Experiences of first-year students in ICT courses: good teaching practices

Report 2014

Monash University (Lead)

Partners
University of Adelaide
University of Newcastle

Project Team
Judy Sheard (Project Leader, Monash)
Michael Morgan (Deputy Leader, Monash)
Matthew Butler (Monash)
Katrina Falkner (Adelaide)
Amali Weerasinghe (Adelaide)
Simon (Newcastle)
Beth Cook (Research Assistant, Newcastle)

Report authors
Judy Sheard, Michael Morgan, Matthew Butler, Katrina Falkner, Amali Weerasinghe, Simon

Abbreviations, acronyms, and terminology

ACDICT	Australian Council of Deans of ICT
ACS	Australian Computer Society
ACM	Association for Computing Machinery
CS	Computer Science
CS1	Computer Science 1
FYE	First-Year Experience
ICT	Information and Communication Technology
IEEE	Institute of Electrical and Electronics Engineers
LMS	Learning Management System
SFIA	Skills Framework for the Information Age
PASS	Peer Assisted Study Sessions
PBL	Problem-Based Learning
PI	Peer Instruction
course	Throughout this report we use the word 'course' to refer to a program of study, typically a degree program
unit	Throughout this report we use the word 'unit' to refer to single item of a course, which leads to a single result for a student; units are also known as subjects; and, confusingly, as courses

Executive summary and recommendations

Introduction

The project 'Experiences of first-year students in ICT courses: good teaching practices' has investigated the teaching of first-year Information and Communications Technology (ICT) students at Australian universities and the influences of teaching practices on students' learning experiences.

The aim of the project was to identify and disseminate good practices in ICT teaching at Australian universities with a specific focus on the first-year experience. Through examining recent research literature, surveying information on existing courses and content, and interviewing academics concerned with design and delivery of the first-year learning experience in Australian universities, we were able to gain a comprehensive view of current teaching practices and were able to outline the unique challenges that our first-year ICT students face. The literature provided an international perspective of teaching first-year ICT students while the academics interviewed were best placed to describe recent Australian trends in teaching practices and initiatives to support our students. The academics were also able to provide a detailed description of the rationale for current practice and to share their thoughts on the impacts of recent changes on their students. Key findings of the literature survey, the course information and the insights gained from the academic participants have been collated in order to provide examples of good practice in the field and to recommend areas for further investigation.

The project was conducted by a team of six academics from three Australian universities in three states. The project was funded by the Australian Council of Deans of ICT (ACDICT).

Project approach

To investigate teaching practices, the project team designed a framework with the following six themes:

- What we teach
- Where we teach
- How we teach
- How we assess
- How we strengthen the learning environment
- How we support our students

Each of these themes contributes to the total learning experience of our ICT students and has a major impact on their educational outcomes. The transition from secondary to tertiary studies is a difficult process for many students and it is therefore important to understand the influences on this experience. The relatively high rate of attrition in ICT courses indicates that there may be challenges that are unique to this field. While there are a number of studies of the first-year experience across the university sector, to investigate these challenges it is necessary to consider the ICT context in the Australian setting. The volume of the literature concerned with specific ICT teaching and support issues indicates that a lot of worthwhile research is being conducted but this research needs to be properly collated and evaluated in order to drive change in practice.

Each of the six project themes was investigated by two or more members of the team. The interview schedule was designed by the full project team, and the interviews were conducted by a research assistant who also created summary notes of the interview data. The collation and drafting of the final project report was carried out by the project leader and deputy, with the assistance of all other team members. Findings from each of the project themes are summarised in the following sections. The Executive Summary concludes with a number of recommendations arising from the report as a whole.

Summary of findings

The evidence presented of innovative and good practice for the six themes was derived from a systematic literature search and interviews of academics at Australian universities. The literature search encompassed 207 papers relevant to the themes, with 57 of these being Australian studies. Thirty academics from 25 Australian universities were interviewed. These included six Group of Eight (go8), three Australian Technology Network (ATN) and five Innovative Research (IRU) universities.

What we teach

The 'What we teach' theme focused on the core curriculums of the first year of ICT courses in Australian universities and the process of curriculum design. Relevant courses from all Australian universities were identified and the units offered to first-year students examined to identify similarities between courses and units as well as key areas of differentiation. The teaching of computer programming was explored in detail as this topic is widely researched and discussed in the literature. Also covered in this theme were factors influencing course and unit design, such as the guiding principles adopted from the Australian Computer Society (ACS) and Association for Computing Machinery (ACM)/ Institute of Electrical and Electronics Engineers (IEEE).

The most common ICT courses are:

- general ICT courses, most with majors
- computer science
- software engineering
- information systems / business information systems

Common units include:

- programming
- database
- systems analysis
- computing fundamentals
- mathematics (predominantly in computer science courses)

The study found that there is little recent literature from the Australian context about what is taught to first-year ICT students; further research is needed into curriculum development to meet the specific needs of first-year ICT students. We identified a need to further explore the role of formal skills frameworks provided by organisations such as ACS, ACM and IEEE, for example the Skills Framework for the Information Age (SFIA), to assist in curriculum development for first-year ICT courses.

The main issue raised in the literature and in the interviews was a lack of consensus on the most appropriate introductory programming language. However, we identified a trend towards increased focus on problem solving rather than on syntax when teaching introductory programming.

Recommendation 1

There has been a perceptible trend towards programming environments where the focus has moved away from syntax to problem solving. This is an

area that needs investigation to determine how students respond to learning programming in these environments.

Recommendation 2

Investigation is required to determine how formal skills frameworks provided by organisations such as ACS, ACM and IEEE can be most usefully applied in curriculum development.

Where we teach

The 'Where we teach' theme focused on the teaching and learning spaces used for first-year ICT courses in Australian universities. It considered the design and use of new teaching spaces and the redesign of existing spaces, either physical or virtual. For virtual teaching spaces, the theme included teaching and learning in situations enabled through the use of mobile and ubiquitous technologies.

The research literature provided several examples of virtual lab software, social networking tools, web-based collaborative environments, and a variety of introductory programming environments. However, we found little research on physical and virtual learning environments for first-year ICT students in the Australian context.

The interviews provided insights into:

- blended and online teaching environments
- current teaching models which have seen reduction in lecturing activities and increase in practical lab work, prompting changes in learning space requirements.
- design philosophies used for physical and virtual spaces
- the impact of new teaching spaces on teaching practices

The layout of physical teaching spaces was reported to be increasingly diverse and flexible. The provision of online resources is more prevalent, resulting in an increase in flexible study options, including the integration of social networking tools to assist the formation of student learning communities. These changes highlighted the need for further research in order to assess their impact on the first-year ICT student experience.

Recommendation 3

Various new physical and virtual learning environments are tailored to the needs of first-year ICT students. Investigation is needed to assess the impact of these environments on student performance and on the student experience.

How we teach

The 'How we teach' theme was concerned with all aspects of the design and delivery of university-level learning experiences to first-year ICT students and associated supporting academic activities. These were frequently discussed in terms of influences on student learning, motivation, and engagement.

Four main topics emerged from the research literature;

- theories and models of teaching and learning
- approaches to teaching
- cooperative and collaborative learning
- social media and learning communities

Our analysis showed that, while there was a significant body of literature devoted to these themes, much of this literature was focused on the programming context. We propose that further research is needed to explore other aspects of the first-year curriculum and to evaluate the comparative effects of various techniques on the student experience.

