideas and concerns that might not emerge from individual interviews.

### 3.3 Administration of Data Collection

At the end of the semester when students submitted their group projects, they were requested to participate in the survey in the classroom. At the beginning of the session, students were provided with the peer evaluation form (Table 1). As the students had no prior knowledge and experience on peer evaluation, the lecturer had to explain the content of the evaluation form in detail, particularly the objectives and method of peer evaluation and the nature of the survey. Students were asked to complete the peer evaluation form in confidence, giving them an opportunity to practise and experience the peer evaluation process. Students were then asked to complete a questionnaire (Table 2), which sought their opinions on the proposed peer evaluation method they just experienced. The completion of the questionnaire was voluntary and done anonymously, and the students were assured that whatever answers and comments they provided would not affect their project marks because the survey is to collect students' views on the proposed peer evaluation approach that to be implemented in the next academic year. All fifty-eight students completed and returned the questionnaires. The entire process took about an hour.

A week later, a student from each project group was selected randomly and invited to participate in a focus group interview. The rationale for the group-mix is that

the participants coming from different teams are unlikely in a position of control over each other. The focus group interview was voluntary, with a total of eight students participating. To provide a quiet, relaxed, and comfortable discussion environment, the interview was conducted in the meeting room, with snacks and drinks provided. The entire process took about one-and-a-half hours and was recorded.

To guarantee that the students would not feel threatened about the consequence of identification, the purpose of the study was explained to them before the interview with the assurance that their feedback was for planning purposes only and would not affect their project marks.

Since the interview was conducted in Cantonese (the native language of the students), the interviewees' responses were first transcribed in Chinese. The statements were then grouped according to the survey questions. For reporting purpose, only the verbatim quotes which best exemplified the essence of the survey questions were chosen and translated into English.

## 4 Results

The analysis is divided into two parts. The first part considers the perceptions of the students on the proposed peer evaluation method and addresses study objective 2, whereas the second part is comprised of the analysis of the suggestions for improvement of the proposed method from the students and is related to study objective 3.

### 4.1 Students' Perceptions of the Proposed Assessment Method

The responses of the students to the first six statements in terms of the five-point scale ranging from 'strongly agree' to 'strongly disagree' are summarised in Table 3.

strongly disagree are summarised in Table 5:					
Questions	SA	A	N	D	SD
Q1: Students should take part in assessing their group members' contribution.	39%	39%	17%	-	5%
	78%			5%	
Q2: I am able to assess fairly my group members' contribution.	22%	39%	23%	11%	5%
	61%			16%	
Q3: It is fair to take the peer evaluation marks into account when allocating marks to each student.	22%	34%	22%	11%	11%
	56%			22%	
Q4: I have understood the project assessment procedure.	33%	44%	18%	-	5%
	77%			5%	
Q5: The assessment form is appropriate for assessment.	17%	56%	17%	5%	5%
	73%			10%	
Q6: Peer evaluation should be used earlier.	39%	34%	17%	5%	5%
	73%			10%	
SA=strongly agree; A=agree; N=netural; D=disagree; SD=strongly disagree					

**Table 3: Students' responses to questions 1-6**

#### 4.1.1 Question 1: Should students assess their group members

When asked whether students should take part in assessing the contributions of their group members, the majority of the students (78 percent) agreed or strongly agreed that they should, whereas 17 percent did not have an opinion on this question. Only 5 percent disagreed.

All the students who took part in the focus group interview confirmed that the effort or the contribution of the members should be assessed as they knew the process of work and were thus the best judges of the activities of the group. Some interviewees added that students' assessing the contribution of their peers to the group work was important because it encouraged the members to share workload and work harder.

'...I believe students will work harder if they know other members can assess them... we students are realistic and will focus on doing things that can get higher marks...'

'I support peer evaluation because I, as a leader, am unable to push members to do work... at the end, I do most of the work ... I believe that with peer evaluation, all group members will cooperate and work harder... the project work will be of higher quality'.

However, an interviewee expressed concern that peer evaluation would damage the friendship. In view of this, he tended to give other members more or less the same percentage.

'... although the percentages allocated by peers were kept confidential, the team mate still can work out the mark you give him/her, say, by asking other members... I fear losing my friends, so I will give the same marks to my friends...'

#### 4.1.2 Questions 2, 3, and 6: Fairness and confidence of student assessors

The results for question 2 indicated that 61 percent of the students felt that they were able to assess the contribution of their members fairly; 23 percent did not have a firm opinion on their ability; and 16 percent felt that they were unable to do it fairly.

Students gave similar answers to question 3: 56 percent of the respondents supported the idea of including peer evaluation marks into final project mark of the students, whereas only 22 percent opposed the idea.

When asked whether peer evaluation should be introduced earlier in this subject (Question 6), 73 percent of students said yes, 17 percent had no opinion, and 10 percent were against the idea.

During the focus group interview, most of the interviewees supported the view that peer evaluation marks should be included in the final mark, and they were confident that they themselves could assess their peers and their own contribution fairly.

'...If peer evaluation marks are included in the final mark, students who work harder will get higher marks, and those who contribute less will get lower mark. It is fairer...I don't mind contributing more...'



‘...as I was completing the form, I focused on recalling my peers’ contribution to the project, what their inputs and outputs are... I didn’t think about who were my good friends... I think I can assess fairly...’

‘... I can assess peers fairly, and I believe my peers can do the same because I trust them...’

However, an interviewee raised the concern that the maturity level, friendship, and trust between the members could affect the reliability of the grading.

‘I intend to rank fairly, but I am an ordinary student. I would give a little higher mark to my closed friends...’

‘If I anticipate some members will rank me lower, I will rank them lower too’.

#### **4.1.3 Questions 4 and 5: Appropriate format and clarity of the procedure**

When asked on the procedure of the peer evaluation (question 4), the replies showed that 77 percent of the students found the explanation clear, and only 5 percent considered that the procedure was not explained clearly.

During the interview, all students confirmed that they clearly understood the assessment method and process, and no further explanation was required.

A total of 73 percent of the respondents felt that the peer evaluation form (Table 1) was appropriate for assessing the performance in and contribution to the project of the member (question 5); 17 percent had no opinion, and only 10 percent considered the form inappropriate.

During the interview, the students expressed that the grading criteria in part A were helpful in standardising the assessment. Part A was also a valuable means of providing formative feedback to peers as students assess each other on a number of specific tasks.

#### **4.1.4 Question 7: Is 50 percent the correct value?**

About half of the respondents felt that 50 percent was the correct value for the peer evaluation; 33 percent considered it to be too much; and the remaining 17 percent felt that it was too low.

In the interview, the students in general felt that the weighting of peer evaluation between 30 percent and 60 percent was acceptable.

‘I think 50 percent is the correct value allocated to peer evaluation’.

‘The 50/50 evaluation is too risky because I don’t know whether my group members can judge fairly. Thirty to forty percent is pretty safe, and I am happy with it...’

‘Peer evaluation should contribute 60 percent of the total mark because students who contribute more will get higher marks’.

#### **4.1.5 Question 8: Reasons for liking/disliking peer evaluation**

Of the 58 students who responded to the survey, 89 percent replied to this question. ‘Peer evaluation is a fair

evaluation tool for students’ contribution’ was their only one reason for liking it. Based on the interview, the other reasons for liking it were ‘students’ contribution to the project is reflected in the final mark’ and ‘it provides useful feedback for improvement’.

Students were also asked what they did not like about the process. None of the respondents replied to this question. In the interview, the students stated that the reasons for disliking the process had been mentioned previously, that is, the perceived subjective nature and the potential of the process to alienate friends.

### **4.2 Suggestions for Improvement**

No additional comments to question 9 were supplied by the respondents. In the interview, a student suggested that peer evaluation should be implemented twice because the formative feedback facilitated better performance in the students during the latter part of the project.

‘The peer evaluation of the project work should be conducted two times: in the middle and at the end. Those who do not contribute enough in the beginning can still work harder in the latter part in order to get a higher mark’.

## **5 Discussions and Conclusion**

The survey results showed that majority of the students expressed a positive agreement to the statements that the students should assess their peers, that the students were able to assess fairly, that it was fair to divide marks, that the students understood the process, that the assessment form was appropriate, that 50 percent is the correct value for peer evaluation, and that peer evaluation should be used earlier.

The focus group interview confirmed and reinforced the questionnaire answers and gave better insight into the opinions of the students on the proposed peer evaluation method. The interviewees felt that it was fairer and important to include them in the assessment of contribution because the group members knew what was happening during the work. They also believed that through peer evaluation, some students would be more willing to work harder because they would be aware that their efforts would affect their own mark. They were confident that they could assess the contributions of the group members fairly and objectively but doubted the objectivity of others, especially in relation to the assessment of friends.

The objectivity of the students can be improved if the lecturer can stay involved in the process and encourage students to participate positively. Initial instructions followed by continuous monitoring of the situation are required to uncover and fix problems as soon as possible (Topping, 2003).

Regarding the allocation of mark, the interviewees generally felt that the peer-assessed weighting of 30 percent to 60 percent was acceptable. The result is consistent with the study of Clark (2005) that the split of an individual assessment worth 40 percent and a team assessment worth 60 percent has the majority support. Orsmond et al. (2000) consider involving students in deciding the weighting of peer evaluation prior to the commencement of the project work would be a possible

option because perception is a necessary condition for promoting student learning. The idea is good but it may be difficult in getting consensus from student groups. Moreover, a too high weighting would call the integrity of the assessment procedure into question, whereas a too low weighting would provide little incentive effect for the student to participate actively in the project work.

The reasons given for disliking the process were its perceived subjective nature and the potential of the process to alienate friends. This issue may be overcome by requesting each student to write personal report detailing his/her contribution to the group project. Each student also has to indicate agreement or disagreement with the reports written by each of their team members. If a significant disagreement is found, individual contribution reports would be discussed at the team meetings with the lecturer. This process is regarded as a fair way of indicating student's contribution and has been used by many scholars (e.g. Herbert, 2007; Gates et al., 2000; Hayes et al., 2003). However, the possible drawback is students may complain on the over-loading of paper work.

With regard to the students' suggestions for improvement, the idea of conducting peer evaluation twice is useful and will be included when modifying the peer evaluation approach. It is because the subject 'Project Work' is a full-semester subject, the interim feedback can be based on a meaning amount of work (for example, about 6 weeks) and could be received early enough to allow other group members to consider and implement behaviour changes in response to peer feedback (Tseng & Tsai, 2007). In fact, similar approach has been adopted by Clear (2010). He advocates the implementation of two summative assessment points in his capstone project, one at the project proposal stage when the project is confirmed, two at the final presentation and portfolio submission stage. This early intervention strategy is considered effective in risk mitigation.

In summary, the available data on student reactions allow an initial assessment of students' perceptions of the proposed peer evaluation approach. The findings indicate that the proposed peer evaluation approach is effective in achieving its explicit goal of fairness in assessment, and subsequently enhance the value of group project. On the other hand, students value the learning they have gained from experiencing in a transparent and well structured peer evaluation process. Therefore, the peer evaluation approach as proposed will be implemented to the computing students in the next academic year.

The limitations of this study are it fails to discuss the cultural factors which may have inhibited peer judgement. The collation and summarizing of student feedbacks create additional heavy workload to the lecturer. The development of a database driven web-based peer assessment system will result in providing timely feedback to students and enabling the lecturer to manage the assessment of larger and more diverse student cohorts. These issues should be addressed in future study.

## 6 Acknowledgement

I am grateful to the four anonymous referees for their constructive comments on the earlier version. I would like to acknowledge the Hong Kong Community College to partly sponsor this project. This project was supported in

part by the College of Professional and Continuing Education (CPCE) research fund under the grant number EZ02.

## 7 References

- Barfield, R. L. (2003). Students' perceptions of and satisfaction with group grades and the group experience in the college classroom. *Assessment Evaluation Higher Education*, 28(4), 356-369.
- Bower, M. & Richards, D. (2006). Collaborative learning: some possibilities and limitations for students and teachers. In *Proceedings of the Conference for the Australasian Society for Computers in Learning in Tertiary Education (ASCILITE 06)*, 79-89.
- Bushel, G. (2006). Moderation of peer assessment in group projects. *Assessment Evaluation Higher Education*, 31(1), 91-108.
- Cederblom, D., Lounsbury, J.W. (1980). An investigation of user acceptance of peer evaluations. *Personnel Psychology*, 33, 567-579.
- Chen, N.S., Wei, C.W., Wu, K.T., & Uden, L. (2009). Effects of high level prompts and peer assessment on online learners' reflection levels. *Computers & Education*, 52(2), 283-291.
- Cheng, W., & Warren, M. (1997). Perceptions before and after a peer assessment exercise. *Studies in Higher Education*, 22(2), 233-239.
- Clark, N. (2005). Evaluating student teams developing unique industry projects. *Proceedings of the 7<sup>th</sup> Australasian Computing Education Conference (ACE05)*, 21-31.
- Clark, N., Davies, P. & Skeers, R. (2005). Self and peer assessment in software engineering projects. *Proceedings of the 7<sup>th</sup> Australasian Computing Education Conference (ACE05)*, 91-100.
- Clear, T. (2007). Computing capstone projects and the role of failure in education. *SIGCSE Bull.*, 39-42.
- Conway, R., Kember, D., Sivan, A., & Wu, M. (1993). Peer assessment of an individual's contribution to a group project. *Assessment and Evaluation in Higher Education*, 18(1), 45-56.
- Dyrud, M.A. (2001). Group projects and peer review. *Business Communication Quarterly*, 64, 106-112.
- Falchikov, N. & Goldfinch, J. (2000). Student peer assessment in higher education: A meta-analysis comparing peer and teacher marks. *Review of Educational Research*, 70, 287-322.
- Fellenz, M. (2006). Towards fairness in assessing student groupwork: A protocol for peer evaluation of individual contributions. *Journal of Management Education*, 30(4), 570-591.
- Gates A.Q., Delgado, N. & Mondragon, O. (2000). A structured approach for managing a practical software engineering course. In *Proceedings of the ASEE/IEEE Frontiers in Education Conference*, 21-26.
- Gatfield, T. (1999). Examining student satisfaction with group projects and peer assessment. *Assessment and Evaluation in Higher Education*, 24(4), 365-377.

- Goldfish, J., & Raeside, R. (1990). Development of a peer assessment technique for obtaining individual marks on a group project. *Assessment and Evaluation in Higher Education*, 15(3), 210-225.
- Hayes, J.H., Lethbridge, T.C. and Port, D. (2003). Evaluating individual contribution toward group software engineering projects. In *Proceedings of International Conference on Software Engineering*, 622-627.
- Herbert, N. (2007). Quantitative peer assessment: Can students be objective? In *Proceedings of the 9<sup>th</sup> Australasian Computing Education Conference (ACE 07)*, 63-71.
- Joyce, D. (2002). Group work at postgraduate level: some issues. In *Proceedings of the 7<sup>th</sup> Annual conference on Innovation and Technology in Computer Science Education*, 220-220.
- Lejk, M. & Wyvill, M. (2001). Peer assessment of contributions to a group project: A comparison of holistic and category-based approaches. *Assessment and Evaluation in Higher Education*, 26(1), 19-39.
- Orsmond, P., Merry, S., & Reiling, K. (2000). The use of student derived marking criteria in peer and self-assessment. *Assessment & Evaluation in Higher Education*, 25(1), 23-38.
- Richards, D. (2009). Designing project-based courses with a focus on group formation and assessment. *ACM Transactions on Computing Education*, 9(1), 2:1-2:40
- Riordan M.P., Riordan, D.A., & St. Pierre, E.K. (2008). Groupthink in accounting education. In A.H. Catanach Jr and D. Feldmann (Eds.). *Advances in Accounting Education: Teaching and Curriculum Innovations*. V.9, 189-204. Emerald Group Publishing Limited.
- Sadler, P.M. & Good, E. (2006). The impact of self- and peer-grading on student learning. *Educational Assessment*, 11, 1-31.
- Sorbral, D.T. (1997). Improving learning skills: A self-help approach. *Higher Education*, 33(1), 39-50.
- Sung, Y.T., Chang, K.E., Chiou, S.K. & Hou H.T. (2005). The design and application of a web-based self- and peer-assessment system, *Computers & Education*, 45(2), 187-202.
- Tenenberg, J. (2008). An institutional analysis of software teams. *International Journal Human-Computer Studies*, 66, 484-494.
- Topping, K.J. (2003). Self and peer assessment in school and university: reliability, validity and utility. In M.S.R. Segers, F.J.R.C. Dochy, & E.C. Cascallar (Eds.), *Optimizing new modes of assessment: In search of qualities and standards*. 55-87. Dordrecht: Kluwer Academic.
- Topping, K.J. (2009). Peer Assessment. *Theory into Practice*, 48, 20-27.
- Tseng, S.C. & Tsai, C.C. (2007). Online paper assessment and the role of the peer feedback: A study of high school computer course. *Computers and Education*, 49, 1161-1174.



# Early Relational Reasoning and the Novice Programmer: Swapping as the “*Hello World*” of Relational Reasoning

**Malcolm Corney**

Faculty of Science and Technology  
Queensland University of  
Technology,  
Brisbane, QLD, Australia

m.corney@qut.edu.au

**Raymond Lister**

Faculty of Engineering and  
Information Technology,  
University of Technology, Sydney,  
Sydney, NSW, Australia

Raymond.Lister@uts.edu.au

**Donna Teague**

Faculty of Science and Technology  
Queensland University of  
Technology,  
Brisbane, QLD, Australia

d.teague@qut.edu.au

## Abstract

We report on a longitudinal research study of the development of novice programmers in their first semester of programming. In the third week, almost half of our sample of students could not answer an explain-in-plain-English question, for code consisting of just three assignment statements, which swapped the values in two variables. We regard code that swaps the values of two variables as the simplest case of where a programming student can manifest a SOLO relational response. Our results demonstrate that the problems many students face with understanding code can begin very early, on relatively trivial code. However, using traditional programming exercises, these problems often go undetected until late in the semester. New approaches are required to detect and fix these problems earlier.

**Keywords:** Novice programmer, SOLO, chunking.

## 1 Introduction

Over the last six years, the BRACElet project has studied the relationship between the ability of novice programmers to write code and explain code. Two of the earliest BRACElet papers (Whalley *et al.*, 2006; Lister *et al.*, 2006) studied how students answered the following explain-in-plain-English question in an end-of-first-semester programming exam:

In plain English, explain what the following segment of Java code does:

```
bool bValid = true;

for (int i = 0 ; i < iMAX-1; i++)
{
    if (iNumbers[i] > iNumbers[i+1])
        bValid = false;
}
```

The BRACElet researchers analysed student responses to this question in terms of the SOLO taxonomy (Biggs and Collis, 1982). Some students of course provided

responses that were inadequate, vague or simply incorrect, but there were also correct and comprehensive responses from students that fell into two SOLO categories:

- **Multistructural:** This is a response in which the student provides a description of what each line of the code does, without linking the lines together.
- **Relational:** This is a response in which the student provides a correct summary of the overall computation performed by the entire piece of code, such as, for the above code “*it checks to see if the array is sorted*”. We refer to the ability to read a piece of code and deduce the overall computation performed by that code as *relational reasoning*.

Since those first two BRACElet papers, replication studies have tried several variations on the format of explain-in-plain-English questions. Lister and Edwards (2010) provided a summary of those variations. From all those studies, it appears that there are some students who are able to provide multistructural responses, but who struggle to perform relational reasoning.

In another BRACElet study, Lopez *et al.* (2008) linked relational reasoning with code writing. They found that a combination of student scores on tracing tasks and the ability to manifest relational reasoning on explain-in-plain-English questions accounted for 46% of the variance on a code writing task. Replication studies have reported similar results (Lister, Fidge and Teague, 2009; Venables, Tan and Lister, 2009; and Lister *et al.*, 2010).

All of the above BRACElet studies used data collected as part of end-of-first-semester exams. Also, most of those studies used explain-in-plain-English questions where the code involved iterative processes on arrays.

### 1.1 Relational Reasoning without Iteration

The motivation for our study came from a colleague, who had taught first-semester programming classes for many years, and who had won teaching awards while doing so. Our colleague made the assertion that the first few weeks of teaching programming are straightforward, but the problems start with the introduction of loops.

That comment by our colleague led us to wonder – does relational reasoning only become a problem for novices when loops are introduced? We then looked at examples of code that textbooks presented to students prior to the introduction of loops. (All of the code-writing problems we examined were in the procedural paradigm.) We found that one common type of example presented to

---

Copyright © 2011, Australian Computer Society, Inc. This paper appeared at the 13th Australasian Computer Education Conference (ACE 2011), Perth, Australia. Conferences in Research and Practice in Information Technology (CRPIT), Vol. 114. J. Hamer and M. de Raadt, Eds. Reproduction for academic, not-for profit purposes permitted provided this text is included.

students could be characterized as *input-calculate-output*. For example, consider a piece of code that calculates the area of a rectangle, given the length and height. Such a piece of code has the following general form:

```
Input a value into a variable;
Input a value into a second variable;
Assign to a third variable the
    result of the calculation;
Output the third variable;
```

Such code has properties that make it easier to understand than the iterative code studied later in semester:

- All the variables are either directly manipulable by the user (i.e. input) or directly observable by the user (i.e. output). Thus all variables have a meaning defined by the “real world” problem to be solved, without reference to any algorithm.
- Given the “real world” definition of each variable, the purpose of each line of code makes sense in isolation from the other lines.
- None of the lines of code involve overwriting a meaningful variable value with a new value. Such a change to a variable would change the exact “real world” problem being solved.

Could it be, then, that the first few weeks of semester, prior to the introduction of loops, are straightforward because the non-iterative code we traditionally present to our students only requires a form of reasoning that is simpler than the relational reasoning required for understanding iterative code? If so, could we identify non-iterative pieces of code that might better prepare students for loops?

The above questions led us to identify the simplest piece of non-iterative code that requires the same form of relational reasoning as iterative code – the swapping of the values between two variables:

```
temp = a;
a = b;
b = temp;
```

Unlike the code for calculating the area of a rectangle:

- the variable `temp` is not an input/output variable but only has meaning in the context of an algorithm;
- each line does not stand alone. For example, the final line does more than simply assign the value of `temp` into `b` – it assigns *the original value of a* into `b`; and
- the meaningful values in variables `a` and `b` are overwritten.

Since some programming textbooks use a “Hello World” program as their very first example, we refer to code that swaps the values of two variables as the “Hello World” of relational reasoning. We then asked the following research question, which we pursue in this paper:

*Would some students struggle in the early part of the semester with the code for swapping the value of two variables, just as earlier BRACElet research had demonstrated that students struggled with iterative code at the end of semester?*

If there are students who struggle with the code for swapping two variables, then the early part of semester does not prepare students well for the iterative code to follow.

Prior to our study, there was an earlier BRACElet study that used an explain-in-plain-English question for swapping two variables (Sheard *et al.*, 2008). However that study, like other BRACElet studies, examined students at the end of the first semester.

## 2 The Learning Context

The students from whom data was collected for this study were enrolled in an introductory computing course. This subject was a breadth-first introduction to building IT systems and was not devoted entirely to the teaching of programming. Other material taught during this subject included introductions to SQL and web-page development. The first six weeks of the 13 week semester were allocated to an introduction to programming, using Python. In those six weeks, the students were expected to reach a point where they could understand and write code involving array/list structures, conditional statements, loops, function definition and use, and recursion. In the remaining seven weeks of the semester, students had further practice with their Python skills, when they used Python in the web-based systems they wrote (e.g. to interact with SQL databases and to perform input/output processing for web pages).

This paper is restricted to an analysis of the performance of the students on the programming component of this subject. For more details on the entire subject, see Corney, Teague and Thomas (2010).

## 3 Method

Students attending the lectures in weeks 3 and 5 were given two short written tests. These two tests are provided in this paper, as the last two pages.

Normal exam conditions applied during these tests. The lecture room was supervised by the first and third authors to ensure that students completed the exams individually. There was no strict time limit on either test. Students were given as long as they needed. After 10 to 15 minutes all students had finished. Most had completed the test well before that time.

Prior to both tests, the students were told that the tests would not contribute to their grade. The students were told that the teachers would use the test results as a guide to what topics required more teaching or improved methods of teaching. Of the test sheets returned by the students for marking, a very small number were entirely blank, and a few were completed but left anonymous. The anonymous tests were excluded from the analysis described below, since we could not collate a student’s performance across tests without knowing their identity.

Following each test, the first author, who taught the subject, talked about each of the questions. He demonstrated an approach to solving the questions, and he provided correct answers to the questions.

### 3.1 The Week 3 Test

This first test was administered in week 3 of the semester, at which time the material being presented in lectures assumed that students could understand the basic

concepts of variables and assignment statements. The test was distributed to students on both sides of a single sheet of paper. We provide this test on the second last page of this research paper, reduced in size to conserve space. A total of 227 students submitted this test.

### 3.1.1 Questions 1 to 3: Screening Questions

The three goals of these screening questions was to test that a student (1) understood variables, (2) understood assignment, and (3) could trace code of similar complexity to the remaining questions in that test. Since this test was administered very early in the semester, we could not assume that students had these skills, particularly as some students enrolled late.

In this paper, we are not interested in what percentage of the class understood variables and assignment statements. Asking these three screening questions is analogous to, in a non-programming research study, first giving an experimental subject a test on their ability to read English. It is sometimes a wise precaution to know that someone can read English before giving that person a test on the real material of interest, when the test on the real material happens to be written in English – it is the experimental subject’s grasp of the real material that we would want to measure, not the experimental subject’s ability to read English. More specifically, our research interest in this paper involves testing whether a novice programmer understands a piece of code as a whole, when the novice understands all the programming constructs in that code. We therefore need to screen to ensure that students understand those constructs. Students who could not successfully answer all three of these questions were eliminated from the analysis we present later in the “Results” section.

A fourth goal in having the screening questions was to ensure that we had a sample of students who had made a genuine effort to answer the questions in the test. Students who scored a perfect score on these screening questions clearly approached the test seriously.

### 3.1.2 Questions 4 and 5: Explain a Swap

Both question 4 and question 5 are explain-in-plain-English questions, and the code in both questions swaps the values of two variables, using a third variable as a temporary variable.

At week 3 of the semester, students had not encountered an explain-in-plain-English question before, so there was a danger that the students might not understand the type of answer we wanted. For that reason, we designed Question 4 so that it would show the students what sort of answer we wanted, in three ways:

- Question 4 begins by providing an example of the type of answer we wanted, *“The purpose of the following three lines of code is to swap the values in variables a and b”*.
- Question 4 specifies that the answer should be *“one sentence”*. Furthermore, the box in which the students are directed to write their answer is meant to indicate that the answer should not be very long.
- Question 4 contains the note *“Tell us what the second set of three lines of code do all by themselves. Do NOT think of those second three lines as being executed after the first three lines of code.”* We

added this note after a pilot test, at a different university. In the pilot, we found that a small number of students gave answers such as *“It restores the variables to their original values”* because the students thought of the second set of three lines as being executed after the first set of three lines.

The aim of Question 5 was to see whether students could generalise from Question 4, and see that Question 5 also swapped values between two variables. We can report that 91% of the 227 students who answered both Question 4 and Question 5 were consistent across those questions – either they answered both of these questions correctly and relationally, or they answered both questions either incorrectly or non-relationally. The remaining 9% were split roughly even, among some who answered Question 4 correctly and relationally, and some who answered Question 5 correctly and relationally. Given that 91% of students answered both questions consistently, we subsequently focussed our analysis on Question 4 and ignored Question 5.

## 3.2 The Week 5 Test

This second test was administered in week 5 of the semester, at which time the material being presented in lectures assumed (in addition to the concepts tested in week 3) that students could understand `if` statements. Like the week 3 test, this test was distributed to students on both sides of a single sheet of paper. We provide this test on the last page of this research paper, reduced in size to conserve space. A total of 176 students submitted this test, of whom 148 had also completed the week 3 test.

### 3.2.1 Question 1: Write a Swap

The first question required students to write code that swaps the values between two variables. Recall that Question 4 in the week 3 test had asked students to explain a piece of code that also swapped two variables. The research interest in asking students to write the swap code was to see whether students, two weeks after the first test, could remember the swap code well enough to write it.

#### 3.2.1.1 Two Temporary Variables

Of the 140 students who answered this question, 8 students (6%) made appropriate use of two temporary variables, instead of the minimum necessary single temporary variable. (Recall that the swap code in the week 3 test used a single temporary variable, so these students had clearly not memorised that code.) When a student’s code with two temporary variables worked correctly, it was marked as correct, since such an answer was not excluded by the phrasing of the question. Another 9% of students also used two temporary variables, but they did so incorrectly.

### 3.2.2 Question 2: Screening Questions

The second question performed a similar screening role in the week 5 test as the first three questions of the week 3 test. That is, the goal of Question 2 was to test that a student could trace code containing `if` statements, which also implied that the student had made an effort on the test. As for the week 3 test, students who did not

successfully answer this question were eliminated from the analysis we present later in the “Results” section.

### 3.2.3 Question 3: Explain a sort of three variables

The third question is an explain-in-plain-English question, in which the code contains `if` statements. In the framing of this question, we took steps similar to those steps we took in the framing of Question 4 in the week 3 test, to ensure that students were clear on what type of answer was required.

## 3.3 End of Semester Exam

At the end of the 13 weeks of semester, the students underwent an examination of the material from the entire semester. In this research paper, we shall focus on three programming-related questions from that exam, which are described below.

### 3.3.1 MCQ: Explain Product of Even Numbers

Of the eight programming-related multiple choice questions in the final exam, only one was an explain-in-plain-English question:

Which best describes the purpose of the following Python function definition?

```
def do_something_with_numbers():
    total = 1
    response = input('Please input an integer: ')
    while (response != 0):
        if response % 2 == 0:
            total = total * response
        response = input('Please input an integer: ')
    return total
```

- (a) It does not do anything as the body of the while loop never executes
- (b) It returns the product of all numbers entered
- (c) It returns the product of all even numbers entered ← *the correct option*
- (d) It returns the product of all odd numbers entered

### 3.3.2 Trace a Swap

One of the programming-related exam questions tested the students again on code that swapped the values in two variables:

What do the variables `value_1`, `value_2` and `value_3` hold after the following Python code is executed? Assume that they are all integer type variables.

```
value_1 = 10
value_2 = 15
value_3 = value_1
value_1 = value_2
value_2 = value_3
```

Students were deemed to have supplied a correct answer if they provided the correct values for all three variables.

### 3.3.3 Write the Reverse of a String

Only one exam question required students to write any Python code:

The following Python source code copies a String:

```
source = 'the cat sat on the mat'
target = ''
for character in source:
    target = target + character
```

Rewrite this code snippet so that the target String contains the source String in reverse order. e.g. 'abc' becomes 'cba'.

A concise, correct answer needed only to be a copy of the above code, with the final line changed to:

```
target = character + target
```

Some students provided a more verbose but correct answer, such as the following:

```
n = 0
target = ''
for character in source:
    target = target + source[len(source) - n - 1]
    n += 1
```

In this paper, we are interested in the students’ conceptual grasp of programming, and not their ability to get code exactly right, first time, under exam conditions. Therefore, we ignored minor bugs. For example, some students provided a similar solution to the more verbose solution provided above, but they made errors in the calculation of the subscript into the sequence `source`. For example, instead of the correct `(len(source) - n - 1)` as in the above solution, some students wrote `(len(source) - n)`. We ignored such errors. We also ignored off-by-one errors in loops.

The most common answer attracting zero marks used code similar to the concise answer given above, but replaced the plus sign in `target = target + character` with a minus sign: `target = target - character`. Since the subtraction operator does not exist for strings, those students were either manifesting a conceptual error, or were making a guess.

## 4 Results

As described earlier, we culled all students who did not correctly answer all four screening questions (i.e. the first three questions in the week 3 test, and the second question from the week 5 test). We also culled all students who had not provided some form of answer to all of the remaining questions, except Question 5 from the week 3 test, since that question was left out of our data analysis. (The reasons for leaving it out were discussed in section 3.1.2). After this culling, 83 students remained. The percentage of these students who answered each test and exam question correctly is shown in Table 1.



Week 3	Week 5		End of Semester Exam		
Explain a swap	write a swap	explain a sort of 3 variables	MCQ, explain the product of even nums	trace a swap	write the reverse of a string
47%	73%	48%	76%	89%	59%

Table 1: The percentage of students who answered each question correctly (n=83)

Week 3 explain a swap	Week 5		End of Semester Exam		
	Column A	Column B	Column C	Column D	Column E
	write a swap	explain a sort of three variables	MCQ, explain the product of even nums	trace a swap	write the reverse of a string
Wrong (n = 44)	57%	36%	64%	82%	41%
Right (n = 39)	92%	62%	90%	97%	79%
$\chi^2$ test	p = 0.001	p = 0.03	p = 0.01	p = 0.03	p = 0.001

Table 2: The performance of students, broken down according to the week 3 explanation question (n=83).

Week 5 write a swap	Week 5	End of Semester Exam		
	Column A	Column B	Column C	Column D
	explain a sort of three variables	MCQ, explain the product of even nums	trace a swap	write the reverse of a string
Wrong (n = 22)	14%	59%	68%	27%
Right (n = 61)	61%	82%	97%	70%
$\chi^2$ test	p = 0.001	p = 0.03	p = 0.001	p = 0.001

Table 3: The performance of students, broken down according to the week 5 writing question (n=83).

#### 4.1 Results for Week 3 Explain a Swap

Table 2 shows the percentage of students who correctly answered questions from the week 5 test and the end of semester exam. These percentages are broken into two rows, according to how students answered the explanation of a swap in Question 4 of the Week 3 test. The row that commences with the word “Right” shows the percentages for the 39 students who correctly answered Week 3 Question 4, while the row commencing “Wrong” shows the percentages for the 44 students who answered incorrectly.

As the bottom row of Table 2 shows, chi square analysis of the raw numbers used to produce the percentages in Table 2 show a statistically significant difference (at the traditional  $p=0.05$  criterion) between the percentages within the “Right” and “Wrong” rows of each column. That is, there is a statistically significant difference in the performance of students on the week 5 test questions, and also on the end of semester exam questions, depending upon whether the students answered Week 3 Question 4 correctly or incorrectly.

It is remarkable that performance on a simple explanation question in week 3 results in a consistent, statistically significant difference in performance in other tasks for the remainder of the semester – problems with relational reasoning start early and persist.

The difference in performance on the week 5 “write a swap” task (i.e. Column A, 57% for wrong vs. 92% for right) is consistent with much of the literature in cognitive psychology. A student who can explain

swapping at week 3 remembers that code as a meaningful “chunk”. A student who cannot explain that code struggles to remember three separate lines of code.

#### 4.2 Results for Week 5 Write a Swap

Table 3 shows the percentage of students who correctly answered questions from the week 5 test and the end of semester exam. These percentages are broken into two rows, according to whether students correctly answered Question 1 of the Week 5 test, “write a swap”. As with Table 2, a chi square analysis showed a statistically significant difference between the “Right” and “Wrong” percentages of each column.

Again, it is remarkable that performance on a simple writing task in week 5 results in a consistent, statistically significant difference in performance on each of the exam questions, especially the dramatically differing performance on writing code to reverse a string (Column D, 27% for wrong vs. 70% for right).

Column A in Table 3 adds to the evidence from BRACElet studies that code writing and code explanation are closely linked cognitive skills. The students who could write the swap code at week 5 (i.e. the row beginning “Right”) performed much better on the other week 5 task, where they had to explain some code (Column A, 14% for Wrong vs. 61% for Right). This result again demonstrates that the ability to “chunk” code into meaningful pieces is important in both writing code and explaining code.

Week 5 explain a sort of three variables	Week 5	End of Semester Exam		
	Column A	Column B	Column C	Column D
	write a swap	MCQ, explain product of even nums	trace a swap	write the reverse of a string
Wrong (n = 43)	56%	65%	84%	44%
Right (n = 40)	93%	88%	95%	75%
$\chi^2$ test	p = 0.001	p = 0.02	p = 0.1	p = 0.01

**Table 4: The performance of students, broken down according to the week 5 explanation question (n=83).**

Prior Programming Experience?	Week 3	Week 5		End of Semester Exam		
	Column A	Column B	Column C	Column D	Column E	Column F
	explain a swap	write a swap	explain a sort of 3 variables	explain product of even nums	trace a swap	write reverse of a string
No (n = 27)	52%	70%	37%	70%	78%	56%
Some (n = 21)	43%	67%	57%	81%	90%	57%
Yes (n = 11)	27%	73%	45%	64%	100%	55%
From Table 1 (n = 83)	47%	73%	48%	76%	89%	59%

**Table 5: The percentage of students who answered each question correctly, based on their prior background in programming (n=59, as 24 of the 83 students did not respond to the survey)**

### 4.3 Results for Week 5 Explain a Sort

Table 4 shows the percentage of students who correctly answered questions from the week 5 test and the end of semester exam. These percentages are broken into two rows, according to whether students correctly answered the Week 5, Question 3 explanation question. As with Tables 2 and 3, a chi square analysis showed a statistically significant difference, at the traditional  $p=0.05$  level, between the percentages within each column, except for “trace a swap” (column C).

### 4.4 Results for End of Semester Trace a Swap

The results in the “Right” and “Wrong” rows of both Table 2 column D and Table 3 column C (both columns for “trace a swap”) show a statistically significant difference. For example, Table 3 column C shows that 97% of students who could write a swap in week 5 could successfully trace the swap code at the end of semester, compared to only 68% of students who could not write the swap in week 5.

Given that all n=83 students in this study had passed a screening test where they successfully answered four tracing problems, why should a tracing problem in the final exam present a problem? Our explanation to that question is as follows. Tracing is an error prone activity. The students who were able to explain the swapping code in week 3, or who could write the swapping code in week 5, were more likely to recognize similar code in the final exam. Consequently, those students might have been able to determine the answer to this question in the final exam without having to trace the code, or at least they could have verified their trace by comparing the result to what they thought it should be. However, the other students

(i.e. those who were not able explain the swapping code in week 3, or who could not write the swapping code in week 5) would have been less likely to recognize that the code was swapping the values of two variables. Such students had no alternative but to derive the answer by tracing the code, and they had no means of checking their answer, other than by tracing the code again.

### 4.5 Prior Knowledge

To assess whether prior programming experience may have been a factor in the above results, we analysed the responses to a survey that the students completed at the beginning of the semester. The survey contained the following questions:

- Have you ever written a computer program before? (Yes, No)
- If you answered “Yes” to the above question, in which language or languages have you written computer programs? (Free form answer)
- With respect to programming, attempt to explain what a variable is. (Free form answer)
- With respect to programming, attempt to explain what a function or a method or a procedure is. (Free form answer)
- With respect to programming, attempt to explain what a parameter or argument is. (Free form answer)

On the basis of the answers to the above survey questions, one of the authors classified all the students into 1 of 3 categories:

- “No” – the student indicated they had not programmed before and did not know what variables, methods and parameters were.

- “Some” – either the student indicated they had not programmed before but gave good answers regarding variables, methods and parameters OR the student indicated they had programmed before but could not answer all other questions; usually the parameter question was the problem.
- “Yes” – the student indicated they had programmed and gave good answers for the other questions.