The topics that emerged from an analysis of the interview data broadly aligned those highlighted in the literature. Most interviewees highlighted rapid changes in traditional methods of on-campus course delivery due to a perceived lack of student engagement, in particular changes to the lecture format and to the balance between lectures and practical labs. Practices such as active learning approaches, flipped classrooms, peer, cooperative, and collaborative techniques, and problem-based learning were frequently discussed. Also mentioned were the integration of social networking tools to promote learning communities and the importance of matching academic staff skills and experience to the needs of first-year students. Again, the focus was predominantly on the programming context, so we propose that other areas of the first-year curriculum and the integration of the curriculum of the whole first year merit future consideration.

Finally there is a need to formally evaluate the effects of many of the innovative teaching practices that have been described in this report. Substantial work has been documented on efforts to improve the relevance and appeal of the ICT curriculum to a wider range of students, including non-ICT students, through the use of social media, visual programming, and problem-based learning techniques. In many cases the initial reports of the techniques are positive, but more rigorous evaluation is required to support evidence-based decision-making on which techniques should be further developed to drive improvements in the first-year learning experience of ICT students.

Recommendation 4

Research into teaching in ICT is overwhelmingly dominated by a focus on techniques to teach introductory programming courses. Research is needed on other areas of the first-year ICT curriculum.

Recommendation 5

This report documents a number of initiatives to increase ICT student engagement in the learning process. There is a clear need for more formal evaluations of the effects of these teaching initiatives in the Australian ICT context and for the collation of examples of good practice for wider dissemination.

Recommendation 6

When allocating teaching responsibilities, careful consideration should always be given to the appropriateness of the staff allocated to first-year courses.

How we assess

The 'How we assess' theme focused on assessment-related issues in first-year courses in Australian universities. Included under this theme were assessment strategies, techniques, and tools. The tools were either instruments to assess students' learning or tools to facilitate the assessment marking process. Different forms of summative and formative assessment were discussed in relation to provision of feedback, verification of student work, and other issues associated with academic integrity.

Six main topics emerged from the literature:

- assessment design and strategies
- exam assessment
- non-exam forms of assessment
- automated assessment
- assessment instruments
- academic integrity

Much of the literature found was related to assessment in the programming context, with a strong emphasis on automated assessment tools.

A key issue raised by interviewees was that the trend for increased online delivery had placed demands on academics to create appropriate assessment tasks for this context and to verify the identity of each student undertaking the assessment. Related to this was a perceived need for tools to automate assessment for large groups and to facilitate provision of feedback. We propose that these issues require further research in order to ensure valid and fair assessment for our first-year students.

Recommendation 7

We found a variety of techniques and tools for assessment of programming but very few in other areas of ICT study. Research is needed on assessment techniques for other areas of the first-year ICT curriculum.

Recommendation 8

The recent adoption of social media has led to innovative forms of assessment; however, there were few studies found of the use of such forms of assessment in first-year ICT courses. This is an area that should be further investigated.

Recommendation 9

The trend for increased online delivery has placed demands on academics to create appropriate assessment tasks for this context and to verify the identity of the student undertaking the assessment. There is a clear need for work in this area.

Recommendation 10

There is a need for tools to automate assessment for large groups and to facilitate provision of feedback to students.

How we strengthen the learning environment

The 'How we strengthen our learning environment' theme focused on approaches to help first-year students in Australian ICT courses become effective learners. This included methods to encourage the development of learning communities and programs to assist students with study skills, teamwork, language and communication skills, and to educate students about academic integrity.

Four main topics emerged from the literature:

- learning communities
- team-work skills
- study skills
- academic integrity

The literature reported a number of programs to assist students with team-work and study skills, and strategies to support their learning, including the formation of learning communities.

The interviewees gave many examples of ways in which they assist students with study skills and time management and educate them about academic integrity. These ranged from initiatives of individual lecturers to formal university-level programs. A few interviewees also mentioned learning communities and explained how they encourage these through the use of discussion forums and social media.

Recommendation 11

Social media play a major role in the lives of the current student cohort. There is a clear need to investigate how these media can be used effectively to strengthen learning support.

Recommendation 12

There is a clear need to understand how educators can develop the communication skills of first-year ICT students, as very little research has been done in this area.

Recommendation 13

A variety of approaches are used to educate students about academic integrity; there is a clear need for work on understanding the effectiveness of these.

How we support our students

The 'How we support our students' theme focused on programs to support students in their social integration into university. This includes programs designed to assist students in their transition from school to university, programs designed to increase social support structures, and programs designed to address equity issues, specifically increasing participation and support for female students and indigenous students.

Four main topics emerged from the literature

- transition support
- social support
- equity programs
- at-risk behaviour analytics

All interviewees indicated that programs for at-risk students are implemented in their universities; however, there were few reports of the results of such programs. Many universities have transition programs for their first-year students. Several interviewees discussed the importance of students having a social cohort, and there were a number of initiatives to provide social support for students.

Recommendation 14

A number of institutions offer support programs for at-risk students, but there is little evidence of results or review of these programs. This indicates a need to discuss further and share knowledge on these types of programs.

Recommendation 15

There is a clear need for the development of learning analytics tools to better assist academic staff in early identification of at-risk students.

Recommendation 16

The trend in equity-based support programs for female students in ICT programs has been towards recruitment rather than retention. There is a clear need to identify good practice retention policies and programs within an Australian context.

General Recommendations

Each theme in this report leads to a number of recommendations. These more general recommendations either overlap several themes or arise from the project as a whole.

Recommendation 17

Our research found many examples of innovative teaching practices across the universities; however, most were used only within a particular institution or by a single academic. It is recommended that avenues be explored for dissemination of these practices.

Recommendation 18

Most innovations found were described without any indications that they had been rigorously evaluated. It is recommended that academics be encouraged to appropriately evaluate any new teaching practice.

Recommendation 19

Accessible training and resources are needed to assist academics to evaluate their teaching innovations.

Recommendation 20

Academics and departments should become and remain aware of innovations, including teaching and assessment tools, that are reported in the literature, and should give due consideration to the value that might accrue from employing these innovations.

Contents

List of tables

Introduction

There are many challenges in teaching essential ICT concepts and skills to first-year students. With the rapid evolution of information and computing technologies, ICT educators contend with continual curriculum changes. Furthermore, recent innovations in teaching approaches and educational technologies mean that educators face many choices in how they teach. This project has investigated trends in the first year of ICT education and current practices in Australian universities, highlighting examples of good practice and issues raised by stakeholders in the field.

The project was conducted in two phases framed in six broad themes related to the first-year ICT student experience: *What we teach*, *Where we teach*, *How we teach*, *How we assess*, *Learning support*, and *Student support*. Each theme was investigated by several project members, reflecting their particular experience and expertise.

Phase 1, Literature review: An examination of current trends and good practice in ICT education nationally and internationally was conducted in the form of a detailed systematic review of relevant research literature. The review covered national project reports and key journals and conferences in computing education.

Phase 2, Survey of current practice: A research assistant conducted extensive interviews with 30 first-year ICT academics from universities in Australia, using an interview script based upon the six themes. All universities that delivered ICT courses were approached. Exemplars of good practice were identified from the the interviews.

This report is structured around the six key themes. Each theme incorporates findings from the literature review and detailed interviews of relevant academics involved in delivering first-year ICT courses. The identified examples of good practices are discussed in terms of their impact on student support and on the learning experience.

Project themes

The investigation of good practices in first-year ICT courses was based upon six themes. Within each theme the different aspects are discussed in relation to issues such as engagement, retention/attrition, students at risk, transition, and graduate attributes.

A broad description of each theme follows.

What we teach

This theme focuses on examining the core content of what is taught to students in the first year of ICT courses across the country. Relevant courses from all Australian universities that offered ICT courses were identified and the units offered to first-year students were examined to determine similarities between courses and units as well as key areas of differentiation. Computer programming was used as one significant case study in examining unit content across courses. Also surveyed in this theme were the underlying motivations for course and unit design, such as the guiding principles adopted from the ACS and ACM/IEEE.

Where we teach

This theme covers the physical and virtual teaching and learning spaces used for first-year courses. It includes the design of new teaching spaces or redesign of existing spaces, either physical or virtual. For virtual teaching spaces, this includes teaching and learning in situations enabled through use of mobile and ubiquitous technologies.

How we teach

This theme covers several aspects: methods of teaching; how tools, resources and technologies are used in teaching first-year courses; and the underlying pedagogies with which they are used. The theme includes the design and use of new tools, resources or methods, and innovative applications of existing ones. The technology aspect includes how teaching platforms, software applications, or programming languages are used. The pedagogy aspect includes the development and implementation of new pedagogies or the innovative use of more established pedagogies.

How we assess

This theme covers assessment-related issues in first-year courses. The theme includes the design and use of new assessment methods and tools, and innovative applications of existing ones. The tools may be instruments to assess students' learning or to facilitate the assessment or marking process.

How we strengthen the learning environment – learning support

This theme is concerned with programs, tools and methods used to support first-year students with their learning. This includes programs to assist students with their study skills or language and communication skills, and to educate students about academic integrity. Also discussed are mentoring and peer-support programs. Other aspects examined are strategies and tools to detect at-risk students and the use of learning analytics to inform intervention strategies.

How we support our students – student support

This theme is concerned with programs to support students in their social integration into university. This includes programs to assist students in their transition from school to university and programs to assist international students. This theme also covers equity issues and programs to increase female participation in ICT education courses.

Report Structure

The body of this report begins with a description of the methodology used to guide the investigation and to collect the data on which the report is based. Here the focus is on the procedure used to conduct the literature review and on the process of conducting the interviews. This is followed by the sections detailing the six themes that are the focus of the report. Each theme is structured into a consistent format. First the scope of the theme is described and then the research literature related to the theme is presented. This is followed by a discussion of current practice in the Australian context, based on the issues and examples of good practice emerging from the interview data. Each theme concludes with a summary, a discussion of future research directions that are suggested in this area, and recommendations arising from our investigation of that theme. The report itself concludes with a summary of the exemplars of good practice highlighted throughout the report.

Methodology

The evidence presented to detail innovative and good practice for the six main themes covered in this report was derived from two primary sources: a systematic literature review of relevant research papers published between 2009 and 2014; and 29 interviews of 30 academics directly concerned with delivery of first-year ICT higher education courses in 25 universities in Australia.

The Literature Review

In order to identify current trends and issues concerning the first-year experience of ICT students in higher education, particularly in the Australian context, a detailed and systematic review of the available literature was conducted. To ensure currency, the scope of the literature was limited to research papers published between 2009 and 2014. Full peer-reviewed research papers published in high-quality academic journals and conferences relevant to the area of study were targeted.

The review began with a series of keyword searches in Google Scholar of relevant terms in the date range from 2009 to 2014. The search settings specified a search for terms anywhere in the full text of each paper. The researchers identified six groups of relevant search terms, and organized them into a tabular form (Table 1). Some terms were considered equivalent; for example, the category *Term 3* lists variations of the single term 'first-year experience' that were derived from the literature. Other term categories were intended to cover different techniques of interest; for example, the group *Term 6* covers a number of research issues considered to be of interest in the first-year experience. As additional terms were identified from the literature, they were added to the table; these newer additions are shown in italics.

Term 1	Term 2	Term 3	Term 4	Term 5	Term 6
Australian higher education	student experience	first-year	computing	transition support	teaching spaces
Australian university	student perceptions	first year	information systems	retention	1st year curriculum
Australian tertiary education sector		1st year	computer science	attrition	pedagogy
		introductory	informatics	at-risk students	peer-assisted learning
		FYE (First Year Experience)	CS1 (computer science 1)	*transition curriculum*	educational technology
		First year transition	ICT		PASS (peer-assisted study sessions)

Table 1: Refined search terms.

Combinations of keyword searches were carried out in Google Scholar and the searches of combinations of terms continued until no new relevant research papers were being identified. Similar keyword searches were also conducted in the IEEE Xplore database and the ACM Digital Library databases.

In order to ensure that no relevant literature was overlooked, a manual search of selected high-quality research journals and conferences in the area of computing education was conducted for the years 2009-2014. This was accomplished by compiling a list of suitable journals and conferences, and scanning their tables of contents for relevant-sounding topics. Table 2 shows a list of the journals searched and indicates the number of relevant papers found; the first number indicates the number of new papers found via the manual search and the second number indicates the number of papers for this publication found in each journal by all search methods. The same process was followed for relevant conferences until no new papers emerged. As a result of the above processes, a total of 258 papers were identified.

Venue	manual search / total
ACM Transactions on Computing Education	18 / 20
IEEE Transactions on Education	2 / 2
Journal of Computing in Higher Education	0 / 14
Journal of Computer Science Education	0 / 3
Computers and Education	14 / 22
Australasian Journal of Educational Technology	7 / 8
International Journal of Technology Enhanced Learning	1/ 1
Research and Practice in Technology Enhanced Learning	0 / 0
Journal of Information Technology Education: Research	11 / 11
Australian Educational Computing	1 / 1
Computers in Education Journal	1 / 1

Table 2: Manual search of selected journals from 2009 to 2014 using paper title relevance

Once all relevant research papers had been identified they were uploaded to Mendeley (http://www.mendeley.com/), an online reference and citation management application, allowing all researchers involved in the project to access to the full text of each paper. All duplicate papers were first removed. Full bibliographic, author keyword and abstract information for each paper was then generated. This allowed the researchers to analyse the relevance of the papers and to categorise them.

An analysis of *Title*, *Author Keywords* and *Abstract* was used to create an initial set of tags for the papers. For example, a paper might be tagged as related to: *1st year*; *Australian*; *higher education*; *pedagogy*; *programming*. These tags were used to categorise the papers into primary theme folders. Each theme author then read the papers in their theme folder in detail and created short notes describing the content of the papers. During this process the papers were tagged with theme numbers to indicate the themes the paper was relevant to. A number of papers were tagged with multiple themes. A full listing of tags is shown below.

01, 02, 03, 04, 05, 06, 1st year, academic integrity, assessment, at-risk, attrition, aust, australia, australian, blended learning environments, cognitive load theory, contributing student pedagogy, cs1, curriculum, database, engagement, experience, forums, gender, gender issues, group work, health sciences, high school, higher ed, ict, inclusion, introductory programming, learning support, learning technologies, literature review, motivation, pedagogy, peer instruction, plagiarism, programming, research methods, retention, social media, student behaviours, student engagement, student experience, student support, study habits, success, teaching spaces, technologies, technology, transition

During the second tagging process a number of papers were excluded from the collection. Typical reasons for exclusion of papers included being outside the relevant date range, insufficient rigor in the research approach, the subject matter concerning ICT courses in a high school context, or a focus on curriculum for later years of ICT courses. Following this process a total of 207 papers were found relevant to the themes, with 57 of these being Australian studies. The scope and subject matter of the collected literature within each theme give an indication of where computing education research is focused and specific areas where further research may be required. Several of these areas are highlighted in the report.

Thirty academics from 25 Australian universities were interviewed. These included six Group of Eight (go8), three Australian Technology Network (ATN) and five Innovative Research Universities (IRU) universities.