Table 5 describes the percentage of students who answered the test and exam questions correctly, broken down according to the above three categories of prior programming experience. Chi square analysis of the raw numbers used to produce each column of Table 5 showed no statistically significant differences (at the traditional  $p=0.05$  criterion) between the percentages shown within each of those columns. We therefore conclude that prior programming experience is not a confounding factor in the results we have reported.

## 5 Discussion: To Read, Write and Understand

### 5.1 Statistics and Causation

We wish to stress that we are not claiming that the ability to write code is dependent upon the ability to explain code. To do so would be to make a well-known fallacy of statistical reasoning commonly stated as “*correlation does not imply causation*”. To use a frivolous example sometimes used in introductory statistics lectures, there may be a statistical relationship between ice cream sales and deaths from drowning, but that is because both are linked by hot weather. More formally, two statistical variables may be related because both variables depend upon a third variable.

A possible third variable that links code writing and code explaining is the ability to understand and/or reason about code. Research on the psychology of programming has demonstrated that, as expertise develops, a programmer’s knowledge is organized into more abstract, flexible forms, which would benefit both code writing and code explaining (Adelson, 1984; Corritore & Wiedenbeck, 1991; Fix, Wiedenbeck & Scholtz, 1993; Mayer, 1981; Shneiderman & Mayer, 1979; Soloway, 1986).

### 5.2 Pedagogical Implications

If understanding and/or reasoning about code is the third variable upon which both code writing and code explaining depend, then the crucial pedagogical question is as follows:

*How can we most efficiently develop our students’ capacity to understand and/or reason about code?*

#### 5.2.1 Learning by Code Writing

Is writing code the most efficient way to learn how to understand and/or reason about code? Clearly, students must write some code, but current pedagogical practise emphasises code writing to such an extent that almost all the active learning exercises we give our students (i.e. laboratory exercises and assignments) require our students to write code. Is fighting the compiler the most time efficient way of improving student understanding of code? Perhaps the most efficient way is a judicious mix

of having students write code and having them read code (and testing their ability to read via tasks such as explain-in-plain-English).

#### 5.2.2 Roles of Variables

If lecturers are to teach relational reasoning explicitly, and if lecturers are going to set and grade students on exercises where the students must read and understand code, then we need a vocabulary for relational reasoning.

One promising vocabulary is “*roles of variables*” (Ben-Ari & Sajaniemi, 2004; Kuittinen & Sajaniemi, 2004; Sajaniemi, 2010). These are a dozen categories for the purpose of a variable in a piece of code. Three of these roles are:

- **Stepper:** is defined as being “*a data entity stepping through a succession of values that can be predicted as soon as the succession starts*”. This role is illustrated by the for-loop control variable “*i*” in the explain-in-plain English question on the first page of this paper.
- **One-way flag:** is defined as being “*a two-valued data entity that cannot get its initial value once its value has been changed*”. This role is illustrated by the variable “*bValid*” in the explain-in-plain English question on the first page of this paper.
- **Temporary:** is defined as being “*a data entity holding some value for a very short time only*”. This role is illustrated by the variable “*temp*” in the code on the second page of this paper, which is code for swapping the values of two variables.

Lecturers could teach these roles, and explain code in terms of these roles. Students could be graded on exercises where they identify the roles of variables in a piece of code, perhaps as part of an explain-in-plain-English question. Our intuition is that a student who can identify the roles of all the variables in a piece of code is close to explaining what the code does (but that is a conjecture that would make for interesting future work).

## 6 Conclusion

Understanding three assignment statements, that swap the values in two variables, is not rocket science. Neither is writing that same code. However, we have shown that, in week 3 of semester, half of the students in our sample have a problem with understanding such a simple piece of code, and two weeks later one half of those students cannot write that same code. Furthermore, as a group, these students who could not answer those questions in weeks 3 and 5 performed relatively worse on programming tasks in the final exam. Thus, from the very early stages of the semester, the students begin to separate into two groups. The students in one group tend to think relationally about code, of their accord. The students in the other group do not tend to think relationally about code. Early detection and treatment of those students in the second group may improve failure rates.

We are not advocating that thinking relationally is an innate skill. Instead, we believe that current pedagogical practice does not help novice programmers learn to think relationally. Today, learning to think relationally about code is an implicit part of the curriculum of

programming. Some of our current students succeed in teaching themselves that implicit part of the curriculum, but many do not. We need to develop pedagogical techniques that transform this implicit component of the curriculum into an explicit part of the curriculum.

Finally, we urge the reader to either use our two in-class tests, or design their own tests that are more to their liking, and collect data from their own class. Not only might the results illuminate the reader's thinking about their own teaching, but replications of our study will determine whether the statistical relationships we have found are widespread, or are the result of some relatively unusual aspect of our teaching environment.

## Acknowledgements

Raymond Lister's participation in this work was partially funded by an Associate Fellowship awarded by the Australian Learning and Teaching Council.

## References

- Adelson, B. (1984) When novices surpass experts: The difficulty of a task may increase with expertise. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, **10**(3), 483-495.
- Ben-Ari, M. and Jorma Sajaniemi, J. (2004) *Roles of variables as seen by CS educators*. 9th Annual SIGCSE Conference on Innovation and Technology in Computer Science Education (ITiCSE). pp. 52-56. <http://doi.acm.org/10.1145/1007996.1008013>
- Biggs, J. B. and Collis, K. F. (1982): *Evaluating the quality of learning: The SOLO taxonomy (Structure of the Observed Learning Outcome)*. New York: Academic Press.
- Corney, M., Teague, D., Thomas, R. (2010) *Engaging Students in Programming*. Twelfth Australasian Computing Education Conference (ACE 2010), Brisbane, Australia, January 2010. CRPIT, **103**. Tony Clear and John Hamer, Eds., ACS. pp. 63-72. <http://crpit.com/confpapers/CRPITV103Corney.pdf>
- Corritore, C. & Wiedenbeck, S. (1991) What Do Novices Learn During Program Comprehension? *Int. J. of Human-Computer Interaction*, **3**(2), 199-222.
- Fix, V., Wiedenbeck, S., and Scholtz, J. (1993) *Mental representations of programs by novices and experts*. In INTERACT '93 and CHI '93 Conferences on Human Factors in Computing Systems. pp. 74-79. <http://doi.acm.org/10.1145/169059.169088>
- Kuittinen, M. and Sajaniemi, J. (2004) *Teaching Roles of Variables in Elementary Programming Courses*. 9th Annual SIGCSE Conference on Innovation and Technology in Computer Science Education (ITiCSE). pp. 57-61.
- Lister, R. And Edwards, J. (2010) *Teaching Novice Computer Programmers: bringing the scholarly approach to Australia – A report on the BRACElet project*. Australian Learning and Teaching Council, Sydney, Australia. ISBN: 1-921856-02-5. Downloadable from <http://www.altc.edu.au/altc-associate-fellow-raymond-lister>
- Lister, R., Simon, B., Thompson, E., Whalley, J. L., and Prasad, C. (2006) *Not seeing the forest for the trees: novice programmers and the SOLO taxonomy*. 11th Annual SIGCSE Conference on Innovation and Technology in Computer Science Education (ITiCSE), 118-122. <http://doi.acm.org/10.1145/1140123.1140157>
- Lister, R., Fidge C. and Teague, D. (2009) *Further evidence of a relationship between explaining, tracing and writing skills in introductory programming*. 14th Annual SIGCSE Conference on Innovation and Technology in Computer Science Education (ITiCSE), pp 161-165. <http://doi.acm.org/10.1145/1595496.1562930>
- Lister, R., Clear, T., Simon, Bouvier, D. J., Carter, P., Eckerdal, A., Jacková, J., Lopez, M., McCartney, R., Robbins, P., Seppälä, O., and Thompson, E. (2010). Naturally occurring data as research instrument: analyzing examination responses to study the novice programmer. *SIGCSE Bull.* **41**, 4 (Jan.), pp. 156-173. DOI=<http://doi.acm.org/10.1145/1709424.1709460>
- Lopez, M., Whalley, J., Robbins, P., and Lister, R. 2008. *Relationships between reading, tracing and writing skills in introductory programming*. Fourth International Workshop on Computing Education Research (ICER), Sydney, Australia. pp. 101-112. <http://doi.acm.org/10.1145/1404520.1404531>
- Mayer, R. (1981) The Psychology of How Novices Learn Computer Programming. *ACM Computing Surveys*, **13**, 1 (March), pp. 121-141. <http://doi.acm.org/10.1145/356835.356841>
- Sajaniemi, J. (2010) Roles of variables. [http://cs.joensuu.fi/~saja/var\\_roles/](http://cs.joensuu.fi/~saja/var_roles/) [Accessed Nov 2010.]
- Sheard, J., Carbone, A., Lister, R., Simon, B., Thompson, E., Whalley, J. (2008) *Going SOLO to Assess Novice Programmers*. 13th Annual SIGCSE Conference on Innovation and Technology in Computer Science Education (ITiCSE), Madrid, Spain. pp. 209-213. <http://doi.acm.org/10.1145/1384271.1384328>.
- Shneiderman, B. & Mayer, R., (1979) Syntactic/semantic interactions in programmer behavior: A model and experimental results. *International Journal of Computer and Information*, **8**(3), pp. 219-238.
- Soloway, E. (1986) Learning to program = learning to construct mechanisms and explanations. *CACM*, **29**, 9 (Sep), 850-858. <http://doi.acm.org/10.1145/6592.6594>
- Venables, A., Tan, G., and Lister, R. (2009). *A closer look at tracing, explaining and code writing skills in the novice programmer*. Fifth International Workshop on Computing Education Research (ICER), Berkeley, USA. pp. 117-128. <http://doi.acm.org/10.1145/1584322.1584336>
- Whalley, J., Lister, R., Thompson, E., Clear, T., Robbins, P., Kumar, P.K.A. and Prasad, C. (2006). *An Australasian Study of Reading and Comprehension Skills in Novice Programmers, using the Bloom and SOLO Taxonomies*. Eighth Australasian Computing Education Conference (ACE2006), Hobart, Australia. CRPIT, **52**, pp. 243-252. ACM International Conference Proceeding Series, Vol. 165. <http://crpit.com/confpapers/CRPITV52Whalley.pdf>

## INB104 Test 1, [Sem 1, 2010 Week 3], page 1

Student's Name \_\_\_\_\_ Student's Number \_\_\_\_\_

*For all questions in this test, you may write down any working out on this test paper, except in the answer boxes. Write ONLY your answer in the answer boxes.*

- Q1. In the boxes provided below, write the values in the variables after the following code has been executed:

```
r = 2
s = 4
r = s
```

The value in `r` is  and the value in `s` is

- Q2. In the boxes provided below, write the values in the variables after the following code has been executed:

```
p = 1
q = 8
p = q
q = p
```

The value in `p` is  and the value in `q` is

- Q3. In the boxes provided below, write the values in the variables after the following code has been executed:

```
x = 5
y = 3
z = 7
x = z
y = x
z = y
```

The value in `x` is  the value in `y` is   
and the value in `z` is

*The rest of the test is on the other side of this piece of paper ...*

## INB104 Test 1, [Sem 1, 2010 Week 3], page 2

Student's Name \_\_\_\_\_ Student's Number \_\_\_\_\_

*For all questions in this test, you may write down any working out on this test paper, except in the answer boxes. Write ONLY your answer in the answer boxes.*

***This is page 2 of the test. The rest of the test is on the other side of this piece of paper.***

- Q4. The purpose of the following three lines of code is to swap the values in variables `a` and `b`:

```
c = a
a = b
b = c
```

The three lines of code below are the same as the lines above, but in a different order:

```
a = b
b = c
c = a
```

In one sentence that you should write in the box below, describe the purpose of those second set of three lines. **NOTE:** Tell us what the second set of three lines of code do all by themselves. Do NOT think of those second three lines as being executed after the first three lines of code.

- Q5. In one sentence that you should write in the box below, describe the purpose of the following three lines of code for any set of values stored in variables `i`, `j` and `k`:

```
j = i
i = k
k = j
```

\*\*\* End of Test \*\*\*

**INB104 Test 2, [Sem 1, 2010 Week 5], page 1**

Student's Name \_\_\_\_\_ Student's Number \_\_\_\_\_

*For all questions in this test, you may write down any working out on this test paper, except in the answer boxes. Write **ONLY** your answer in the answer boxes.*

- Q1. Suppose you have two integer variables, called *p* and *q*. **In the box below write code to swap the values in those two variables.** You may declare and use any extra variables required to make the swap. Give each extra variable a meaningful name that reflects its purpose.

- Q2. This question is about the following code, where the variables *p*, *q*, *r* and *s* all have integer values:

```
if (p < q):
    if (q > 4):
        s = 5
    else:
        s = 6
```

Assume that, **before** the above code is executed, the values in the four variables are:

*p* = 1      *q* = 2      *r* = 3      *s* = 4

After the codes is executed, the value in variable *s* is

**INB104 Test 2, [Sem 1, 2010 Week 5], page 2**

Student's Name \_\_\_\_\_ Student's Number \_\_\_\_\_

*For all questions in this test, you may write down any working out on this test paper, except in the answer boxes. Write **ONLY** your answer in the answer boxes.*

- Q3. If you were asked to describe the purpose of the code below, a good answer would be “*It prints the smaller of the two values stored in the variables *a* and *b**”.

```
if (a < b):
    print a
else:
    print b
```

**In one sentence that you should write in the empty box below, describe the purpose of the following code.**

Do **NOT** give a line-by-line description of what the code does. Instead, tell us the purpose of the code, like the purpose given for the code in the above example (i.e. “*It prints the smaller of the two values stored in the variables *a* and *b**”).

Assume that the variables *y1*, *y2* and *y3* are all variables with integer values.

In each of the three boxes that contain sentences beginning with “Code to swap the values ...”, assume that appropriate code is provided instead of the box – do **NOT** write that code.

```
if (y1 < y2):
```

Code to swap the values in *y1* and *y2* goes here.

```
if (y2 < y3):
```

Code to swap the values in *y2* and *y3* goes here.

```
if (y1 < y2):
```

Code to swap the values in *y1* and *y2* goes here.

```
print y1
print y2
print y3
```

\*\*\* End of Test \*\*\*

# My Students Don't Learn the Way I Do

**Michael de Raadt**

University of Southern Queensland  
deraadt@usq.edu.au

**Simon**

University of Newcastle  
simon@newcastle.edu.au

## Abstract

An understanding of how students learn can facilitate the design and creation of better teaching materials. Instruments that attempt to measure learning styles are many and varied, and their use can be controversial. This paper reports on a study of the learning styles of students in an introductory programming course, and a comparison with the learning style of their instructor. We conclude that it is not necessary to cover all possible learning styles, so long as students' learning preferences *include* the modalities presented in materials.

**Keywords:** learning styles, VARK, computing education, introductory programming

## 1 Introduction

It would be ideal if the materials created by an instructor were suitable for all students in a course. An introductory programming course will typically include students from a variety of disciplines, who will presumably not all have the same preferred manner of learning. It is difficult to know a great deal about students in such a course, and consequently how to cater for them.

This paper begins by describing some identified learning styles, the inventories designed to measure them, and some past research involving learning styles in the computing education domain. An exploratory study of introductory programming students' learning styles is described, with the broad goals of comparing the learning preferences of students and instructor, and of determining what learning preferences should be catered to by the course materials. Results of the study are then reported and discussed.

### 1.1 Learning Styles

It is generally agreed that students learn in different ways. However, there is little consensus on the best way to measure students' *learning styles*. There are many learning style inventories and questionnaires, some of which are strongly validated while others are as unreliable as a personality quiz in a glossy magazine. The inclusion of learning styles assessment within educational practice has been criticised by some psychologists due to the lack of evidence to justify the validity of the instruments used (Pashler et al., 2009). Yet this criticism is not universally warranted, as some learning style inventories have been heavily tested. Some claim reliable

internal consistency, which is one measure of validity. Some learning style inventories are based on established psychological theories. An excellent review of popular learning style inventories, with measurements and comparisons, is presented by the Learning and Skills Research Centre of the UK (Coffield et al., 2004).

One obvious difference between inventories is the choice of dimensions that they attempt to measure. Jackson's Learning Styles Profiler proposes four learning types linked to personality: initiator, reasoner, analyst and implementer (Jackson, 2002). The Herrmann Brain Dominance Instrument looks at use of parts of the brain to classify learners (or, more correctly, thinkers) as theorists, organisers, innovators or humanitarians (Herrmann, 1989). The validity of placing learners 'in a box' by categorising them is accepted by some researchers and rejected by others, but is not the focus of this study.

Some inventories, such as the Myers-Briggs Type Indicator (Myers & McCaulley, 1985), assume that learning style is stable, a relatively unchanging part of a learner. Others, for example Kolb's Learning Style Inventory (Kolb, 1999), assume that learning style is a changing aspect of the learner, influenced by the learning environment. Again, this distinction is not being questioned here.

Some learning style inventories attempt to collect together other inventories. Some do so to provide a sensible combination, while others seem to try to cover as many dimensions of a learner as possible. The Index of Learning Styles, formulated by Felder and Silverman, combines some of the work of Kolb and Myers-Briggs to categorise learners as active/reflective, sensing/intuitive, visual/verbal, and sequential/global (Felder, 1996).

Related to learning style is learning approach. While learning styles describe how learners prefer to receive instruction, a learning approach defines how students undertake learning. For example, Biggs (1987) categorised learners as taking on a deep or surface approach in a course. Later, Entwistle (1998) extended this to include, most significantly, a strategic approach, and a number of other categories. This study will focus only on learning styles, not on learning approaches.

Learning styles are measured by asking the learner to complete a questionnaire, choosing responses they feel are appropriate for themselves. Through analysis of responses, a student's learning preferences can be identified according to the dimensions of the particular inventory.

### 1.2 The VARK Questionnaire

The VARK Questionnaire (Fleming, 1995) measures preference for receiving instructional material. VARK differentiates preferences as:

- Visual,



**Figure 1. The VARK Questionnaire presented online**

- Aural,
- Read/Write, and
- Kinesthetic.

A learner with a visual preference prefers to learn by seeing graphs, charts, and flow diagrams. “Sometimes they will draw maps of their learning sequences or create patterns of information. They are sensitive to different or changing spatial arrangements and can work easily with symbols” (Fleming, 1995, p. 2). A student with an aural preference is best served when information is communicated through speech. Students with a read/write preference receive information best when it is presented in written form. In the generally understood sense, kinesthetic learners prefer to learn using their tactile senses. The VARK sense is a little more general: “kinesthetic individuals prefer to learn through direct practice, which may also involve the other perceptual modes” (Leite et al., 2010, p. 326).

The VARK preferences can be related to sensory modalities (eg tactile, visual and auditory), which are studied experimentally within educational psychology (Conway & Christiansen, 2005; Mayer, 1997).

The VARK Questionnaire is available online (Fleming, 2009b) and free for all to use. The questionnaire consists of 16 questions, each with four possible responses. A participant must answer at least 12 questions and can choose from none to all four of the responses for each question (see Figure 1).

After analysis of responses, a learner can be identified as having a single preference, being bimodal, trimodal, or accepting all four modes (this is referred to as a VARK preference). Analysis of the tens of thousands of completed online questionnaires (Fleming, 2009c) produces the distribution of participants with these combinations shown in Table 1.

The distribution of single preferences can be broken into specific preferences and presented together with multiple preferences to create an overall picture of typical

**Table 1. Single and multiple VARK preferences**

Profile	Proportion
Single preference (V, A, R, K)	37.0%
Bi-modal (VA,VR,VK,AR,AK,RK)	15.5%
Tri-modal (VAR,VAK,ARK,VRK)	12.5%
All four modes (VARK)	35.0%

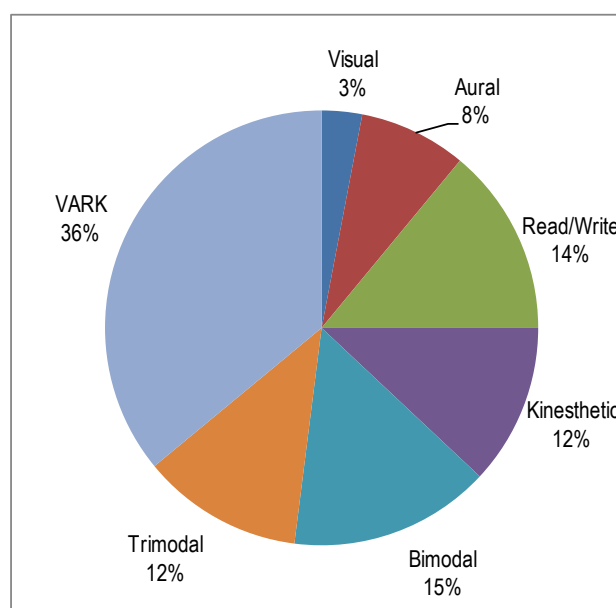
learning preferences as shown in Figure 2.

At the end of the questionnaire, participants are given their calculated profile. They are also given the opportunity to respond to this information, to state whether they believe the calculated profile matches their perceived learning preference. Results of agreement are reported as a measure of validity on the VARK site (Fleming, 2009a). Such results are certainly not a formally accepted measure of validity, and would perhaps be better described as a measure of internal consistency, but general agreement is certainly more encouraging than general disagreement. The responses and the proportion of responding participants from the general population who have chosen each response are shown below.

- Match = 58%
- Don't Know = 38%
- No match = 4%

The VARK Questionnaire is claimed to be “an excellent vehicle for opening a dialog on the differences that might exist in the way individuals prefer to learn, but the statistical properties are not sufficient for its use as a research tool” (Fleming, 2009c). In that light, measurements gained from the VARK Questionnaire (and perhaps from all such inventory related tools) should be taken as a way to begin understanding how students’ learning preferences can be catered for in a course, and not used to predict or infer outcomes for individual students.

The VARK Questionnaire has recently undergone testing for validity: “many individuals are interested in using the VARK as an input or outcome measure in research, and it is these uses that require a more rigorous evaluation of its psychometric properties” (Leite et al., 2010, p. 324). The questionnaire was found to be a valid instrument for identifying learning preferences: “those who wish to use the instrument as a way of helping students identify their preferences should feel



**Figure 2. Typical VARK learning style preferences (Fleming, 2009c)**



comfortable in this use” (p. 336). However, Leite and his colleagues did not suggest that the VARK Questionnaire could be used as an instrument to predict success in learning. It would therefore be invalid to attempt to infer student attributes based on VARK preferences. In our study, student preferences are measured as a general consideration to inform our teaching, not as a way of concluding anything about individual students.

### 1.3 Learning Styles in Computing Education

A number of instructors of computing have used learning style questionnaires with their students and have reported their findings.

Thomas et al. (2002) used the Felder-Silverman Learning Style Index to measure the learning styles of their students in a sequence of two introductory programming courses. They compared learning styles captured at the beginning of the sequence with exam and final marks. They found significant differences in exam marks between reflective and active learners, with reflective learners performing significantly better. Also, verbal learners statistically out-performed visual learners in the exam. In overall results, on the four dimensions of the Felder-Silverman LSI, students who were reflective, intuitive, verbal, global learners performed better than students from other groups. Using these findings, Thomas et al. indicated that they would create materials that would “appeal to different kinds of learners’ preferences” (p. 36) and encourage students to reflect on their learning styles and strengthen less preferred modes of learning. They also suggested they would follow up with a further analysis in later offerings, to see if their efforts had improved student outcomes. In later cohorts, the mix of learning preferences had changed and was no longer comparable to the original study (Thomas, 2009).

Researchers from the University of Joensuu (now merged into the University of Eastern Finland) measured the learning preferences of students enrolled across a number of undergraduate computing courses (Bednarik & Fränti, 2004). The group used the VARK Questionnaire to determine students’ learning preferences with a view to exploring whether students with a particular preference would perform better across the courses they were studying. Their results showed no strong evidence that students who preferred a single modality would be more successful across courses. Students with a balanced VARK profile had slightly better performance. The group also measured the learning preferences of teachers, and found a variety of learning styles among these staff. In each course, students with profiles that matched those of the instructor tended to perform better than other students.

A multi-national study (de Raadt et al., 2005) looked for predictors of student success in introductory programming courses at eleven institutions in three countries. A number of cognitive and articulation tests were applied, including Biggs’s approach to learning (Biggs, 1987). No single measure emerged as a significant predictor of success, but the strongest correlation with success was found to be the students’ learning approach: learners with a deep approach to learning were found to be more successful and those with a surface approach were found to be less successful.

Apart from these few studies there does not appear to have been much exploration of learning preferences in computing education. While our data arises naturally from the course and the way it was taught, the scarcity of other work was a motivating factor in our subsequent analysis of the data.

## 2 The Study

We have collected and analysed the learning preferences of both on-campus students and external students over three offerings of a first programming course.

### 2.1 Motivation

Awareness of the variety of learning preferences in a particular cohort allows an instructor to design or adjust course materials to suit that cohort. Zualkernan (2007) describes how the Felder-Silverman Learning Style Index can be used to evaluate the resources for introductory programming students, and how this can be fine-tuned to improve learning outcomes.

If a cohort includes a large proportion of students with a particular learning preference, it would be wise to cater for that learning preference. Where there is a variety of learning preferences, materials should be presented in multiple formats.

In the study described by Bednarik & Fränti (2004), students with learning preferences similar to their instructor’s generally performed better. It is therefore useful to know the instructor’s learning preferences and whether this matches those of the students in the course. At the same time, an awareness of differences in learning preferences between instructor and students can help to ensure that particular groups of students are not advantaged or disadvantaged.

Our study began as an exploratory one, guided by the following questions.

1. What were the students’ learning preferences?
2. How did these preferences compare to those of the instructor?
3. What modalities presented in materials would cater for students’ preferences?

### 2.2 Method

Learning styles were measured in order to determine whether materials presented to students suited their learning preferences and whether the learning profiles of students matched that of their instructor. This was conducted assuming that preferences shift from one cohort to another, so results applicable to the current cohort will not necessarily represent future cohorts. Measurement of learning style is not an empirical science, so the results were not used as part of a statistical proof or for any form of inference.

The VARK Questionnaire was chosen to measure learning preferences because:

- it is a validated tool;
- it is simple to calculate and compare results;
- it is free, online and easily accessed by students; and
- results are easily related to materials presented to students.

Students were invited to complete the questionnaire during an initial workshop and to report (copy and paste) in a discussion forum their score for each preference and their learning preference calculated by the VARK site. These results were collected and recalculated to check that values were reported accurately. Students were also asked if they agreed with their rating and if they thought it reflected how they learn.

This collection of learning preferences was carried out as part of the teaching, not as part of a research project. The principal purpose was in fact to encourage participation in the online forum for the course. There was no inducement for students to take part, and students who declined to do so were not penalised in any way.

### 2.3 Context

The course used as a context for this study is an introductory programming course conducted by a single instructor.

The course is offered in two modes: on-campus and external. On-campus students attend workshop classes in a computer laboratory. Each workshop consists of short periods of didactic teaching mixed with practical exercises. External students follow the same format, but the didactic teaching is presented as written text with complementary recorded video snippets. Over the three semesters that the study has been running there have been 394 students, with 71% enrolled as external students and 29% studying on campus. Students in both modes were invited to complete the VARK Questionnaire and submit their results.

The initial design for materials had a focus on kinesthetic learning through practical exercises. Written material was also presented, mixed with occasional visual diagrams. Recorded video snippets containing slides and instructor audio and video were inserted into the materials as an alternative to text.

## 3 Results

There were 394 students enrolled over the three semesters. Of these, 226 voluntarily completed the VARK Questionnaire and posted their learning preference results, giving a participation rate of 57%.

Participants' acceptance of their calculated learning style was analysed. Learning styles data was used to explore two principal issues: the match between the learning preferences of the students and the instructor, and the general pattern of the students' learning preferences.

### 3.1 Agreement with Learning Style

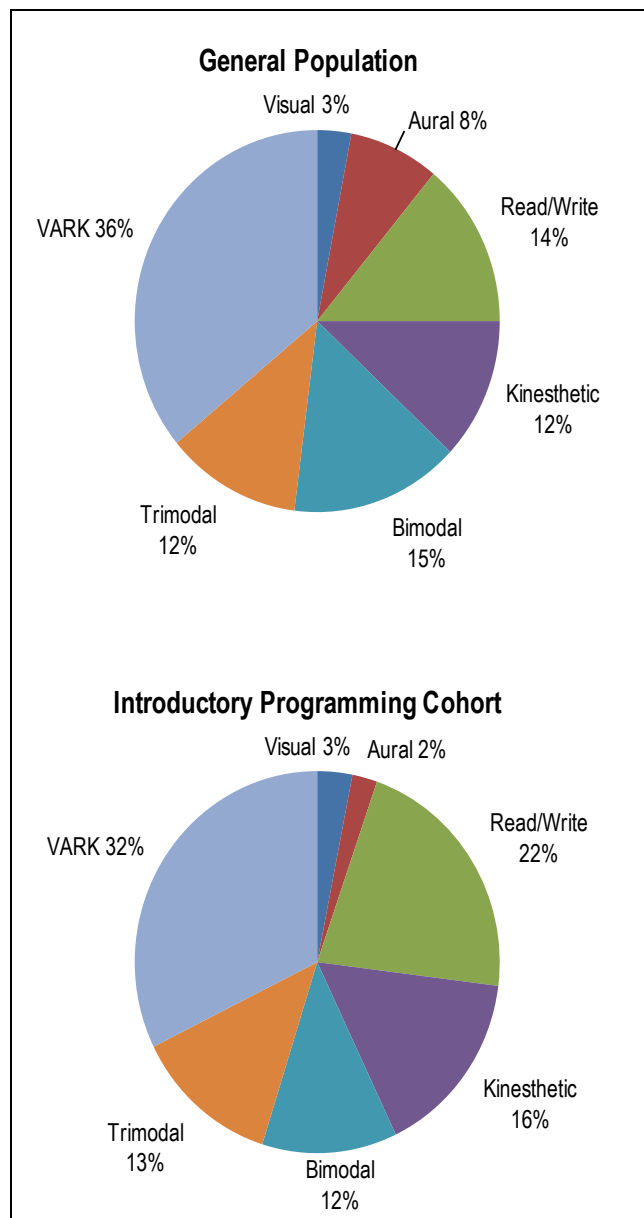
Participants were asked if they agreed with their calculated learning style. Positive responses can be used as a small measure of the validity of the VARK instrument. Over the three cohorts, 78% of students who posted their learning style also gave their agreement. Unsure students were grouped with students who gave an indefinite response in the "don't know" category. Results are shown in Figure 3 alongside the levels of agreement given by the general population as reported at the VARK site.

**Table 2. Agreement with calculated learning style**

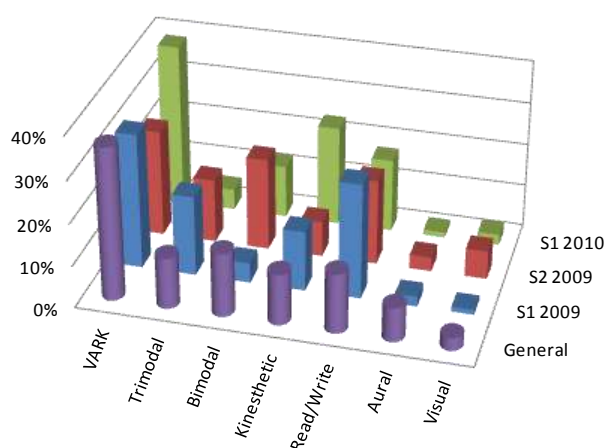
Response	Participants	General Population
Yes	72%	58%
Don't know	18%	38%
No	10%	4%

**Table 3. Learning Preferences of students by mode**

Preference	On-campus (n=60)	External (n=166)	All
Visual	2%	2%	2%
Aural	2%	2%	2%
Read/Write	29%	19%	22%
Kinesthetic	19%	16%	16%
Bimodal	13%	13%	13%
Trimodal	10%	16%	14%
VARK	25%	32%	30%



**Figure 3. Comparison of participants (n=226) to general population**



**Figure 4: Comparison of all three cohorts (n=85, 60, 81) and general population**

Compared to the general population, participating students were more confident about whether they agreed with their calculated learning style, with fewer indefinite “don’t know” responses. There were more definite “yes” and “no” responses, with the yes responses strongly outweighing the no responses.

### 3.2 Instructor’s Learning Preference

After completing the VARK Questionnaire, the instructor’s learning preference was determined to be a single preference for kinesthetic learning. This indicates that the instructor has a preference for learning through doing (perhaps not abnormal for a programming instructor). It should be noted that this is the instructor’s learning preference and not necessarily his teaching preference, although it seems likely that these would be related.

Table 3 shows that only 16% of students share the instructor’s preference for kinesthetic learning.

### 3.3 Student Learning Preferences

The learning preferences, shown in Table 3, show sizeable populations of multi-modal VARK students (30%), and read/write students (22%).

Figure 3 shows a comparison between students in the course and the general population as reported on the VARK site. Students in the course were more likely to have a read/write preference than normal. Fewer students in this cohort have an aural preference, compared to the general population. However, there appear to be more similarities than differences between the two breakdowns, suggesting that the learning preferences of students in this course are not markedly different from those of the ‘general population’ of tens of thousands who have taken the VARK test.

The initial conclusion from this breakdown of preferences was that the kinesthetic aspect of the course materials was appropriate, but that the read/write aspect should not be overlooked. This conclusion will be further discussed in the discussion section.

#### 3.3.1 Separate Cohorts

At first sight, the learning preferences of separate cohorts did not appear to match. Figure 4 shows the preferences

**Table 4: Example VARK scores from a selection of students (vs, s, and m indicate very strong, strong, and mild)**

Calculated Student Preference	Student Scores			
	Visual	Aural	Read/write	Kinesthetic
Kinesthetic (vs)	1	3	2	13
Kinesthetic (s)	6	6	5	13
Kinesthetic (m)	4	8	8	13
Kinesthetic (m)	1	4	4	7
Bimodal	5	7	12	12
Bimodal	1	6	2	7
Bimodal	2	3	5	6
Trimodal	15	2	14	12
Trimodal	4	5	5	2
VARK	13	9	13	12
VARK	6	4	6	6
VARK	3	12	9	6
VARK	3	4	5	5

of each of the three cohorts, again alongside those from the VARK general population.

There are some clear differences between cohorts; for example, a high proportion of read/write preferences in S1 2009 (first semester of 2009) and of kinesthetic in S1 2010 (first semester of 2010). But at the same time there are similarities between the cohorts, similarities that should be mined for useful information.

In all three cohorts the proportion of read/write preferences is noticeably higher than that of the general population. This perhaps not surprising, as a read/write preference is possibly associated with academic success of the sort required to enter a university course.

The high proportion of multimodal or VARK preferences appears comparable with that of the greater population, as does the low proportion of visual preferences.

But in fact the most useful feature of the learning preference analysis is one that is hidden in Figure 4. The real lesson for the designer of learning materials lies in the bimodal and trimodal preferences, and will be elaborated in the following section.

#### 3.3.2 Bimodal and trimodal preferences

At first we readily accepted the broad preference groupings that are reflected in most of the tables and figures to this point. People’s learning preferences can be single-mode (visual, aural, read/write, or kinesthetic), bimodal (any two of those four), trimodal (any three of those four), or VARK (all four). Single-mode preferences are accompanied by a strength indicator of mild, strong, or very strong.

The algorithm for distinguishing preferences is given on the VARK website and the breakdown provided to users includes an explicit score for each of the four primary preferences, allowing further analysis.

Table 4 shows the calculated preferences and the individual scores for a selection of students in order to demonstrate how preferences are determined. The

strength ratings of the first four students show that the strength of the preference depends not on the absolute score in that preference but on the differences between that preference and the others. A preference is given when the difference stepping down between a stronger and next strongest modality is less than or equal to a proportional margin. The margin is greater when there are more points overall; the more points the greater the allowed margin (1 to 4). For example, the first four students are kinesthetic because their kinesthetic scores are higher than their other scores by a proportional margin. A student shows a multi-modal preference when there are smaller steps between modalities. For example, the first trimodal student totalled 48 points, allowing a margin of 4 points between modalities; this succeeds from 15 to 14 and 14 to 12, but not 12 to 2, so the student has a trimodal preference. It is not clear why this method was chosen; one might ask why the third VARK student is considered VARK when their strongest preference differs greatly from their weakest.

Of more interest, though, is that a single-mode preference by no means excludes the other modes. While the third and fourth students in Table 4 have a clear preference for the kinesthetic, they would also appear to be quite comfortable with both aural and read/write materials, whereas the first student is markedly less so. This sort of information is obscured by the simple single-mode classification.

But what the instructor really needs to know in order to develop good teaching materials is not which modes the students prefer, but which modes they do not prefer. If a particular learning resource were purely read/write, it would probably disadvantage the first kinesthetic student and the second bimodal student in Table 4, but the rest of the students would appear able to cope with it. If a learning resource were purely visual, it would probably cause difficulty for half a dozen of the students – including two with the multi-modal VARK preferences. Therefore we need to consider not which learning preferences are present in a cohort, but which are absent: which students will be unable to cope with material designed for specific learning preferences?

Rather than observing that 27% of the students have a single-mode read/write preference, we ask how many students include a read/write preference in either single-mode read/write, bimodal, trimodal, or VARK. The answer is a rather more impressive 73% of all students tested. This, and the figures for the other three modes, are summarised in Table 5.

It is clear from Table 5 that if learning materials were designed to suit read/write students only, 27% of students would not learn well from them. Likewise, if learning materials were designed to suit kinesthetic students only,

**Table 5: proportion of students showing each preference either as a single mode or as part of a multiple mode**

Preference	Students
Visual	50%
Aural	44%
Read/write	73%
Kinesthetic	67%

33% of students would face some difficulty. We now explore which students, if any, would be disadvantaged by learning materials designed to suit both read/write and kinesthetic preferences.

If materials were designed to suit both of these groups, the 73% of students whose preferences include read/write and the overlapping 67% of students whose preferences include kinesthetic should be able to learn effectively from them. Who does that leave? It leaves the students with single-mode visual preference, the students with single-mode aural preference, and the students with bimodal visual and aural preferences. Every other student has preferences that include either read/write, kinesthetic, or both. Of the 226 students who completed the VARK test, only 14 fall into this visual/aural set, as summarised in Table 6.

Inspection of the preferences in Table 6 shows that in fact 11 of the students have reasonable scores in both read/write and kinesthetic, two have reasonable scores in read/write, and one has a reasonable score in kinesthetic. In other words, while these 14 students (6% of the total) have clear preferences for the visual, the aural, or both, they should be reasonably able to cope with materials designed to cover both read/write and kinesthetic preferences.

## 4 Discussion

The conclusions of this study discuss the level of student agreement with measured VARK preferences, alignment with the instructor's learning style, and an examination of how participating students' preferences can be catered for.

### 4.1 Agreement

Of the participants who indicated agreement, 72% agreed with their calculated learning style and only 10% disagreed. This level of agreement is above that indicated by the general population, providing some confidence in the results gathered in this study.