The Interviews

The purpose of the interviews was to collect detailed information about teaching practices and factors impacting the first-year experience of ICT students in the Australian higher education context. In order to gain this information the project targeted academic staff directly involved in the design, coordination and delivery of first-year courses, as these participants were likely to provide the required insights into the first-year experience and to be in a position to highlight recent changes and examples of good practice. An examination of the student perspective on the first-year ICT experience was beyond the scope of this project in terms of the funding and time available.

Participants were selected from each participating university in Australia that delivered an ICT course. Where possible, project members nominated relevant people at various universities. Where this could not be done, the contact details listed on faculty and degree websites were used to initiate e-mail contact. Participants were recruited by contacting the concerned faculties and requesting the contact details of the staff most involved with the delivery of the first-year ICT programs.

Ethics approval for the research was sought and granted from the three universities conducting the project. After contact via email, an explanatory statement and consent form were sent to the selected participants. Consent for the audio recording of the interviews was also sought, and was recorded at the start of each interview. The anonymity of participants was ensured when reporting the results by allocating a unique identifier code for each participant. This ensured that participants could speak frankly and openly about issues such as the challenges they faced, the solutions developed and the results achieved. The identifying code scheme consisted of the letter U, a number assigned to the person's university, and, where required, a further letter corresponding to an individual academic; for example, U7b was the second academic interviewed from university 7.

The interview script (see Appendix C) developed by the full project team consisted of a number of semi-structured questions, and the interviewer was encouraged to ask follow-up questions if interesting practices or new issues emerged. The script was trialed in two pilot phone interviews, and slight modifications were made to reduce duplication of the topics covered and to reduce the likely interview time. The revised script was used for all subsequent interviews.

All interviews were conducted by telephone by the research assistant Beth Cook, during February and March 2014, at a time convenient to the interviewee concerned. A consistent approach was assured by the fact that all interviews were conducted by the same person. Interviewees were sent the list of questions prior to the interview so that they would be aware of the nature of the questions to be covered. In total 29 interviews were conducted involving 30 academics, as two academics from one university opted to be interviewed at the same time. The 30 academics represented 25 Australian universities including six Group of Eight (go8), three Australian Technology Network (ATN) and five Innovative Research (IRU) universities.

Twenty-nine interviews were recorded, ranging in duration from 16 to 74 minutes and averaging 53 minutes. Detailed summary notes were taken during each interview. After each interview the notes were elaborated upon and organised into the six themes, with notes of the approximate times at which the discussion could be found in the audio recording. The interview notes were then examined to find important issues and to identify possible case studies of good practice for further investigation. Detailed quotes from relevant interviews were subsequently transcribed as required.

Using the data collected with the methodology described above, the following sections of the report describe our findings for each of the six main themes.

What we teach

Overview

The ‘What we teach’ theme focuses on the core curriculums of the first year of ICT courses in Australian universities and the process of curriculum design. Relevant courses from all Australian universities are identified and the units offered to first-year students examined to identify similarities between courses and units as well as key areas of differentiation. The teaching of computer programming is explored in detail as this topic is widely researched and discussed in the literature. Also, covered in this theme are factors influencing course and unit design, such as the guiding principles adopted from the ACS and ACM/IEEE. Examples of good practice in course design are highlighted, as well as innovations in specific unit curriculum.

We begin by presenting a literature review of first-year ICT university course curriculums, highlighting any Australian studies. This is followed by an overview of Australian ICT courses and a review of course content, as found in a web survey. Findings from the interviews of ICT academics are then presented to provide in-depth perspectives on content and design of first-year ICT courses.

1. Literature Perspectives

In the literature search, conducted as described in the methodology section, 28 research papers were found related to the theme of ‘What we teach’ in the context of ICT university courses. Thirteen papers were focused on the first year of ICT courses and ten papers were set in the Australian context. However, only three papers were set in both Australian and first-year contexts (Corney et al 2010; Mason et al, 2012; Mason & Cooper, 2014) and all three of these papers relate specifically to programming. See Table 3 for a list of the Australian papers for this theme.

Approximately half the papers found discuss higher-level curriculum design issues within university ICT courses. These papers typically present guides and frameworks for using noted ICT charters (such as ACS, ACM, IEEE, and SFIA) in curriculum design, often highlighting specific case studies of recently redesigned curriculums (Adegbehingbe & Obono 2012; Koohang et al, 2010; Herbert et al, 2013a). Because of this, the literature is often not focused on the first-year context. While discussion of curriculum design can identify certain needs for structuring courses with supporting progressions, these papers typically discuss design of an entire three- or four-year curriculum.

Moves to adopt SFIA in curriculum design are evident in the more recent papers. Several Australian universities appear to have adopted this framework as a key charter in redesigning their curriculums, with the University of Tasmania being a well-documented example of this (Herbert et al, 2013a; 2013b; 2013c; 2014). The SFIA framework is of importance in its presentation not only of core skills as they relate to industry but also of levels of responsibility, which can be aligned to different year levels in a course (von Konsky et al, 2014). Consequently, these papers provide some insight into curriculum design within the first-year context.

The publications relating most closely to the first-year context deal with narrower fields of study within the first year. For example, discussion of programming curriculum and issues in most cases relates specifically to novice programmers, thus usually the first-year context. Indeed, programming was clearly the best represented context, with 11 papers relating specifically to curriculum issues within this area of study. Mason et al (2012) and Mason & Cooper (2014) provide a comprehensive analysis of trends in introductory programming courses in Australian universities. They note a fragmentation of choice of the programming language being used, and a reduction in the use of Java as a language in introductory programming courses. Issues raised by other researchers relate mainly to the choice of programming language and environment (Fincher et al, 2010; Stefik & Siebert, 2013), and restructure of curriculum to better support novice programmers (Corney et al, 2010; Hu et al, 2013; Thota & Whitfield, 2010). The narrower focus suggests that notions of what we teach are more easily placed in the context of a specific year and unit, while broader curriculum issues (both design and content) will focus on whole courses.

Other specific contexts for discussion of curriculum issues were found, although much less prevalent than those relating to programming. Subject areas found include computer systems (Benkrid & Clayton, 2012; Patitsas et al, 2010) and software development (Thomas et al, 2010). Other sub-themes that were found in the literature relating to curriculum include investigation of gender issues (Koppi et al, 2012) and career progression and its implications for curriculum design (von Konsky et al, 2014).

In summary, there is little recent literature about what is taught to first-year students in the Australian context. While there is research relating to curriculum development in higher-education ICT courses, it tends not to address specific first-year issues, which are typically reported on in relation to specific topics such as programming. This suggests that there is scope for further research relating to how curriculum is developed in consideration of the needs of first-year students.

Sub-topic	Australian-focused references
Curriculum design	Herbert, N., Dermoudy, J., Ellis, L., Cameron-Jones, M., Chinthammit, W., Lewis, I., de Salas, K. L. & Springer, M. (2013a). Stakeholder-led curriculum redesign.
	Herbert, N., Lewis, I., & Salas, K. De. (2013b). Career outcomes and SFIA as tools to design ICT curriculum.
	Herbert, N., Salas, K. De, Lewis, I., Cameron-jones, M., Chinthammit, W., Dermoudy, J., Ellis, L. & Springer, M. (2013c). Identifying career outcomes as the first step in ICT curricula development.
	Herbert, N., Salas, K. De, Lewis, I., Dermoudy, J., & Ellis, L. (2014). ICT curriculum and course structure : the great balancing act.
	von Konsky, B. R., Jones, A., & Miller, C. (2014). Visualising career progression for ICT professionals and the implications for ICT curriculum design in higher education.
First-year curriculum	Corney, M., Teague, D., & Thomas, R. N. (2010). Engaging students in programming.
	Koppi, T., Roberts, M., & Naghdy, G. (2012). Perceptions of a gender-inclusive curriculum amongst Australian information and communications technology academics.
	Thomas, R. N., Cordiner, M., & Corney, D. (2010). An adaptable framework for the teaching and assessment of software development across year levels.
Programming languages	Mason, R., Cooper, G., & de Raadt, M. (2012). Trends in introductory programming courses in Australian universities – languages, environments and pedagogy.
	Mason, R., & Cooper, G. (2014). Introductory programming courses in Australia and New Zealand in 2013 – trends and reasons.