**Table 6: individual VARK scores for all students whose preferences do not include read/write or VARK**

Preference	Visual	Aural	Read/write	Kinesthetic
Aural	2	8	4	6
Aural	2	7	5	2
Aural	2	8	6	4
Aural	5	11	4	4
Aural	3	10	6	6
Visual	13	5	6	5
Visual	8	3	3	2
Visual	12	3	7	8
Visual	14	6	6	6
Visual	13	7	8	8
Visual	11	1	8	6
Visual	7	2	3	4
VA bimodal	7	6	3	4
VA bimodal	6	5	3	3

## 4.2 Difference between instructor and students

There was only a 16% alignment between the instructor's kinesthetic learning preference and those of students in the course. However, where the instructor had a single learning preference (kinesthetic), many of the students had multi-modal profiles which included kinesthetic as a modality to some degree. Indeed, subsequent inspection shows that with VK, VAK, VRK, AK, ARK, RK and VARK students, who all include kinesthetic among their preferences, the overlap is a rather more impressive 67% of all students.

The instructor is confident that the design of the course materials was influenced by his kinesthetic learning preference. Initially, there was a plan to reduce the importance of textual content of the course materials. After coming to know the learning preferences of students, and particularly the strong read/write preference of over a quarter of the first cohort, the textual content was instead enhanced rather than reduced.

This study showed that it is unwise to assume that students in a cohort share the same learning preference as their instructor. As we observe in the title of this paper, my students don't learn the way I do – and it would be a mistake to design my learning materials on the assumption that they do.

When focusing on the difference between students enrolled in different study modes, interesting differences emerged. A much larger proportion of on-campus students than of external students prefer kinesthetic learning. It may be that these students have chosen to study on campus, in some part, because of this aspect of their learning. It was surprising that a larger proportion of on-campus students had a read/write preference, compared to external students, who are often presented with materials primarily as text in external courses. A larger number of external students had a broader VARK preference than on-campus students. Perhaps students willing to undertake external study have to be better prepared to accept materials in a variety of forms. However, these conclusions might be putting the cart before the horse. Do students choose to study in a particular mode because of their learning preferences or because of other circumstances such as geographical location, family, employment, etc? If it is the latter, we should be wary of attributing any significance to the relationship between the students' modes of study and their learning preferences.

## 4.3 Catering for students' learning styles

While our initial focus was the foregoing comparison between the lecturer's and students' learning styles, we now believe that the real message of this work lies not in the analysis of what preferences students have but in the complementary analysis of what preferences they do not have. When a system pigeonholes students learning styles or preferences into single categories, an inevitable conclusion is that in order to be effective, learning materials must cover all of those categories. The great advantage of the VARK system, which we did not realise until we were well into the analysis phase of our work, is its recognition that single preferences do not necessarily suit the bulk of the population, and its provision of actual

scores in each category so that further analysis can be conducted.

While it might indeed be a good idea to design learning materials to suit all possible VARK modalities, or all preferences in any other system, we are now in a position to conclude that learning materials designed to cover both read/write and kinesthetic preferences will suit all students in the three cohorts we have studied. Learning materials do not have to be all things to all people: it is enough that they be at least one usable thing to each student.

Given the similarity between the VARK distribution of our three cohorts and the general population, we believe that our conclusion can be generalised to other courses, unless their instructors conduct VARK assessments and find that their profiles differ significantly. If learning materials are designed with good textual material to suit the read/write modality and good hands-on experience to suit the kinesthetic, such materials should suit the learning preferences of virtually all of the students.

## 5 Acknowledgements

We are grateful to the anonymous reviewers of this paper. The points that they raised, particularly the more critical ones, drew our attention to parts of the paper that required further explanation to avoid confusing readers.

## 6 References

- Bednarik, R., & Fränti, P. (2004): Survival of students with different learning preferences. *Proceedings of the Seventh Baltic Sea Conference on Computing Education Research (Koli Calling 2004)*, Koli, Finland. 121 - 125.
- Biggs, J. B. (1987): *Student Approaches to Learning and Studying*. Melbourne, Australian Council for Educational Research.
- Coffield, F., Moseley, D., Hall, E., & Ecclestone, K. (2004): *Learning styles and pedagogy in post-16 learning: A systematic and critical review*. UK, Learning and Skills Research Centre.
- Conway, C. M., & Christiansen, M. H. (2005): Modality-Constrained Statistical Learning of Tactile, Visual, and Auditory Sequences. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, 31(1):24 - 39.
- de Raadt, M., Hamilton, M., Lister, R., Tutty, J., Baker, B., Box, I., et al. (2005): Approaches to Learning in Computer Programming Students and their Effect on Success. *Higher Education in a Changing World: Research and Development in Higher Education*, 28:407 - 414.
- Entwistle, N. (1998): Improving teaching through research on student learning. In J. Forrest (Ed.), *University teaching: international perspectives*. New York: Garland.
- Felder, R. M. (1996): Matters of Style. *American Society of Electrical Engineers: Prism*, 6(4):18 - 23.
- Fleming, N. D. (1995): I'm different; not dumb: Modes of presentation (V.A.R.K.) in the tertiary classroom *Proceedings of the 1995 Annual Conference of the Higher Education and Research Development Society of Australasia (HERDSA)*. 308 - 313.

- Fleming, N. D. VARK Frequently Asked Questions, <http://www.vark-learn.com/english/page.asp?p=faq>. Accessed August 17 2009.
- Fleming, N. D. The VARK Questionnaire, <http://www.vark-learn.com/english/page.asp?p=questionnaire>. Accessed 17 August 2009.
- Fleming, N. D. VARK Research & Statistics, <http://www.vark-learn.com/english/page.asp?p=research>. Accessed 17 August 2009.
- Herrmann, N. (1989): *The creative brain*. North Carolina, Brain Books.
- Jackson, C. J. (2002): *Learning Styles and its measurement: An applied neuropsychological model of learning for business and education*. UK, PSi-Press.
- Kolb, D. A. (1999): *The Kolb Learning Style Inventory, Version 3*. Boston, USA, Hay Group.
- Leite, W. L., Svinicki, M., & Shi, Y. (2010): Attempted Validation of the Scores of the VARK: Learning Styles Inventory With Multitrait–Multimethod Confirmatory Factor Analysis Models. *Educational and Psychological Measurement*, **70**(2):323 - 339.
- Mayer, R. E. (1997): Multimedia Learning: Are We Asking the Right Questions? *Educational Psychologist*, **32**(1):1 - 19.
- Myers, I. B., & McCaulley, M. H. (1985): *Manual: a guide to the development and use of the Myers-Briggs Type Indicator*. Palo Alto, USA, Consulting Psychologists Press.
- Pashler, H., McDaniel, M., Rohrer, D., & Bjork, R. (2009): Learning Styles: Concepts and Evidence. *Psychological Science in the Public Interest*, **9**(3):106 - 119.
- Thomas, L. (2009): Personal Communication.
- Thomas, L., Ratcliffe, M., Woodbury, J., & Jarman, E. (2002): Learning Styles and Performance in the Introductory Programming Sequence. *ACM SIGCSE Bulletin*, **34**(1):33 - 37.
- Zuallkernan, I. A. (2007): Using Solomon-Felder Learning Style Index to Evaluate Pedagogical Resources for Introductory Programming Classes. *Proceedings of the 29th International Conference on Software Engineering (ICSE'07)*, Minneapolis, USA. 723 - 726.

# A unit testing approach to building novice programmers' skills and confidence

Jacqueline L. Whalley and Anne Philpott

School of Computing and Mathematical Sciences

AUT University

PO Box 92006, Auckland 1142, New Zealand

jwhalley, aphilpott@aut.ac.nz

## Abstract

This paper discusses the integration of unit tests into a first semester programming course. The students were supplied with unit tests to support their learning and assessments. A questionnaire was completed by the student cohort about their use and perceptions of these unit tests. As a result of both the students and our experiences we examine the advantages and disadvantages of introducing unit tests early and make some pedagogical recommendations for the introduction and use of unit tests in first year programming.

**Keywords:** testing, novice programmers, assessment.

## 1 Introduction

Novice programming students struggle to take a structured approach to gaining a full understanding of what is required of their program. This initial inadequate consideration then leads to haphazard tinkering in an attempt to produce a piece of code that does what it should (Perkins and Martin, 1986). For many students the barrier to identifying and fixing bugs or extending and refactoring code is a negative experience and they simply "abandon all hope of solving the problem on their own" (Perkins et al., 1989, p.265). Several factors motivated the early introduction of unit tests. Student numbers in our introductory programming course had more than doubled over the last two years, increasing the student to teacher ratio considerably. One of the most noticeable consequences of this change was that students were not receiving immediate feedback during their laboratory sessions and often waited long periods of time to receive assistance. Students were also receiving less individualised feedback and support when undertaking their take home programming assignment. At the same time the sole lecturer on the course was swamped with marking. This resulted in delays in students receiving critical early feedback. As a result of the increased marking load student tutors were employed to assist with the laboratory teaching and assignment marking. Along with the introduction of tutors there was then the need to ensure that the marking was consistent across markers.

It was anticipated that the introduction of unit tests would have several advantages. In particular, that

marking would be more efficient and consistent and that students would be provided with a mechanism to get immediate feedback on their code writing.

It has been suggested that students should have more success if we can move them away from a *trial and error* approach to writing code and towards a *reflection in action approach* (Schön 1983). Subsequently, Edwards (2006) noted that software testing tends to move students towards a reflective approach to programming. We propose that by supplying students with unit tests we provide structure for them that supports them in reflective tasks such as requirements extraction and semantic bug finding. One of the primary goals, of this work, was therefore to provide a tool that guides the students through their initial design process. However, Ginat (2007) found that test instances supplied later and fed to the students one at a time had led to hasty design and futile patching. In the approach reported here foreknowledge of relevant test cases is provided and we believe that this should trigger the observation and consideration behaviour that Ginat found lacking in his experience of using unit tests with novice programmers.

## 2 Background

Test units have been used in introductory programming courses in the past. However, generally the approach reported in the literature is focused on test driven development (TDD), with students writing the tests. Desai et al. (2008) provide a useful survey of results from these studies. Marrero and Settle (2005) introduced a test first early approach to their assessments for novice programmers. As the first stage of their assessment students were required to write unit tests and submit them ahead of the actual code submission. They suggest that despite the extra load of learning to write unit tests, an emphasis on testing is beneficial. An evaluation of a test-driven learning approach, where again students were required to write their own unit tests, found that there was a strong reluctance on the part of novice programmers to adopt a test-first approach (Janzen and Saiedian 2008).

Edwards (2003) identified several roadblocks to integrating TDD into an early programming paper:

1. Knowledge
  - a. Students are not ready for testing until they have basic programming skills
2. Time
  - a. Is limited and there isn't enough time to also teach testing
  - b. Teachers already have their hands full grading code correctness, it may not be feasible to also grade unit tests



### 3. Value

- a. Students must be able to see the value in the non-functional code

Studies in general have found that novice programmers find TDD hard largely because they are struggling to find purpose in the functional code, so testing it is difficult. Despite these difficulties studies to date have reported that courses using TDD increased student confidence (Edwards 2004, Janzen 2008, Kaufmann 2003, Muller 2001) and increased code comprehension (Jones 2004, Muller 2002, Janzen 2006).

Because the students in the course are assumed to have no prior programming experience, and, in light of the roadblocks to integrating TDD, it was decided to supply the test units rather than require the students to write them. We propose that this is a way to introduce good practice early without adding too much additional learning load for the students. The focus is on students using the unit tests as a tool that will help them see the value in testing their code and on the students being able to read and comprehend unit tests.

Concurrent with our investigations Cardell-Oliver et al. (2010) were investigating the use of automated testing tools to improve the quality of programs produced by software engineering students. The suite of tools they investigated included JUnit for assessing the correctness and Checkstyle to evaluate the readability of student code. For lab exercises they provided students with unit tests, but for the assignment they required the students to write unit tests. On reflection they suggested that for first year students using instructor test cases is more appropriate than an approach where students are required to write the test cases. Their experience led them to believe that this approach focuses students' attention on the quality of their programs. Additionally, they found that, with supplied unit tests, most students were able to write fully compliant functionally correct code. This work lends support to our current approach to integration of testing early with instructor supplied unit tests.

### 3 Teaching practice

First semester programming students attended two one hour lectures and a two hour supervised laboratory session each week. They were taught Java using an objects-first approach and the BlueJ development environment. All the lab exercises that required code writing were supported by JUnit test cases that were supplied to the students. In the third week of the course the students were given a hands-on, instructor led introduction to the running and reading of test units. They were also introduced to common novice errors that might result in compiler errors and test case failure. The BlueJ debugger was used to demonstrate how to discover the source of the error using the unit test. A large part of this session was devoted to learning how to read the unit test cases, elicit the specification for a method and interpret assert statements and test failures. Students were taught that unit tests are a tool.

The lab exercises supported with unit tests were based on a kiwi world project that simulated kiwis and their zoo enclosure. For the kiwi project students were provided with existing code in the first week and in each subsequent week built on that project. For each lab they

were supplied with a new set of unit tests that they needed to copy into their project to test the new functionality they would be adding.

Because the unit tests were tightly coupled with the code the students were testing we found it useful to provide the unit tests with the test cases commented out. As the students completed each task they then uncommented the related test cases and compiled and ran the tests.

The students also worked on a three stage take-home assignment. For this assignment the students were provided with a set of unit tests and starting code for each stage. The assignment required the modelling of an organic pig farm with a waste management system that supplied fertiliser to the farm.

In the first stage the unit tests supplied had detailed inline comments and test case header comments to assist the students to read and comprehend the unit tests. Additionally a detailed assignment specification that specified in words the method signatures and any rules that the method output should meet was provided. At the second stage the assessment prescriptor provided was more general and lacked detailed descriptions of the method signatures so students had to read the unit tests before starting to write their code in order to elicit some simple requirements for their methods. From stage 2 extended inline comments and header comments were removed from the unit tests.

For the final stage of the assignment the prescriptor was a general overview of each problem to be solved. The students needed to read and understand the unit tests in order to get a full specification of the code to write, including the algorithmic rules.

## 4 Results

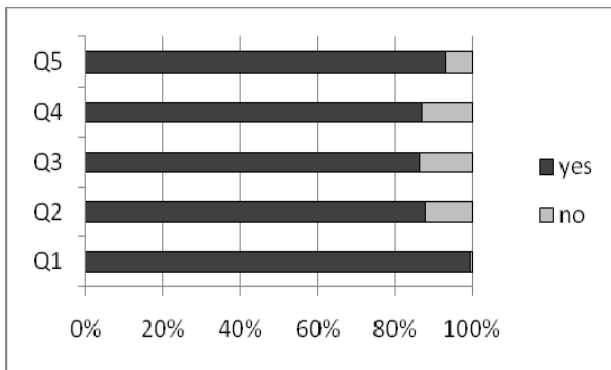
In the tenth week of the 12 week semester students were asked to complete a questionnaire (see Appendix A) about their use and perceptions of unit tests. We asked them to focus solely on the use of unit tests in their assignment. Of the 172 students in the class, 133 students provided responses for analysis (Figure 1).

Question 1 asked the students whether or not they used the unit tests. Ninety nine percent of the students responded that they used the unit tests supplied. Of that 99%, 88% said they read the related unit test cases (Q2) when preparing to answer a question and 86% said that this helped them to decide what the code they wrote needed to do (Q3). This high percentage of students indicating they used the tests is not surprising given that from stage two the assignment structure forced the students to elicit some of the requirements from the tests.

When asked if they found the unit tests helpful for debugging code 87% of the students said that they were helpful (Q4).

There was a general consensus, 94% of students, that unit tests gave the students confidence that their code was correct and complete (Q5). The remaining students for whom it did not provide confidence said that they found unit tests hard to understand and read.

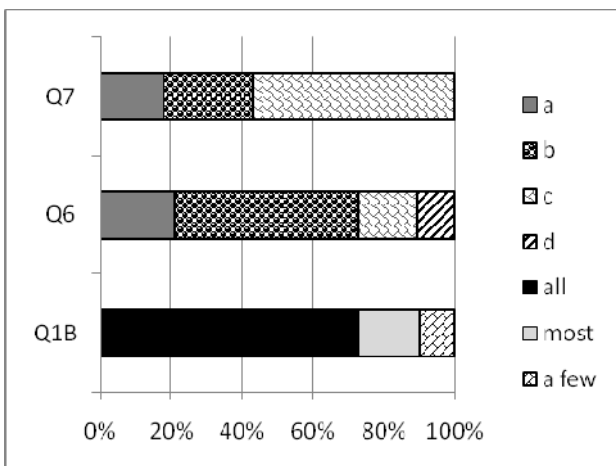




**Figure 1: Responses to Questions 1-5 ( $n = 133$ )**

To elicit how the unit tests were being used the students were asked how frequently they used unit tests and how they used the unit tests. Question 1B asked, if they used the unit tests, how many of the unit tests they ran. Seventy two percent of students responded that they ran all of the tests, 17% ran most of the tests and the remaining students said that they ran a few of the tests.

Question 6 asked at what point in the development of their code the students ran the unit tests cases: (a) as soon as the functionality the method tests was implemented, (b) on completion of the question, (c) on completion of a related set of questions or (d) on the completion of the assignment. The majority of students responded that they ran the relevant unit test cases when they had completed a question while a minority of students ran them at the end of the assignment (Figure 2).



**Figure 2: % responses by question ( $n = 133$ )**

In Question 7, the students were asked how often they ran the unit tests for a marked exercise: (a) 0-5 times, (b) 5-10 times, or (c) 10 or more times. This question appears to have been misinterpreted by some students. As they ran the unit tests at the end of each question they must have run them more than 5 times, given that there were more than 5 questions, but some responded 0-5 times. Despite this complication 56% of the students stated that they had run the tests more than ten times showing that the tests were used frequently during the development process.

The students were also asked three open questions about their reaction to the use of unit tests in their assignments (Questions 8-10, Appendix A).

A content analysis approach was adopted to evaluate the student responses (Stemler, 2001). The categories for content analysis for each question were established using the emergent coding guidelines (Haney et al., 1998). This categorisation system was used by two independent raters to classify student responses. There was very strong inter-rater agreement on the classification of the responses to all three questions<sup>1</sup>.

It should be noted that some students made more than one statement in their response to a question. Each statement was classified to a category. Therefore the total number of ratings for a question can exceed the number of students.

The following subsections report on the results of the application of the content analysis to the student responses. The implications for teaching practice are discussed in a later section.

#### 4.1 Question 8

Twenty one percent of students reported that they had found the unit tests helpful for debugging their code.

*"Helps as a starting point for debugging"*

Twenty-six percent of students reported that unit testing helped them overcome the difficulty of knowing what was required. They mentioned finding it useful for identification of the correct method signature and understanding parameters, return types and values.

*"If I was in doubt about a task I read the unit test first."*

Thirty one percent of respondents reported that using unit tests helped them confirm the correctness of their code. Respondents in this category also mentioned that being able to confirm the correctness of their code gave them confidence and motivated them to complete the assignment.

*"It gives me confidence <that> I got the question right ... If it is wrong at least it will show me which part is incorrect."*

*"The unit tests help me feel confident my code was correct and complete...<they> let me know that I am on the right track."*

*"...knowing the previous answer is shown to be correct it give a lot of heart to answer the more tougher ones."*

Some students mentioned finding unit tests particularly useful to identify code that runs but does not fully meet the specifications.

*"When I make a semantic error it was easier to identify using the unit tests"*

Seventy one percent of students provided an answer to question 8.

#### 4.2 Question 9

The great majority of the students did not mention anything that they did not like or find helpful about unit testing.

<sup>1</sup> Q8  $\kappa = 0.915$ ; 95% confidence interval from 0.862 to 0.969.

Q9  $\kappa = 0.934$ , 95% confidence interval from 0.861 to 1.008.

Q10  $\kappa = 0.939$ , 95% confidence interval from 0.880 to 0.998.

Sixteen percent reported that they found the unit tests or the results of the tests complicated and the error messages difficult to understand. They said that they had difficulty learning from the unit tests results. Some had problems knowing what was being tested and comprehending the unit test code. Almost four percent of students mention frustration with unit tests repeatedly failing their code.

*"A little complicated to get your head around it at first"*

*"When errors and a fail occurred I could not always understand the tests"*

Two percent of students did not like the mismatch between the specification provided by the unit tests and the assignment prescriptor. One student was not happy about losing marks for code quality when all the unit tests passed.

### 4.3 Question 10

Most students did not respond to the opportunity to make any further comment about the use of unit tests. Twenty five percent of students gave an additional positive judgement regarding the use of unit tests while another five percent gave a further general negative comment about the use of unit tests in their assignments.

Five percent of students made recommendations regarding the teaching and presentation of unit tests for their assignments. The recommendations included:

- Providing more comments in the unit test code.
- Providing more explanation or teaching of unit tests
- Wanting the opportunity to write their own unit tests.

## 5 Implications for teaching

We identified several advantages and a significant overhead in this approach. The advantages identified included:

- Students are more confident of their submission as they now know their code meets correctness requirements.
- Students gain more immediate feedback as they work. This immediate feedback is important as it means that students are not left floundering in uncertainty and are able to recognise when they need help, and to seek that help.

*"They are a great way of getting feedback and direction while writing code particularly when you are out of school."*

- Increased student confidence was observed and there was an increased engagement in online peer support.

*"Thank you for including unit tests. This helped me understand java programming better."*

- The course leader was confident about the consistency and correctness of marking by student tutors.

It should be noted that unit tests are not a substitute for student-instructor interaction and we found that our lab

sessions and on-line discussion forums were busier than usual just as observed by Cardell-Oliver et al. (2010). We believe that this is because the unit tests provided a means not only for students to identify problems in their code but also a means to express the error in a way that allowed others to answer their questions.

The significant overhead was the increased preparation necessary to provide a clear and complete set of unit tests for all lab and assessment work. However this had the advantage of highlighting issues regarding the assessment requirements that may have otherwise been missed.

### 5.1 Presentation of material

Our results indicate that the adopted approach of explicitly teaching the reading and running of tests early in the course was effective for most of the students. However a minority of students (16%) still indicated they found the tests or the generated error messages difficult to understand. It may be beneficial to explicitly assess test reading and error message interpretation early to identify and provide additional support to this group.

### 5.2 Motivating reading and use

Use of the tests must be made as straightforward as possible for the students. Encouraging test reading by enforcing discovery of requirements from the test seems to have been an effective strategy and no doubt contributed to 99% reporting using the tests.

Sequencing of tasks and blocks of tests can prove a challenge in assessment design and this contributed to the preparation overhead identified. When students complete a part, or even a subpart of a question, the tests, or lines within tests that need to be enabled should be clear. We believe that commented out lines or tests supported by clear indicators of the question part they applied to was a better approach than the confusion that might have arisen from multiple version of tests being provided. Any mistakes in sequencing or lack of clarity in the comments however did confuse students.

*"I found unit tests/questions out of order frustrating when they were dependant on each other"*

### 5.3 Test coverage

It may be necessary to write some tests cases that would not normally be required to support this approach. This can arise from a need to support task order or allow for students who don't complete all questions or parts of a question. For example, an accessor method might normally be adequately tested by a constructor test and the use of the accessor when testing other public methods. Students answering a question that requires them to write an accessor method however will require explicit tests that allow them to confirm the correctness of this method and discourage hard coding of return values, an approach taken by some students in this cohort. The unit tests written therefore need to not only cover standard test cases (e.g. data boundaries etc.) but need to take account of task sequence and common novice errors.

### 5.4 Ensuring quality not just correctness

Care must be taken to avoid an observed tendency to approach assignments in a tick list fashion – tests-pass

good – move on to next task. Students who take this approach fail to take that important reflection in action step and ask themselves "can I write my code better?". A well designed marking schema and the use of supporting tools such as Checkstyle can mitigate this. Clear expectations about code quality and readability should also be communicated to the students at the start.

## 5.5 Hack and Tinker

While unit tests help the majority of students in the class and give them focus, the weak students still tinker with their code rather than using the guidance of the unit tests. We believe that this is because their knowledge of programming is so fragile they cannot use or understand even the simplest constructs and therefore cannot use or understand the unit tests. We have observed that these students often use the unit test code as a template for their code and copy and paste the unit test code into the class and tinker hoping to get something working.

## 6 Conclusion

Taking a unit testing approach to learning, teaching and assessment of novice programming was effort well spent as we believe it benefited both the learners and the designers and assessors of the course. In particular we consider that using unit tests in this manner provides support for the often neglected middle learners. By far the majority of experiences reported by these students were positive. We acknowledge that extra effort is required in the preparation of course material but the gain in the quality and clarity of the assessments designed was a bonus we had not predicted.

The unit testing approach however did not decrease the time required for marking as much as we had hoped. There is still a substantive part of the marking time being used to give feedback on the quality of the student code. Supplementary tools, as previously mentioned, may be necessary before there is a significant reduction in the marking time.

The ability to read and interpret unit tests was assessed in the final examination but, given the 16% of students reporting that tests were difficult to read or understand, it may be more helpful to assess this skill earlier in the course and follow up those who are struggling.

We have reported on the students perceptions of the tests and their use. It would also be useful to investigate the way in which students actually use the tests. The impact of this approach on students' subsequent behaviour would also be interesting. Do students who used the provided unit tests demonstrate a better appreciation of the value of such tests in the future? Does the early test reading behaviour enforced affect their subsequent ability to write good unit tests?

## 7 References

Cardell-Oliver, R., Zhang, X., Barady, R., Lim, Y.H., Naveed, A. and Woodings, T. (2010): Automated Feedback for Quality Assurance in Software Engineering Education. *Proc. of the 21<sup>st</sup> Australasian Software Engineering Conference*, Auckland, New Zealand, 157-164.

Desai, C., Janzen, D. and Savage K. (2008): A survey of evidence for test-driven development in academia. *ACM SIGCSE Bulletin*, **40**(2): 97–101.

Edwards, S. (2003): Using Test-Driven Development in the Classroom: Providing Students with Automatic, Concrete Feedback on Performance. *Proc. of the International Conference on Education and Information Systems: Technologies and Applications (EISTA'03)*, International Institute of Informatics and Systemics, 421-426.

Edwards, S. (2004). Using Software Testing to Move Students from Trial-and-Error to Reflection-in-Action. *ACM SIGCSE Bulletin*, **36**(1):26-30.

Ginat, D. (2007): Hasty design, futile patching and the elaboration of rigor. *Proc. of the 12th Annual SIGCSE Conference on innovation and Technology in Computer Science Education* (Dundee, Scotland, June 25 - 27, 2007). ITiCSE '07. ACM, New York, NY, 161-165.

Haney, W., Russell, M., Gulek, C. and Fierros, E. (1998): Drawing on Education: Using student drawings to promote middle school improvement. *Schools in the Middle*, **7**(3): 38-43.

Janzen, D. and Saiedian, H. (2006): Test-Driven Learning: Intrinsic Integration of Testing into the CS/SE curriculum. *Proc of the 37<sup>th</sup> SIGCSE Technical Symposium on Computer Science Education*, 254-258. ACM Press.

Janzen, D. and Saiedian, H. (2008): Test-Driven Learning in Early Programming Courses. *Proc of the 39<sup>th</sup> SIGCSE Technical Symposium on Computer Science Education*, Portland, Oregon, USA, 532-536, ACM Press.

Jones, C. (2004). Test-Driven Development Goes to School. *Journal of Computing Sciences in Colleges*, **20**(1): 220-231.

Kaufmann, R. and Janzen, D. (2003): Implications of Test-Driven Development: A Pilot Study. *Proc. ACM SIGPLAN Conference Object-Oriented Programming Systems, Languages and Applications*, Anaheim, California, USA, 298-299. ACM Press.

Marrero, W. and Settle, A. (2005): Testing First: Emphasizing Testing in Early Programming Courses. *ACM SIGCSE Bulletin*, **37**(3): 4-8.

Muller, M. and Tichy, W. (2001): Case Study: Extreme Programming in a University Environment. *Proc of the 23<sup>rd</sup> International Conference on Software Engineering (ICSE)*: 537-544.

Muller, M and Hagner, O. (2002): Experiment about Test-First Programming. *IEEE Proc, Software*. **149**(5):131-136.

Perkins, D. N., Hancock, C., Hobbs R., Martin, F. and Simmons R. (1989): Conditions of Learning in Novice Programmers. E. Soloway and J. C. Spohrer, (eds.), *Studying the Novice Programmer*, 261-279.

Perkins, D.N. and Martin, F. (1986): Fragile knowledge and neglected strategies in novice programmers. E. Soloway and S. Iyengar (eds.), *Empirical studies of programmers, First Workshop*, Norwood, NJ: Ablex, 213–229.

- Schön, D. (1983): *The Reflecting Practitioner: How Professionals Think In Action*. London, Temple Smith.
- Stemler, S. (2001): An Overview of Content Analysis Practical Assessment, Research & Evaluation, **7**(17). <http://PAREonline.net/getvn.asp?v=7&n=17>. Accessed 06 Nov 2010.

## Appendix A

### The questionnaire

(Note: The spaces provided for student responses has been removed)

1. Did you run any of the unit tests when completing the Marked Exercises? **YES | NO**
  - A. If not, why not?
  - B. If you did how many unit tests did you run?  
**ALL|MOST|A FEW**
2. Did you read the related unit test(s) when preparing to answer a marked exercise question? **YES | NO**
3. Did the related unit tests help you to decide what the code you wrote needed to do? **YES | NO**
4. Were unit tests helpful when you needed to debug code? **YES | NO**
5. Did using the unit tests help you to feel confident that your code was correct and complete? **YES | NO**
6. When did you run the unit test methods? (circle the most appropriate option)
  - A. As soon as the functionality the method tests is implemented
  - B. On completion of the question
  - C. On completion of a set of related questions
  - D. On completion of the marked exercise (assignment)
7. How often did you run the unit tests for a marked exercise? (circle the most appropriate option)
  - A. 0-5 times
  - B. 5-10 times
  - C. More than 10 times
8. Identify a situation when you found a unit test useful. Write a brief account of that situation in the box below.
9. Is there anything you did not like or find helpful about using the unit tests?
10. Is there anything else that you would like to mention about the use of unit tests in your Programming 1 assignments?

# Work-Integrated Learning in ICT Degrees

Chris Pilgrim

Faculty of Information and Communication Technologies

Swinburne University of Technology

PO Box 218, Hawthorn 3122, Victoria

cpilgrim@swin.edu.au

## Abstract

Work-Integrated Learning (WIL) is acknowledged by universities, professional societies, government and industry as a valuable model of learning that provides significant benefits to students. Despite this agreement there remain some differences of opinion between universities, industry and the professional society regarding implementation of WIL. This paper reports on the outcomes of a university forum that discussed various aspects of WIL programs. A recommendation for a stakeholder approach to WIL is proposed that would ensure that all motivations and expectations are made explicit and that the primacy of student learning outcomes is maintained.

*Keywords:* Work Integrated Learning

## 1 Introduction

Many Australian universities have a strong history of engagement with the ICT industry regarding the design and implementation of their degree programs. Modes of interaction include industry-based learning, industry-linked projects and the use of industry examples in case studies and scenarios. Authentic engagement with industry has the potential to bring significant benefits to all stakeholders including students, the university, industry and the nation in general.

The term ‘Work-Integrated Learning’ (WIL) is now frequently used to describe the various forms of learning experiences that aim to develop student’s professional capabilities and knowledge of the workplace. The recent Australian Learning and Teaching Council (ALTC) national scoping study for Work Integrated Learning (WIL) (Patrick et al, 2009) reported on the broad and growing picture of WIL across Australia and ways of improving student learning experiences in relation to WIL. The project regarded the term ‘Work Integrated Learning’ as an umbrella term that included a “range of approaches and strategies that integrate theory with the practice of work within a purposefully designed curriculum”.

It is commonly acknowledged that WIL brings mutual benefits for students, universities and industry. Bates et al (2007) suggest that WIL provides students with an opportunity to test the theoretical knowledge learnt at university and to put it into action in the complex and pressurized environment of the real professional world.

For graduates, previous industry experience provides significant salary advantages with the Graduate Careers Council reporting that computer science graduates (collective category for IT graduates) with previous full-time employment experience earned a median starting salary AUD\$13,000 higher than those without previous experience (GCA, 2010). In a survey of Australian universities, Smith et al (2008) found that many lecturers identified Industry-Based Learning as the single best feature of their degrees, primarily because it realized the alignment of their programs to industry. Other more recent surveys found that ICT graduates in the workplace strongly believe that there needs to be some form of work-integrated learning to address both what was missing from their courses and what needed improvement, and that ICT employers believe that students need more work placements to gain industry experience (Koppi and Naghdy, 2009).

The Federal Government also acknowledges the value of WIL with the then Minister for Employment Participation (O’Connor, 2008) noting the “immeasurable” value of integrating real work experience into academic programs including recognized benefits to students: “By integrating practice and theory, students develop those important ‘softer’ skills greatly valued by employers, such as team work, self-management and initiative”, and other stakeholders: “Students are able to make an immediate and meaningful contribution to increasing productivity and prosperity—for industries, businesses and the nation as a whole.”

Despite the many examples of good practices and the significant potential benefits of WIL, the ALTC sponsored ICT Scoping Study (Koppi and Naghdy, 2009) found that ICT graduates and ICT employers identified common deficiencies in the workplace readiness of new graduates particularly in relation to the development of essential generic skills such as interpersonal and professional communications, business awareness and problem-solving abilities. The ICT Scoping Study suggested that these deficiencies could be addressed in large by appropriate workplace experience with a formal recommendation that universities and industry leaders to “investigate the possibilities for greater work-integrated learning by all students of ICT, and develop a scheme that has local and national applicability”. This paper reports on progress on a component of an Australian Learning and Teaching Council project (Ogunbona, 2009) that aims to address several of the recommendations from the 2008 ICT Scoping Study (Koppi and Naghdy, 2009). In particular, the paper focuses on the recommendation regarding the learning experiences that will enhance the work-readiness of graduates.

## 2 Context and Consultation Process

The inaugural Australian Council of Deans of ICT (ACDICT) Learning and Teaching Network Forum was held in July 2010 with the aim of disseminating new developments and sharing good practice. The forum was attended by over 20 Associate Deans Learning and Teaching or their equivalents in the ICT disciplines of the ACDICT member universities. One of the key sessions at the forum related to Work Integrated Learning in ICT degrees. The purpose of the WIL session was to consult with the forum participants through small group and plenary processes on a range of issues relating to WIL in order to identify any common position that universities in general have on WIL. The WIL issues covered at the forum were derived from a recent newspaper article entitled “Sector split over on-the-job year for IT students” (Mather, 2010) that reported on a proposal from the Australian Computer Society for compulsory work placements for information technology degrees. The particular issues covered during the session related to the value of WIL, compulsory WIL, the tension between the needs of industry versus education and finally models of WIL. The session included some group work in order to seek input on each issue. The subsequent discussions were recorded by a scribe and audio recording. Each issue is presented in the following sections.

## 3 The Value of WIL

The newspaper article (Mather, 2010) primarily focused on the value that WIL brings to industry with comments such as “Industry is looking for graduates who are capable of being productive in the shortest time possible” and “Graduates need to be more work ready, particularly as the information and communications technology sector braces for a flurry of activity associated with the roll-out of the national broadband network and e-health initiatives”. From an industry perspective there appears a strong consensus that traditional forms of WIL such as Industry-Based Learning (IBL) placements and internships are highly desirable and effective in developing the employability attributes of graduates.

The representatives at the ACDICT workshop unanimously agreed that from a University perspective WIL is beneficial in developing certain ‘professional attributes’ in students with many universities regarding WIL and IBL as key features of their degrees. There were some concerns raised during the ACDICT workshop regarding the need for some ‘hard evidence’ of the value of WIL in the context of student outcomes beyond the direct employment benefits and also that such employment benefits should not necessarily be seen as the primary goal of university education. This broader view of the value of WIL within higher education was raised in the newspaper article (Mather, 2010) who quoted the ACS chief executive, Bruce Lakin commenting that “Among the 17 deans.... the majority support it (WIL) but some take a view that says, “We’re not in the business of teaching vocational skills, we’re about teaching the philosophy of learning, the meaning of life””. This particular issue was highlighted during the ACDICT workshop with some participants taking the view that whilst WIL does provide a direct employment advantage, employability comes from having something

‘interesting’ to offer, for instance a research portfolio or an international experience. Those participants commenting on this issue made the point that students should be presented with a range of options of alternative learning experiences.

## 4 Compulsory WIL

The newspaper article (Mather, 2010) was written in response to the proposition from the Australian Computer Society (ACS) that information technology degrees should be extended by up to 12 months to include a compulsory work placement period. The article noted that industry placements are mandated in other professions such as engineering, teaching and nursing, however there was no such requirement for IT degrees. The article indicated that the motivation behind the ACS position that industry placements should be compulsory was linked to the need for graduates to be productive in the shortest time possible.

The ACDICT workshop participants unanimously rejected the call for work placements to become a compulsory component of IT degrees. The key concerns raised during the workshop related to equity and access to placements as well as issues relating to student diversity and personal life preferences. It was also noted that the ACS position was based on a view that WIL experiences were limited only to models that involved workplace experiences such as industry-based learning or internship programs. Industry acceptance of the value of other models of WIL will be addressed in a later section of this paper.

The equity and access issues raised during the ACDICT workshop included the visa limitations that are imposed on international students that generally precluded them from participating in full-time paid work including industry-based learning. In addition it was noted that many regional and rural universities and/or university campuses would have difficulty in sourcing appropriate industry placements within their immediate location. The ALTC national scoping study for Work Integrated Learning (Patrick et al, 2009) also identified the issue of ensuring equity and access noting that not all students have easy or equal access to WIL experiences with access dependant on university and/or degree program priorities. These two factors present a significant barrier to the implementation of a mandatory work experience requirement for IT degrees with any solution requiring a substantial package of government and industry funding and support to ensure that no student or university is disadvantaged.

The issues relating to student diversity and interests that were raised during the ACDICT workshop present a broader range of challenges for any compulsory work placement initiatives. The first of these issues related to generational change, in particular the different attitudes and values that Generation Y students bring to all aspects of life including education. Students born in the 1980’s and 1990’s are now known as the Gen Y students. These Gen Y students are the target age cohort that are involved university workplace learning experiences such as IBL and internships. The key behavioral attributes that characterizes Generation Y include their cynicism towards what they are told by older generations and their

rejection of the things that their parents took for granted including lifelong relationships, continual employment and home ownership (Nimon, 2007). Generation Y students also display different approaches to learning with a desire for “immediacy” (Nimon, 2007) and a strong preference for flexibility of learning options including modes that meet their individual preferences and needs. In addition, Generation Y students value learning that occurs outside the classroom and seek to have this recognized (Skene, 2007).

The ACDICT workshop participants indicated that the vast majority of undergraduate students have significant part-time work obligations. This view is supported by research that found that half of all Gen Ys in full-time study also have paid jobs compared to the late 1980s when around two-thirds of full-time students devoted all their attention to their studies and did not undertake paid work (AMP-NATSEM 2007). The workshop participants indicated that the value of part-time work is critical to Gen Y students as it funds their technology lifestyle requirements within the flexible work-life balance that Gen Y desire. Many students are therefore reluctant to give up their part-time work for a work-placement or internship as they may not be able to resume their part-time positions afterwards and it has been suggested that some are reluctant to reduce flexibility in their life styles.

The ACDICT workshop participants also suggested that students may be acquiring the desired soft-skills and the professional attributes through their part-time work hence are achieving some of the learning outcomes of a work placement or internship. It was suggested that some form of formal credentialisation of the part-time work might be considered in order to recognise the value that this work brings.

The final issue that was raised at the ACDICT workshop regarding the proposal for compulsory work-placements related to the diversity of the student cohort, particularly in relation to the academic and personal capabilities of students and their suitability for a traditional work placement. The workshop participants indicated that many universities have eligibility criteria for work placements or internships that require a certain level of academic achievement (e.g. a Credit average). In addition many universities require the students to undergo a vetting process that may include an interview to ensure that the student has the required interpersonal and communication skills demanded by employers. These criteria are applied to provide some assurance that the placement or internship will be successful from the perspectives of all stakeholders and are based on many years of experience in the cases of some universities.

There was a concern expressed by some participants that the relationship with industry partners involved in work placements or internship programs may be put at risk if universities were compelled to involve ‘pass-level’ students as they might place an unreasonable burden on the industry partners who would need to introduce additional support and supervisory procedures to manage students who may not have the yet developed the capability to work without direct supervision and support.

The newspaper article (Mather, 2010) also noted that the achievement level of the student has to be a consideration in any work placement process but used an

example from the opposite end of the spectrum “If you’ve got some really technically smart person, you are going to upset them if you stick them on a help desk for a month”.

## 5 Training vs Education

The newspaper article (Mather, 2010) mentioned a range of stakeholders including industry, government, universities, professional societies and finally the student themselves. The article identified the benefits that some stakeholders were aiming to achieve through participation in work-placements or internship programs, in particular the benefits for industry in having access to a supply of graduates who are “capable of being productive in the shortest time possible”. This was contrasted with a concern expressed by universities who aimed to achieve a balance between teaching theory and vocational practice.

The 2008 ICT Scoping Study accurately described the tension that currently exists between universities and industry regarding the design of curriculum for ICT degree programs. Many universities have focused on the development of the foundations that will enable graduates to acquire ICT skills relevant for their work with the aim of developing a graduate with life-long learning skills. This approach is contrary to the “application driven outcome-based curriculum” (Shoikova & Dwishev, 2004) in which there is a focus on training in the contemporary tools and techniques used in the corporate and industry environments. The ICT Scoping Study (Koppi and Naghdy, 2009) indicated that an approach to curriculum and learning that focuses on fundamentals rather than training has resulted in a situation where employers believe that “universities are not interested in meeting industry requirements” and subsequently universities conclude that industry is “remote from and skeptical about university education”.

The ACDICT workshop participants recounted stories of the different motivations of organizations involved in work placements and internships with examples ranging from those organizations who work as genuine partners with the student’s interest as a priority to some organizations that have been banned from any future involvement in WIL. WIL need not preclude tangible task-related outcomes for organizations but the focus must remain on providing the student with a holistic learning experience with intentional and agreed learning outcomes.

## 6 Models of WIL

The ALTC national scoping study for Work Integrated Learning (Patrick et al, 2009) reported on the broad and growing picture of WIL across Australia and ways of improving student learning experiences in relation to WIL. The project regarded WIL as an umbrella term that included a “range of approaches and strategies that integrate theory with the practice of work within a purposefully designed curriculum”.

The number of programs that provide students in various professional fields with practical experience in the workplace have proliferated in recent times (Bates et al, 2007). This increase in interest in WIL has seen the development and adoption of a range of models of WIL extending from the traditional work experience placement to new virtual or simulated WIL experiences. In his

speech to the World Association for Cooperative Education Asia Pacific Conference (O'Connor, 2008), the Hon Brendan O'Connor representing the Minister for Education recognized that WIL now comes in many different forms including "research, internships, studying abroad, student teaching, clinical rotations, community service or volunteer work, industry attachments or placements, sandwich programs, and professional work placements". These models of WIL may be classified on a continuum from the traditional external, industry-based WIL experiences such as work experience placements and internships to internal, university-based experiences such as project work, case studies and experiential learning opportunities.

The newspaper article (Mather, 2010) discussed at the ACDICT workshop interchangeably used the terms 'work placement', 'industry placements', 'workplace integrated learning' and 'work integrated learning' to describe a singular model of WIL that involves students working at the premise of the host organisation on a full-time basis for an extended period of usually between 3 to 12 months. This traditional view of WIL is contrasted by the University community and the ALTC who see WIL as "occurring in a workplace, in the community, within the university, and real or simulated, as long as the experience is authentic, relevant and meaningfully assessed and evaluated" (Boud and Symes, 2000).

The ACDICT workshop participants spoke with enthusiasm of the different models of WIL including simulated work environments, industry linked projects and part-time variations of WIL that provide the flexibility that meet the diversity of student capabilities and interests, including international and students with significant part-time jobs. As the demand for WIL increases further alternative models are likely to be developed and refined in order to offer students a range of options for gaining workplace experience that suits different student motivations and capabilities and different university resourcing models and priorities.

## 7 Discussion

The plenary discussions at the ACDICT workshop confirmed strong support for Work Integrated Learning as a desirable feature of undergraduate IT degree programs. The benefits that WIL brings to student learning include providing an opportunity to develop and practice soft-skills and to expand the student's understanding of workplace dynamics and the IT industry. However, the workshop noted that the key value of WIL is the opportunity for students to improve their understanding of professional responsibility. Importantly, it is noted that these benefits relate primarily to the student rather than the other stakeholders involved in WIL.

University support for WIL was contingent upon the recognition of the multiplicity of models of WIL including traditional work experience placements and internships as well as innovative models such as industry-linked project work, case studies and experiential learning opportunities.

The workshop participants agreed that the concept of compulsory work experience placements as proposed in the newspaper article (Mather, 2010) was not viable

given the diversity of universities and students. There was some level of agreement that undergraduate IT courses should provide a range of WIL options for students with support for the idea that WIL experiences should be provided early in the curriculum in order to develop student's understanding of the industry that they will most likely work in.

Implementation of a range of WIL experiences throughout the curriculum in an undergraduate IT degree would require a different level of engagement with other stakeholders including industry and professional societies. The ACS Body of Knowledge (BOK) (Gregor et al, 2008) calls for a 'common vision' that can be shared by all stakeholders including industry, government, educators, academic disciplinary bodies, the community, students and professional standards bodies. The ACS BOK encourages a partnership between universities and industry where by curriculum designers in universities should determine what is required of professionals in the workforce when designing programs and industry should be involved in program design through advisory committees.

The approach to curriculum design recommended by the ACS BOK is an appropriate mixture of top-down and bottom-up considerations. The top-down focus should ensure that designers consider the roles that their graduates will undertake after graduation with an emphasis on the development of a professional who should be able to work across the boundaries of traditional disciplines. This is balanced with a bottom-up consideration of ensuring that underlying knowledge, principles and theories are covered rather than the extreme position where programs are designed to address short-lived market trends or skills gaps.

One of the most encouraging aspects to the ACS BOK document is the development of schema that categorizes ICT roles into "Technology Building, Technology Resources, Service Management and Outcomes Management". This development will aid in the partnership approach to curriculum design by establishing a common vocabulary and understanding between the stakeholders.

Taking a stakeholder approach to WIL is critical to its further development and adoption. The ALTC national scoping study on WIL identified a broad range of stakeholders involved in providing or benefiting from WIL experiences, including students, university academic and professional staff, employers, professional associations, and government (Patrick et al, 2009). Employers benefit from participation in WIL through recruitment opportunities, universities benefit through improved student learning, engagement and retention (Patrick et al, 2009) and the government and the wider community benefit through the development of a graduate workforce who can make an "immediate and meaningful contribution to increasing productivity and prosperity" (O'Connor, 2008).

The ALTC WIL Scoping Study (Patrick et al, 2009) also recommended an integrated stakeholder approach to the planning and implementation of WIL that would be based on "formalised relationships and a common understanding of the associated responsibilities and level of commitment required" where there are "clear



agreements and the recognition of needs as well as mutual benefit and costs". Smith et al (2008) link the quality of any Work Integrated Learning program to a "dynamic interplay of stakeholder needs (such as academic disciplines and departments contributing to the curriculum, the expectations of industry and professional associations, and the students).

O'Connor (2008) also mentioned the "critical" requirement for a stakeholder approach in his speech but extends this to the notion of a "genuine partnership" between students, employers and education providers for effective collaborative education programs.

The partnership concept is also suggested by Orrell (2004) who notes that partnerships are a distinguishing feature of any effective work-placement programs. In addition, Orrell indicates that the continuing success of WIL programs also requires identification and attainment of explicit benefits for each partner and "if the benefit fails for any party, the partnership ceases to be effective". Orrell comments that the motivations of host organizations ranged from those who had a 'value-added' ethos in which the placement is evaluated on tangible, short-term returns for the organization, to the 'stakeholder' ethos which emphasizes learning with a long-term view seeking benefits for all parties.

Bates et al (2007) discussed the WIL partnership model by examining the respective responsibilities of each partner. They suggested that WIL is a three-way partnership between the student, workplace, and educational institution with specific responsibilities for each partner. The student must take responsibility for their own learning during a placement, the university has the responsibility for ensuring the WIL curriculum provides students with learning opportunities including a requirement for reflective learning, the academic supervisor has responsibility for mentoring, support and feedback, the workplace organisation has the responsibility for providing a relevant and suitable project for the student to focus on and a suitable induction process for introducing students to the specific workplace.

Given these requirements and the necessity of the management of partnership relationships, Work Integrated Learning should be intentional, organized and real-world rather than opportunistic or contrived. WIL requires a formal educational structure that defines the roles of the student, the teacher/supervisor, the curriculum emphasis and teaching methodologies (Calway and Murphy, 2006).

## 8 Conclusions

Work Integrated Learning brings multiple benefits to student learning including the development of soft-skills, workplace and industry knowledge and professional responsibility. The primacy of the student in WIL is regarded as a central success factor for the implementation of WIL.

The contemporary view of WIL is that it includes a broad range of models including the traditional work experience placements and internships as well as innovative models such as industry-linked projects and experiential learning opportunities.

The proposal for mandatory work experience placements was regarded by participants at the ACDICT workshop as not viable or appropriate given the diversity of universities and students. Instead the focus should be on the key issue of identifying and managing different stakeholder motivations and expectations and the requirement to ensure that WIL is an integral part of the entire undergraduate curriculum.

## 9 References

- AMP.NATSEM (2007). Generation whY?, *AMP. NATSEM Income and Wealth Report*, Issue 17. July.
- Bates A., Bates B., and Bates C., 2007. Preparing students for the professional workplace: who has responsibility for what? *Asia-Pacific Journal of Cooperative Education*, 2007, 8(2), pp 121-129.
- Boud, D. & Symes, C. (2000). Learning for real: work-based education in universities. In C. Symes & J. McIntyre (Eds), *Working Knowledge: the new vocationalism and higher education*, pp.14-29. Buckingham: Open University Press.
- Calway, B and Murphy, G. (2006). Education for Professionals through Work-Integrated Learning, *Proceedings of the Australian Association for Research in Education (AARE) International Education Research Conference*, Adelaide.
- Gregor, S., von Kinsky, B.R., Hart, R., and Wilson, D., (2008). *The ICT Profession and the ICT Body of Knowledge (Vers. 5.0)*, Australian Computer Society, Sydney, Australia.
- Koppi, T. and Naghdy, F., (2009). Managing educational change in the ICT discipline at the tertiary education level.  
[http://www.altc.edu.au/carrick/webdav/site/carricksite/users/siteadmin/public/grants\\_dbi\\_report\\_changeICT\\_uow\\_mar09.pdf](http://www.altc.edu.au/carrick/webdav/site/carricksite/users/siteadmin/public/grants_dbi_report_changeICT_uow_mar09.pdf)
- Mather, J (2010), Sector split over on-the-job year for IT students, *Australian Financial Review*, May 30th 2010.
- Nimon, S. (2007) Generation Y and Higher Education: The Other Y2K, *Journal of Institutional Research*, 13(1), pp24-41.
- O'Connor, B. (2008). Work Integrated Learning (WIL): Transforming Futures Practice... Pedagogy... Partnership, Address to: World Association for Cooperative Education (WACE) Asia Pacific Conference, 1 Oct, 2008  
[http://www.deewr.gov.au/Ministers/OConnor/Media/Speeches/Pages/Article\\_081003\\_124044.aspx](http://www.deewr.gov.au/Ministers/OConnor/Media/Speeches/Pages/Article_081003_124044.aspx)
- Ogunbona, P (2009), ALTC Project PP9-1274, "Addressing ICT curriculum recommendations from surveys of academics, workplace graduates and employers". <http://www.altc.edu.au/project-addressing-ict-curriculum-recommendations-uow-2009>.
- Orrell, J. (2004). Work-integrated Learning Programmes: Management and Educational Quality, *Proceedings of the Australian Universities Quality Forum*.
- Patrick, C., Peach, D., Pocknee, C., Webb, F., Fletcher, M., Pretto, G., (2008). The WIL [Work Integrated Learning] report: A national scoping study [Australian Learning and Teaching Council (ALTC) Final report].

- Shoikova, E. & Dwishev, V. (2004). University - Industry network, *Proceedings of the 27th Int'l Spring Seminar on Electronics Technology IEEE*, pp 510-514
- Skene, J., Cluett, L. & Hogan, J. (2007). Engaging Gen Y students at university: What web tools do they have, how do they use them and what do they want? *Proceedings of the 10th Pacific Rim First Year in Higher Education, Regenerate, Engage, Experiment*.
- Smith, R., Mackay, D., Holt D. & Challis, D. (2008). Expanding the realm of best practices in cooperative industry-based learning in information systems and information technology: an inter-institutional investigation in Australian higher education, *Asia-Pacific Journal of Cooperative Education*, 9(1), 73-80

# How Active are Students in Online Discussion Forums?

Dip Nandi, Margaret Hamilton, James Harland and Geoff Warburton

School of Computer Science and Information Technology

RMIT University

Melbourne 3000, Australia

[dip.nandi, Margaret.hamilton, james.harland]@rmit.edu.au

geoffw@cs.rmit.edu.au

## Abstract

The role of discussion forums is an essential part of online courses in tertiary education. Activities in discussion forums help learners to share and gain knowledge from each other. In fully online courses, discussion forums are often the only medium of interaction. However, merely setting up discussion forums does not ensure that learners interact with each other actively and investigation into the type of participation is required to ensure quality participation. This paper provides a general overview of how fully online students participate in discussion forums and the correlation between their activity online and achievement in terms of grades. The main benefit of this research is that it provides a benchmark for the trend of participation expected of the fully online introductory information technology and programming students. Investigating the participation and the factors behind online behaviour can provide guidelines for continual development of online learning systems.

The results of the data analysis reveal that a high number of students are not accessing or posting in the discussion board. Results also show that there is a correlation between activity of students' in online forums and the grades they achieve. Discussion about the findings of data analysis and the lessons learned from this research are presented in this paper.

**Keywords:** Online learning system, online activity, assessment, diversity, asynchronous discussion forums.

## 1 Introduction

Since the introduction of internet enabled online learning, discussion forums have been used to ensure interaction between learners and instructors (Sharples, 2000; Farmer 2004). Currently, a major focus has been put onto the better use of the technology to support online learning in particular with the introduction of Web 2.0 technologies (Thompson, 2007). But the way in which online interaction/participation can be designed has yet to be adequately investigated.

Our overall research aim is to investigate the factors that lead to effective online interaction in fully online introductory Computer Science and Information Technology (IT) courses and propose a framework with

design principles for online interaction. The first step to achieve this is to measure how active students are in online discussion forums and the correlation between this activity and the overall marks obtained in the subject if there are any. This is the first step in a longer process to determine the appropriate pedagogical issues and design parameters for fully online courses.

This paper draws from existing literature and presents the findings of quantitative data analysis regarding the activity of online students in two fully online introductory Information Technology (IT) and Programming courses and the correlation between online activity and marks achieved. The data for this research was collected throughout a study period in 2009.

In particular this paper draws from existing literature about online learning and use of discussion forums, presents and discusses the finding of data analysis to answer the following questions;

- (1) *"How active are students in online discussion forums?"*
- (2) *"Does the online discussion forum activity have an effect on the marks obtained?"*

## 2 Literature Review

The "learning" context has changed significantly over the years and the emphasis is nowadays on learner-centeredness and peer-based activities. The advancement of technology and learners' advanced computer skills has made it possible for online learning to develop quickly. Interactions between teachers and learners are now more often happening online (Sheard, Ceddia, Hurst and Tuovinen, 2003). Online learning increases the opportunities for learner participation and enhances the participation of learners who may feel more inhibited to engage in discussions in a traditional classroom setting (Dengler, 2008). This has prompted an increase in the amount of research being performed on online learning environments.

### 2.1 Online Learning and Interactivity

Online learning systems have been described as web based learning environments consisting of digitally formatted content resources via the use of the World Wide Web (Zhu and McKnight, 2001). The online learning system is regarded as a communication device to provide communication link between the instructor and students where they can actively interact (Chang and Fisher, 2001; Piguet and Peraya, 2000).

"Interaction" has been recognized as the most significant attribute in any online system or course (Maor and Volet, 2007; Al-Mahmood and McLoughlin, 2004;

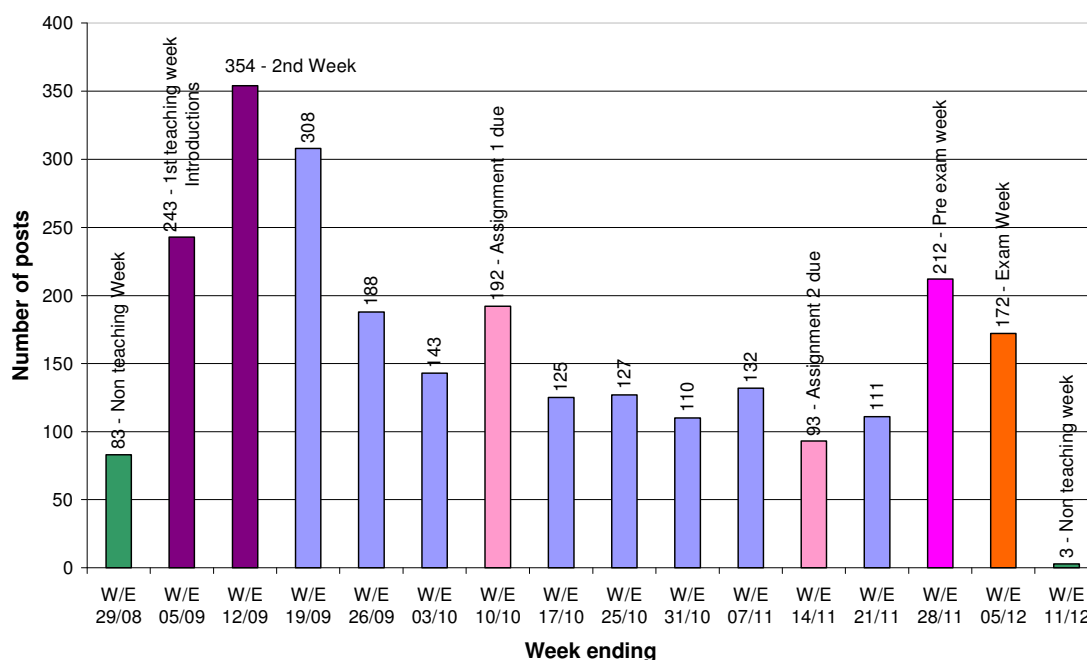


Figure 1: Posts by week (Intro to IT)

Sharples, 2000). Both the conversation theory of learning (Pask, 1975) and social constructive learning theory of learning with technology (Brown and Campione, 1996) emphasize the fact that learning, to be successful, requires continuous conversation and interaction, not just between teacher and learner, but also amongst the learners. Also learners need time to act and reflect. Consequently, educators should consider interactivity when designing online learning strategies (Maor and Volet, 2007).

The challenge of teaching with technology is to create a learning environment that allows and supports the full range of learning strategies. Al-Mahmood and McLoughlin (2004) state that teaching is a sort of conversation and the importance should be on effective interaction with students through technology. Online teaching is not about broadcasting but two-way communications are required to make it blended within a classroom learning environment.

To ensure that a classroom type interaction between students and teachers takes place in online learning environment, asynchronous and synchronous discussion forums can be used. Discussion forums are an effective way of exchanging ideas and sharing knowledge among learners and instructors. The asynchronous discussion forums are preferred over synchronous discussion forums because of the nature of communication. Synchronous discussions can be arranged in online courses through the use of tools like “Blackboard Chat”, “Elluminate”, “Skype” etc. However in synchronous discussion forums, the communication is instant and it is not always possible for all students to participate in the forum because of time commitments. Hence synchronous discussions in online courses attract low student participation.

The asynchronous nature of discussion in a web-based course incorporates interactive communication that is unique and different from the face-to-face classroom type instruction. It also eliminates the constraints of time and space. This type of system facilitates the requirements of

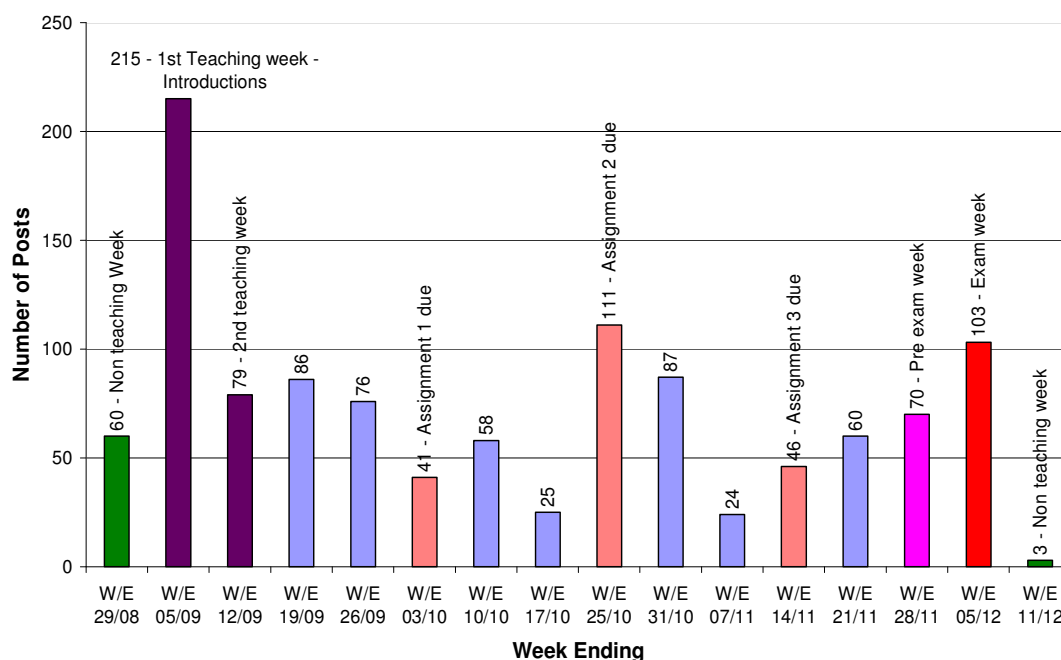
people with family and work responsibilities, transport problems, physical disabilities etc to have quality education online (Sher, 2009).

## 2.2 Discussion Forums and Participation

The discussion forum is the ubiquitous communication tool within online learning environments and hence significantly shapes the kind of communication that takes place. The asynchronous nature of discussion in a web-based course incorporates interactive communication that is unique and different from face-to-face classroom type instruction and also eliminates the constraints of time and space.

Discussion forums have frequently been used successfully as communication tools in online learning environments to facilitate interaction between students to share knowledge (Rovai, 2002) (Bradshaw and Hinton, 2004; Berner, 2003). Discussion forums also provide an effective opportunity to exchange ideas and share knowledge amongst learners and instructors (Tallent-Runnels et al., 2006; Levine, 2007). Because of their potential benefit, online discussion forums are becoming a common feature even in face-to-face courses as they allow students and instructors to communicate with each other regardless of time and space.

In online courses, students are encouraged to participate in discussion forums to demonstrate their capability to carry on a discussion by sharing their knowledge of the topic. The use and benefits of these forums vary immensely, covering topics as diverse as learner or teacher-led discussions, debates, collaboration around set tasks or projects, or set activities (Berner, 2003; Rovai, 2002; Rovai and Jordan, 2004; Bradshaw and Hinton, 2004; Gerbic, 2006). Forums are also used for posting comments on readings, prior to submitting a formal review of the reading, as a memory trigger (looking back at old discussions), to find role models, to get some form of immediate peer review, or for making



**Figure 2: Posts by week (Intro to Programming)**

connections with each other. These activities allow learners to think critically, discuss the topic intimately and learn from others.

Broadly speaking, the above mentioned benefits can be termed as quality online engagement, but on the other hand, research has shown that participation in online discussion forums is not always equal (Poole, 2000; Guzdial and Carroll, 2002; Leh, 2002; Russo and Benson, 2005; Salmon, 2003). There are three main levels of participation (Salmon, 2003):

- Firstly, some are “lurkers” i.e. who just read the messages and do not participate. They may learn by reading the posts and incorporating the ideas into their assignments (Guzdial and Carroll, 2002);
- Secondly, some learners read the messages and treat them as a notice board posting their own position having limited interactivity;
- Thirdly, the participation is interactive and to its full potential (Ho, 2002).

The above mentioned models of students’ participation in online discussion forums provide an outlook for the expected behaviour of online students and they need to be investigated with fully online introductory Computer Science/IT students.

### 2.3 Research Questions

From the above discussion we can summarize that there is a definite need to investigate the current activity of students in online forums. This leads us to our first research question which is

(1) “How active are students in online discussion forums?”

The above mentioned research question also draws us to investigate whether there is a correlation between the level of activity in online forums and the grades they achieve in that course. So our second research question is

(2) “How does the level of activity in online forums affect the grades students achieve in the online course?”

Investigating the above mentioned research questions will provide us with a clear picture of the diverse styles of leaning of online students. It will identify who the “lurkers” are and who are interacting actively whether the particular behaviour has any impact on grades or not. This investigation will enable us to fulfil our overall research aim of identifying the factors that lead to effective interaction in fully online courses.

### 3 Methodology

We have mentioned previously that our overall research aim is to investigate the factors that lead to effective online interaction in fully online introductory Computer Science/IT courses. A step towards achieving that research goal is to find out the distribution of activity of online students throughout a study period and the correlation between this activity and grades achieved.

Hence to conduct the research we have chosen two fully online introductory Computer Science/IT courses. One of them is an Introduction to Programming course and the other is an Introduction to IT course.

The Introduction to IT course covers general IT concepts e.g. computer fundamentals, operating systems and applications, internet, spreadsheets etc. This course has students from various degrees including Bachelor of Technology, Bachelor of Business IT, Bachelor of Indigenous Studies, and Bachelor of Accountancy.

The Introduction to Programming course covers introductory concepts of programming through the use of two programming languages; Alice and Java. Students enrolled in this course are only from the Bachelor of Technology degree.

Both the courses are conducted in a fully online environment and there are absolutely no face-to-face

classes. We collected data from Blackboard, the University's Learning Management System.

We collected data throughout the study period which began in September 2009 and ended in November 2009. Both the courses have online discussion forums where students are encouraged to participate and interact with each other.

To determine the distribution of student activity we recorded the number of accesses and posts by the students throughout the study period. At the end of the study period, assignment and final examination results for each student were recorded. Using these assessment results we investigated if there is a correlation between the level of student activity in discussion forums and the grades they achieved.

#### 4 Data Analysis and Findings

The introductory IT course had 299 students enrolled whereas there were 346 enrolments in the introductory programming course. The students were located in many parts of Australia and also different parts of the world while studying the courses. The age of the students ranged in between 20 to 70 which represent diversity in maturity and motivations of the students.

The introduction to IT course had two extra tutors apart from the instructor and the discussion board participation in the course was assessed where 10 percent of the total mark was allocated for participation. Whereas the introduction to programming course was conducted by the instructor only with no tutor support and the participation was not assessed.

Figure 1 provides a broad-spectrum overview of the number of posts each week by the students in the introduction to IT course. It indicates that there are a high number of posts by students during the first couple of teaching weeks where students may have tried to get accustomed of the course details. The number of posts gradually decreased after the initial teaching weeks and again rises during weeks when assignments and examinations are due.

Figure 2 provides an overview for the number of posts each week by the students in the introduction to programming course. It initially provides a similar scenario like figure 1 where there are soaring numbers of posts in the first couple of weeks. However the number of posts gradually declined and was quite low before the first assignment was due, unlike the trends in the introductory IT course. The trend remained same throughout the study period as assignment weeks failed to attract high number of posts as seen in the introductory IT course.

Figures 3 and 4 provide the number of posts by the students in the two courses respectively. Both the figures reveal there are a high number of students who have zero posts (174 in Introduction to IT and 218 in Introduction to Programming) that is; they did not post at all during the study period. Although the graphs were prepared with different distributions of number of students, the trend of posts is very similar.

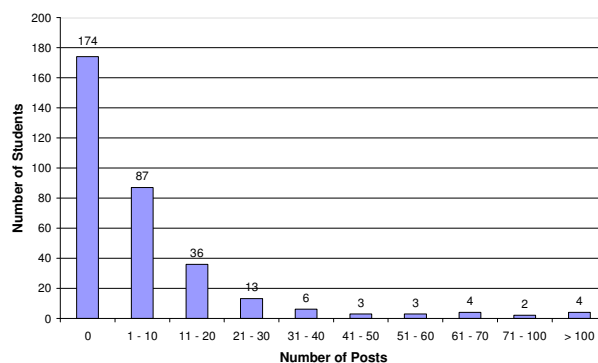


Figure 3: Posts by students (Intro to IT)

When ascertaining the activity of the students in discussion boards, using only the actual number of posts may not provide the overall picture.

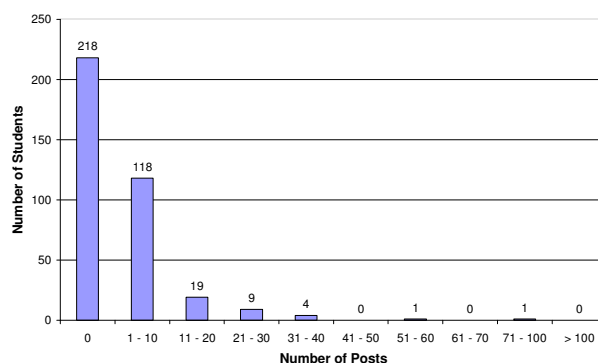


Figure 4: Posts by students (Intro to Programming)

As well as students that have posted in the forum, there may also be a significant number of "lurkers" present (Salmon, 2003; Guzdial and Carroll, 2002) in both the courses. Hence we have graphed the number of accesses by students in discussion forums against the number of students in figures 5 and 6 for Introduction to IT and Introduction to Programming courses respectively.

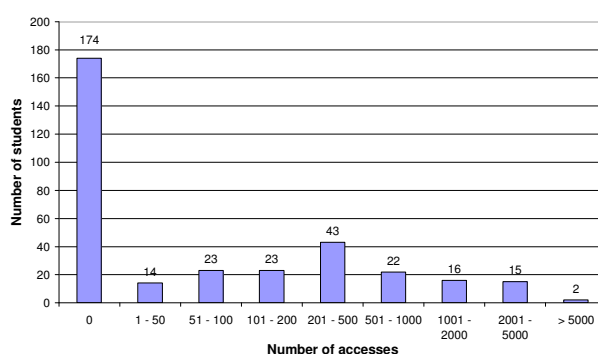
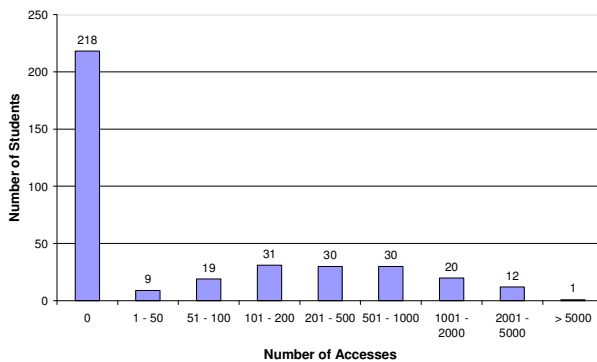


Figure 5: Accesses by students (Intro to IT)

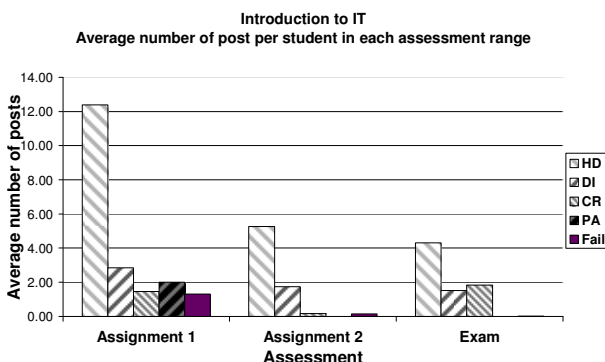
Comparing figures 5 and 6 indicates that there is a similarity between the trends of student accesses in both subjects; also there are a high number of students who did not access the discussion board. Not accessing and not posting might be a bit surprising for the introductory IT course as the participation in the discussion board was assessed and ten marks were allocated for posting on the discussion board.



**Figure 6: Accesses by students (Intro to Programming)**

Comparing the posts and access data in the same course reveals a coincidence. Almost 58 percent (174/299\*100) of students did not access and post in the Introduction to IT subject. This rate of “0” accessing and posting is around 63 percent (218/346\*100) in the Introduction to Programming subject. Consequently only around 40 percent of the total students in both the subjects accessed and posted in the online discussion forum.

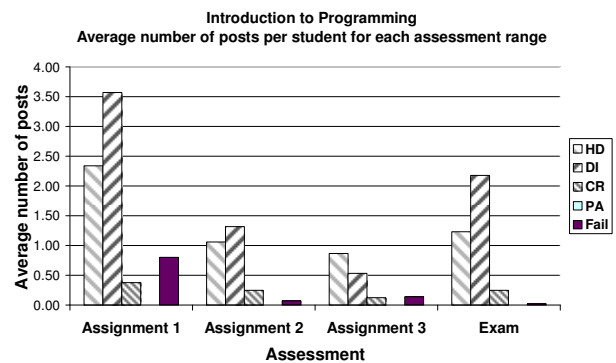
Further investigating the data closely demonstrated that students who accessed at least once also posted in the discussion forum.



**Figure 7: Grades achieved (Intro to IT)**

Figure 7 and 8 provides a general overview of the average number of student postings in the discussion board against their performance in each of the assessments in the study period. This allowed us to investigate the second research question for this paper. With this question we investigated if there is a trend between the level of activity in online forums and the grades students achieve in each course. In general, most of the students with higher number of posts achieved Distinction or High Distinction in the assignments and final assessment. The grades of High Distinction (HD) refers to marks which are in between 80-100, whereas Distinction (DI) refers to 70-79, Credit refers to 60-69, Pass (PA) refers to 50-59 and Fail refers to the of 0-49.

There is a definite correlation as most of the students who posted for a few number of times either failed or just passed the course. This establishes the fact that high achieving students participated in the online discussion forum more actively than other students did.



**Figure 8: Grades achieved (Intro to Programming)**

## 5 Discussion

The data analysis and findings section has provided a general overview of the activity of the students in the two online courses. It also shows the number of students posting and accessing the forums over the study period. There are differences in the number of accesses and posts between the two courses and we consider various possibilities below, in terms of the content, the students themselves, the instructors and assessment.

### 5.1 Impact of Content

There is a difference in the overall number of posts by students throughout the semester in the two courses which we now consider. Although the Introduction to Programming course had more enrolments (346) than the Introduction to IT course (299), the overall rate of posting was higher in the introduction to IT course with around 41 percent. This rate was around 36 percent in the Introduction to Programming course. This disparity could be explained by the dissimilarity in the content of the courses plus the assessment given for participating.

The content for the programming course is more prescriptive. As the course content is algorithmic and more narrowly focused, the opportunity for direct discussion and asking questions is limited. It was noticeable from the observation that, often a single solution by a student to a problem raised by another student or the instructor had ended the discussion at that point. The same situation applied to assignments also. Once the solution is obtained, there was very little discussion to follow which may explain the reason for generally low number of posts during the weeks when assignments are due.

On the contrary, the introduction to IT course covers the basic topics from general information technology covering a vast area from both hardware and software. Often there was a lot to discuss about these topics from different angles. While discussing online, students pointed towards examples and real world situations from the past and current use of information technology in their personal and work life which broadens the discussion. This can cause a jump in the number of posts. The identical situation applies for assignment weeks also where students discuss different solutions for the problems in the assignments causing a sharp rise in the number of posts. This situation points towards the fact that the content of the course has a bearing on the overall



number of posts and direction of discussion. In the course like programming, it is often difficult for instructors or tutors also to extend discussions to attract more participation from students.

## 5.2 Diversity of Students

The diversity in participation can be defined by the research carried out by Sheard, Ramakrishna and Miller, (2003) who pointed out that the background, maturity and motivations of learners have an impact on the online participation. As mentioned previously, students from different degrees e.g. Bachelor of Technology, Bachelor of Business IT, Bachelor of Indigenous Studies and Bachelor of Accountancy were all enrolled in the IT course. Many of them were not pursuing direct Information Technology related studies. Hence those students needed to participate more in the discussion forum to get used to the basic IT- related topics taught in the course.

The Introduction to Programming course only had students from Bachelor of Technology degree. Individually there might have been a bit of diversity in the area of previous study or experience in programming; however they all were pursuing the same degree which was typically technology or programming. As a result they were more accustomed to handle the concepts of the programming course and needed less attention and interaction with others.

## 5.3 Difference in Tutor Support

As discussed in section 4 the Introductory IT course had tutor support which was not the case in the introductory programming course. Hence the students had more tutors to ask questions of and receive more feedback from tutors which resulted in higher levels of participation in terms of number. This is one of factors that Gerbic (2006) and Weaver (2005) identified as motivators of online participation of learners. This phenomenon explains that although online learning is more learners centred, there is still the typical traditional glimpse of instructor dependency in the discussion forum. The more feedback students get from the instructors, the more they interact with the instructors and other students and are inspired by the presence of the instructors.

## 5.4 Impact of Assessment

Research suggests that the strongest motivator for participation is to add some form of incentive as learners generally perceive that what is valued is what is assessed (Burkett, Leard and Spector, 2004; Laurillard, 2002; Leh, 2002; Ramsden 2003; Sheard, Ramakrishna and Miller, 2003; Seo, 2007). The phenomenon of lurkers is also most evident in online discussion forums where participation and engagement is not compulsory (Sheard et al., 2003; Sheard, Ramakrishna and Miller, 2003).

Participation in the discussion forum in the introductory IT course was mandatory as it was assessed and worth 10 percent of the final mark. That could be one of reasons for the higher number of posts. Students had to post to get the marks and so it was highly valued by the students. The lower rate of participation in the introductory programming course could be explained by

the fact that the forum participation was not assessed and students only posted when they were really in need of some assistance in solving problems.