Table 3: 'What we teach' literature focused in the Australian context

2. ICT Courses in Australia

A summary of ICT courses in Australian universities can be found in Appendix A. All but one university (University of Notre Dame) offer an ICT or related degree. While most degree offerings are located in capital cities, a substantial number are offered in rural locations, and a number in off-campus mode.

The faculties that offer ICT degrees are predominantly Information Technology, Science, Engineering, or Business (or faculties that are a combination of these disciplines). There are now very few dedicated ICT faculties in Australian universities. Different ICT degrees are in some cases taught within different faculties in the same university, depending on the context of the degree. For example, a Computer Science degree may be located within an Engineering or Science faculty or department, while an Information Systems degree may be located within a Business faculty or department. In most cases, however, one faculty takes ownership for all ICT-related degrees.

The degrees offered by Australian universities typically fall into one of the following broad categories/contexts:

- general ICT
- ICT with a major or specialisation. Majors typically include
    - games programming

    - software/application development (including mobile)
    - security
    - networks
    - web design and development
    - multimedia
- software engineering
- computer science
- business information systems

General ICT courses, most with majors, make up the majority of courses offered. Computer science ranks second, software engineering third, and information systems / business information systems fourth. There are also a number of miscellaneous ICT courses focusing on other specialist areas such as multimedia, game development, cyber-security and engineering.

In keeping with our focus, we consider units situated in the first year of a typical progression in these courses. Units studied in first year depend on the particular course being taken; however, there is some consistency in units undertaken by students in their first year of ICT study. A summary is shown in Appendix B. Common units include:

- programming
- database
- systems analysis
- computing fundamentals
- mathematics (predominantly in computer science courses)

Programming and database are the units most frequently studied by first-year ICT students.

3. Current Practice in Australia

The interview questions related to the theme of 'What we teach' sought added insights into the nature of first-year ICT courses in terms of student demographics, the development of the teaching curriculum and, more specifically, programming languages taught.

i. Demographics of first year of ICT courses

Enrolments in the first year of ICT courses vary considerably across Australia, ranging from approximately 100 to 500 students. According to interviewees it is often difficult to gauge exactly how many students are in the first year of a course, as different students enter the courses by different pathways, some of which will attract credit for designated units. Many interviewees made informed estimates of the numbers on the basis of enrolment numbers in units that were core for first-year students, along with the course information of those students. Based on the interviewees' responses, just over 5000 first-year students were estimated to be enrolled in ICT courses across the 25 universities contacted.

The mix of students also varied considerably across the universities. Many interviewees were not privy to the breakdown of local versus international students, but most were able to give informed estimates, again based on class demographics. In view of the uncertainty of these estimates, we present only the broad picture. Six institutions indicated very low numbers of overseas students (less than 10%), while another six indicated that 50% or more of their first-year cohort were international students.

Between these extremes, the majority of interviewees (7) estimated their international enrolments as 20-30% of their cohorts. There would appear to be scope for research into internationalisation of the teaching curriculum, not only because of these demographic estimates, but also because of the international nature of ICT.

ii. Curriculum design

Interviewees were asked whether the design of their courses was influenced by any external curriculums. Most interviewees indicated that their courses are accredited by the Australian Computer Society. Many mentioned that their course designs were influenced or inspired by external bodies such as the ACS, ACM, and IEEE as well as industry companies like CISCO. Although these organisations played an important role in the consideration of their curriculum design, interviewees were often unsure exactly how the frameworks provided by these organisations were specifically used. An illustrative response:

> *"The degree programs are a combination. It is not directly taken from the ACM/IEEE computer science curriculum but they were used as input into the design of the course. So we used the ACM/IEEE curriculum as well as the ACS guidelines. The courses are ACS accredited." (U1)*

There is little literature on the exact role of bodies such as ACS, ACM, and IEEE in curriculum design, suggesting an opportunity for research to seek greater insights into the role of such formal bodies in the design and development of the tertiary curriculum.

The use of SFIA in curriculum design was notably absent from the interviews. Recent literature suggests that it can play a major role in the design of courses, so it was of interest that it was not mentioned by any interviewees. This is likely to change in the near future, as SFIA gains awareness through both the ACS and published literature.

iii. Programming languages

Interviewees were asked what programming languages are introduced to students in their first-year ICT courses. The most common languages were Java (16) and Python (12). Java has been well documented as a language used to teach students programming both at a foundation level and also as an introduction to object-oriented programming. While it remains a popular choice, a number of interviewees reported recent moves away from Java as an introductory language, in many cases to Python. Interviewee U4 explained this shift in languages:

> *"Java was seen as having too much excess baggage to get people off the ground that just wanted to learn the basics. They didn't go into object-oriented or object-based programming so the need for all of the concepts around object-oriented programming weren't necessary and so instead they wanted to build the strength in the fundamentals and the wisdom was that Python would be better."*

Another interviewee echoed these sentiments, noting that:

> *"We are considering at the moment moving away from Java and maybe going to something like Python. We've used Java for a fair while but it's losing relevance in a lot of areas and is a quite bloated language. Something like Python is more elegant and sophisticated in some ways and enforces some good program structure and at least as good at formatting, so it's better for the first-year students to introduce them to the programming concepts." (U6)*

In contrast, interviewee U7b indicated a move from C++ to Java as the introductory programming language, *"Changed from C++ to Java, very popular in industry, slightly easier."*

Concerns have been raised in the literature about the significant learning challenges faced by novice programmers starting with an object-oriented language such as Java, and some responses in the interviews appear to address these concerns. A number of interviewees discussed their shift to Python, while others had moved to less traditional languages and environments such as Processing, Gamemaker, and Scribble (a variant of the Scratch programming environment). The literature also includes the move to environments such as Alice. These examples appear to place the emphasis on problem solving rather than language syntax or complex programming paradigms; however, little research has been found that describes the learning outcomes of these changes.

One interviewee said that the move from Java to Scribble, a visual programming language, was to "*get students to focus on solving problems rather than concentrating on syntax*" (U15b). A program is constructed in Scribble by assembling visual blocks representing code segments, a process that shields novice programming students from syntax and code and allows them to focus on programming logic. This is seen as a more accessible environment than a traditional programming language for introducing fundamental programming concepts to novice programmers. As interviewee U15b explains:

> *"It was a fair undertaking, and it was a fairly big decision to say let's not start students in a syntactic language like Java. I mean there is always the question of which language do you choose. So it was a very concerted effort to get away from that and to say no we need to focus on creating problem solvers first."*

Interviewee U15b observed that the student evaluations for the unit have been really good, but the important consideration is how the students will perform in subsequent units. Students study at least one more programming language in their course, for example, Python, Java or C++. The transition to these subsequent programming units is currently of some concern, and the effects of the change are currently being formally evaluated (Good Practice Example 1).