## 5.5 Level of Activity and Achievement

It is important to ascertain the trend between activity and achievement to decide whether or not the online learning environment is totally student-centred and whether this isolated environment provides a barrier in achieving good results in online courses. We found there is a trend between student activity in online discussion forums and the grades they achieved in each assessment. This is clearly evident from figures 7 and 8. All the assessment marks of high achieving students were quite consistent with their participation throughout the semester. Generally the number of posts dropped as the study period progressed, but the trend remained the same. The students, who posted more, got higher marks in each assignment and in the exam than others, with this trend being same for both the courses. However, looking at this trend, it can not be concluded that active participation is the only reason behind higher marks in assessments. There may be other factors that may influence the marks of the students and will be further investigated in future research.

## 6 Lessons Learned

Several lessons are learned from this research regarding the general behaviour of the online students and its impact. Some of the lessons learned from this research are discussed in the following sections.

### 6.1 Managing the Content Sequentially

The way course content is managed has an impact on how students' participate in the discussion board. For example, if all the assignments are released during the first weeks of the course, then most of the students may start discussing all the assignments well before the due date of the first assignment. One of the reasons for that can be a "scare" factor regarding the assignments; this factor works in the minds of the students, being online and isolated makes this factor more prominent. They start to consider the assignments as a hurdle and attempt to complete them as soon as possible. In this way, the focus of the students will be on the assignments rather than learning the basic concepts of the subjects. For these courses, all the assignments were released together at the beginning of the study period and this could explain why a lot of postings occur during the first few weeks and once all the assignment problems are clarified, the participation rate decreases. As a result of our study and observation, we believe that it is better to release the assignments periodically. By releasing the assignments periodically, the focus of the students can be diverted towards learning the subject matter sequentially which can provide them with a strong background on the subject material.

### 6.2 Managing the Expectation

Managing the expectation is another lesson that we learned from this research. This expectation can be of two types: The expectations of the instructors and the



expectations of the students. As we discussed earlier, the expectations of the instructors on how the students participate online has to be dependent on the content of the subject. In a course like Introduction to Information technology, where there is a vast opportunity for discussion, the expectation can be around 5-6 posts per week by the students. This number can be a bit too high for courses like Introduction to Programming where there might be fewer prospects for the discussion to broaden.

From the above discussion we can see that the rate of “lurking” could be different for various subjects depending on the number of possibilities to answer, time of posting and release of content. In this research, we found there were no students who accessed but did not post over the study period. However there were students who accessed more, but posted less, at different times, as shown in Figures 3, 4, 5 and 6. In a subject like Introduction to Programming, there might be a lot of “lurkers”, because the solution of a problem might already be there and lot of students might just view it and not post. Hence the expected participation needs to vary depending on the content of the subject.

The expectations of the students are another aspect to consider. There is a tendency that students want the instructors only to answer their questions and get regular feedback. These factors might have been a reason for higher number of posts in the IT course than the programming course. However instructors do need to consider their way of providing feedback and responses to the students’ questions. This role of the instructor contributes towards the ultimate learning i.e. deep learning or surface learning.

### 6.3 Role of the Instructor

The role of the instructor is one aspect that needs to be taken into consideration while looking at the participation of the students. The way instructors moderate discussion forums has an impact on how the students’ participate. The type of moderation has a major impact on the direction and number of student participation. If the instructor directly answers the questions from the students, then the discussion ends there on the spot. Whereas broadening the discussion through hints, clues and directions makes the students participate consistently in a higher rate.

### 6.4 Preference of Students

Students prefer to build a learning community early on in first few weeks. They introduce themselves and sometimes post their personal email addresses so that they can chat about the subject matter informally. Research shows that strong sense of community not only increases persistence of students in online programmes, but also enhances information flow, learning support, group commitment, collaboration, and learning satisfaction (Dede, 1996; Wellman, 1999). Hence students need to be encouraged to follow this practice of building online communities which ultimately leads to effective collaborative learning.

The preference of the students regarding the use of tools for online participation can have an impact on their posting online. This fact came up from close

observation of the course that some students may prefer to use direct synchronous chat or audio tools to ask questions of the instructor and get an instant answer rather than posting on the discussion board and waiting for someone to answer. On the contrary, most students prefer to use the asynchronous discussion board where they can post questions and comments anonymously. Hence student preferences need to be taken into consideration while investigating online activity and participation. This trend of student preference will be further investigated during the qualitative analysis phase of our research and will be reported in future publications.

## 7 Conclusion

This paper has provided a general overview of the activity of students in the online discussion forums in two introductory courses in a fully online learning environment. By acknowledging the importance of interactivity in online discussion forums, this paper has presented the findings from data analysis carried out with the students of two fully online courses. In addition, the possible reasons for the sort of findings and some of the lesson learned through this research have also been discussed. There are certain factors that contribute towards student participation online and this research has contributed towards identifying those factors.

Our overall research goal is to identify the factors that lead to effective online participation in introductory IT/Programming courses. The first part of that process was to investigate the trend of activity of students in these fully online courses and find out if there is a correlation between activity and grades achieved or not. Hence the major focus of this paper was to present the “big picture” showing the general activity of students in online forums.

Results of our data analysis show a high percentage of students do not access the discussion forums or post at all throughout the semester. However the results also show that it is essential to participate consistently to achieve a high grade. As we have seen from this paper, there are many factors that contribute towards students’ active participation online.

The key lessons learned from this research are that managing the course content and expectations have a large impact on how students participate on online discussion forums. This research has presented the expected behaviour of fully online students in discussion forums. The type of moderations carried out by the instructors and the preference of the students also shape the online discussion.

We will take the findings of this research into our next step and carry out a qualitative analysis of the participation of the students and instructors to investigate the type of participation. Qualitative analysis is required to explore student’s and instructor’s perceptions to capture the views and go deep into the thoughts (Lechner, 2001). By combining the findings of the qualitative analysis and the quantitative analysis from this research, we will attempt to develop a comprehensive framework for effective online interaction.

For active and effective discussion forum participation, there needs to be a comprehensively defined framework that can assist the instructors and

students. We intend to develop a comprehensive framework with design principles for online interaction that can act as guidelines to help and encourage them to participate online; and guidelines for instructors on how to set up the online discussion forums for effective learning. The clearer the criteria for interaction in online forums, the more effectively academics will be able to make use of online interactions and discussions as an educational tool.

## 8 References

- Al-Mahmood, R. and McLoughlin, C. (2004): Re-learning through e-learning: Changing conceptions of teaching through online experience. In R. Atkinson, C. McBeath, D. Jonas-Dwyer & R. Phillips (Eds), *Beyond the comfort zone: Proceedings of the 21st ASCILITE Conference* (pp. 37-47). Perth, 5-8 December. <http://www.ascilite.org.au/conferences/perth04/procs/al-mahmood.html> accessed 1/8/09
- Berner, R. T. (2003): The benefits of discussion board discussion in a literature of journalism course, *The Technology Source*, Sep/Oct. <http://ts.mivu.org/default.asp?show=article&id=1036>
- Bradshaw, J. and Hinton, L. (2004): Benefits of an online discussion list in a traditional distance education course. *Turkish Online Journal of Distance Education*, 5(3) <http://tojde.anadolu.edu.tr/tojde15/articles/hinton.htm>
- Brown, A. and Campione J. (1996): Psychological theory and design of innovative learning environments: on procedures principles and systems. In L. Schauble & R. Glaser (Eds.), *Innovations in learning: new environments for education* (pp. 289-325). Mahwah, NJ: Erlbaum.
- Burkett, R. Leard, C and Spector, B (2004): Using an Electronic Bulletin Board in Science Teacher Education: Issues and Trade-offs. *The Journal of Interactive Online Learning*, vol 3, no. 1, pp 1-9.
- Chang, V. and Fisher, D. (2001): The Validation and Application of a New Learning Environment Instrument to Evaluate Online Learning in Higher Education. Paper presented at the *Australian Association for Research in Education*, Fremantle, 2-6 December. Accessed 1/8/09 from <http://www.aare.edu.au/01pap/cha01098.htm>.
- Dede, C. (1996): The evolution of distance education: Emerging technologies and distributed learning. *American Journal of Distance Education*, 10(2), 4-36.
- Dengler, M. (2008): Classroom active learning complemented by an online discussion forum to teach sustainability. *Journal of Geography in Higher Education*, 32 (3): 481-494, 2008
- Farmer, J. (2004): Communication dynamics: Discussion boards, weblogs and the development of communities of inquiry in online learning environments. In R. Atkinson, C. McBeath, D. Jonas-Dwyer & Phillips, R (Eds), *Beyond the comfort zone: Proceedings of the 21st ASCILITE Conference* (pp. 274-283). Perth, 5-8 December. <http://www.ascilite.org.au/conferences/perth04/procs/farmer.html>, accessed 1/8/09
- Gerbic, P. (2006): To post or not to post: Undergraduate student perceptions about participating in online discussions. In *Who's learning? Whose technology? Proceedings ascilite Sydney 2006*. [http://www.ascilite.org.au/conferences/sydney06/proceeding/pdf\\_papers/p124.pdf](http://www.ascilite.org.au/conferences/sydney06/proceeding/pdf_papers/p124.pdf), accessed 1/8/09
- Guzdial, M. and Carroll, K. (2002): Explaining the lack of dialogue in computer-supported collaborative learning. Paper presented at the *Computer Supported Collaborative Learning Conference*, CSCL 2002. <http://newmedia.colorado.edu/cscl/18.html>, accessed 1/8/09
- Ho, S. (2002): Evaluating students' participation in online discussions. Paper presented at the *Australian World Wide Web Conference (AUSWEB)*, Sunshine Coast, Queensland, Australia
- Laurillard, D. (2002): *Rethinking university teaching: a framework for the effective use of learning technologies* (2nd Ed.). London and New York: Routledge.
- Lechner, S. K. (2001): Evaluation of Teaching and Learning Strategies, *Medical Education Online*, 6(4).
- Leh, A. (2002): Action Research on Hybrid Courses and their Online Communities. *Education Media International*, vol 39, no. 1, pp 31-38.
- Levine, S. (2007): The Online Discussion Board. *New Directions for Adult and Continuing Education*, vol 2007, no. 113, pp 67-74.
- Maor, D. and Volet, S. (2007): Interactivity in professional learning: a review of research based studies. *Australasian journal of educational technology*, Vol 23(2) pp 227-247.
- Piguet, A. and Peraya, D. (2000): Creating Web-integrated learning environments: An analysis of WebCT authoring tools in respect to usability. *Australian Journal of Educational Technology*, 16(3), 302-314.
- Pask, G. (1975): Minds and media in education and entertainment: some theoretical comments illustrated by the design and operation of a system exteriorizing and manipulating individual these. In R. Trappl & G. Pask (Eds.), *Progress in cybernetics and systems research* (Vol. 4, pp. 38-50). Washington and London: Hemisphere.
- Poole, D. (2000): Student Participation in a Discussion-Oriented Online Course: A Case Study. *Journal of Research on Computing in Education*, vol 33, no. 2, pp 162-177.
- Ramsden, P. (2003). *Learning to teach in higher education* (2nd Ed.). London: Kogan Page.
- Rovai, P.A., (2002): Building sense of community at a distance, *International Review of Research in Open and Distance Learning*. <http://www.irrodl.org/content/v3.1/rovai.html>
- Russo, T. and Benson, S. (2005): Learning with Invisible Others: Perceptions of Online Presence and their Relationship to Cognitive and Affective Learning.

- Educational Technology & Society*, vol 8, no.1 pp 54-62.
- Salmon, G. (2003): *E-tivities: The key to active online learning*. London: Kogan Page
- Seo, K. (2007): Utilizing Peer Moderating in Online Discussions: Addressing the Controversy between Teacher Moderation and Nonmoderation. *The American Journal of Distance Education*, vol 21, no. 1, pp 21-36.
- Tallent-Runnels, M., Thomas, J., Lan, W., Cooper, S., Ahern, T., Shaw, S. and Liu, X. (2006): Teaching Courses Online: A Review of the Research. *Review of Educational Research*, vol 76, no. 1, pp 93-135.
- Thompson, J. (2007). Is Education 1.0 ready for Web 2.0 students?. *Innovate 3 (4)*. Accessed August 10, 2010 <http://www.innovateonline.info/index.php?view=article&id=393>
- Sher, A., (2009): Assessing the relationship of student-instructor and student-student interaction to student learning and satisfaction in Web-based Online Learning Environment, *Journal of Interactive Online Learning*, Volume 8, Number 2, Summer 2009
- Sharples, M. (2000): The design of personal mobile technologies for lifelong learning. *Computers and Education*, 34 (2000), 177-193. Phillips, R (Eds), *Beyond the comfort zone: Proceedings of the 21st ASCILITE Conference* (pp. 274-283). Perth, 5-8 December. <http://www.ascilite.org.au/conferences/perth04/procs/farmer.html>, accessed 1/8/09
- Sheard, J., Ceddia, J., Hurst, J. and Tuovinen, J. (2003): Inferring Student Learning Behavior from Website Interactions: A Usage Analysis. *Education and Information Technologies* 8:3, 245-266, 2003. 2003 Kluwer Academic Publishers. Manufactured in the Netherlands.
- Sheard, J., Ramakrishnan, S. and Miller J. (2003): Modeling learner and educator interactions in an electronic learning community. *Australian Journal of Educational Technology*, 2003, 19(2), 211-226.
- Weaver, C. (2005): What encourages student participation in online discussions? *Unpublished PhD thesis*, University of Southern Queensland, Towomba, Australia.
- Wellman, B. (1999): The network community: An introduction to networks in the global village. In Wellman, B. (Ed.) *Networks in the Global Village*. (pp.1-48). Boulder, CO: Westview Press.
- Zhu, E., and McKnight, R., (2001): Principles of online design, Accessed July 20 2008 from Florida Gulf Coast University, *Office of Instructional Technology* Web site: <http://www.fgcu.edu/onlinedesign/>



# Helping first year novice programming students PASS

Adrian Devey  
Office of Deputy Vice Chancellor (Education)  
Monash University  
adrian.devey@monash.edu

Angela Carbone  
Faculty of Information Technology  
Monash University  
angela.carbone@monash.edu

## Abstract

In this paper, we report the results of introducing a face-to-face Peer Assisted Study Scheme (PASS) and an electronic Peer Assisted Study Scheme (ePASS) into a first year introductory programming unit, core to four undergraduate IT degrees. PASS is a program of supplementary instruction through which successful senior students facilitate weekly face-to-face study sessions in targeted key first year units. Sessions are open to all students enrolled in the target units and participation in PASS is voluntary. ePASS is a discussion board monitored by PASS Leaders. Results show that although attendance at face-to-face sessions is less than 25%, the students that attended found the sessions very useful. Of those that participated in PASS, over 40% attended to help them achieve an excellent grade, and not purely to obtain a pass grade. It was therefore not surprising that the proportion of regular PASS attendees who failed the target unit was much lower than in the unit overall, while the proportion of regular PASS attendees achieving Distinctions and High Distinctions was higher. ePASS did not meet our objectives either in terms of the volume or nature of use.

**Keywords:** Peer assisted study, first year programming.

## 1 Introduction

Many studies have shown that first year students find computer programming a hard skill to learn (Bonar & Soloway, 1983; Lahtinen, Ala-Mutka, & Jarvinen, 2005; Lewandowski, Gutschow, McCartney, Sanders, & Shinnars-Kennedy, 2005; Lister, Adam, Fitzgerald, Fone, Hamer, Lindholm, McCartney, Mostrom, Sanders, Seppala, Simon, & Tomas, 2004). Although many new IDEs have been developed to help students develop programming skill such as BlueJ (Kölling & Rosenberg, 2001) and Alice (Dann, Cooper, & Pausch, 2006) the reality is that the fail rate at first year is usually greater than 30% (Dehnadi & Bornat, 2006).

For the past three years first year programming at Monash University has experienced a fail rate of between 36-40% each year. Admittedly, the high rate of failure is usually accompanied by a high grades. Robins (2010) provides an explanation of this characteristic bimodal grade distribution.

To help students succeed at university, a range of approaches have been adopted, including active learning such as adapting the Triesman model (Chinn, Martin and Spencer 2007) and contributing student pedagogy (Hamer et al. 2008). Different models of peer-assisted learning have been applied to computer science education, including curricular (Wills & Finkel, 1994; Machanisk, 2007) and extra-curricular activities, such as student mentoring (Boyer et al. 2010; D'Souza et al. 2008; Miller & Kay 2002).

Monash University decided to trial a Peer Assisted Study Scheme (PASS), one model of extra-curricular peer-assisted academic mentoring. PASS targets "high risk" first year units. These are units that have demonstrated their difficulty over time regardless of the staff who teach them or the material used (Arendale, 1993), core first year science, engineering, IT and business units are typical. Subsequently, the Faculty of Information Technology decided to be part of the initial trial to introduce PASS into its first year programming unit.

Students who have succeeded and done well in the target unit (the "PASS Leaders") help junior students make a successful adjustment to studying at university. In the Faculty, PASS used second year high achieving students who have already demonstrated strong competency in the subject to provide additional support for first year computer programming students.

PASS targets "at risk" units rather than "at risk" students, avoiding the stigma of being a remedial program (UMKC, 2010). Attendance is voluntary, but all students in targeted units are encouraged to attend, not just those who are struggling. In addition to assisting students with unit content ("What to learn"), PASS aims to develop the study skills required for success at university ("How to learn"). PASS achieves these objectives through a student-centred and collaborative approach to learning that also generates social benefits for participating students (van der Meer & Scott, 2009).

PASS sessions are regularly scheduled and facilitated by PASS Leaders. Leaders are paid to run the PASS sessions, and they encourage collaborative study techniques specific to the discipline. Students meet on a

weekly basis and engage as a group with the unit material. Typical activities include clarifying lecture notes, discussing key concepts, developing effective study techniques, and working through practice problems as a group. The PASS Leaders also monitor a Moodle discussion board – known as ePASS.

Our research investigates the use and value of the PASS and ePASS services provided to these students. We also examine the type of student likely to engage with the service and why they do so; to pass the unit or as a way to increase their final grade. Our findings will provide lecturers with insights into how students use such resources so that they can better design support services to engage students in learning.

## 2 Previous Work

The work of Tinto, Gardner and Kuh (Kuh, Kinzie, Schuh, & Whitt, 2005; Tinto, 1987) has provided new perspectives on student success at universities. Universities which once measured their prestige from their ability to weed out students, are now measuring success on how they are able to support students to achieve goals.

The PASS Program is based on the Supplemental Instruction model initially developed at University of Missouri, Kansas City (Martin & Blanc, 1981), the theoretical foundations of which have been well described (see for example McGuire, 2006; Crouchman, 2008). In Australia, The University of Wollongong (UoW) hosts the National PASS Office, and the UoW PASS program is cited by Australian University Quality Agency on its Good Practice Database (AUQA, 2010). Within Australia's Group of Eight universities, the Universities of Sydney, Melbourne and Queensland have well established programs, with UNSW introducing the scheme in 2008.

Studies into the effectiveness of PASS in Australia and overseas have found that students need to attend PASS sessions on a regular basis to gain any real benefit (O'Brien, 2006; Murray, 1996; Bidgood, 1994; Kochenour *et al.*, 1997), with regular attendance usually defined as 50% of sessions in a semester. Therefore, one of the measures of the success of Monash PASS program is the proportion of the total number of students enrolled in the target units who attend five or more PASS sessions in a semester. Data from UoW shows that regular PASS participants had retention rates and final marks up to ten percentage points higher than first year students who did not participate (UoW, 2010). The University of Queensland reports similar results (Miller *et al.*, 2004). Improved pass rates benefit both student's retention and progression, two of the key Learning and Teaching Performance criteria applied to domestic first year undergraduate students.

## 3 Study Context

This study investigates the benefits of PASS and ePASS to first year novice programming students enrolled in a

core first year unit<sup>1</sup>, titled FIT1002 Computer Programming. The unit is taught as part of a common core unit across four Information Technology (IT) degrees, and across six campuses. PASS and ePASS were introduced in Semester 1, 2010 at two of the campuses teaching FIT1002 that had the largest student intakes.

### 3.1 Recruitment of PASS leaders

To recruit PASS leaders, all second and third year students, who received a Distinction or High Distinction in the target unit and a Credit average overall were sent an email and invited to apply. The shortlisted applicants were asked the following interview questions:

1. What sort of problems (academic, social, other) do you think students have in the transition from high school to university?
2. What sort of problems do you think students doing this course/unit face?
3. One goal of PASS is to help students become independent learners. How would you contribute to this?
4. How would you create a positive learning environment in your sessions?
5. Can you tell us about a recent group or leadership role you've been involved with?

### 3.2 PASS Leader training program

The PASS Leaders were required to successfully complete a two-day training program prior to semester starting. Day 1 focused on four modules: (1) How PASS works and why it is successful, (2) The role of the PASS Leader, (3) Dealing with different student types and (4) The practicalities of running a PASS session. Day 2 provided the leaders with an opportunity to practice the delivery of their activities, then review and evaluate them.

Module 1 contextualised PASS in terms of effective learning practice in higher education. Module 2 focused on differentiating the role of the PASS Leader from that of a tutor, and on establishing, fostering and maintaining good relationships with students. Module 3 focused on identifying potential difficulties that students might have, strategies for overcoming these, and familiarisation of other support services offered at the University. Module 4 covered setting objectives for a PASS session, selecting activities and structuring a session.

The programme highlighted two important points that distinguished PASS sessions from the regular classes in the unit:

- i. PASS Leaders do not re-teach content. Rather, the Leader's role is to facilitate the group gaining a greater understanding of content through discussion, with reference to lecture notes, the textbook and other study materials. Thus, the Leader will assist students to locate the information they need to answer their own question and to become independent learners. For that reason, Leaders do not make themselves available to students outside PASS sessions, except via ePASS.

<sup>1</sup> "Unit" refers to a single semester program of study. In other contexts, "course" or "subject" may be used.

- ii. PASS Leaders do not provide individual assistance with assessment tasks, nor are PASS sessions used for students to work on tasks set by the lecturer. In the tutorials and labs students can ask questions related to their assignments and tutors can assist students on particular assignment problems. Students that participate in PASS do not work on these problems so are not provided with more time on assessment tasks. Acceptable Leader assistance on assessment tasks includes discussing the meaning of question verbs or how to approach different types of examination question.

### 3.3 Unit organisation

Students enrolled in the unit attend one 2-hour lecture, one 1-hour tutorial and one 2-hour laboratory class per week. The unit is delivered across thirteen weeks, the last week reserved for revision. The unit covered foundational programming concepts and taught Java as its first year programming language. Students attend lectures on-campus. All students are provided with the following set of learning resources via Moodle: unit guide, lecture notes and audio recording of lectures, summary sheets, tutorial exercises and solutions, lab exercises and solutions, assignments, quizzes and an online discussion forum. PASS sessions were scheduled straight after the lecture or at a convenient time during the week. Students registered for PASS via Moodle signup sheets.

## 4 Study Design

PASS was advertised to students in several ways. Announcements were made to students in the Week 1 and 2 lectures by the PASS Leaders, describing the scheme and outlining its benefits. A notice was also placed on the unit's news forum advising students to register via the Moodle sign-up sheets, and tutors recommended PASS to students that they perceived would benefit from the additional support.

Face-to-face PASS sessions commenced in Week 3, and ran until Week 12, giving ten sessions per semester. Each of the four PASS Leaders facilitated two one-hour PASS sessions each week. Thus, four sessions were delivered each week at the Caulfield campus and four at the Clayton campus. At Caulfield, a further two sessions were delivered in Weeks 14 and 15 as revision.

ePASS was a basic online version of peer-assisted learning in FIT1002 in which a separate discussion board was created on the unit's Moodle site. Students in the unit were told that the text-only ePASS discussion board would be monitored only by the PASS Leaders, and would be quite separate from the regular discussion board monitored by academic staff.

ePASS was set up to mirror the face-to-face PASS sessions in two important regards: students were told that the PASS Leaders could not assist them with specific assessment-related issues (such questions should be directed to the regular FIT1002 discussion board); and students were encouraged to use ePASS to ask questions broader in nature than those related to the content of the unit – questions on “How to Learn” rather than just on

“What to Learn”. Students might not feel comfortable posting such transition questions on the regular discussion board monitored by academic staff. As with their face-to-face PASS sessions, the Leaders were instructed to point students in the right direction to answer their own questions (by referring them, for example, to the relevant lecture slides or section of the textbook), and to encourage students to contribute ideas.

Given concerns over relatively low attendance in the face-to-face PASS sessions in FIT1002, it was hoped that ePASS would appeal to a range of students: (i) those who could not attend face-to-face sessions due to timetable clashes, family, work and other outside commitments; (ii) those who might prefer the relative anonymity afforded by online interaction; (iii) those at the University's smaller, outer metropolitan Berwick campus. In two previous attempts, it had not been possible to sustain face-to-face PASS sessions at Berwick due to smaller enrolments in FIT1002. It was hoped that ePASS would provide a way in which students at Berwick could make contact with the PASS leaders at the Clayton and Caulfield campuses.

The four PASS Leaders were rostered to monitor ePASS. On their day(s), the Leader had to check and respond to any new postings, ensuring that all student postings were provided with a response within 24 hours.

### 4.1 Data Collection Method

Data was collected from participating students in Weeks 3 and 12 of a 13-week semester, the first and final weeks of the PASS program, using voluntary, anonymous surveys. Postings on the ePASS discussion forum were also used as data. Approval from the Monash University Human Research Ethics Committee was obtained to use the data collected from PASS and ePASS.

Data was gathered by the one investigator who had no teaching or assessment role in this unit. The lecturer of the unit did not have access to the individual survey data and only had access to summaries of the data after the unit was over and grades had been submitted. It was not possible to identify students from this data.

### 4.2 Survey questionnaire

The initial Week 3 survey was distributed to students in their first PASS session (Appendix 1). The questionnaire focused on students' expectations of PASS, their confidence about the unit and their reasons for signing up. In an attempt to gain an understanding of whether it is the more confident students who attended PASS to get a better mark, students were asked to indicate their confidence in the unit on a five-point rating-scale, in which a “1” indicated that the student felt they needed to attend PASS simply to pass the target unit, though to a “5” which indicated that the student felt confident they would pass the unit, but were attending PASS because they wanted a better grade (van der Meer & Scott, 2009). Students were also asked whether they had ever studied programming before. The final questions asked students

to note concerns they had either about the unit or, more generally, about being a successful student at university.

The Week 12 survey (Appendix 2) explored the value of PASS to students. The survey comprised the following questions:

- How many PASS sessions have you attended?
- How would you describe PASS to future students in FIT1002?

Using a 5 point rating scale, with “5” indicating “very”, and “1” “not at all” students were asked:

- How confident are you of achieving your academic goal in FIT1002? (Note that for some students, their academic goal was simply to pass the unit. Other students were hoping to achieve a Distinction or High Distinction)
- How useful has PASS been in helping you reach your academic goal?

Students were also asked to indicate, on a 5-point Likert scale, how PASS helped them during the semester, in terms of:

- understanding unit content,
- increasing confidence,
- developing transferable study skills,
- having the confidence to ask questions,
- making friends,
- using study time more effectively,
- feeling more enthusiastic about studying IT,
- making it easier to adjust to studying at university.

In addition to the above questions the survey asked students to nominate what they liked best about PASS; what they would like to see more of in PASS; and whether they would recommend PASS to their friends.

### 4.3 Data Analysis

Survey results were collected by the investigator and analysed under the themes of student expectations, evaluation and relationship between final results and PASS attendance.

## 5 Results

In this section an analysis of the survey data is reported. First we report on the student attendance in the PASS sessions. We analyse the students' responses to the Week 3 survey and provide insights into their expectations of PASS and reasons for attending. The Week 12 results are used to evaluate the usefulness of PASS. The final section reports on students' final grades, categorised by PASS attendance.

### 5.1 Student attendance

Attendance records maintained by the PASS Leaders on each campus show that 28 students of the 133 students enrolled in FIT1002 at the Caulfield campus attended five or more sessions in the semester, as did 11 students of the 241 enrolled at Clayton campus. This means that 21% of the Caulfield campus students regularly attended PASS, while the figure for the Clayton campus was just 5%.

### 5.2 Expectations of PASS (Week 3 Survey)

There were 42 respondents to the Week 3 survey. When asked about their confidence in FIT1002 and to what extent they needed PASS to get through the unit, only ten of the 42 respondents (24%) placed themselves at the lower end of the confidence scale (1 or 2 out of 5), as illustrated below in Figure 1. Eighteen of the 42 (48%) placed themselves at the higher end of the confidence scale (4 or 5). Overall,  $M = 3.17$ ,  $SD = 1.21$ .

Students were also asked whether they had previously studied programming, and 18 of the 42 (43%) respondents stated that they had. There appeared to be a positive relationship between relevant prior experience and confidence of doing well in FIT1002, as can be seen in Figure 1, below. Amongst the 18 students with previous programming experience,  $M = 3.50$ ,  $SD = 1.20$ ; amongst the 24 students with no previous experience  $M = 2.92$ ,  $SD = 1.18$ . Eleven of the 18 students (61%) who rated their own confidence in the unit as high (4 or 5) reported previous programming study.

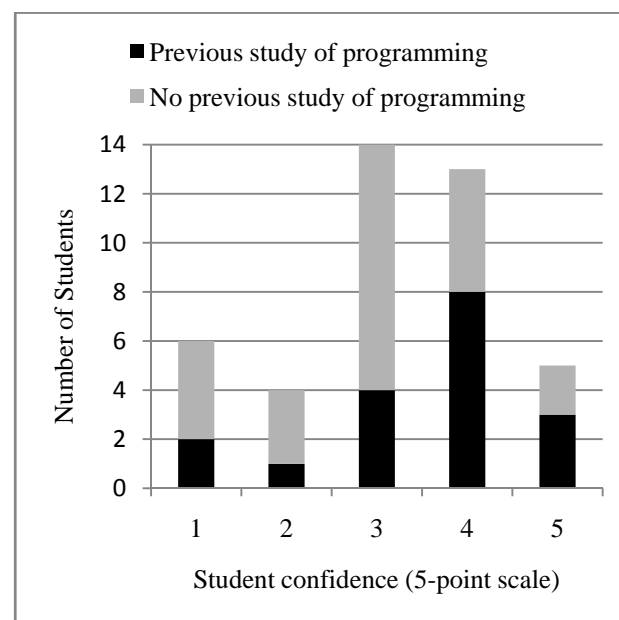


Figure 1: Student confidence in FIT1002, Caulfield and Clayton, by relevant previous study ( $n = 42$ ,  $M = 3.17$ )

Twenty-nine students completed the open-ended item on specific concerns with the unit. Irrespective of prior programming experience, students who expressed low confidence wrote very broad comments, such as “Everything”, “Steep learning curve” or “How to learn it all?”. However, across the whole confidence spectrum students expressed “Time management” and “Not falling behind” as concerns. By far, the greatest number of comments, particularly amongst the more confident students, focused specifically on concerns with the syntax, code and functions of Java.

Only 16 students completed the open-ended item on broader concerns with adjusting to study at university. Again, irrespective of previous programming experience, the most common concern was time management and keeping up with the workload. Several students wrote about needing to develop effective study skills, while the



third most common concern related specifically to doing well in assignments and final exams. In terms of broader academic transition issues, then, there was a stronger focus on “How to Learn” – particularly with regard to developing effective time management skills.

### 5.3 Evaluation of PASS (week 12 Survey)

In Week 12, 36 respondents completed the anonymous PASS evaluation questionnaire.

Students were asked to indicate their confidence of achieving their own academic goal in FIT1002, whether this might be getting through the unit with a pass, or aiming for a high distinction. Students were also asked to indicate the extent to which they felt attending PASS had helped them achieve their academic goal (bearing in mind that they had not yet sat their final examination). Figure 2, below, illustrates the responses to these survey items according to the number of sessions attended.

As can be seen, there is a demonstrated gap in student confidence of achieving their academic goal in FIT1002 between the 17 students who attended 8-10 PASS sessions (mean response 4.18) and the 19 students who attended fewer sessions (mean response 3.47). It was gratifying to see, however, that all respondents rated the usefulness of PASS well above 4 on a 5-point scale.

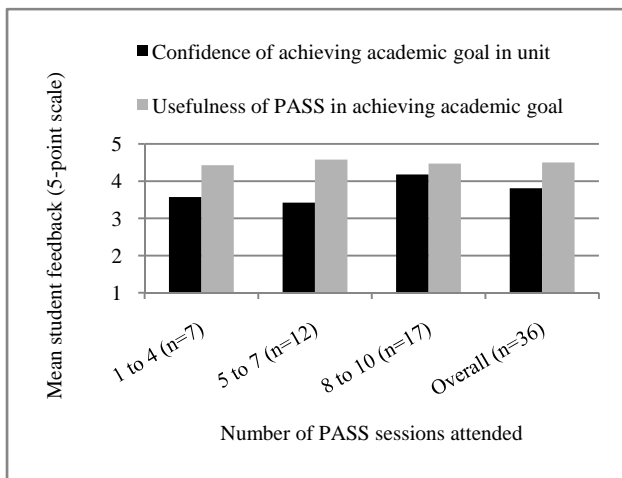


Figure 2: Feedback from FIT1002 students on achieving their academic goals in the unit and the value of attending PASS, by number of PASS sessions attended

Figure 3, below, illustrates the feedback received from students in response to the eight items on the usefulness of the face-to-face PASS sessions using a 5-point Likert scale. As can be seen, feedback was consistently positive across the three goals of the program, the “What to learn” (student understanding of key concepts, confidence in the unit and confidence to ask questions); the “How to Learn?” (development of transferable study skills and effective use of study time); and easing transition to university (making friends, feeling enthusiastic about studying IT). In addition, when asked whether they would recommend PASS to their friends, all 36 students said that they would.

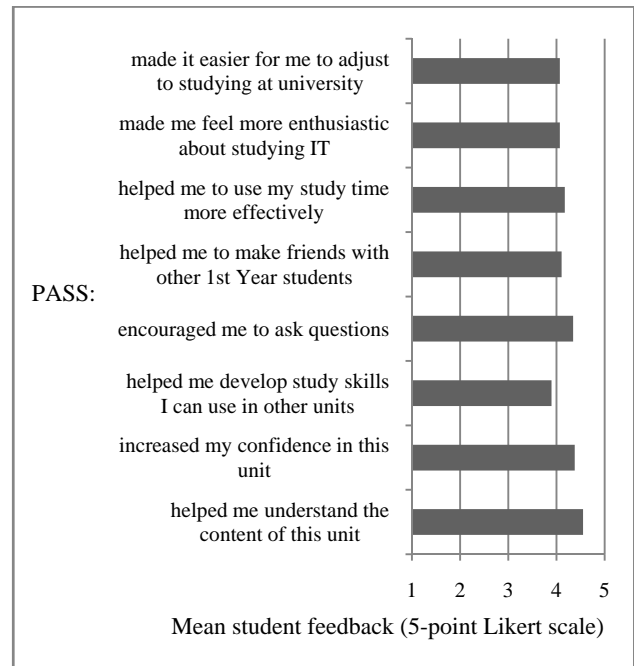


Figure 3: Feedback on PASS in FIT1002, students who attended at least five sessions (n=29), Semester 1, 2010

When asked how they would describe PASS in IT to future first year students, the students focused on PASS as a technique for improving their grades, assisting their learning and as a revision tool.

Student comments that support PASS as technique to improve their grades include:

- “An opportunity to excel in a unit”
- “Imperative if you wish to maintain an HD standard”
- “Guaranteed your marks will go up”

Others perceived PASS as a way of consolidating what was covered in lectures, and teasing out difficult concepts:

- “Very informative and helps explains what was discussed in the lecture”
- “Brilliant. It’s like a private, personal class to discuss areas of difficulty”
- “Great study group. Good place to ask any questions or concerns you have”
- “An opportunity to share knowledge and understanding with your peers. You will be able to have your questions answered and gaps in your understanding filled in a comfortable, friendly environment”
- “A helpful hand in grasping the tumultuous content of the Java language”

A few mentioned PASS as a revision tool to test understanding:

- “A brilliant way to recap what you’ve learned as well as a place to ask questions”

Open-ended comments on what students liked about PASS in FIT1002 included the opportunity to make

friends, form study groups, ask questions and explain ideas to friends. Example comments include:

- “Helping other students to further my own understanding of the content. Even though I was very comfortable with the unit, with PASS I was able to solidify ideas through practice”
- “Basically like a study group with the class genius”
- “Everyone working together”, “Making friends”
- “Students who attend ask questions I wouldn’t have thought of”

Students felt that the leaders provided quality assistance, in an environment that was fun and relaxed:

- “The fact that you could ask a “stupid” question and it will get answered without being put down”
- “The PASS Leaders understand our current issues”
- “A very supportive environment”
- “Help was always available”
- “The personal feel”

Overall students enjoyed the PASS experience and felt that PASS helped improved their learning and understanding:

- “Able to talk through problems slowly which helped me understand things better”
- “The simplicity – it made complex stuff simple and easy!”
- “Gaining confidence”
- “It was fun!”

Open-ended comments on what students would like to see more of in PASS included: more sessions per week, longer sessions and embedding PASS in other units.

#### 5.4 PASS attendance and students’ final results

Figure 4 below illustrates the proportion of students achieving different final grades in the target unit, by PASS attendance.

The graph highlights the fact that the proportion of regular PASS attendees who failed the target unit was considerably lower (15%) than in the unit overall (29%). The proportion of regular PASS attendees achieving Distinction and High Distinction was notably higher (54% to 40% overall). Regular PASS attendees were those students that attended 5 or more PASS sessions.

There is no claim here of any causal relationship between attending PASS regularly and achieving a higher final grade. Because students self-select to attend, it is the most motivated and engaged students who choose to attend, rather than students who might be feeling less engaged with the unit. No attempt has been made to try to isolate the impact on final grades of regularly attending PASS from other relevant factors, either academic (such as university entrance scores) or demographic (including gender, domestic/international status, socioeconomic status, first in family to attend university).

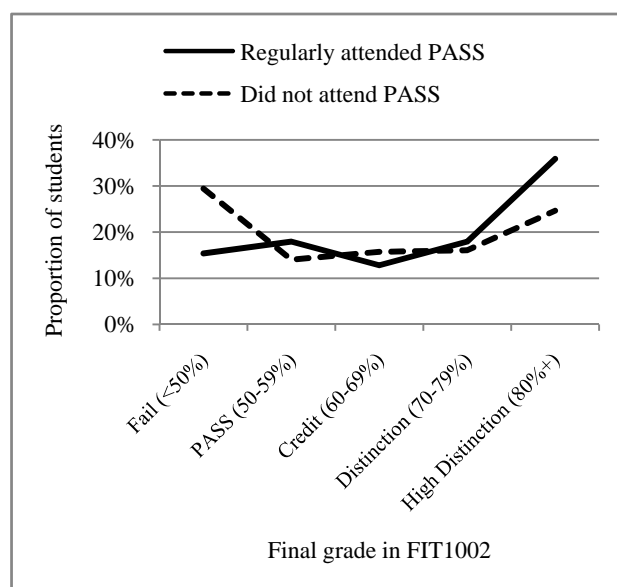


Figure 4: PASS attendance and final grades, FIT1002, Caulfield and Clayton combined, Semester 1, 2010

Regular attendees would be expected to benefit from the supplemental instruction. Indeed, the hour a week dedicated to programming study is one of the selling points used in Week 1 to encourage students to register for PASS.

#### 5.5 Students’ use of ePASS

There were 21 postings on the FIT1002 ePASS discussion board between Weeks 2 and 12 of first semester. Only five individual students posted questions on ePASS, along with three of the four PASS Leaders, and one tutor.

Only eight discussion topics were initiated throughout the semester, and half of these were administrative in nature, posted by the PASS Leaders themselves (a welcome, and three notices of changes to session times). Of the four programming-related questions posted by students, only one, on the topic of *string immutability*, with six postings, resembled the type of online group discussion we had hoped ePASS would encourage.

Of the five students who made use of ePASS, only two also regularly attended face-to-face PASS sessions. All five students passed the unit.

#### 6 Conclusion and Discussion

Studies into the effectiveness of PASS at other Australian universities have found that students need to attend at least five sessions in a semester to gain any real benefit from the program. This is supported by the data gathered in FIT1002. One of the measures of the success of PASS is the proportion of the total number of students enrolled in the target unit who attended five or more PASS sessions in a semester. Attendance in IT PASS sessions has always been considerably lower than in the other disciplines at Monash in which PASS is offered (ranging from 40% in first year biology units to 25% in first year engineering units).

So the key issue that needs to be resolved for PASS to continue in IT is poor attendance. The 29% failure rate in FIT1002 at the campuses in which face-to-face PASS ran in Semester 1 is evidence of the need for PASS, and the positive feedback from the small number of students who did attend regularly is evidence of the value of the program.

A list of possible ways to raise student awareness of PASS and increase attendance might include:

1. Obtaining testimonials on the benefits of PASS from IT students who regularly attended, and use these to help promote the program.
2. Making weekly reminders of PASS in lectures and on Moodle.
3. Raising tutor/demonstrator awareness of PASS and encouraging them to refer students to PASS if they see them struggling.
4. Issuing letters to students repeating FIT1002 describing PASS and encouraging them to attend.

Although PASS targets “at risk units” it is the students that disengage with the unit and who fail that make it an at risk unit. Given that much of the attendance records and in semester results are recorded on an LMS, a more efficient and automated way to identify at risk students might be useful in attracting more students to attend PASS sessions.

ePASS did not meet our objectives in terms of either the volume or nature of use, and there are several possible explanations for this. The first is that the unit had another, parallel online discussion forum in which students could interact with each other and with their tutors, and the distinction between the regular discussion board and the ePASS discussion board was probably not clear to the students. Second, the asynchronous, text-only model used in the trial of ePASS has severe limitations in terms of achieving the objectives of the PASS program.

Monash University has recently adopted Google Apps, and plans are in development for the trial of a more sophisticated model of ePASS for Semester 1, 2011, involving the scheduling of weekly, online evening ePASS sessions using text and audio chat functions and Google documents that are open to real-time group editing.

More broadly, Monash is currently trialling a number of models of in-class peer-assisted learning (PAL). This will look more at “horizontal” PAL (1st Year students helping each other), rather than the vertical PAL used in PASS (second and third year students helping first year students). Two arguments in favour of a move to in-class PAL are that it would benefit a greater proportion of the students in the unit (i.e. those who attend the tutorials), and remove the potential timetable clashes that prevent some students from attending PASS. In-class PAL might also encourage students to form study groups outside class, taking the benefits of PAL beyond the classroom.

In-class PAL, however, does have potential limitations. Two of the advantages of PASS being extra-curricular and purely peer-facilitated (i.e. there are no members of academic staff present) are that the PASS sessions are separated from assessment, meaning that students can (and do) ask any question without fear of being judged by their assessors. Furthermore, unless contact hours were increased, the provision of in-class PAL, whether horizontal or vertical, would not provide the same guaranteed additional study time as did the extra-curricular PASS sessions.

## 7 References

- Arendale, D. (1993). ‘Supplemental Instruction: Improving student performance and reducing attrition’, in *Educational Programs that Work: The Catalogue of the National Diffusion Network* (19th ed). Longmont, CO: Sopris West, Inc.
- AUQA (2010). ‘Peer Assisted Study Sessions (PASS) Program, University of Wollongong’, AUQA Good Practice Database [http://www.auqa.edu.au/gp/search/detail.php?gp\\_id=2608](http://www.auqa.edu.au/gp/search/detail.php?gp_id=2608), accessed 12<sup>th</sup> August, 2010.
- Ausubel, D. P. (1967). *Learning Theory and Classroom Practice*. Toronto, Ontario: Ontario Institute for Studies in Education.
- Bidgood, P. (1994). The success of Supplemental Instruction: Statistical evidence. In C. Rust, & J. Wallace (Eds.), *Helping students to learn from each other: Supplemental Instruction* (pp. 71-79). Birmingham, England: Staff and Educational Development Association.
- Bonar, J., & Soloway, E. (1983). *Uncovering the principles of novice programming*. Paper presented at the Tenth ACM SIGACT-SIGPLAN Symposium on Principles of Programming Languages, Austin, Texas, USA.
- Boyer, K., Thomas, E., Rorrer, A., Cooper, D., Vouk, M. (2010). *Increasing technical excellence, leadership and commitment of computing students through identity-based mentoring*. Proceedings of the 41st ACM technical symposium on Computer science education (SIGCSE’10), Milwaukee, Wisconsin, USA p167-171
- Chinn, D., Martin, K., and Spencer, C., (2007) *Treisman workshops and student performance in CS* Proceedings of the 38th SIGCSE technical symposium on Computer science education Covington, Kentucky, USA p 203 - 207
- Crouchman, J. (2008). ‘Who am I now? Accommodating New Higher Education Diversity in Supplemental Instruction’. *Australasian Journal of Peer Learning*, 1(1).
- Dann, W., Cooper, S., & Pausch, R. (2006). *Learning to Program with Alice*.: Prentice Hall.
- Dehnadi, S. & Bornat, R. (2006). *The camel has two humps*. Paper presented at the Little PPIG 2006 Workshop, Coventry, UK. Retrieved October,

- 2010 from  
<http://www.eis.mdx.ac.uk/research/PhDArea/saeed/paper1.pdf>
- D'Souza, D., Hamilton, M., Harland, J., Muir, P., Thevathayan, C. & Walker, C. (2008). *Transforming Learning of Programming: A Mentoring Project*. Paper presented at the Australasian Computing Education Conference, Wollongong, Australia, p75-84
- Hamer, J., Cutts, Q., Jackova, J., Luxton-Reilly, A., McCartney, R., Purchase, H., Riedesel, C., Saeli, M., Sanders, K., Sheard, J. (2008). Contributing student pedagogy. *ACM SIGCSE Bulletin*. Volume 40, Issue 4. p194-212
- Kochenour, E. O., Jolley, D. S., Kaup, J. G., Patrick, D. L., Roach, K. D., & Wenzler, L. A. (1997). Supplemental Instruction: An effective component of student affairs programming. *Journal of College Student Development*, 38(6), 577-586
- Kölling, M., & Rosenberg, J. (2001). *Guidelines for teaching object orientation with Java*. Paper presented at the Sixth Annual Conference on Innovation and Technology in Computer Science Education (ITiCSE 2001), University of Kent, Canterbury, United Kingdom.
- Kuh, G., Kinzie, J., Schuh, J., & Whitt, E. (2005). *Student success in college: Creating conditions that matter*. San Francisco: Jossey-Bass.
- Lahtinen, E., Ala-Mutka, K., & Jarvinen, H.-M. (2005, 27-29 June). *A study of the difficulties of novice programmers*. Proceedings of 10th Annual Conference on Innovation and Technology in Computer Science Education, Universidade Nova de Lisboa, Monte da Caparica, Lisbon, Portugal.
- Lewandowski, G., Gutschow, A., McCartney, R., Sanders, K., & Shinnars-Kennedy, D. (2005). What novice programmers don't know. *Proceedings of the 2005 international workshop on Computing education research (ICER 2005)*, 1-12.
- Lister, R., Adam, E. S., Fitzgerald, S., Fone, W., Hamer, J., Lindholm, M., et al. (2004, 28-30th June). *A multi-national study of reading and tracing skills in novice programmers*. Paper presented at the Working Group Reports from the Innovation and Technology in Computer Science Education Conference (ITiCSE-WGR 2004), Leeds, United Kingdom.
- McGuire, S. (2006). 'The Impact of Supplemental Instruction on Teaching Students How to Learn'. *New Directions for Teaching and Learning*, 106.
- Machanick, P. (2007). 'A Social Construction Approach to Computer Science Engineering'. *Computer Science Education*, 17(1).
- [Martin, D. C.](#), & Blanc, R. A. (1981). 'The learning center's role in retention: Integrating student support services with departmental instruction'. *Journal of Developmental and Remedial Education*, 4(3).
- Miller, A. and Kay, J. (2002). *A mentor program in CS1*. Proceedings of the 7th Annual Conference on Innovation and Technology in Computer Science Education (ITiCSE'02). Aarhus, Denmark, p9-13
- Miller, V., Oldfield, E., & Bulmer, M. (2004) *Peer Assisted Study Sessions (PASS) in first year chemistry and statistics courses: insights and evaluations*. UniServe Science
- O'Brien, M. (2006). *An analysis of the effectiveness of the Peer Assisted Study Sessions (PASS) program at the University of Wollongong: Controlling for self-selection*. School of Economics: University of Wollongong.
- Robins, A. (2010). Learning edge momentum: a new account of outcomes in CS1. *Computer Science Education*, Volume 20, Issue 1, 2010, Pages 37 – 71
- Scholarly Inquiry Symposium Proceedings, The University of Sydney, 30-35. Murray, M. H. (1996). *Alternative to lecturer-centred teaching enhances student learning and costs no more*. Academic Staff Development Unit Update, November. Brisbane: Queensland University of Technology.
- Tinto, V. (1987). *Leaving college: Rethinking the causes and cures of student attrition*. Chicago: University of Chicago Press.
- van der Meer, J. & Scott, C. (2009). 'Students' Experiences and Perceptions of Peer Assisted Study Sessions: Towards Ongoing Improvement'. *Australasian Journal of Peer Learning*, 2(1).
- UMKC (2010). 'Overview of Supplemental Instruction', International Center for Supplemental Instruction website  
<http://www.umkc.edu/cad/si/overview.shtml>  
 accessed 12<sup>th</sup> August, 2010
- UoW (2010). PASS Program Results,  
<http://www.uow.edu.au/student/services/pass/evaluation/index.html>, accessed 12<sup>th</sup> August, 2010.
- Wills, C. & Finkel, D. (1994). 'Experience with Peer Learning in an Introductory Computer Science Course'. *Computer Science Education*, 5(2).

## Appendix 1

### *f2f* PASS in FIT1002

What do you hope to get out of PASS?



MONASH University



Please tick one of the boxes below to indicate your academic goal for attending the PASS sessions:

I'm worried about FIT1002. I think I'll need PASS to get through FIT1002

<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
--------------------------	--------------------------	--------------------------	--------------------------	--------------------------

I'm confident I'll pass FIT1002, but I'm here to try to get a better grade

What questions or concerns do you have about FIT1002?

Have you studied computer programming before?

Have you used Java before?

What questions or concerns do you have about being a successful student at uni?

## Appendix 2

### PASS Student Evaluation Questionnaire

FIT1002 Semester 1, 2010

How many PASS sessions did you attend?

1 – 4

5 – 7

8 – 10

How would you describe PASS to future students in FIT1002?

How confident are you of achieving your academic goal in FIT1002? Please tick one of the boxes below.

<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
--------------------------	--------------------------	--------------------------	--------------------------	--------------------------

Not at all confident ←————→ Very confident

How useful has PASS been in helping you reach your academic goal for FIT1002? Please tick one of the boxes below.

<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
--------------------------	--------------------------	--------------------------	--------------------------	--------------------------

PASS has not helped me at all ←————→ PASS has been very useful

How has PASS helped you this semester?

**Please tick the boxes to indicate your response to the statements below:**

PASS:

- helped me understand the content of this unit
- increased my confidence in this unit
- helped me develop study skills I can use in other units
- encouraged me to ask questions
- helped me to make friends with other 1<sup>st</sup> Year students
- helped me to use my study time more effectively
- made me feel more enthusiastic about studying IT
- made it easier for me to adjust to studying at university

Strongly agree	Agree	Neutral	Disagree	Strongly disagree
5	4	3	2	1
5	4	3	2	1
5	4	3	2	1
5	4	3	2	1
5	4	3	2	1
5	4	3	2	1
5	4	3	2	1
5	4	3	2	1

What did you like best about PASS?

What would you like to see more of in PASS?

Would you recommend PASS to your friends?

Yes

No

Thank you for completing this questionnaire

# Assessing Professional Skills in Engineering Education

**Åsa Cajander and Mats Daniels**

Department of Information technology  
Uppsala University  
PO Box 337, 751 05 Uppsala,  
Sweden  
[asa.cajander@it.uu.se](mailto:asa.cajander@it.uu.se)  
[matsd@it.uu.se](mailto:matsd@it.uu.se)

**Roger McDermott**

School of Computing,  
Robert Gordon University  
Aberdeen, AB25 1HG  
United Kingdom  
[roger.mcdermott@rgu.ac.uk](mailto:roger.mcdermott@rgu.ac.uk)

**Brian R. von Konsky**

Curtin University  
GPO Box U1987  
Perth, Western Australia 6845  
Australia  
[B.vonKonsky@curtin.edu.au](mailto:B.vonKonsky@curtin.edu.au)

## Abstract<sup>1</sup>

This paper addresses the issue of developing and assessing professional skills in higher education programs. This includes defining and assessing these skills, in the contexts of an individual course unit and for an entire degree program. Identifying forms of assessment that are seen as authentic, meaningful and understandable by the students, teaching staff and curriculum developers are of utmost importance if professional skills are to be accepted and included in the formal curriculum. This can be particularly important in programs that aim to offer students a truly collaborative learning experience in a culturally diverse team. Reflections are presented as one example of an assessment method that fits this requirement. Building assessment based on the notion of threshold concepts is introduced in the context of an open ended group project course unit at Uppsala University.

**Keywords:** Professional skills, assessment, reflections, open ended group problems, threshold concepts

## 1 Introduction

There is general agreement that university students should develop professional skills and be able to demonstrate them as they enter the work force as emerging professionals in their discipline. These are typically described in the learning goals of tertiary educational programs, particularly in professional disciplines like engineering. This is often driven at a national level by accreditation requirements such as those of the Accreditation Board for Engineering and Technology (ABET 2009) in the United States, the Australian Computer Society in Australia, and the British Computer Society in the United Kingdom (2010).

Often, however, teaching teams are more comfortable placing emphasis on the development of technical skills. Limited room in the curricula, the view that professional skills are not core to e.g. the discipline of computer science, or that instructors lack experience with these topics are sometimes cited as reasons for reduced or limited emphasis on these important skills (Spradling et al., 2009).

Another problem is that many educators have an intuitive grasp of what professional skills are, but struggle to give a clear definition of them and to define rubrics for their assessment. This can be further complicated by the plethora of names for professional skills, e.g. soft skills, transferable skills, and employable skills.

The authors believe that the relative reluctance to deal with professional skills at the individual course unit level is strongly related to an uncertainty among teachers on how to integrate, teach, and assess professionals skills in the curriculum as expressed in e.g. (McKenzie et al. 2004).

Large projects unit are an obvious place to develop and assess professional skills, particularly in those cases where the project is run as an open ended group project (Faulkner, Daniels, and Newman 2006, von Konsky and Ivins 2008). This paper provides a case study describing such a unit at Uppsala University.

This paper will also address an approach for holistically integrating professional skills into the curriculum, while simultaneously highlighting their importance to stakeholders. These include educational designers, instructors, project supervisors, and students.

The approach described in this paper involves:

- The specification of the professional skills to be developed at different levels of the educational program.
- Ensuring that academic staff and supervisors have the relevant skills to guide students in their development as emerging professionals.
- The provision of authentic learning experiences and environments.
- The implementation of an appropriate framework to assess student learning and the actual attainment of professional skills.

The paper draws on experiences from Curtin University, which implements an institution-wide process for defining, contextualizing and embedding professional skills into the formal curriculum of all degree programs it offers.

The paper also highlight the use of reflections in which students self-assess their attainment of professional skills, which have been a useful tool at the authors' institutions. This will be especially illustrated with work at Robert Gordon University.

The use of reflections at Uppsala University will be presented in the context of a course unit called *IT in Society* (Laxer et al. 2009). This unit will be used to illustrate issues and solutions related to the specification and

---

Copyright © 2011, Australian Computer Society, Inc. This paper appeared at the 13th Australasian Computer Education Conference (ACE 2011), Perth, Australia. Conferences in Research and Practice in Information Technology (CRPIT), Vol. 114. J. Hamer and M. de Raadt, Eds. Reproduction for academic, not-for profit purposes permitted provided this text is included.





Figure 1. Bookmark describing Curtin University's Graduate Attributes and the Triple-i curriculum

assessment of professional skills in an open ended group project. This is discussed in the context of building assessment on the notion of threshold concepts (Meyer and Land 2003, 2005) to differentiate between genuinely possessing the professional skills in question as opposed to merely being able to talk about them.

## 2 Professional Skills in an Educational Setting

A number of recent developments in UK Higher Education have tended to emphasise the development of skills for lifelong learning. This has led to a renewal of interest in issues such as student employability and the role of university curricula in expanding students' capacity to learn.

This is exemplified by the Scottish National Enhancement Themes programme (Lines 2010), which currently has a focus on the development of Graduate Attributes (Barrie 2004), drawing heavily on work done at Australian universities such as Curtin and Sydney. Part of the application of this initiative is the embedding of

reflective practice as a major component in the promotion of lifelong learning skills.

The work of Schön (Schön 1983, 1987) proposed a direct link between the use of critical reflection and successful professional practice. Reflection, self-evaluation and self-assessment are characteristics that distinguish expert practitioners from novices and so the development of a capacity to reflect on practice should be an essential element of any preparation for a professional career.

### 2.1 Activities at Curtin University

In conjunction with an institution-wide curriculum renewal project called C2010, Curtin University in Perth, Western Australia implemented a process of Comprehensive Course Review (CCR). The goal was to examine all teaching programs at the University to ensure that each program is of high educational quality, pedagogically sound, and sustainable.

As part of the process, teaching teams map the curriculum for an entire degree program, showing how



## Professional Practice 401 Unit Learning Outcomes.

Unit Learning Outcome (ULO)	Graduate Attributes Developed	Bloom's level
1. <b>Analyze</b> user requirements and document them.	Thinking Skills Communication Skills	Analysis
2. <b>Establish</b> goals and a work plan to track progress with respect to management and technical roles on the team, and including metrics to measure goal attainment.	Thinking skills	Analysis
3. <b>Manage</b> on-going project progress, making efficient use of available resources and planning tools.	Thinking skills Technology skills Communication skills	Analysis
4. <b>Provide</b> constructive feedback to other team members	Professional skills Thinking skills Communication skills	Evaluate
5. <b>Reflect</b> on goal outcomes associated with your assigned management and technical roles.	Thinking skills	Evaluate

**Table 1: Mapping between subject specific ULO and Graduate Attributes for a hypothetical unit**

## Professional practice 402 Assessment Mapping

Assessment	ULO Assessed	Weighting
Client interview	ULO# 1	10%
Requirements document	ULO# 1, 2	30%
Final management report	ULO # 2, 3, 4, 5	60%

**Table 2. Mapping between associated assessments and ULO for a hypothetical unit**

the University's nine graduate attributes are embedded and contextualized in the context of the given discipline for each subject and for the program as a whole.

A summary of Curtin's graduate attributes and Triple-i curriculum experiences that develop them are shown in Figure 1. The figure is from a bookmark, routinely distributed to both staff and students.

In the Curtin context, the graduate attribute for professional skills includes teamwork and leadership skills, professional behavior and ethical practices. The Curtin CCR process include analysis data that incorporate student evaluations conducted at the end of each semester, results from the Course Evaluation Questionnaire (CEQ) (Wilson et al. 1997), and surveys capturing the perceptions of recent graduates and their employers regarding skills developed during the degree program.

An output of the CCR process is an updated curriculum map. The map shows the relationship between Curtin's graduate attributes and discipline specific professional competencies defined by professional bodies like Engineers Australia. For each unit in the degree program, the curriculum map lists 5 or 6 Unit Learning Outcomes (ULOs) that are intended to clearly define what the students must do to demonstrate learning, the Bloom's Taxonomy Level at which each outcome is demonstrated, the associated graduate attributes developed, and the assessments that measure their attainment.

Bloom's original taxonomy defined six levels of thinking, each requiring increasing levels of cognition. The six levels are knowledge recall, comprehension, application, analysis, synthesis, and evaluation (Bloom 1956).

Tables 1 and 2 show a portion of an abstracted curriculum map for a hypothetical course unit called

Professional Practice 401, Table 1 shows the mapping between ULO and graduate attributes. Table 2 shows the associated mappings between assessments and ULOs. Assessment rubrics are also considered as part of the review, although these are not shown in the table nor included in the curriculum map.

Care must be taken when writing ULO statements to ensure that verbs convey the required level of thinking. Selection of verbs is usually based on Bloom's taxonomy. For example, an outcome statement that says "*understand* project management standards" does not convey what the student must do to demonstrate that they have understood these standards. The outcome statement "*describe* project management standards" requires a low level of thinking on Bloom's scale. In contrast, the outcome statement "*implement* project management standards" would require higher order thinking skills. The statement "*evaluate* project risks when selecting and implementing appropriate project management standards" would require even higher order thinking.

The curriculum map describes where teaching teams and curriculum developers *intend* for the graduate attributes to be developed. A new electronic portfolio, called the iPortfolio, closes the loop on curriculum design. That is, the iPortfolio captures what students have *actually* learned, based on self-reflection and evidence provided by students (von Konsky et al. 2010, Oliver et al. 2009, von Konsky et al. 2009).

### 3 Reflection as a Means to Assess Professional Skills

The connection between the development of professional skills and the capacity to reflect on experience is found in work on positive learning dispositions, e.g. Claxton's 'four Rs': resilience, resourcefulness, reflectiveness and reciprocity (Claxton 2002). This is a useful classification

for the development of 'leaning how to learn' and the extension to lifelong learning skills. The disposition of reflectiveness naturally finds counterparts in a network of concepts such as metacognition, self-regulation, self-direction, and self-efficacy (Higgins, 2009).

Further links between the development of professional skills and reflection is found in the work of Nicol and his co-workers (Nicol et al. 2006, 2009) on formative assessment and feedback. Nicol situates his work in the context of the enhancement of self-regulated learning, defined as:

*'an active constructive process whereby learners set goals for their learning and monitor, regulate, and control their cognition, motivation, and behaviour, guided and constrained by their goals and the contextual features of the environment.'* (Pintrich and Zucho 2002)

This approach was incorporated into the REAP project (REAP 2007) and has been influential in motivating curriculum change in Scottish Higher Education.

Some form of learning journal (whether paper-based, electronic, or simply a set of discrete reflections on learning) is a prime candidate for a vehicle to facilitate the development of self-assessment and reflection (Moon 2006).

While the use of paper-based journals or lab-books may well be more familiar to engineering disciplines, the social features of a blog provide an important additional element that serves to encourage dialogue between tutors and students about the learning process. In particular, the commenting facility plays an important pedagogical role in promoting the development of social and academic support networks and student self-regulation.

A number of pedagogical benefits result.

- Timely feedback allows students to discern the strengths and weaknesses of their performance. It provides an opportunity to make decisions about how they may subsequently modify their own work and so increase learning autonomy.
- The action of supplying commentary on work done by peers provides students with the opportunity to develop the capacity to make objective judgements with reference to externally-set marking criteria.
- This ongoing student-teacher and student-student dialogue also serves to clarify the subtler (and often unstated) characteristics of what counts as "good performance" in the context of a particular assignment.
- Individual students can monitor the relationship between their own understanding of high performance and that of their teacher and also their peers. This is a significant factor in the development by students of appropriate mental models of the learning process.
- On a practical side, advice and academic support from peers may be articulated at a more appropriate level and be perceived as less of a threat to student self-esteem.
- The alternative perspective that such peer feedback may present can serve to motivate

perseverance on tasks and provide a degree of mutual support and validation for efforts made.

- The repetitive nature of tasks like blogging also increases time-on-task and allows students to iterate the feedback cycle in a natural way.

This link between successful reflective practice and increased learning autonomy suggests that the narrative structure of blogs may be used profitably to encourage an atmosphere of developmental improvement. Students come to realise that the relationship between their current state of knowledge and the established subject matter does indeed evolve. This understanding that the acquisition of expertise does not happen instantaneously and that their conceptual model of a topic will change, evolve and deepen over time is an important characteristic of mature learners.

Finally, blogs give a useful two-way feedback mechanism that allows students themselves to offer commentary on the provision and suitability of teaching activities. They can therefore be used to provide high quality information to teachers about the nature of the student experience. Such information may go well beyond academic concerns and offer insights into the social, economic and intellectual milieu of the student which may, for example, affect the way in which the course is delivered or simply increase the teacher's appreciation of the (variety of) student experiences.

### 3.1 Use of Reflections at Robert Gordon University

Within this overall context, work done in the School of Computing at the Robert Gordon University has a particular focus on the use of blogs to capture student reflection on their first year experience (McDermott et al. 2010). The activities are embedded into the curriculum within a two-semester course unit investigating professional skills. Each student is required to keep an individual blog and post a minimum of two hundred words per week describing their learning experience on each of the units they study. This forms part of the raw material for an e-portfolio of work that would accompany the student throughout their course of study and could, potentially, form the basis of further reflective activities in later years.

In addition to posting their own reflections, there was a requirement that individual students make a substantive comment on two other posts each week. The academic goals of the blogging activity were carefully explained to students and a default template for the presentation of reflective comments was distributed providing some basic scaffolding for these exercises. This consisted of a number of questions in which the student was asked to identify the major learning objectives covered that week, detail new information or skills assimilated, comment on any learning strategies adopted, and describe any significant affective reactions to the classes the student had experienced.

### 3.2 Assessing Professional Skills Through Reflection

*Robert Gordon University*

While students are identified as driven by assessment (Biggs 1999), there are a number of issues surrounding identification and appraisal of reflection that complicate a straightforward alignment of learning objectives with the desired goal of promoting this kind of activity.

The first of these is that despite widespread agreement in the literature that the development of metacognitive skills is important, there is nevertheless a lack of clarity or precision in the terminology used. Concepts such as reflection, reflective thinking, and critical thinking are defined in different ways by different authors and it is not always apparent how these overlap, or their relationship to other ideas relating to student empowerment (such as self-regulation and self-direction).

This lack of precision in the terminology also manifests itself in the wide variety of theoretical frameworks that underpin schemes to identify and assess reflective work (e.g. Boud et al. 1985, Mezirow 1991, Hatton and Smith 1995, Wong et al. 1995, Scanlon and Chernomas 1997, Kember et al. 1999, Moon 2000, Kember et al. 2008).

A second issue pertains to student engagement with such reflective activities. The majority of students find such activities difficult to practice, and many teachers find them difficult to promote. While this may, in part, be due to a long acquisition time for the capacity to critically reflect, it also appears that activities which are designed to promote the skill lack focus.

For this reason, in addition to requiring students to participate in the reflective blogging exercises, the course unit described above also provided structured opportunities to develop and enhance the graduate attributes mentioned earlier.

The initial exercises were discursive in nature and focussed on the purpose of the course unit and the idea of graduate attributes (over and above subject-based technical competencies). These were then followed by an introduction to the computing infrastructure relevant to the unit, e.g. the blogging environment. Further activities engaged with issues in the psychology of learning. The Hatton-Smith categorisation of reflective writing (Hatton and Smith 1995) was also described. Blogs were reviewed using the Hatton-Smith framework, which classifies writing into four levels of increasing sophistication of reflective activity, see Table 3.

As may have been anticipated, analysis of the data indicated a natural trajectory for written work throughout the year (McDermott et al. 2010). While most students started at the descriptive writing stage, the vast majority progressed to descriptive reflection, with a number of students regularly engaging in dialogic and even critical reflection. Comments from questionnaires showed that a majority of students felt positively about the need for reflection. Moreover, they also suggested that student satisfaction concerning feedback was also positive, contributing to increased satisfaction measures with course as a whole.

Level of Reflection	Indicator
1. Descriptive Writing	The student simply describes experience without significant attempts at analysis. Although essentially non-reflective, it can nevertheless serve as a foundation for later, more complex activity.
2. Descriptive Reflection	The student attempts to provide reasons for their learning experiences based upon quasi-reflective personal judgements.
3. Dialogic Reflection	The student enters into a personal discourse to explore possible reasons for observed outcomes.
4. Critical Reflection	In this context, critical reflection was taken to be demonstrated by the elaboration of reasons for personal learning decisions and experiences which takes into account a mature understanding of the psychological and pedagogical factors affecting the learning process.

**Table 3. Hatton and Smith Framework for Reflective Writing (Hatton and Smith 1995).**

*Curtin University*

Writing critical reflections describing the outcomes associated with a task or learning experience requires higher order thinking; more so than merely reporting on the task or learning experience in a descriptive manner. In Table 1, for example, ULO 5 reads “*reflect* on goal outcomes associated with your project management and technical roles.” To demonstrate this outcome, the rubric for the assessment articulates that the student must be able to state whether or not the goal was achieved, how they know this, and what was learned from the experience that can be applied during similar experiences in the future.

My iPortfolio | Admin | Feedback | Settings | Help | Logout | Close Window

# iPortfolio

## Engineer Edgar (Example iPortfolio)

About Me
My Ratings
My Course
My Employment
My Journals
My Networks
My Showcases

	Helpful	Harmful
<b>Internal</b>	<b>STRENGTHS</b> <ul style="list-style-type: none"> <li>Well equipped shops and labs</li> <li>Chemical team skills</li> <li>Enthusiastic volunteers</li> <li>Multiple year levels</li> <li>Learn new skills</li> <li>Reinforce existing skills</li> <li>Work in an interdisciplinary context</li> </ul>	<b>WEAKNESSES</b> <ul style="list-style-type: none"> <li>Different timetables</li> <li>Different assessment due dates</li> <li>Academic advisors not identified</li> <li>Corporate sponsors not identified</li> <li>Interfaculty cooperation issues</li> <li>Lack of funding from Curtin</li> <li>Lack of succession plan for future</li> <li>No communication plan</li> <li>No management plan for design decisions</li> </ul>
<b>External</b>	<b>OPPORTUNITIES</b> <ul style="list-style-type: none"> <li>Work experience</li> <li>Industry exposure</li> <li>Networking</li> <li>Professional development</li> <li>Learn new skills</li> </ul>	<b>THREATS</b> <ul style="list-style-type: none"> <li>Better resourced team at UWA team</li> <li>Lack of external funding</li> <li>Sickness or team members during critical development stages</li> </ul>

**SWOT Analysis**

**Earned Value Report**

Click on any piece of evidence to see it larger.

### Curtin Motor Sports Team Engineering Project Management

*Situation*

Each year, the Curtin Motor Sports Team designs and manufactures a high performance motor vehicle for competition.

*Task*

In addition to my role in the shop, I served as the Engineering Project Manager, managing a diverse interdisciplinary engineering team.

*Action*

In this role, I led a series of planning workshops in which a *SWOT analysis* was conducted that identified strengths and skills of the team, potential risks, and a work breakdown structure giving the *significant tasks* to be undertaken. Based on this, I developed a formal project plan, and used it to monitor team progress against key performance indicators.

*Result*

As a result, we delivered our vehicle on-time and under budget, meeting all performance targets.

*Lessons Learned*

I learned that it is necessary to bring resources and people together for mutual team benefit and success, while optimising and co-ordinating individual strengths and skills.

*Evidence Provided*

1. SWOT Analysis
2. Earned Value report showing project progress over time against planned progress and actual effort

**Figure 2. Example reflection in Curtin University's iPortfolio using the STAR-L template (Curtin University 2010, Queensland University of Technology 2010).**

Curtin's iPortfolio includes a range of optional templates that assist students to write structured reflections. These includes the STAR-L template, in which students write reflections that includes the:

- *Situation*- the event that gave rise to the learning experience;
- *Task*- a description of the learning experience;
- *Action*- the specific action taken by the student in implementing the task;
- *Result*- the outcome of that action; and
- *Lessons learned*- what can be applied from this experience in the future, including what would be done the same, and would be done differently.

An example is shown in Figure 2 for an extracurricular project management learning experience. The reflection in the example is accompanied by artifacts. Note that by themselves, the artifacts only tell part of the story. The reflection is required to put the artifacts into context, and demonstrate what has been learned as a result of the activity.

#### **4 Professional Skills in an Open Ended Group Project (OEGP) Course Unit**

The Open Ended Group Project (OEGP) framework referred to in this paper is described in (Faulkner, Daniels, and Newman 2006). It is based on a similar view of learning as underpins Problem Based Learning (Kolb 1984, Kolmos and Algreen-Ussing 2001), Situated Cognition (Brown, Collins, and Duguid 1989), Practice fields (Barab and Duffy 2000), and Communities of Practice (Wenger 1998). It is furthermore closely related to ideas concerning use of ill-structured problem solving (Jonassen 1997). This is also discussed in general by for example Rittel and Webber (1973) who call these problems "wicked problems". Schön (1983) describes these wicked problems as belonging to the swampy lowland where predefined methods and techniques are of no use in the problem solving process. Schön describes the different nature of problems like this:

*"high, hard ground where practitioners can make effective use of research-based theory and technique" as well as "swampy lowland where situations are confusing 'messes' incapable of technical solution"* (Schön, 1983).

The actual implementation of an OEGP can to no surprise vary considerably depending on a number of factors, e.g.:

- Where it occurs in the academic program (i.e. which year/semester).
- Number of students involved.
- Time available for the OEGP.
- Academic credit offered for the work.
- Method by which groups are formed and managed.
- Type of task chosen as the problem.
- Inter-relationship between the groups.
- Educational 'objectives' or 'intended learning outcomes'.

Most variants will however involve use of several professional skills, with limited control over which and to what degree for any given student. This is a natural consequence of the OEGP idea and one that provides a challenge when it comes to assessment. An OEGP course unit, IT in Society (Laxer et al. 2009), will be presented and assessing the ability to truly collaborate in a culturally diverse team will be investigated.

#### 4.1 The IT in Society (ITiS) Course Unit

The ITiS course unit at Uppsala University is run in collaboration with a course unit in US (Communication in a Global Society) and is offered to students in the first semester of the fourth year. The unit accounts for half of the study load for a student during that semester in the IT engineering degree program. A goal of the ITiS unit is that the students should be able to constructively participate in a project dealing with a complex and multifaceted problem set in a real environment.

Since 2002, the setting has been the Uppsala Academic hospital, and since 2004 all students have been involved in the same project. The collaboration with the US students at Rose-Hulman Institute of Technology started 2005. The number of students has varied from 20 to 45 over the years.

#### 4.2 Using Reflections in Earlier Instances

Assessing student goal achievement regarding ability to function in a culturally diverse team has been collaborative, and involved reflections and direct observation during the project. There was a practical reason for using reflections in that it seemed to be a good candidate to address the problem with students seldom seeing their own role in problematic issues, and especially in cases where they viewed the international collaboration as a burden. This choice was also influenced by the emphasis that Fincher and Petre (2001) put on the value of reflection in computer science project work.

The educational value of being able to reflect has been addressed already in the paper and is clearly described in work on the reflective practice model by Schön (1987). He observes that professional work involves an ongoing process of reflective practice involving self monitoring, continual improvement and action cycles (plan, act, observe, reflect). There was thus an educational benefit to use reflective work to assess professional skills, such as collaboration, in the unit. This can in fact be expressed even stronger in that the ability to reflect is a prerequisite

for a professional skill such as the ability to truly collaborate in culturally diverse teams.

Reflection is an action that was first introduced as a written and oral individual final report at the end of the unit. These reports offered students an opportunity to reflect upon and demonstrate what they had learnt about the process of global collaboration, the results they had achieved, the problems they had successfully overcome, what they had gained personally and professionally from the experience and where they still had to develop. This report and the follow-up individual meeting was not merely descriptive of the project, but included a broader critical dimension as befits a final year degree course. Many gave insightful descriptions on their performance and learning, e.g. *"I think I took many opportunities to get to learn new things and also to practice what I already know."* This action has been kept with some slight variations in the phrasing of the instructions given to the students.

The value derived from the final reflections led to introducing weekly individual reflections throughout the unit. The high volume led to slow responses from the teachers and it was problematic to post issues to reflect on that were relevant for all students. This led to a reduction of the number of reflections as well as using peer feedback in some instances and also using both individual and group reflections. These changes had a positive effect on the quality of the reflections as reported by the teachers. The value of the reflections is reported as moderately high, (3.5 out of 5) in the course evaluations.

Reflection has also been done in the form of students being active in producing a paper (Cajander et al. 2009) describing their learning experiences in the ITiS unit. The value of a research framework for understanding the role of technology in collaborations (Clear 2009) in terms of improving ability to reflect is reported in (Cajander, Clear, and Daniels 2009).

#### 4.3 Extending the Scope of Assessment in the Next Instance

Assessing using reflections has been valuable, but there is a perceived limit when it comes to assessing the ability to function effectively in a culturally diverse team. The ability to reflect, in the full sense of the word as captured in Table 3, on true collaboration in such a team is not enough to ensure that a student is able to "truly" function in such a setting. This is an example of where there is a difference between knowing the theory related to a professional skill and being able to practice the skill. Both are essential in order to possess the professional skill, i.e. to be a craftsman in the discipline relies on being able to draw upon a mix of theory and practice. Reflections are typically on action, rather than "reflection in action" (Schön 1987), and as such more suited to assess the theory part.

The process of reflection in action is according to Schön (1987) central to the "art" (professional skill) by which practitioners (professionals) deals with situations of uncertainty, instability and value conflict. This seems to indicate that *reflections in action* also would be a suitable means to assess also the actual possession of a professional skill. This is also the basis for our next step

in developing assessment methods in the ITiS course unit. It should however be noted that many aspects of a professional skill is of a tacit type (Polanyi 1958) and thus almost impossible to capture in a reflection.

A suggestion on how to address this issue based on the notion of threshold concepts (Meyer and Land 2003, 2005) is outlined below. The approach will be tried in the upcoming instance of the ITiS unit.

#### 4.3.1 View of Learning in an OEGP

The view of how learning take place, the epistemology, in this example is constructivism (Piaget 1970) in which learning is seen as a social process. The immersion of the learner in a complex realistic real world problem is seen as instrumental for creating the context for learning. The need for discussion is paramount in addressing open ended problem and the, for learning vital, social process is a natural component of an OEGP setting. Selecting a real world problem stems from the concern of finding a problem that is relevant for the learner. A good problem is defined in (Brooks and Brooks 1999) as one that:

- Requires students to make and test at least one prediction.
- Can be solved using only equipment and facilities that are available.
- Is realistically complex.
- Benefits from a group effort.
- Is seen as relevant and interesting by students.