The introduction of programming languages focused on mobile development platforms is a relatively recent inclusion in the programming curriculum prompted by current industry trends. Interviewees U24 (two interviewees were involved in this interview at the same time) described the introduction of Objective C and XML as the programming languages for smartphone/tablet development in iOS:

> *"We actually have started introducing some new programming languages. We now include objective C .... We now also teach XML and we've introduced smartphones and iPads into our learning space too."*

This further demonstrates the diversity of approaches that are currently being explored in introductory programming units. *"We introduced the Mac to replace the tablet PCs two years ago and they were introduced so we could teach iOS languages."* In part this change was made to appeal to students by targeting a computing environment, in the form of mobile devices such as smartphones and tablets, with which the students engaged on a regular basis (Good Practice Example 2). In terms of research, a formal evaluation and comparison of the range of approaches currently being trialed in the Australian context would be of benefit.

Some universities place the introduction to programming into a web development context, using web-scripting languages such as Javascript and HTML. Other languages mentioned included Visual Basic, C, C# and ActionScript (Flash). One interviewee indicated that a number of languages are covered across their degrees, but not in the first programming unit:

> *"What we do in the first semester. We teach it in a language neutral fashion… We deliver the material in language neutral fashion so it's about the programming concepts not specifically about the one language. We teach them the way to do something in general not in a particular language. Then we have material that helps them learn how to apply those concepts in a particular language." (U1)*

What the literature and especially the interviews highlight is that there appears to be little consensus as to what programming language or environment best supports novice programmers. Many institutions recognise the inherent difficulties for novice programmers, but the quest for the ideal learning approach appears far from over.

4. Future Directions and Recommendations

Based on the literature identified and the interviews conducted, a number of opportunities and future directions arise with regard to first-year ICT students and what curriculum and technologies they encounter.

We found little consistency with regard to the programming languages that are introduced to new programmers in ICT courses. While Java and Python are very prominent across the Australian universities of the people we interviewed, there appears to be no consensus on the best approach to take with novice programmers. This is also reflected in the literature, with research often highlighting the problematic nature of introducing both programming concepts and syntax. While we identified good practice in the use of a visual programming language and languages for mobile devices, further investigation is warranted on the adoption of programming environments that appear to shift the learner's focus away from syntax and onto problem-solving.

Recommendation 1

There has been a perceptible trend towards programming environments where the focus has moved away from syntax to problem solving. This is an area that needs investigation to determine how students respond to learning programming in these environments.

The other main scope for further research is in the use of formal skills frameworks provided by organisations such as ACS, ACM and IEEE. There is little literature and little understanding by the interviewees exactly how course curriculums are developed with these frameworks in mind. As discussed, there are a number of recent publications regarding SFIA and its role in curriculum development, and literature such as this may present an opportunity for more formal acknowledgement of these frameworks in this area.

Recommendation 2

Investigation is required to determine how formal skills frameworks provided by organisations such as ACS, ACM and IEEE can be most usefully applied in curriculum development.

Where we teach

Overview

The 'Where we teach' theme focuses on the teaching and learning spaces used for first-year ICT courses in Australian universities. It considers the design and use of new teaching spaces and redesign of existing spaces, either physical or virtual. For virtual teaching spaces, this theme includes teaching and learning in situations enabled through the use of mobile and ubiquitous technologies.

To provide a context for this theme we first explore trends and innovations in university teaching spaces, particularly first-year ICT programs, as reported in the literature. Findings from the interviews of ICT academics are then presented to provide insights into the use and design of teaching spaces used in the first year of Australian ICT courses.

1. Literature Perspectives

The systematic literature review found 13 papers that were concerned with the 'Where we teach' theme. All of the papers were set in the higher education sector and in the context of programming – all but one of them in introductory programming; two were Australian studies. See Table 4 for a list of the Australian papers for this theme.

Topic	Australian-focused references
Virtual teaching spaces	Alammary, A., Carbone, A., & Sheard, J. (2012). Implementation of a smart lab for teachers of novice programmers.
	Maleko, M., Hamilton, M., & D'Souza, D. (2012). Novices' perceptions and experiences of a mobile social learning environment for learning of programming.

Table 4: 'Where we teach' literature focused in the Australian context

The papers found for this theme report studies of a variety of different teaching and learning spaces. Govender (2009) explored the lecture setting in an investigation of the influence of the learning context on how students approach the task of learning to program and their ultimate success. Cheryan et al (2011) investigated the effect of virtual learning environment design on male and female students' interest and anticipated success in an introductory computer science course. Both studies concluded that context was an important factor in students' success in learning to program.

A study by Howles (2009) compared the impact of different learning environments on student retention. The findings revealed that a change from a studio environment (20 students with access to computers) to an active learning environment (40 students without computers) did not negatively impact student retention.

Australian researchers Alammary et al (2012) describe the implementation of a virtual 'smart lab' for assisting programming lab class teachers. The smart lab monitors students' progress as they perform programming tasks, enabling instructors to readily respond to individual students and assess the overall progress of the class. An evaluation demonstrated the usefulness of the smart lab in providing timely and appropriate feedback to the teachers. Another Australian study by Maleko et al (2012) explored novices' perceptions and experiences of a mobile social learning environment designed to enhance student-to-student interactions. A key finding of this study is that most students engaged more with their learning and with colleagues in the mobile social environment than in the face-to-face environment. Small learning communities were formed, enabling students to interact regardless of their physical location or the time of day.

Considerable resources have been expended on the development of environments to support the teaching and learning of programming, and a number of these have been specifically designed for introductory programming students. There are many studies of the use of these environments for engaging students in the learning process and helping them to learn to program. Verginis et al (2011) studied a web-based learning environment, SCALE (Supporting Collaboration and Adaption in a Learning Environment), and found it valuable for supporting learning in introductory computer science. Moons and Backer (2012) present an interactive programming environment, EVizor (Educational Visualization of the Object Oriented Run-time), implemented as a Netbeans plugin. The EVizor system visualises program execution and incorporates explanations and embedded quizzes. The system design is founded on constructivist and cognitivist learning theories. A series of evaluations and experiments showed that it is useful in helping students understand program behaviour.

Fincher and Utting (2010) introduce Alice (Cooper, 2010), Scratch (Maloney et al, 2010) and Greenfoot (Kölling, 2010), three environments widely used in introductory programming courses, each of which has a different focus and approach. The design rationale and pedagogical approach that each supports are explained in a series of articles by the designers. Wellman et al, (2009) introduced Alice into an introductory programming course to increase students' interest in computer science. They report that students were motivated and engaged in the learning activities. However, Garlick and Cankaya (2010) had a different experience. In an experimental study they found that students who used Alice in their introductory programming course had lower performance and responded less favourably to students who were given traditional instruction.

In summary, there are very few examples of recent literature discussing the first-year ICT learning environment in the Australian context, therefore further research is needed in this area. Current research focuses on specific examples of virtual lab software, the inclusion of social networking tools to promote learning communities, web-based collaborative learning environments, and a variety of introductory programming environments. There is a need to conduct further research on both physical and virtual learning environments that are tailored to the needs of first-year students in the ICT context.