#### 4.3.2 Threshold Concepts

The notion of threshold concepts has been explained in work by Meyer and Land (2003, 2005). It is a concept that has properties suitable for reasoning about learning and investigation on how to assess professional skills.

A threshold concept is defined in Meyer and Land (2003) in the following way:

*A threshold concept can be considered as akin to a portal, opening up a new and previously inaccessible way of thinking about something. It represents a transformed way of understanding, or interpreting, or viewing something without which the learner cannot progress. As a consequence of comprehending a threshold concept, there may thus be a transformed internal view of subject matter, subject landscape, or even world view.*

Threshold concepts are *integrative* and tie concepts together in new ways and *irreversible* in that they are difficult to unlearn. However, they might also be *troublesome* as they are seen as alien, difficult, or counter-intuitive.

Professional skills can be seen as prime candidates for being identified as threshold concepts and discussions about genuinely possessing a skill, integrating it and transforming thus comes natural. In the context of this paper it is perhaps the difference between mimicry and genuine understanding of the threshold concept that is the most interesting aspect of the notion. The transformation when acquiring a professional skill may be either sudden or take place over a considerable period. This transformative stage of development and learning is

named *liminality* by Meyer and Land. Liminality in this context can be understood as the period preceding the actual ‘crossing’ of the threshold. The liminal state might involve puzzlement and confusion. In the liminal state people may imitate the language and behaviours, prior to full understanding of a discipline or area of expertise. Cousin (2006) describes this confusion in an interesting way:

*“In short, there is no simple passage in learning from ‘easy’ to ‘difficult’; mastery of a threshold concept often involves messy journeys back, forth and across a conceptual terrain.”*

Meyer and Land (2005) point out that scaffolding may create a proxy for the threshold concept that can lead to mimicry or to the student being in the liminal state described above. To capture this difference in a student is the aim of the changed assessment method in the upcoming instance.

#### 4.3.3 Plan for Implementation

Experience from previous instances of the ITiS unit suggest that the occurrence of a major shift in direction of the project lead to students obtaining a higher level of professional skills. The instances in question were in the first case when the customer halfway through the project realized the potential of the students and wanted them to change direction and in the second case the teachers wanted the students to take radically different approach to structuring their final report. Both cases resulted in a better product and improved learning in terms of professional skills as seen from the teacher point of view, but it is however unclear to what degree the students realized this.

A form of constructive controversy (Johnson and Johnson 2007, 2009) was introduced in order to try to create a similar learning opportunity (Daniels and Cajander 2010), but with a weaker result as compared to the two instances that inspired the approach. One hypothesis is that the approach needs to be strengthened with a better ability to reflect among the students and furthermore that the motivation to change direction is experienced as genuine among the students.

In short, the idea is that a change of direction will unsettle the students and force them to use professional skills in an intense manner. This increase in intensity will allow a better accessibility to reflecting on these skills. Selection and introduction of suitable threshold concepts, e.g. the skill “ability to genuinely collaborate”, will influence the design of the constructive controversy event and will be used as lenses for the students to observe how they use their skills and provide a reference for assessing how well the students understand and master these skills.

That the students master how to write reflections will be crucially important for the success of this approach. Less critical, but still important is that they also have an understanding of the notion of threshold concepts. The same goes for the teachers in order for them to perform assessment, and perhaps especially be able to distinguish between mimicry and genuine transformation.

## 5 Conclusions

The value of reflections as a means to assess professional skills in higher education has been addressed, both at individual course and whole education study program levels. The potential for essential improvements regarding developing and assessing professional skills rely on a raised awareness, and increased capability, among all involved in the educational process. That is, from students, through TA's and teachers, up to coordinators of education programs and education institution boards.

There are many different aspects of professional skills, and careful application of reflection based assessment techniques is found to be promising. An outline of building on the notion of threshold concepts has been presented. There are many aspects of threshold concepts that relate to acquiring professional skills and building on this in developing assessment methods is promising.

## References

- ABET (2009): *2009-2010 Criteria for accrediting engineering programs*, ABET Inc, Baltimore.
- Barab, S. and Duffy, T. (2000): From practice fields to communities of practice, *Theoretical Foundations of Learning Environments*, eds. Jonassen, D. and Land, S., Lawrence Erlbaum Ass. Inc, 25-56
- Barrie S.C. (2004): A research-based approach to generic graduate attributes policy. *Higher Education Research and Development*. vol. 23 (3), 261-275
- Biggs, J. (1999): What the Student Does: teaching for enhanced learning, *Higher Education Research & Development*, vol.18: 1, 57-75
- Bloom B. S. (1956). *Taxonomy of Educational Objectives. Handbook I: The Cognitive Domain*. New York: David McKay Co Inc.
- Boud, D., Keogh, R., and Walker, D. (1985): Promoting reflection in learning: A model. In D. Boud, R. Keogh, & D. Walker (Eds.), *Reflection: Turning experience into learning* (pp. 18-40). Kogan, London
- British Computer Society (2010): *Guidelines on Course Accreditation*, BCS, Swindon, UK
- Brooks, M. and Brooks, J. (1999): The courage to be constructivist, *Educational leadership*, vol. 57, no. 3
- Brown, J., Collins, A., and Daguid, P. (1989): Situated cognition and the culture of learning, *Educational Researcher*, vol 18, no 1., 32-42
- Cajander, Å., Clear, T., Daniels, M., Edlund, J., Hamrin, P., Laxer, C., and Persson, M. (2009): Students analyzing their collaboration in an international Open Ended Group Project, *ASEE/IEEE Frontiers in Education*, San Antonio, USA
- Cajander, Å., Clear, T. and Daniels, M. (2009): Introducing an External Mentor in an International Open Ended Group Project, *ASEE/IEEE Frontiers in Education*, San Antonio, USA
- Claxton, G.L., (2002): *Building Learning Power: Helping Young People Become Better Learners*, TLO: Bristol
- Clear, T. (2009): Supporting the Work of Global Virtual Teams: The Role of Technology-Use Mediation, *Computing and Mathematical Sciences*, Auckland: Auckland University of Technology, New Zealand
- Cousin, G. (2006). An introduction to threshold concepts. Planet Special Issue on Threshold Concepts and Troublesome Knowledge(17), 4 5.
- Curtin University (2010) Engineer Edgar (Example iPortfolio), [https://iportfolio.curtin.edu.au/view/my\\_showcases.cfm?user=1361&entry=842](https://iportfolio.curtin.edu.au/view/my_showcases.cfm?user=1361&entry=842) , Accessed 23 August 2010
- Daniels, M. and Cajander, Å. (2010): Constructive Controversy as a way to create "true collaboration" in an Open Ended Group Project Setting, *CRIPT*, vol 103 73-78
- Faulkner, X., Daniels, M., and Newman, I. (2006): The Open Ended Group Project: A way of including diversity in the IT curriculum, *Diversity in Information Technology Education: Issues and controversies*, ed. Trajkovski, G., Information Science Publishing, 166-195
- Fincher, S., Petre, M., and Clark, M. (2001): *Computer Science Project Work Principles and Pragmatics*, Springer Verlag, London
- Hatton, N. and Smith, D. (1995): Reflection in Teacher Education – Towards Definition and Implementation, *Teaching and Teacher Education*, Vol. 11, No. 1, 33-49
- Higgins, S. (2009): *Learning to Learn. Beyond Current Horizons*. Durham: Beyond Current Horizons Project
- Johnson, D. and Johnson, R. (2007): *Creative constructive controversy: Intellectual challenge in the classroom*, 4<sup>th</sup> ed, Edina
- Johnson, D. and Johnson, R. (2009): Energizing learning: The instructional power of conflict, *Educational Researcher*, vol. 38, no. 1, 37-51
- Jonassen, D. (1997): Instructional design models for well-structured and ill-structured problem-solving learning outcomes, *Educational technology Research and Development*, vol 45, no 1, 65-94
- Kember, D., Jones, A., Loke, A., McKay, J., Sinclair, K., Tse, H., Webb, C., Wong, F., Wong, M. & Yeung, E. (1999) Determining the level of reflective thinking from students' written journals using a coding scheme based on the work of Mezirow, *International Journal of Lifelong Education*, vol. 18(1), 18-30
- Kember, D., McKay, J., Sinclair, K. and Wong, F.K.Y. (2008): A four-category scheme for coding and assessing the level of reflection in written work, *Assessment and Evaluation in Higher Education*, vol. 33: 4, 369-379
- Kolb, D. (1984): *Experiential learning: Experience as the source of learning and development*, Prentice Hall



- Kolmos, A. and Algreen-Ussing, H. (2001): Implementing problem-based and project organized curriculum, *Das Hochschulwesen*, 1, 15-20
- Laxer, C., Daniels, M. Cajander, Å., and Wollowski, M. (2009): Evolution of an international collaborative student project, *CRPIT*, vol. 95, 111-118
- Lines, D. (2010): *Things that make a difference: lessons from the Enhancement Themes, The Quality Assurance Agency for Higher Education*, ISBN 9781849791182, and <http://www.enhancementthemes.ac.uk>
- McDermott, R., Brindley, G., and Eccleston, G. (2010): Developing tools to encourage reflection in first year student blogs. *ACM Innovation and Technology in Computer Science Education*, Ankara, Turkey
- McKenzei, L., Trevisan, M., Davies, D., Beyerlein, S., and Huang, Y. (2004): Capstone design courses and assessment: A national study, *ASEE Annual Conference*, Salt Lake City
- Meyer, J. and Land, R. (2003): Threshold concepts and troublesome knowledge: Linkages to ways of thinking and practicing within the disciplines, in Rust, C. (ed.), *Improving student learning: Improving student learning theory and practice – Ten years on*, Oxford center for staff and learning development
- Meyer, J. and Land, R. (2005). Threshold concepts and troublesome knowledge (2): Epistemological considerations and a conceptual framework for teaching and learning. *Higher education*, vol 49, 373-388
- Mezirow, J. (1991): *Transformative Dimensions of Adult Learning*. Jossey-Bass, San Francisco
- Moon, J. (2000): *Reflection in learning & professional development: theory & practice*, Routledge, London
- Moon, J., (2006): *Learning Journals: A Handbook for Reflective Practice and Professional Development*, 2<sup>nd</sup> ed, Routledge, London
- Nicol, D.J. & Macfarlane-Dick, D. (2006). Formative assessment and self-regulated learning: A model and seven principles of good feedback practice. *Studies in Higher Education*, vol. 31(2), 199-218
- Nicol, D. (2009), Assessment for learner self-regulation: Enhancing achievement in the first year using learning technologies. *Assessment and Evaluation in Higher Education*, vol. 34 (3) 335 -352.
- Oliver, B., von Konsky, B., Jones, S., Ferns, S., and Tucker, B (2009): Curtin's iPortfolio: facilitating student learning within and beyond the formal curriculum, *Learning Communities: International Journal of Learning in Social Contexts*, ISSN 1329-1440, December e-Portfolio Edition, [http://www.cdu.edu.au/centres/spil/publications\\_ijlsc.html](http://www.cdu.edu.au/centres/spil/publications_ijlsc.html)
- Piaget, J. (1970): *Science of Education and the Psychology of the Child*, Orion
- Pintrich, P. R. and Zusho, A. (2002): Student motivation and self-regulated learning in the college classroom, in: J. C. Smart & W.G. Tierney (Eds) *Higher Education: handbook of theory and research* (vol. XVII) (New York, Agathon Press)
- Polanyi, M. (1958): *Personal knowledge: Towards a post-critical philosophy*, University of Chicago Press
- Queensland University of Technology (2010) Using STR L to show evidence of your experiences, <http://www.careers.qut.edu.au/student/resource/UsingS TARL.pdf> accessed 23 August 2010
- REAP: Re-Engineering Assessment Practices in Scottish Higher Education, <http://www.reap.ac.uk>
- Rittel, H. W. J., & Webber, M. M. (1973). Dilemmas in a general theory of planning. *Policy sciences*, 4(2), 155 169.
- Scanlon, J. M. and Chernomas, W. M. (1997): Developing the reflective teacher, *Journal of Advanced Nursing*, vol. 25(6), 1138–1143
- Schön, D. (1983): *The Reflective Practitioner*, Jossey Bass, San Francisco
- Schön, D. (1987): *Educating the Reflective Practitioner*, Jossey Bass, San Francisco
- Spradling, C., L.-K. Soh, and Ansorge, C.J. (2009): A comprehensive survey on the status of social and professional issues in United States undergraduate computer science programs and recommendations. *Computer science education*, 2009. **19**(3): p. 137-153.
- von Konsky, B., Oliver, B., Nikolettatos, P, and Wilkinson, H. (2010): Showcase You on iTunes U: The iPortfolio enables student self-assessment of key capabilities and the public showcase of achievements, Learning Forum London 2010, European Institute for E-Learning, London, UK
- von Konsky, B., Oliver, B., and Ramdin, A. (2009): The iPortfolio: Capture, Reflect, Share, Educause Australasia 2009: Innovate, Collaborate, Sustain (Educause Australasia 2009), Perth, Australia
- von Konsky, B. and Ivins, J. (2008): Assessing the Capability and Maturity of Capstone Software Engineering Projects, *CRIPT*, vol. 78, 22-25
- Wenger, E. (1998): *Communities of practice: Learning, meaning, and identity*, Cambridge University Press
- Wilson, K.L., Lizzio, A., and Ramsden, P, (1997): *The development, validation and application of the Course Experience Questionnaire*. *Studies in Higher Education*, **22**(1): p. 33-53.
- Wong, F.K.Y., Kember, D., Chung, L.Y.F. and Yan, L. (1995): Assessing the level of reflection from reflective journals. *Journal of Advanced Nursing*, vol 22: 48-57



# MyPyTutor: an interactive tutorial system for Python

Peter J. Robinson

School of Information Technology and Electrical Engineering,  
The University of Queensland,  
Brisbane, Australia,  
Email: pjr@itee.uq.edu.au

## Abstract

MyPyTutor is an interactive tutorial system for the Python programming language. The system provides support for both students doing tutorial problems and tutorial developers. MyPyTutor can be used in two modes: stand-alone or with online support. For courses where MyPyTutor is used for assessment it can be set up so that students submit correct solutions to problems online for marks.

*Keywords:* Automatic marking, interactive tutorials, Python, code analysis

## 1 Introduction

For some years we have been using xTutor from MIT (Ehrmann et al., 2006) for interactive tutorials, first for Scheme and more recently for Python. One of the disadvantages of xTutor is that the student enters Python code into a text box that provided little support for the kinds of support expected in an IDE such as syntax highlighting, layout and error feedback. It is also difficult to provide questions about GUIs where the student can see the results of their code. Also, students need to be registered, i.e. have a username and password, in order to use xTutor and students need to be logged on to the system in order to do the tutorial problems. On the other hand xTutor supports several kinds of tutorial questions other than coding questions such as multiple choice questions.

Python installations come with Tkinter (a GUI library based on Tcl/Tk) and an IDE written in Tkinter called IDLE. Most of our students, particularly Windows users, use IDLE for writing and testing their code. Furthermore, we use IDLE in lectures when writing code. One of the key design decisions for MyPyTutor is to provide, as much as possible, a familiar interface to students. Consequently, MyPyTutor is written in Tkinter and uses the same code edit window as is used in IDLE. Further, various kinds of errors, such as syntax errors, produce the same error highlighting in the edit window as when using IDLE. Also, by using Tkinter we are able to write tutorial questions about GUIs (written in Tkinter) where, as part of checking the code, the result of running the code can appear as a window, giving the student visual feedback.

Unlike xTutor, where everything is run on a server, MyPyTutor is an application that students download

to their machines. The collection of tutorial problems are also downloaded. This means that MyPyTutor can be run in a stand-alone mode if it is simply used as a resource for students, or if a student wants to work on problems without being connected. If MyPyTutor is used for assessment then it can be set up so that students can log on and submit answers to problems.

MyPyTutor supports only coding problems, i.e. students write their answer as code and the system checks their code for correctness.

In order to provide more feedback to students other than just telling them if they are right or wrong, we often use parsing of student code to either give more information about where the student went wrong or possibly provide suggestions on better ways of solving the problem. These techniques are not specific to MyPyTutor but could be used in any interactive system in any language for which examination of the parse tree is easy to do.

As discussed in Douce et al. (2005), most modern automatic systems for assessing student programs are web based (like xTutor) and so typically have similar advantages and disadvantages as discussed above for xTutor. Douce et al. point out that security is an issue for automatic marking systems. In particular, malicious or badly behaving student code can be problematic. This can be avoided by, for example, sandboxing the test code (as is done in xTutor). MyPyTutor does not suffer from this problem as all test code is run on the student machine.

MyPyTutor is currently being used for very introductory problems and so the testing, apart from analysis of the parse tree, is quite straightforward. In principle, though, it would be possible to write more sophisticated testing code as used in other systems such as AutoGrader (Helmick, 2007) and Web-CAT (Edwards, 2003).

In the next section we describe the student view of MyPyTutor. In Section 3 we describe the developers and course administrators view. In Section 4 we discuss the design of tutorial problems and in Section 5 we consider testing using parsing. In Section 6 we describe the implementation and we conclude with some remarks.

## 2 Student View

MyPyTutor consists of two windows. One window is for editing code and is the same as the IDLE edit window apart from some minor changes to the menus. The other window consists of two text widgets. The top one displays the problem being worked on as a rendered HTML document. The bottom text widget is used to output the results of testing the student code.

It is used to display a “Correct” message if the code successfully passed the tests or some kind of error or warning message otherwise. Any output from

student code will also be displayed in this widget. The Problems menu allows the student to choose a problem to work on. If MyPyTutor is set up for online use, the Online menu is present and allows the user to log on and off, change their password, submit the current (correct) problem answer, upload/download their (partial) solution to the problem, and view the submission status of the problems.

Each student can configure MyPyTutor by choosing fonts and the folders where the tutorial problems and the student answers for each problem is to be found. MyPyTutor supports multiple problem sets for students doing more than one course using MyPyTutor.

Typically, students start up the application and choose a problem to work on. If they have previously saved their code for the problem then the code will be automatically loaded from the answers folder on their machine. They might wish to overwrite this by downloading a previously uploaded version of their code (if they are logged on).

After working on the code they can check their code for correctness. If the code passes the test, and they are logged on, they can submit their answer to the server for marks. If the code fails the test they can edit their code and check again.

When students check correct code, MyPyTutor will respond with a message saying that the code is correct, and if MyPyTutor is set up for answer submission then it will also remind the students to submit their answers. If the code is incorrect, MyPyTutor responds by displaying an error message of some kind. All exceptions produced by student code are displayed and exceptions such as syntax errors or name errors will cause the offending line of code to be highlighted. The other error messages displayed are based on the results of testing the code. This could simply be a message saying that the code is producing the wrong value and what the correct value should be. It could be an error message that gives more feedback on where the student has gone wrong based on, for example, the use of parsing (see later). In some cases the system might provide some feedback as a warning and then continue testing. This typically occurs when the analysis of the student code suggests that the student may be, for example, using the wrong test for termination of recursion. In this situation it may be possible that the student code is correct but not using the expected test.

When MyPyTutor is set up for online use (and for marks), due dates are associated with each section in the Tutorials menu. If the student submits after the due date, the problem is marked as late and is not counted towards marks.

Figures 1 and 2 given an example of MyPyTutor in action. The user has chosen a problem to work on and has pressed the hint button to reveal the hint. The user has added the last line in the edit window (the other lines are pre-loaded). The user has checked the code and the output shows the result of checking the code. When the user corrects this problem by replacing the print with a return then, on checking again, the output window will display an error message such as "Wrong: for input 7 the correct result is 49, you got 14". This problem is used as an example in Section 5.

### 3 Developer and Administrator View

The developer is the person who creates individual tutorial problems and collects some or all of these problems into a tutorial set. The administrator is the person running a course where MyPyTutor is used for assessment. The MyPyTutor system comes with

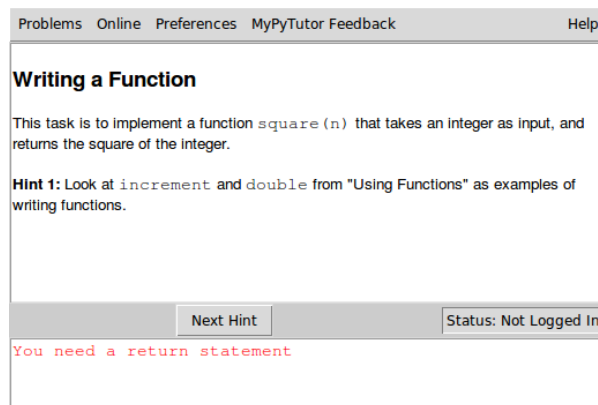


Figure 1: MyPyTutor Example: Problem Description and Output Window

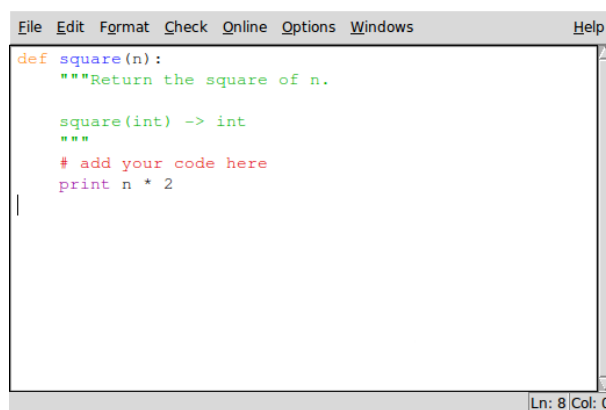


Figure 2: MyPyTutor Example: Edit Window

several tools to support the developer and administrator. All these tools are written in Tkinter and share some of the MyPyTutor code. The developer has the use of three tools: a problem creator tool, a problem checking tool, and a tutorial creator tool.

The problem creator tool is used to construct tutorial problems. It has two edit windows. One window is where the problem description is written in HTML source. The other window is where the testing code is written.

The problem testing tool is a cut-down version of MyPyTutor that the developer can use to test to see if the problem he/she has written behaves correctly from a student point of view. These two tools are designed to work together, making it easy to edit and check problems.

The tutorial creator tool is used by the developer to create a collection of tutorial problems to be used in a course. This consists of an editor window in which the developer can list the tutorial sections and each problem within each section together with any auxiliary files (e.g. GIF files for images in the HTML). If the system is intended for online use, the first line is the URL of the MyPyTutor server and due dates are added to the section headers.

When the developer has finished designing the tutorial set, the tutorial set can then be exported for use by the student. This consists of copying all the required files to the designated folder, encrypting the test code part of each problem, and zipping the contents of the folder.

The system has been designed in such a way that, if it is used in online mode, any modifications to the problems can be uploaded to the server and when a student next logs on, the updated problem set will be downloaded to replace the old version. This means

that any errors found can be corrected and an update problem set distributed to students is a straightforward manner.

The administrator tool is used to manage student information when MyPyTutor is used in online mode. The administrator typically starts by registering students (i.e. create usernames and passwords) individually or from a class list in a particular format. At any time, the administrator can change the password of a student. The administrator can search for matching patterns in the user file (e.g. search for family name or username).

As was discussed earlier, if a student submits a problem late then no marks are awarded for that problem. The administrator has the ability to unset the late flag on a problem so that it will be counted.

The administrator can also extract results from the server. At the end of the course (or indeed at any time) the administrator can collect the marks for all the students and can also obtain information on the percentage of the class that have submitted each question. As a very rough guide to determining how hard students found a particular problem the average number of times a given problem was checked in the current session can be displayed. Although this is a very rough estimate of difficulty, it seems to provide a good guide in practice. This information might be used by the problem designer, for example, to reword the problem description or add extra hints for problems students appear to be having problems with.

#### 4 Problem Design

Writing the problem description is reasonably straightforward. It is written in HTML source with a restricted number of tags. Images in GIF format can be embedded in the problem description. It's also possible to add hints. If a problem has one or more hints then a Hints button appears and each time the button is pressed the next hint is displayed in the problem description widget. Hints are also written in restricted HTML.

The other component of problem design is the test code. Any number of different tests can be run on the student code and any test can be repeated a number of times (for example, if random numbers are used to generate values). Tests can be straightforward, for example, simply comparing the answer produced by student code with the expected answer. The tests can also be quite sophisticated. For example, the student code can be parsed and tests carried out based on the structure of the code (see later).

For the introductory course in which MyPyTutor is currently being used, the problems are all very straightforward and so, if simply testing the answer is all that is required, the test code could be very simple. However, experience with using xTutor suggests that simply giving the results of the tests to beginner students is not very helpful. In response, many of the problems use parsing to provide better feedback. As students and problems become more sophisticated, parsing is probably less necessary and so the testing code would typically be much simpler.

An example of test code is given in Figure 3. The problem asks the student to assign to the variable `prod` the product of the variables `item1` and `item2`.

The comments such as `#{global}#` are headers for the different blocks of the test code. The main part of the testing code is the test blocks (with the `#{test}#` header). There needs to be at least one of these blocks. When the student code is checked, the code of each test block is run until either an error occurs or all tests have completed. There are three blocks that may appear before the first test block.

```
#{global}#
import random
#{test}#
#{start}#
item1_save = random.randint(2,100)
item2_save = random.randint(2,100)
#{init}#
item1 = item1_save
item2 = item2_save

#{code}#
import sys
if prod == item1_save*item2_save:
    correct()
else:
    print_error("Wrong - You got %s \
the correct result is %d \n\
The right hand of the assignment should be \
an expression involving item1 and item2" \
% (str(prod), item1_save*item2_save))
```

Figure 3: Test Code Example

The `#{global}#` block contains code that is common to all tests. In this case it imports the random module but it could be, for example, a class definition to be used in testing or some function definitions students are expected to use. The `#{preload}#` block (not used in this example) contains code that is copied into the edit window when this problem is chosen. It might, for example, be a partially defined function that the student needs to complete (Figure 2 shows an example). MyPyTutor uses a timeout (default one second) to stop runaway student code after the given time. For some problems this may not be enough time. The block of the form `#{timeout = secs}#` (with no body) allows the designer to specify the timeout for running the tests.

Each test block can contain three subblocks: the start block, the init block, and the code block. The first two are optional. When a test is executed, the code in the start block is first executed in an environment that cannot be accessed when running the student code. Then the init code is run in environment that has access to the start environment. Then the student code is executed followed by the code in the code block. The reason for this rather complicated setup is to prevent, in this example, the student code

```
prod, item1, item2 = 0, 0, 0
```

from being a solution.

By keeping a copy of the values of `item1` and `item2` in an environment that can't be accessed by the student code, then the proper test can be carried out. This is discussed in more detail in Section 6.

When using random numbers for testing it is often necessary to run a given test multiple times in order to be reasonably confident the student code is correct. This can be done by using the header `#{test repeats = 3}#`, say, rather than `#{test}#`.

Exception handling is an important issue when testing student code. To hide details of the test code all exceptions produced by the testing code (that are not explicitly handled in the test code) are trapped and the error message 'Error: report to maintainer' is output. This should be avoided at all costs as it gives no information to the student about the cause of the problem. It is therefore important to structure the test code so this does not happen.

Below is a fragment of test code for a problem that calls a function `f` defined by the student.

```
try:
```

```

    result = f(test_arg)
    ok = True
except Exception, e:
    print_exception(e)
    ok = False

if not ok:
    pass
elif result == .....

```

In this example, if no exception is thrown by the student code then the result is tested for correctness. If, however, an exception is thrown the exception is passed to a MyPyTutor function that displays the exception and further testing is short-circuited.

Now consider an example where an integer value is expected as the result of calling a student defined function (as in the above example). It is tempting to write test code as follows

```

if result != 42:
    print_error("Wrong: you got %d \
the correct answer is 42" % result)

```

where `print_error` is a MyPyTutor function that displays the message. The problem is that, if the student returns something other than an integer (`None`, for example), then an exception will be thrown by the formatting. This can be avoided by using either `str` or `repr` as follows.

```

if result != 42:
    print_error("Wrong: you got %s \
the correct answer is 42" % repr(result))

```

Another place where exceptions can crop up is in walking the parse tree of the student code. If the problem designer assumes too much about the shape of the student code then it's easy to produce an exception if the student writes "unexpected" code. So, although parsing student code can provide excellent feedback, the parsing code must be carefully crafted to avoid this problem.

## 5 Parsing Student Code

Parsing is a powerful tool for finding certain kinds of errors and for providing more detailed feedback to students than simply saying what the student code produced and what the correct answer was. The ideas presented below are not specific to MyPyTutor or to Python.

In the following example we present fragments of a relatively straightforward example using parsing. The tutorial problem is to write a square function that takes an integer and returns its square. This is an introductory example and as such the student might be having trouble with return statements, the use of arithmetic operators and the use of formal parameters to functions. By walking the parse tree of the student code we can check if this is being done correctly or provide some feedback.

Python comes with a module called `compiler` that has a function for constructing the abstract syntax tree (AST) of the student code. It also has a function that takes an AST and a code visitor object (defined by the problem designer) and visits nodes according to the methods defined in the code visitor class. An example of such a class is given in Figure 4.

The code for using the parse tree is given below.

```

ast = compiler.parse(user_text)
visitor = CodeVisitor()
compiler.walk(ast, visitor)
if not visitor.has_return:

```

```

class CodeVisitor:
    def __init__(self):
        self.arg1 = None
        self.in_defn = False
        self.has_return = False
        self.use_mult = False
        self.use_power = False
        self.in_mult = False
        self.in_power = False
        self.use_arg1 = False

    def visitFunction(self,t):
        if t.name == 'square':
            self.in_defn = True
            if len(t.argnames) == 1:
                self.arg1 = t.argnames[0]
            for n in t.getChildNodes():
                compiler.walk(n, self)
            self.in_defn = False

    def visitReturn(self,t):
        if self.in_defn:
            self.has_return = True
            for n in t.getChildNodes():
                compiler.walk(n, self)

    def visitPower(self,t):
        if self.in_defn:
            self.use_power = True
            self.in_power = True
            for n in t.getChildNodes():
                compiler.walk(n, self)
            self.in_power = False

    def visitMul(self,t):
        if self.in_defn:
            self.use_mult = True
            self.in_mult = True
            for n in t.getChildNodes():
                compiler.walk(n, self)
            self.in_mult = False

    def visitName(self,t):
        if self.in_defn and \
            (self.in_power or \
             self.in_mult):
            self.use_arg1 = \
                self.arg1 == t.name

```

Figure 4: Code Visitor Class

```

    print_error('You need a return \
statement')
elif not (visitor.use_mult or \
          visitor.use_power):
    print_error('You should use either \
* or **')
elif not visitor.use_arg1:
    print_warning('Are you using the \
variable %s in the body of the \
definition?'% visitor.arg1)

```

By doing this we give feedback if, for example, a return statement is not used. This is a common mistake by beginners who often use a print statement rather than a return statement.

As another example, if the question asked the student to write a for loop to solve a problem but the student produced a correct solution using a while loop then straightforward testing is not enough to detect the non-use of a for loop. By parsing we can give feedback that a for loop should be used.

## 6 Implementation

MyPyTutor is implemented in Python using the Tkinter GUI library. One important aspect of the implementation of MyPyTutor relates to the fact that the application itself and the tutorial problems are loaded on to the student machine. This differs from server or web based systems like xTutor where all the code lives on the server. Because the xTutor code is on the server, the student does not have access to the implementation of xTutor or the problem testing code. In MyPyTutor, however, the student has all the code on their machine. Consequently, attempts to hide the application code and the problem testing code has been made. Firstly, apart from the top-level interface to the MyPyTutor application, the support code is compiled to pyc files. We believe this makes it sufficiently difficult for students to decompile the support code in order to understand how the code works.

In order to hide the problem testing code, when the tutorial developer exports the set of tutorial problems, the testing code for each problem is encrypted using a moderately strong and fast algorithm. When a student chooses to work on a given problem MyPyTutor renders the HTML problem description and decrypts the testing code ready for use.

Admittedly this is “security by obfuscation” but we feel it is good enough to hide the code from all but the most talented students.

The MyPyTutor code edit window inherits from the IDLE code edit window. By doing this, students use the same editor for doing MyPyTutor problems as they do when doing their assignments or other Python coding. Furthermore, MyPyTutor traps exceptions such as Syntax errors, extracts the line number for the error and highlights that line in the same way as is done in IDLE.

For rendering the HTML for the problem description, the HTMLParser module is used. We currently support the following tags.

```
h1 h2 h3 h4 h5
b i t t
br p ul li
pre
img
span
```

The `img` tag is of the form `img src="filename.gif"` - i.e. the image must be a GIF file. Currently, only `span` with a colour style of `red`, `green` or `blue` is supported.

The key part of the implementation of MyPyTutor is the testing of student code. When the testing code is run on the student code, first a global environment is created.

```
global_env = \
    {'user_text': self.user_text,
     'print_warning': self.print_warning,
     'print_error': self.print_error,
     'print_exception': self.print_exception,
     'correct': self.correct,
     'master': self.master}
```

This environment provides the testing code with access to MyPyTutor code: `user_text` is the student code; `print_warning` displays a warning in the output text widget; `print_error` and `print_exception` display messages and set a flag that will terminate testing; `correct` displays a message that the code is correct and sets a flag to signal this; and `master` provides a mechanism for students to write GUI code that will be displayed in a window.

```
1: test_globals = global_env.copy()
2: try:
3:     exec test.get('start', '') in \
4:         test_globals
5: except Exception, e:
6:     raise TestError(e)
7: save_globals = test_globals.copy()
8: locs = {}
9: try:
10:    exec test.get('init', '') in \
11:        test_globals, locs
12: except Exception, e:
13:    raise TestError(str(e))
14: exec self.user_text in locs
15: test_globals.update(locs)
16: test_globals.update(save_globals)
17: try:
18:    exec test.get('code', '') in \
19:        test_globals
20: except NameError, e:
21:    raise(NameError(e))
22: except Exception, e:
23:    raise TestError(str(e))
```

Figure 5: MyPyTutor Testing Code

The core part of the testing code is listed in Figure 5. First the global environment is copied into another environment that will be modified during testing. On lines 3 and 4 the start code (if present) is evaluated in the `test_globals` environment, updating that environment. That environment is then copied to `save_globals` for later use. Then on line 8 an empty (local) environment is created and on lines 10 and 11, the init code (if present) is evaluated relative to the given global and local environments, updating the local environment. Next, on line 14, the student code is evaluated in the local environment, updating that environment. On lines 15 and 16, the local environment is merged with the global environment and, just in case the student redefines something in `global_env` or in the start code, `test_globals` is then updated again with `save_globals`. Finally, on line 18 and 19, the test code is evaluated in the `test_globals` environment.

By manipulating environments in this way, the student cannot gain access to, or modify, the information used for testing but the testing code can access all required information needed to test the student code.

When MyPyTutor is run with an online component it is possible to automatically update both the tutorial problems and even MyPyTutor itself. Whenever a student logs on, the date stamp of the tutorial problems is compared with the date stamp on the server. If the server has a more recent version then it lets MyPyTutor know that a newer version is available and in response MyPyTutor downloads the zip file and overwrites the old version with the new one. The student is notified that an update has happened. The same thing happens with MyPyTutor itself, except the version numbers are used for comparison.

## 7 Remarks

MyPyTutor was used for the first time for assessment in a first year software engineering course in semester 1, 2010. Several errors were discovered with the tutorial problems. Some errors were to do with poor wording and others were to do with code testing as discussed earlier. Because of the automatic update facilities of MyPyTutor it was easy to fix errors and

distribute a new version quickly. The ability to update MyPyTutor itself also turned out to be helpful as students discovered several errors when using the application.

MyPyTutor provides a feedback facility so that students can provide feedback on both MyPyTutor itself and on specific tutorial problems. Feedback from students was mostly very positive.

More details of MyPyTutor, developer tools, and the current database of problems can be obtained from the author.

## References

- Stephen C. Ehrmann, Steven W. Gilbert and Flora McMartin (2006), Factors Affecting the Adoption of Faculty-Developed Academic Software: A Study of Five iCampus Projects, TLT Group, <http://icampus.mit.edu/index.shtml> (last referenced 13/08/2010).
- Christopher Douce, David Livingstone and James Orwell (2005), Automatic Test-Based Assessment of Programming: A Review, *in* 'Journal of Educational Resources in Computing', Vol. 5, No. 3, ACM Press, New York.
- S.H. Edwards (2003), Using test-driven development in the classroom: providing students with automatic, concrete feedback on performance, *in* 'Proceedings of the International Conference on Education and Information Systems: Technologies and Applications', pp. 421-426.
- Michael T. Helmick (2007), Interface-based Programming Assignments and Automatic Grading of Java Programs, *in* 'ITiCSE 07: Proceedings of the 12th annual conference on Innovation and technology in computer science education', ACM Press, New York, pp. 63-67.

# A Pedagogically Rich Interactive On-line Learning Platform for Network Technology Students in Thailand

W. Makasiranondh, S.P. Maj, D. Veal

School of Computer and Security Science

Edith Cowan University

2 Bradford St., Mt. Lawley 6050, Western Australia

wmakasir@our.ecu.edu.au, p.maj@ecu.edu.au, d.veal@ecu.edu.au

## Abstract

Internetworking enables communication between networks and forms the foundation of the Internet. Internetworking teaching is typically conducted in a traditional face-to-face classroom, but nowadays it can be conducted online. Online learning environments have many advantages that include allowing remote students' access to not only curriculum but also lecturers and other enrolled students. However, unlike some other disciplines, teaching internetworking courses online is problematic because students need to be given access to internetworking equipment. It is technically possible to provide remote access to online students in order to compensate for the lack of direct physical equipment access, which normally is offered to traditional students. However the standard method of remote access only provides students with a limited text based method of configuring internetworking devices. Internetwork simulators are of value but they cannot provide students experience working with real devices. A pedagogically rich, interactive on-line learning environment using low-cost, assistive multi-media based technologies was therefore developed. This paper presents details of the platform and results of its deployment from an Australian university to a small group of students in Thailand.

**Keywords:** E-learning, remote access laboratory, network education, internetworking education, State Model Diagram, distance learning, pedagogy

## 1 Introduction

On-line learning, also referred to as e-learning, is an essential part of many modern university courses. This mode of instruction is not only cost effective but it also provides educational opportunities to students on a global scale. Some students, those who are geographically isolated or have competing commitments, for example, might be precluded from a more traditional educational environment. Furthermore, many students have grown up with communication technologies that have influenced their preferred learning style (Dede 2005). The 'net generation' is believed to have developed aptitudes and expectations based upon their daily use of technologies such as email and instant messaging (Gulatee and

Combes 2008). On-line curricula typically provide both asynchronous (email, podcasting, discussion boards) and synchronous (instant messaging, voice over IP) methods of communication to students. Online-learning offered alternatives.

### 1.1 Network technology education

Within the field of network technology education, practical, hands-on skills are of paramount importance; particular issues may be arisen with this requirement. Ideally students should be provided with the opportunity to interact with network devices. Hands-on activities are suggested as an important component in learning (DiCerbo 2009). Hands-on workshops not only enhance learning but also provide students with practical skills that are demanded by potential employers. This can also be an important factor in enabling students to obtain initial employment in the industry. In order to provide curricula relevant to employer expectation the world's largest suppliers of network equipment, Cisco, developed the Cisco Network Academy Program (CNAP). CNAP defines the global standard by which students can learn about and be assessed in network technology and is offered in over 11,000 academies in 162 countries with over 500,000 students worldwide. It is the most widely used network curriculum and the international standard by which professional competency can be measured. CNAP regards hands-on skills as a key graduating factor.