2. Current Practice in Australia

The interview questions related to the theme of 'Where we teach' sought detailed information about teaching spaces in Australian universities and how they are used. In addition to describing the physical teaching spaces, interviewees were asked to provide information about their teaching in online or blended environments. Their responses gave insights into current teaching models and into the physical and virtual spaces where teaching is conducted. The responses to these questions are discussed under the main topics that were identified from the analysis of the interview data.

i. Teaching models

An important factor in a discussion of where we teach is the teaching model that is used. The most common teaching models used in the universities in our study are the traditional *lecture/laboratory* and *lecture/tutorial/laboratory* combinations. However, there were indications that a number of institutions had moved or were in the process of moving to different models, often involving a shift from physical to virtual teaching spaces. Many interviewees mentioned recent changes to lectures. Interviewee U21 described a radical change where a new degree has been implemented with only a single introductory lecture. Subsequently, students are provided with audio video clips and a text book in paper or electronic form. Tutorial classes are either on-campus or online.

A number of interviewees indicated that the teaching time devoted to lectures has been reduced. For example, interviewee U10 stated:

> *"So we used to have a very standard model of 3 lectures a week and 1 practical session and then we moved it to 3 lectures a fortnight and 1 practical session and 1 collaborative workshop session every week."*

In another example interviewee U7b indicated that they had

> *"Cut down lecture 2 hours to 1, less talking at the students, the boring stuff. Gone with a tutorial and a practical session, more hands on stuff particularly for the first-years."*

In addition, "*All recordings lectures and materials go onto an online Blackboard forum*." So students can access them when convenient.

Several interviewees mentioned the reduction of lecture time in order to increase practical lab sessions. For example, interviewee U24 commented:

> *"first-year programming a special case. … Combined lecture and practical into a workshop. For online students they submit weekly tasks to the lecturer and she checks and gives feedback within 24 or 48 hours".*

In this case the lecturer combined the traditional lecture and practical session into a 3- or 4-hour session (2 hours, a 1-hour break, then another 1 or 2 hours) and called it a workshop. Interviewee U24 observes enigmatically that "*Workshop mode equals flipped classroom minus the pre-class activities*." Although the reduction in lecture time and the corresponding increase in practical sessions was seen to be more resource-intensive it was also seen to be more productive in terms of increased student engagement and therefore increased student retention (Good Practice Example 3).

The most common teaching innovation discussed by interviewees was *blended learning*, and this was having an influence on the way teaching space is used. From the interviewees' comments, however, it is apparent that there are various understandings of the term 'blended learning' and a variety of ways in which this teaching model is implemented. A couple of interviewees used the term to mean the provision of online resources to both on-campus and online students. Several interviewees were exploring the 'flipped classroom' model, where the homework and class activities are reversed. Interviewee U18 said that first-year students had reacted negatively to this teaching model. She felt that the first-year students were not organised enough to watch the videos on their own and she questioned the suitability of this model for first-year students. In a more extreme example, interviewee U7a indicated that they favoured "*Small lectures, big tutorials. Light presentation and heavy practicals*." They indicated that they had *"Removed face to face lectures, some years ago"* and placed "*More emphasis on tutorials with the support of online modules using videos".* U7a further explained that "*Students need to look at video lectures and background readings before [the] tutorial*."

ii. Physical teaching spaces

Interviewees gave descriptions of their various physical teaching spaces. Lectures are typically held in theatres with capacities ranging from 100 to 400 students. Tutorials are usually held in classrooms holding 30 to 40 students. Laboratory classes are typically held in computer labs with space for 20 to 30 students, although a couple of interviewees mentioned labs of 40 to 50 students.

Most interviewees agreed that lecture theatres are less than ideal teaching and learning spaces. Many interviewees raised the issue of lack of student attendance at lectures. While there is a general shift towards reducing time spent in lectures or replacing lectures with more practical classes, there is also a considerable effort being made to improve the learning experience in lectures. Some have introduced new teaching models for lectures and others employ a variety of techniques to motivate and engage the students. These issues are discussed further in the 'How we teach' chapter.

Recording of lectures is now commonplace, with half the interviewees indicating that all lectures are recorded at their institution. Some interviewees stated that lecture recording is mandatory while others mentioned an opt-out policy. At a couple of institutions, where lecture recording systems are not readily available, some individuals record their own lectures. Only a couple of interviewees do not record their lectures in some way. The most common recording system is ECHO 360; others in use are Blackboard Collaborate and Lectopia. The availability of lecture recordings (and in some cases tutorial classes) has reduced the impetus for students to attend on-campus.

Most innovation in the design of physical teaching spaces is apparent in the computer labs where practical classes are held. Computer labs are traditionally set up with straight rows of tables and a computer for each student. At a couple of institutions there are variations on this arrangement. In one institution the lab has multiple fronts and in another the computers are placed around the four walls of the lab with the teacher in the centre. However, a number of institutions have made more radical changes to their computer labs, redesigning them into collaborative learning spaces. One interviewee described a room with tables seating 4 to 6 students, each with a large screen and one keyboard. Another described a similar teaching space with facilities for displaying the work of each group on a central screen for the whole class to view. Some of these labs hold more students than traditional labs and have been designed as flexible learning spaces (Good Practice Example 4).

A few interviewees mentioned more radical designs in teaching spaces. At one institution there are dual teaching spaces where students can move from a classroom setup to a computer lab in a large room divided by a partition. Another, smaller, institution uses only one type of teaching space. The room seats 50-60 students at eight sets of reconfigurable tables. This flexible teaching space has multiple fronts with a data display unit, fixed and mobile white boards and multiple power points around the perimeter of the room and hanging from the ceiling. One interviewee, describing a radical shift away from the traditional teaching model to a blended learning model, said that their learning spaces include "*libraries, site inspection and even corridor meeting, tearooms and virtual teaching environments*" (U7a).

iii. Virtual teaching spaces

Some interviewees acknowledged the increasing importance of virtual teaching spaces. Online learning is happening at most institutions, either with units taught only in online mode or with units taught online in combination with on-campus teaching. A number of interviewees mentioned small cohorts of online students in their on-campus units. Several indicated that all their units are available both on-campus and online, with students having access to teaching resources made available to both cohorts. They saw no difference between the resources provided to their on-campus and off-campus students. As interviewee U24 commented:

> *"I think we have two main teaching spaces – one is the physical space and one is virtual space. The virtual space is constructed with as much care to the design as the physical space is."*

All institutions use a form of Learning Management System (LMS) where typically all course materials are placed. The most commonly used LMS are Blackboard and Moodle. A couple of interviewees emphasised that these are not really learning environments but just delivery platforms for course content. One institution uses Captivate Workshop for delivery of learning objects. A couple of interviewees mentioned other online environments developed for use in specific courses. ViLLE (a visual learning tool) is a collaborative education platform developed specifically for learning programming, and IVLE (Informatics Virtual Learning Environment) is an online interactive instructional system for use in teaching programming and algorithmic problem solving.

iv. Summary

The interview data highlighted the trend for a reduction of lecturing activities and a corresponding increase in practical work. The format of the lecture is also changing, in response to decreased student engagement with this form of instruction. The layout of physical teaching spaces including computing laboratories is becoming more diverse and flexible. In addition, online provision of resources is becoming more prevalent, thereby improving flexible study options for students.

3. Future Directions and Recommendations

The good practices identified in this theme were concerned with the design and use of teaching spaces to engage students in active learning experiences. As indicated in the discussion of the literature review data, there is relatively little specific research on the physical and virtual learning spaces tailored specifically for the needs of first-year ICT students in the Australian context. This contrasts strongly with the significant changes to practice highlighted by the interviewees, including changes to the balance between lectures and practical labs, the changing nature of the layout of computing laboratories, the integration of social networking tools into online environments, and the increased provision of access the online learning resources. Further research is needed to assess the impact of these changes to the teaching environment on student performance and on the student experience.

Recommendation 3

Various new physical and virtual learning environments are tailored to the needs of first-year ICT students. Investigation is needed to assess the impact of these environments on student performance and on the student experience.