In order to build hands-on skills, a proper teaching facility and laboratory equipment is required. Even though low cost network equipment is available to be purchased on CNAP, or second-hand on the internet, network laboratories normally require class sets which incur on-going technical support and maintenance. Such capital expenditure is likely to be beyond the means of many institutions in developing countries.

Various simulation tools can be of use; however, they cannot provide students with the opportunity to interact with actual internetworking devices. Furthermore, simulation results may be limited to the quality of simulation tools, which are not recommended to be used solely as a replacement of the actual laboratory (Cisco 2009).

Hence providing hands-on skills to students via remote access is an important challenge; this in itself includes potential technical problems, namely the bandwidth and reliability of the communication links. While it is possible to provide on-line students with remote access to internetworking devices, users also must interact with actual devices by using only the text-based Command Line Interface (CLI). The CLI is complex, verbose and

syntactically difficult to use. CLI uses words to describe the status and behaviour of the laboratory devices. Words, or symbolic description, are the most advanced stage of learning from cognitive revolution theories according to Bruner (1966). Hence, the CLI alone may be unsuitable to be utilised as an educational tool, as it is intended to be used by experienced professionals in the field.

In order to address the problems of students using the CLI, State Model Diagrams (SMDs) were developed and introduced by Maj (Maj and Kohli 2004). SMDs are a diagrammatic method for representing network devices and protocols (Maj, Murphy and Kohli 2004). According to Maj, this diagrammatic method of interacting with internetworking devices has been clearly demonstrated to enhance learning (Maj, Kohli and Fetherston 2005). The diagrams intrinsically demonstrate concurrent relationships. For example, the diagrams show not only the interface MAC and IP addresses but also the associated Address Resolution Protocol (ARP) table. Students are therefore able to observe relationships, which are the basis of higher order learning in the SOLO taxonomy (Biggs and Collis 1989). Student learning based on SMDs demonstrates a richer conceptual understanding strongly aligned with that of an expert (Maj, Kohli and Fetherston 2005).

Furthermore, as a diagrammatic technique, SMDs are independent of the language of instruction. For example, an SMD-based network curriculum was evaluated in the context of Japanese Professional Graduate Schools. According to Akamatsu, Ohtsuki and Maj (2007),

*"The results strongly suggest that using these methods the students constructed an advanced understanding of network concepts. The results suggest that these diagrams strongly encourage 'deep' multi-structural understanding".*

Meanwhile, learning style in a developing country may be limited to traditional face-to-face classrooms, due to a lack of appropriate technological infrastructure. Students who shift from a traditional learning style to distance education may suffer from a lack of immediate feedback and interaction within a traditional classroom setting (Barnes 2003). Introducing a remote access laboratory for students from developing countries that provides only a limited degree of interaction with a learning environment may result in suboptimal educational experiences. Such a situation might not occur if a higher degree of interaction were possible.

Therefore, the availability of remote access laboratories incorporating the use of SMDs could benefit developing countries. This paper will investigate the suitability of introducing remote access internetworking laboratories in Thailand. In section two, the paper will review the previous literature of implementing such laboratories, particularly those attempted within network technology courses. We will also look at the current situation of network technology education within Thailand. Section three will be used to describe the research process. Section four will demonstrate the findings of the study, followed by a discussion of the results in section five. Conclusions of the study will be provided in section six.

## 2 Previous work

### 2.1 Remote access laboratory in general

The use of remote laboratories in other engineering education disciplines has been well established. According to Machotka, Nedić, Nafalski and Göl (2010), and Nafalski, Nedić, Machotka, Göl, Ferreira and Gustavsson (2010), the use of online remote laboratories can lead to collaboration between universities. Providing a remote laboratory along with a traditional hands-on laboratory was proved to be a valuable solution (Melkonyan, Akopian and Chen 2009). These were all successful examples of integrating remote access laboratory in online or e-learning environments for distance education. Such distance environment can be very important in societies.

In order to provide remote access laboratory successfully, Tomov (2008) has noted two essential elements; action and response. Action will allow users to control laboratory equipment remotely; while response will report the status of the equipment to the users and let them perceive the laboratory practice results. This therefore means that there should be responsive and meaningful feedback from the devices in exercises.

### 2.2 Remote access laboratories in internetworking education

A number of implementations of remote access laboratories have been used in the field of internetworking education. Commercial tools such as Netlab+ can provide access to real networking hardware (Prieto-Blazquez, Arnedo-Moreno and Herrera-Joancomarti 2008). However, the cost effectiveness of commercial tools is still an issue. One alternative to reduce the cost may consider an option to increasing the number of users. According to Jakab et al. (2009) sharing equipment by remote access is routinely conducted by universities.

However, the use of a primitive remote access laboratory may lead to frustration. Yet, comparative studies of remote access laboratories and traditional laboratories were undertaken and the conclusions favoured the use of remote access laboratories (Aravena and Ramos 2009, Lawson and Stackpole 2006). One of the factors may be that distance laboratory can be more suitable for a wider range of students. For instance, an example of providing remote laboratory to vision-impaired students has also been investigated (Murray and Armstrong, 2009).

It may be concluded that the field of remote access laboratory provision has been extensively investigated. This has mainly been focussed upon providing access to a physical laboratory which is an action element according to Tomov (2008). However, most remote access laboratories investigated were based upon text-based interaction with network devices.

### 2.3 Internetworking education in Thailand

Thailand also has a long term focus on building a strong and effective e-learning facility as part of the country's main development plan. Although, Thai universities commenced the development of e-learning in 1994, the



Master planning for educational ICT usage was only commenced in 2004 (Laohajarsang 2009). This plan was endorsed in the national policy statement delivered by the prime minister in 2008, as a means of supporting further education (Vejjajiva, 2008). The corresponding policy from the Ministry of Education of Thailand also promoted e-learning throughout the education system from primary school to university level (Ministry of Education of Thailand, 2010). The current investment plan for the financial year of 2010 to 2012 also reflects this trend by continuing to support the building of e-learning facilities as well as the development of digital courseware (Ministry of Education of Thailand, 2009). However, this plan appears to have been affected by the global financial crisis. Meanwhile, in the last decade Thailand has experienced a lack of technological facilities to deploy e-learning, such as national broadband internet, limited bandwidth of local network and relatively low numbers of computers throughout the educational system (Sirinaruemit, 2004).

Furthermore, the take-up of e-learning of within internetworking education in Thailand, compared to that in Australia, can be indicated by considering the statistics of the Cisco Network Academy Program (CNAP) for both of these countries. According to Cisco (2009), if we compare the number of institutes using the CNAP program and the higher education institution list provided by the International Association of Universities (IAU), we find that the majority, around 90%, of Australian institutions are schools or vocational education campuses. Very few are higher level educational institutions. In contrast, only 13% of the institutions using CNAP in Thailand are not tertiary. Australia has embedded e-learning technologies in the school systems to a greater extent than Thailand has yet been able to achieve. More specifically, Australia is using technology to introduce school students to the field of internetworking education at a much earlier age. Thailand is yet to take up this challenge. Several reasons may have contributed to this situation, for example, the availability of networking equipment for schools, computer facilities, the training of teachers and technical personnel. E-learning by electronic media in Thailand is still considered as novel and under development (Lertkulvanich et al. 2008).

However, Thai universities have started to encompass e-learning technologies. Suanpang and Petocz's (2006) study found positive results arose from the provision of e-learning in standard units offered by a university. Significantly improved grades were achieved by students who were enrolled in the online mode of their courses. However, examples of e-learning application to the area of internetworking education are still limited. Therefore, introducing other more affordable forms of providing internetworking education may assist in enabling its take-up at earlier educational levels in Thailand. These considerations led the authors to collaborate with a Thai university to undertake the following research.

### 3 Research

#### 3.1 Objective of the study

The objective of this study was to investigate the suitability of introducing remote access laboratories in

Thailand by designing and using a multi-media based teaching platform that employs commonly used low-cost technologies to enhance the learning experience of remote students.

#### 3.2 Experimental design

The dedicated internetworking laboratory was located in an Australian university. This laboratory had multiple class sets of internetworking devices. An Access server made it possible to provide access for individual remote students in Thailand to specific devices in the Australian laboratory (Figure 1). The server was configured to provide secure access over the Internet. The only technical requirement for the on-line students was a PC and an Internet connection.

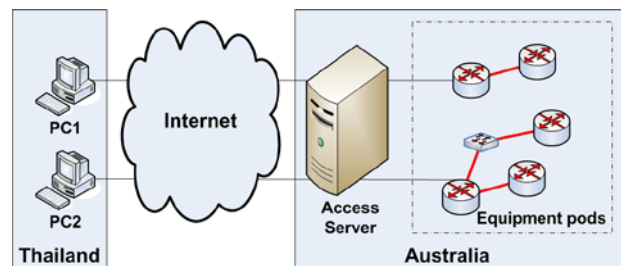


Figure 1: Diagram showing connectivity

As the laboratory operated in a normal mode, the standard method of configuring internetworking devices was the Command Line Interface (CLI). An example of the *show ip route* command is shown in Figure 2.

```
Router3>
Router3#en
Router3#show ip route
Codes: C - connected, S - static, R - RIP, M - mobile, B - BGP
        D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
        N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
        E1 - OSPF external type 1, E2 - OSPF external type 2
        i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2
        ia - IS-IS inter area, * - candidate default, U - per-user static route
        o - ODR, P - periodic downloaded static route

Gateway of last resort is not set

20.0.0.0/24 is subnetted, 1 subnets
C    20.1.1.0 is directly connected, Ethernet1/0
R    200.200.2.0/24 [120/1] via 192.168.2.2, 00:00:22, FastEthernet0/0
C    200.200.3.0/24 is directly connected, Loopback0
R    10.0.0.0/8 [120/1] via 192.168.2.2, 00:00:22, FastEthernet0/0
C    192.168.2.0/24 is directly connected, FastEthernet0/0
Router3#
```

Figure 2: Example of input and output CLI

However, the output from multiple CLIs can be captured using a single SMD (Figure 4). This is important because students are able to observe how the different protocols interact.

The SOLO taxonomy category's definitions of learning are: unistructural, multistructural, relational and extended abstract (Biggs and Collis 1989). Relational learning is the integration of several aspects so that the whole has a coherent structure and associated meaning. SMDs are able to provide the basis of relational or higher order learning.

Typically interaction with a physical object is the initial phase of model development that is later modified to a more conceptual construct. It is important therefore for students to actually 'see' the internetworking devices they configured by means of webcam. This is an element which supports the enactive-learning-stage, by letting students learn by the interaction of physical objects (Barnes 2003).

Lecture material was presented in Australia to the Thai students via WebEx which incorporates Voice over IP (VOIP) and integrated Webcam. WebEx is a commonly used, low-cost method for delivering web-based conferences.

The Thai students were therefore provided with an integrated learning platform, all of which was displayed on a single PC screen (Figure 4).

20.1.1.1 ("Router3")			
Layer 3 Internet	ARP Table		
	Interface	MAC	IP
	1	00-0C-30-B4-35-60	192.168.2.1
	2	00-0C-30-B4-35-70	20.1.1.1
Layer 2 network	Route Table		
	dest	interface	metric
	10.0.0.0	1	1
	20.1.1.0	2	0
Layer 1 line	Address table		
	interface	address	
	E1/0	20.1.1.1	
	Fa0/0	192.168.2.1	

Figure 3: SMD diagram of router

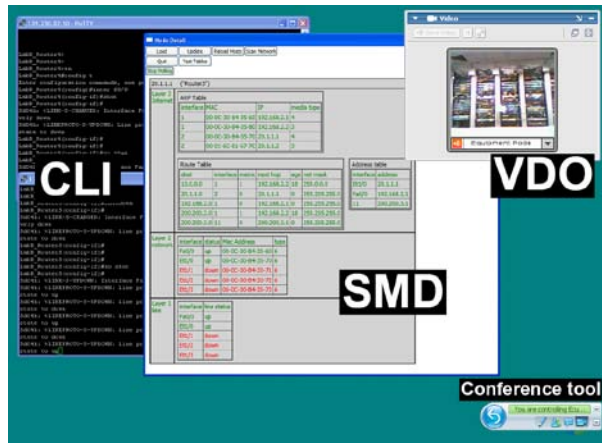


Figure 4: PC's WebEx screen shot showing Webcam, CLI and SMD

### 3.3 Data collection and analysis

The learning platform was evaluated by students, who were enrolling in Cisco Certified Network Associate Program (CCNA). Participants had previous experience in working with Cisco simulation software, Packet Tracer. Therefore, the participants had some familiarity with the equipment and the content of the exercise used in the study.

The participants were asked to sit in the online classroom for two hours in total. The first session was used for an introduction and housekeeping. Then, a lecture about a simple topic in networking technology

was given through online web conference software. After the lecture, the participants were able to use the laboratory through the online connection. The remotely located Thai participants were asked to configure network equipment according to the learning material from the lecture. After configuration they were able to see responses from the command line interface (CLI), a video camera and state model diagrams (SMDs). At the end of the session participants were invited to complete a questionnaire asking for their opinion on using SMDs in a remote environment. The questionnaire consisted of closed questions, which are using five point Likert scale (from 1, strongly disagree, to 5, strongly agree). Open ended questions were also available for the participants to write any further details about their opinions.

Due to the distance between researchers and participants, the initial data collection was undertaken via a paper-based questionnaire. The data analysis report was pre-designed before the actual data collection process by using a web-based tool (Qualtrics). This tool could then be reused with a larger number of participants in the future study phases.

### 3.4 Questionnaire

The questionnaire used in this survey is consisted essentially of three parts. The first part contained questions pertaining to the participant experiences of the current facilities provided by their university. The second part contained questions relating to the participants' experiences of using the remote laboratory. Whilst the third part of the questionnaire asked about participants experiences of using the SMD application. Some key questions are presented below.

- Participants' home university laboratory:  
Q1: Do you have the access you would like to your own university laboratory?
- Opinions on remote access laboratory:  
Q2: Do you think your university should provide remote access facilities and use it for teaching computer networking?  
Q3: When comparing the remote lab with traditional laboratory access, by which mode would you prefer to be taught?  
Q4: When comparing the remote lab with simulation software, by which mode would you prefer to be taught?  
Q5: When considering the effectiveness of remote instruction compared to an instructor available locally, which mode of instruction you prefer?  
Q6: Do you think you have more freedom by working with a remote instructor without any supervision from a local instructor?  
Q7: Do you need a local instructor to be with you when the remote instructor was available to help during the remote lab time?
- Experience of using SMD online:  
Q8: What are your preferable combinations of learning tools form the following: CLI, Video stream of real network equipment, SMD application?  
Q9: Do SMDs assist retention of knowledge gained via the laboratory?

Q10: Do you think that SMD is a useful tool for when you learn via remote internetworking environment?

### 3.5 Data sample

There were twelve participants interested in attending the remote teaching and laboratory session. There was a time difference between Thailand and Perth of only one hour. A combined two hour laboratory and lecture session was conducted independently from the participants' original coursework. However, only seven participants had completed the questionnaire by the time this paper was written and there was only limited number of responses. These preliminary investigations were undertaken to discover possible issues prior to the commencement of a larger scale study.

## 4 Results

The results from Q1 shows 43% of respondents do not feel satisfied with their current level of access to laboratory network equipment. The main reason was that students are allowed to use the equipment only in the scheduled class time.

Consequently, 71% of respondents felt that their institutions should provide further access via a remote access facility (Q2). The volunteers indicated remote access was a preferable method of self practice when compared to the traditional face-to-face laboratory (Q3), by 85% comparing with 15%. When comparing student attitudes to remote access laboratories and simulation software (Q4), 42% percent of the respondents preferred a remote access laboratory, while increasingly 28% still preferred the use of simulation software.

Furthermore, results from Q5 shows 42% percent of respondents agreed that they prefer to have the same instructor available locally when they are using a remote laboratory. Correspondingly, 56% did not enjoy the freedom of practicing in a remote laboratory without supervision from local instructors (Q6). One of the comments from the respondents is as stated below:

*"We need a local instructor to stay with us, as the remote instructor may not be able to rectify any usage problem on time."*

More than 85% of the respondents feel that it is necessary to have a local instructor with them during the laboratory time, even though the remote instructor was present (Q7).

When asked the participants to give a score (from 1 – 5 of Likert five point scale) to the preferred combination of learning components that should be available on the remote access network laboratory (Q8), the majority would like SMDs, video webcam and CLI to be available. Table 1 shows the mean score and standard deviation of each choice. A choice which contained only CLI has the lowest mean score amongst all combinations. Most of the participants would not like to have only CLI available on the remote access laboratory, even though they are already familiar with Cisco's device commands. However, within the lower range of score deviation (0.76 - 0.79), the combination of providing CLI and video webcam showing real-time equipment has the highest mean.

Appreciation score	Mean	SD
CLI only	2.71	0.76
CLI and Webcam	3.57	0.79
CLI and SMD	3.43	0.79
CLI, SMD and Webcam	4	1.15

Table 1: Preferable learning components

When asked about the pedagogical value of SMD (Q9), 71% of the respondents agreed that SMDs help them retain the knowledge from the laboratory exercise. This confirmed the benefit of SMD to the students according to Akamatsu et al. (2007). Particularly, a similar proportion of the respondents agreed that SMDs helped them during their learning process in the remote access laboratory (Q10).

The data gained from this preliminary study may indicate benefits of using a remote access laboratory and clarify the preferred learning style of Thai students. The result may also identify student perceptions of the presented remote laboratories when compared with traditional and simulation laboratories.

## 5 Discussion

### 5.1 Discussion and future work

The results may indicate a high demand from Thai network students for laboratory facilities to be used for practice using internetworking devices. They also indicate the necessity of providing other means of practising for students. Further results from Q5-Q7 show that Thai students may lack experience of working with a remote access laboratories and learning in an e-learning environment. Students tend to prefer to study in a traditional style by using real equipment available on site with a local instructor. Such traditional mode of delivery is what Thai universities currently attempt to provide. Furthermore, Thai learning styles may rely more on the presence of local teachers and indicate the extra responsibility placed upon remote lecturers. Interestingly, this extra need may be shown by the participant's comment below:

*"The remote instructors may have some difficulty to control the local student to pay attention to the class."*

However, when it comes to offering practice time for laboratory exercises, the participants also realize the value of practicing by using the remote access laboratory. The laboratory's availability outside the scheduled class times can offer more flexible access to the students.

When asked to compare the simulation laboratory with the remote access laboratory, although the majority, 42% of participants still favoured the remote laboratory, some participants indicated that they would still prefer to use simulation software. This may be because the simulation software can offer more flexibility of access, even surpassing the remote access laboratories. Furthermore, the students have been exposed to the simulation software for a long period and may be used to it. Also, the lack of

accuracy of the simulation tools may not be a concern from the student perspective at this stage. This can be illustrated by the following participant's feedback: *"Using simulation software is similar to using real equipment in every detail."* However, the majority still prefers the remote access laboratory.

When using the provided remote access equipment, students may not realize that they are actually working on real equipment. A comment to illustrate this point is: *"I prefer to use real equipment rather than remote access laboratory"*. This also pointed out the need of providing more responsive media to the laboratory's interface in order to make the students feel the situation was more realistic.

When considering the part of result from Table 1 that has only the lower range of deviation, the participants were interested to have the combination of CLI and video webcam showing the equipment when they were doing exercises. This may be because they were seeking a similar working environment to the face-to-face laboratory, where they can see the actual equipment. The same result can also indicate that the participants may not be familiar with the SMD software used in this study, as the software was introduced as a new teaching medium. This could suggest the future work to employ an extension of learning session for building tools familiarity.

The differences between the mean scores in Table 1 provide an indication that CLI alone in the remote access laboratory was not an effective solution. This could support the consideration of symbolical CLIs lower pedagogical value. Integrating other means of teaching media such as SMDs may benefit the student learning process. This has demonstrated the need for multimedia pedagogy-rich learning environments for remote students, who may lack the encouragement often provided in an actual laboratory.

Therefore, the internetworking distance learning situation of Thailand still needs more improvement. Especially, educational institutions need to correct students' perceptions as a necessary requirement for studying online courses (Gulatee and Combes 2008). Research shows that Thai students' learning styles differ from those of Western students; they appreciate group learning (Selvarajah, Chelliah, Meyer, Pio and Anurit 2010). One of the main problems of Thai students may relate to cultural factors. They may lack both the ability to learn independently and critical thinking skills and tend, therefore, to rely more on local lecturers. This obstacle may have a larger effect on their online learning, as online learning needs more self discipline and independence. Precautions may be needed when instructors are trying to implement a remote access laboratory in a fully unsupervised learning model with Thai students.

Moreover, lectures and the demonstration of the laboratory exercises may need to be delivered by traditional modes to suit student requirements; the independent practice session may be conducted by means of the remote access laboratory. However, the development and application of such a facility should also consider the different cultural requirements and learning styles.

## 5.2 Problems and lesson learn

When remote laboratory were provided to distant students in this study we faced a variety of issues.

Firstly, the usage of traditional remote access methods that provide only the one-way CLI configuration screen to the distant students may not suitable for class demonstration. This was especially when students need an instant response from remote instructor of their immediate configuration.

Secondly, communication was always an issue in our case. We had some disconnection problems and realized that we should have other standby networks as backup.

Lastly, time availability was also another problem as our laboratory is quite packed during the semester time. Fortunately, Thai universities operate within a different period of the year and we could use this gap to better utilise the equipment.

## 6 Conclusion

This paper is an initial investigation in order to investigate the technical issues with providing concurrent access to multiple remote students.

This study found that network technology students in Thailand face the problem of lack of practice equipment and look for other means of support, either from a remote access laboratory or simulation tools. This study also raises the concern on implementing a fully distance learning environment for network technology classes in Thailand as it may introducing different challenges. Especially, the challenges may relate to support and guidance within the learning environment. For example, 85% of the respondents requested extra guidance from a local instructor even though the remote instructor was present. Also, the study was concerned with the value of traditional remote access configuration methods (CLI) when using with network technology classes. CLI, on one hand, is not a fully appreciated teaching tool from student perceptions by having the lowest appreciation score amongst all four alternatives of 2.71 from 5 point scale; even though the respondents in this study are familiar with CLI and have already attained a level of professionalism. Remote access laboratories may well need to provide more than text-based CLI access. Pedagogically-rich, multimedia learning environments, which offer multiple learning materials to suit different learning styles, should be considered and incorporated into remote access laboratories for networking education. SMDs and webcams, on the other hand, have been introduced in this study to help compensate for the difficulties of a remote learning environment. This study shows that the integration of both tools in the remote access laboratory will benefit distance learners. For example, 70% of the respondents agreed that SMD is necessary for them in remote access environment.

Therefore the remote access laboratory could be of benefit to computer networking education in Thailand as a whole. Although simulation software has the advantages of portability and accessibility, issues of accuracy still remain. The remote access laboratory, on the other hand, may offer a better degree of availability than the traditional laboratory; however, students may need time to adapt to the new teaching environment.

Educational providers, especially in developing countries, may need to understand current student perceptions of online facility usage and focus on building other skills necessary for the students to study independently. Further research into these areas is recommended by the authors.

## 7 Acknowledgement

The authors would like to thank Dr. Chompu Nuangjamnong for her most valuable contribution to the paper. Her assistance on collaborative working with Thai students is very much appreciated. The authors also would like to thank Dr. Judy Clayden for her most valuable editing assistance.

## 8 References

- Akamatsu, T., Ohtsuki, K. and Maj, S. P. (2007): A teaching model for computer networking technology in professional graduate schools. *8th International conference on information technology based higher education and training*, Kumamoto, Japan,
- Aravena, M. A. and Ramos, A. A. (2009): Use of a remote network lab as an aid to support teaching computer. *CLEI electronic journal*.
- Armstrong, H. and Murray, I. (2007): Remote and local delivery of Cisco education for the vision-impaired. *SIGCSE Bull.*, **39**(3):78-81.
- Barnes, S. B. (2003): Computer-mediated communication: human-to-human communication across the Internet. Allyn and Bacon.
- Biggs, J. and Collis, K. (1989): Towards a model of school-based curriculum development and assessment using the solo taxonomy. *Australian journal of education*, **33**(2):151-163.
- Bruner, J. S. (1966): Toward a theory of instruction. Belknap press of Harvard University.
- Cisco (2009): Cisco Academy Netspace. <http://www.cisco.com/web/learning/netacad/Unica/NetSpaceRedirect.html>. Accessed 27 November 2009
- Dede, C. (2005): Planning for "neomillennial" learning styles: Implications for investments in technology and faculty. EDUCAUSE Publishers.
- Dicerbo, K. E. (2009): Hands-On Instruction in the Cisco Networking Academy. *Networking and Services, 2009. ICNS '09. Fifth International Conference on*.
- Gulatee, Y. and Combes, B. (2008): Identifying social barriers in teaching computer science topics in a wholly online environment. *Fifth International conference on Science, Mathematics and Technology Education*, Udon Thani, Thailand: 173 - 182,
- International Association of Universities Information Database: List of Universities of the world. <http://www.iau-aiu.net/onlinedatabases/list.html>.
- Jakab, F., Janitor, J. and Nagy, M. (2009): Virtual Lab in a Distributed International Environment - SVC EDINET. *Networking and Services, 2009. ICNS '09. Fifth International Conference on*.
- Laohajaratsang, T. (2009): E-Learning Readiness in the Academic Sector of Thailand. *International Journal on E-Learning*, **8**(4):539-547.
- Lawson, E. A. and Stackpole, W. (2006): Does a virtual networking laboratory result in similar student achievement and satisfaction? *Proceedings of the 7th conference on Information technology education*, Minneapolis, Minnesota, USA, ACM.
- Lertkulvanich, S., Buranajant, N. and Sombunsukho, S. (2008): A creation of virtual classroom for teaching and learning management with e-learning. *Special issue on ICT for education development*. Bangkok.
- Machotka, J., Nedić, Z., Nafalski, A. and Göl, Ö. (2010): Collaboration in the remote laboratory NetLab. IN Pudlowski, Z. J. (Ed. *1st WIETE annual conference on Engineering and Technology Education*. Pattaya, Thailand, World Institute for Engineering and Technology Education (WIETE).
- Maj, S. P. and Kohli, G. (2004): A new state model for internetworks technology. *Issues in informing science and information technology*, **1**,
- Maj, S. P., Kohli, G. and Fetherston, T. (2005): A pedagogical evaluation of new state model diagrams for teaching internetwork technologies. *Proceedings of the Twenty-eighth Australasian conference on Computer Science - Volume 38*, Newcastle, Australia, Australian Computer Society, Inc.
- Maj, S. P., Murphy, G. and Kohli, G. (2004): State models for internetworking technologies. *Frontiers in Education, 2004. FIE 2004. 34th Annual*.
- Melkonyan, A., Akopian, D. and Chen, C. L. P. (2009): Work in progress - real-time remote Internet-based communication laboratory. *Frontiers in Education Conference, 2009. FIE '09. 39th IEEE*.
- Ministry of Education of Thailand (2009): Investment Plans under the 2nd stimulus package of economic reform (2010 - 2012). [http://www.moe.go.th/English/policy&plan/InvestmentPlans-SP2\\_2010-2012\\_.pdf](http://www.moe.go.th/English/policy&plan/InvestmentPlans-SP2_2010-2012_.pdf). Accessed 9 August 2010
- Ministry of Education of Thailand (2010): Education Policies of Mr. Chinnaworn Boonyakiat, Minister of Education of Thailand. <http://www.moe.go.th/English/policy&plan/8policies.pdf>. Accessed 9 August 2010
- Murray, I. and Armstrong, H. (2009): Remote Laboratory Access for Students with Vision Impairment. *Networking and Services, 2009. ICNS '09. Fifth International Conference on*.
- Nafalski, A., Nedić, Z., Machotka, J., Göl, Ö., Ferreira, J. M. M. and Gustavsson, I. (2010): Student and staff experiences with international collaboration in the remote laboratory NetLab. IN Pudlowski, Z. J. (Ed. *1st WIETE annual conference on Engineering and Technology Education*. Pattaya, Thailand, World Institute for Engineering and Technology Education (WIETE).
- Prieto-Blazquez, J., Arnedo-Moreno, J. and Herrera-Joancomarti, J. (2008): An Integrated Structure for a Virtual Networking Laboratory. *Industrial Electronics, IEEE Transactions on*, **55**(6):2334-2342.
- Qualtrics <http://www.qualtrics.com/>.

- Selvarajah, C., Chelliah, J., Meyer, D., Pio, E. and Anurit, P. (2010): The impact of social motivation on cooperative learning and assessment preferences. *Journal of Management and Organization*, **16**(1):113 - 126.
- Sirinaruemit, P. (2004): Trends and Fources for eLearning in Thailand. *International Journal of the computer, the internet and management*, **12**(2):132 - 137.
- Suanpang, P. and Petocz, P. (2006): E-Learning in Thailand: An Analysis and Case Study. *International Journal on E-Learning*, **5**(3):415-438.
- Tomov, O. (2008): Virtual labs with remote access to a real hardware equipment in the computer systems education. *Proceedings of the 9th International Conference on Computer Systems and Technologies and Workshop for PhD Students in Computing*, Gabrovo, Bulgaria, ACM.
- Vejjajiva, A. (2008): Policy statement of the council of ministers.  
<http://www.moe.go.th/English/policy&plan/policy-statement-abhisit-ENG.pdf>. Accessed 9 August 2010

## Author Index

Barnes, David J., 29  
Berry, Geoff, 79  
Buckland, Richard, 19

Cajander, Åsa, 145  
Carbone, Angela, 135  
Cardell-Oliver, Rachel, 55  
Clear, Tony, 37  
Corney, Malcolm, 95

Daniels, Mats, 73, 145  
de Raadt, Michael, iii, 105  
Devey, Adrian, 135

Hamer, John, iii  
Hamilton, Margaret, 125  
Harland, James, 125  
Ho, Kai Wing, 3  
Ho, Shuk Ying, 3

Lam, S.S., 63  
Lister, Raymond, 9, 95

Maj, S. P., 161  
Makasiranondh, Woratat, 161  
McDermott, Roger, 145

Nandi, Dip, 125  
Ngai, Eric W.T., 63

Philpott, Anne, 113  
Pilgrim, Chris, 119  
Poon, Jandia K. L., 87  
Poon, Jandia Kam-ling, 63

Quartly, Marian, 79

Robbins, Phil, 37  
Robinson, Peter, 155

Sheard, Judy, 79  
Shinners-Kennedy, Dermot, 29  
Simon, 47, 105

Teague, Donna, 95  
Thompson, Errol, 37

Veal, David, 161  
von Kinsky, Brian, 145

Warburton, Geoff, 125  
Whalley, Jacqueline, 37, 113  
Wiggberg, Mattias, 73



## Recent Volumes in the CRPIT Series

ISSN 1445-1336

Listed below are some of the latest volumes published in the ACS Series *Conferences in Research and Practice in Information Technology*. The full text of most papers (in either PDF or Postscript format) is available at the series website <http://crpit.com>.

- |  |  |
|--|--|
| <p><b>Volume 91 - Computer Science 2009</b><br/>           Edited by Bernard Mans Macquarie University.<br/>           January, 2009. 978-1-920682-72-9.</p>   | <p>Contains the proceedings of the Thirty-Second Australasian Computer Science Conference (ACSC2009), Wellington, New Zealand, January 2009.</p>                               |
| <p><b>Volume 92 - Database Technologies 2009</b><br/>           Edited by Xuemin Lin, University of New South Wales and Athman Bouguettaya, CSIRO. January, 2009. 978-1-920682-73-6.</p>   | <p>Contains the proceedings of the Twentieth Australasian Database Conference (ADC2009), Wellington, New Zealand, January 2009.</p>  |
| <p><b>Volume 93 - User Interfaces 2009</b><br/>           Edited by Paul Calder Flinders University and Gerald Weber University of Auckland. January, 2009. 978-1-920682-74-3.</p>   | <p>Contains the proceedings of the Tenth Australasian User Interface Conference (AUIC2009), Wellington, New Zealand, January 2009.</p>   |
| <p><b>Volume 94 - Theory of Computing 2009</b><br/>           Edited by Prabhhu Manyem, University of Ballarat and Rod Downey, Victoria University of Wellington. January, 2009. 978-1-920682-75-0.</p>  | <p>Contains the proceedings of the Fifteenth Computing: The Australasian Theory Symposium (CATS2009), Wellington, New Zealand, January 2009.</p>                               |
| <p><b>Volume 95 - Computing Education 2009</b><br/>           Edited by Margaret Hamilton, RMIT University and Tony Clear, Auckland University of Technology. January, 2009. 978-1-920682-76-7.</p>  | <p>Contains the proceedings of the Eleventh Australasian Computing Education Conference (ACE2009), Wellington, New Zealand, January 2009.</p>                                  |
| <p><b>Volume 96 - Conceptual Modelling 2009</b><br/>           Edited by Markus Kirchberg, Institute for Infocomm Research, A*STAR, Singapore and Sebastian Link, Victoria University of Wellington, New Zealand. January, 2009. 978-1-920682-77-4.</p>                | <p>Contains the proceedings of the Fifth Asia-Pacific Conference on Conceptual Modelling (APCCM2008), Wollongong, NSW, Australia, January 2008.</p>                            |
| <p><b>Volume 97 - Health Data and Knowledge Management 2009</b><br/>           Edited by James R. Warren, University of Auckland. January, 2009. 978-1-920682-78-1.</p>  | <p>Contains the proceedings of the Third Australasian Workshop on Health Data and Knowledge Management (HDKM 2009), Wellington, New Zealand, January 2009.</p>                 |
| <p><b>Volume 98 - Information Security 2009</b><br/>           Edited by Ljiljana Brankovic, University of Newcastle and Willy Susilo, University of Wollongong. January, 2009. 978-1-920682-79-8.</p>   | <p>Contains the proceedings of the Australasian Information Security Conference (AISC 2009), Wellington, New Zealand, January 2009.</p>  |
| <p><b>Volume 99 - Grid Computing and e-Research 2009</b><br/>           Edited by Paul Roe and Wayne Kelly, QUT. January, 2009. 978-1-920682-80-4.</p>   | <p>Contains the proceedings of the Australasian Workshop on Grid Computing and e-Research (AusGrid 2009), Wellington, New Zealand, January 2009.</p>                           |
| <p><b>Volume 100 - Safety Critical Systems and Software 2007</b><br/>           Edited by Tony Cant, Defence Science and Technology Organisation, Australia. December, 2008. 978-1-920682-81-1.</p>  | <p>Contains the proceedings of the 13th Australian Conference on Safety Critical Systems and Software, Canberra, Australia, December, 2008.</p>                                |
| <p><b>Volume 101 - Data Mining and Analytics 2009</b><br/>           Edited by Paul J. Kennedy, University of Technology, Sydney, Kok-Leong Ong, Deakin University and Peter Christen, The Australian National University. November, 2009. 978-1-920682-82-8.</p>      | <p>Contains the proceedings of the 8th Australasian Data Mining Conference (AusDM 2009), Melbourne, Victoria, Australia, November, 2009.</p>                                   |
| <p><b>Volume 102 - Computer Science 2010</b><br/>           Edited by Bernard Mans, Macquarie University, Australia and Mark Reynolds, University of Western Australia, Australia. January, 2010. 978-1-920682-83-5.</p>   | <p>Contains the proceedings of the Thirty-Third Australasian Computer Science Conference (ACSC 2010), Brisbane, Queensland, Australia, January 2010.</p>                       |
| <p><b>Volume 103 - Computing Education 2010</b><br/>           Edited by Tony Clear, Auckland University of Technology, New Zealand and John Hamer, University of Auckland, New Zealand. January, 2010. 978-1-920682-84-2.</p>   | <p>Contains the proceedings of the Twelfth Australasian Computing Education Conference (ACE 2010), Brisbane, Queensland, Australia, January 2010.</p>                          |
| <p><b>Volume 104 - Database Technologies 2010</b><br/>           Edited by Heng Tao Shen, University of Queensland, Australia and Athman Bouguettaya, CSIRO ICT Centre, Australia. January, 2010. 978-1-920682-85-9.</p>   | <p>Contains the proceedings of the Twenty-First Australasian Database Conference (ADC 2010), Brisbane, Queensland, Australia, January 2010.</p>                                |
| <p><b>Volume 105 - Information Security 2010</b><br/>           Edited by Colin Boyd, Queensland University of Technology, Australia and Willy Susilo, University of Wollongong, Australia. January, 2010. 978-1-920682-86-6.</p>                                      | <p>Contains the proceedings of the Eight Australasian Information Security Conference (AISC 2010), Brisbane, Queensland, Australia, January 2010.</p>                          |
| <p><b>Volume 106 - User Interfaces 2010</b><br/>           Edited by Christof Lutteroth, University of Auckland, New Zealand and Paul Calder Flinders University, Australia. January, 2010. 978-1-920682-87-3.</p>   | <p>Contains the proceedings of the Eleventh Australasian User Interface Conference (AUIC2010), Brisbane, Queensland, Australia, January 2010.</p>                              |
| <p><b>Volume 107 - Parallel and Distributed Computing 2010 2010</b><br/>           Edited by Jinjun Chen, Swinburne University of Technology, Australia and Rajiv Ranjan, University of New South Wales, Australia. January, 2010. 978-1-920682-88-0.</p>              | <p>Contains the proceedings of the Eighth Australasian Symposium on Parallel and Distributed Computing (AusPDC 2010), Brisbane, Queensland, Australia, January 2010.</p>       |
| <p><b>Volume 108 - Health Informatics and Knowledge Management 2010</b><br/>           Edited by Anthony Maeder, University of Western Sydney, Australia and David Hansen, CSIRO Australian e-Health Research Centre, Australia. January, 2010. 978-1-920682-89-7.</p> | <p>Contains the proceedings of the Fourth Australasian Workshop on Health Informatics and Knowledge Management (HIKM 2010), Brisbane, Queensland, Australia, January 2010.</p> |



**Volume 109 - Theory of Computing 2010**

Edited by Taso Viglas, University of Sydney, Australia and Alex Potanin, Victoria University of Wellington, New Zealand. January, 2010. 978-1-920682-90-3.

Contains the proceedings of the Sixteenth Computing: The Australasian Theory Symposium (CATS 2010), Brisbane, Queensland, Australia, January 2010.

**Volume 110 - Conceptual Modelling 2010**

Edited by Sebastian Link, Victoria University of Wellington, New Zealand and Aditya Ghose, University of Wollongong, Australia. January, 2010. 978-1-920682-92-7.

Contains the proceedings of the Seventh Asia-Pacific Conference on Conceptual Modelling (APCCM2010), Brisbane, Queensland, Australia, January 2010.

**Volume 112 - Advances in Ontologies 2009**

Edited by Edited by Thomas Meyer, Meraka Institute, South Africa and Kerry Taylor, CSIRO ICT Centre, Australia. December, 2009. 978-1-920682-91-0.

Contains the proceedings of the Australasian Ontology Workshop 2009 (AOW 2009), Melbourne, Australia, December, 2009.