How we teach

Overview

The theme 'How we teach' is concerned with all aspects of the design and delivery of university-level learning experiences to first-year ICT students, along with associated supporting academic activities. The theme considers the theories influencing pedagogical decisions, approaches to teaching, collaborative and cooperative learning, teaching tools, and resources. These are discussed in terms of influences on student learning, motivation and engagement.

To provide a context for this theme we first explore teaching theories, models, approaches, and tools, particularly in first-year ICT programs, as reported in the literature. Findings from the interviews of ICT academics are then presented to provide an insight into current teaching practices in the first year of Australian ICT courses.

1. Literature Perspectives

The systematic literature review identified 57 papers that were considered relevant to the theme of 'How we teach'. The literature in this theme was grouped into four topics:

1. theories and models of teaching and learning – theoretical influences on pedagogy
2. approaches to teaching – including teaching techniques, tools, technologies and resources, and the use of games for learning activities
3. cooperative and collaborative learning – including teaching approaches to facilitate and encourage cooperative and collaborative learning
4. social media and learning communities – use of various forms of social media in teaching programs

All papers were set in the Higher Education sector and in the ICT discipline. Most papers were concerned with teaching in first-year courses. Fifty papers (88%) dealt with the teaching of programming, in particular introductory programming. Eleven were Australian studies, as listed in Table 5. The following review is organised according to the four topics that emerged from analysis of these papers.

Topic	Australian-focused references
Theories and models of learning	Mason, R., & Cooper, G. (2012). Why the bottom 10% just can't do it – mental effort measures and implications for introductory programming courses.
	Corney, M., Teague, D., Ahadi, A., & Lister, R. (2012). Some empirical results for neo-Piagetian reasoning in novice programmers and the relationship to code explanation questions.
	Lister, R. (2011). Concrete and other neo-Piagetian forms of reasoning in the novice programmer.
	Teague, D., & Lister, R. (2014). Longitudinal think aloud study of a novice programmer.
	Cain, A. & Woodward, C. J. (2013). Examining student reflections from a constructively aligned introductory programming unit.
Tools, technologies, resources	Risco, S., & Reye, J. (2012). Evaluation of an intelligent tutoring system used for teaching RAD in a database environment.
	Egan, M. H., & Mcdonald, C. (2014). Program visualization and explanation for novice C programmers.
Cooperative and collaborative learning	Falkner, K., & Munro, D. S. (2009). Easing the transition : a collaborative learning approach.
	Corney, M., Teague, D., & Thomas, R. N. (2010). Engaging students in programming.
Social media	Terrell, J., Richardson, J., & Hamilton, M. (2011). Using web 2.0 to teach web 2.0: a case study in aligning teaching, learning and assessment with professional practice.
	Guo, Z., & Stevens, K. J. (2011). Factors influencing perceived usefulness of wikis for group collaborative learning by first-year students.

Table 5: 'How we teach' literature focused in the Australian context

i. Theories and models of learning

A number of researchers have explored theoretical bases for teaching and learning in the ICT discipline, all in the context of introductory programming.

An Australian study by Mason and Cooper (2012) investigated lecturers' perceptions of the mental effort required for different aspects of their programming units. Interpreting the findings using cognitive load theory (Sweller, 1999), the authors propose that many low-performance students fail to learn due to cognitive overload. Skudder and Luxton-Reilly (2014) reviewed the use of worked examples in computer science. They evaluated different types of worked example in terms of the cognitive load on the learner, and recommend example-problem pairs and faded worked examples as most suitable for novices.

A number of researchers have challenged the 'programming gene' view that people are either programmers or not programmers. Robins (2010) investigated possible reasons for the bimodal grade distribution that some believe is typically found in introductory programming courses. He proposes that this is caused by the 'learning edge momentum' (LEM) effect whereby success in learning a concept helps in learning subsequent closely related concepts. In the programming domain, where concepts are tightly integrated, the LEM effect drives students to extreme learning outcomes. A group of Australian researchers have explored the learning of programming from a neo-Piagetian perspective (Lister, 2011; Corney et al, 2012; Teague & Lister, 2014). From a series of empirical studies they propose that novice programming students pass through neo-

Piagetian stages of sensorimotor, preoperational, and concrete operational before reaching the formal operational stage where they can operate as competent programmers. They recommend that introductory programming teachers use a neo-Piagetian perspective in their instruction where they consider the reasoning levels of their students.

A couple of studies have used Dweck's (2000) 'mindset' theory in introductory programming teaching programs. Dweck identified that learners may have 'fixed' or 'growth' mindsets, which have implications for their learning. Students with a growth mindset focus on learning goals and continue to focus on learning even after failures. By contrast, students with a fixed mindset focus on performance goals, and want to be seen as achieving well at all times. Through several interventions implemented in an introductory programming course, Cutts et al (2010) found that they were able to shift students from fixed to growth mindsets, resulting in a significant improvement in their learning. An intervention program by Hanks et al (2009) reported less success.

Dann et al (2012) report an application of the theory of 'mediated transfer' (Salomon & Perkins, 1988) in the design of an introductory programming course. The purpose was to aid students in transfer of their knowledge of programming concepts learnt in Alice 3 to the Java context. Using this approach they found dramatic improvement in students' final examination performances.

A couple of papers report the use of Biggs' model of 'constructive alignment' (Biggs, 1996) as a framework for design of introductory programming units. Thota and Whitfield (2010) and Australian researchers Cain and Woodward (2012) describe the design of their courses and present results from action research studies. They discuss the implications of the use of constructive alignment as a framework for course design.

A comprehensive review by Sorva (2013) summarises the research on challenges faced by novice programmers in understanding program execution. Based on findings, he proposes that the 'notional machine' should be used explicitly in introductory programming to help novices understand the runtime dynamics of programs. Ma et al (2011) investigated novice students' metal models of programming concepts, finding that many held non-viable mental models of key concepts. Through a teaching approach using visualisation of program execution they found that they could challenge and change students' misconceptions and help them develop a better understanding of key concepts.

ii. Approaches to teaching

Different approaches to teaching form a broad topic encompassing the use of techniques, tools, technologies, games, and resources in teaching first-year ICT courses.

Teaching techniques

A variety of teaching techniques for first-year ICT courses were found, all but one in the context of programming. These were typically introduced to improve students' skills and knowledge of a particular learning outcome and/or to motivate and engage students in the learning process.

- Caspersen and Kölling (2009) present STREAM, a programming process for novice programmers. This process was derived from a stepwise improvement framework that the authors developed by unifying current good practices in software development. STREAM has been used in two universities, and a study indicates that it helped in the development of students' software development competencies.

- Apiola et al (2012) present CSLE (Creative-Supporting Learning Environment), a theoretical framework for the design of a course designed to support students' creative activities. The framework was trialled with a programming course using robotics, and an evaluation indicated that students gained many creative experiences during the course.
- Hu et al (2012; 2013) describe an approach to teaching introductory programming using the concepts of 'goals' and 'plans'. They propose a notation and a programming process incorporating these concepts. An evaluation of the approach using an experimental method indicates a positive improvement in students' programming exam performance.
- Pears (2010) discusses the concept of program quality and students' conceptions of program quality. He describes an approach used in an introductory computing course designed to give students an understanding of program quality. An assessment of student code produced for their project work indicated a level of quality above what is normally produced by first-year students.
- Hertz and Jump (2013) present 'program memory traces', a paper-based approach for code tracing that models program execution in the computer's memory. A study of the use of this approach in an introductory programming class showed improvement in students' programming ability, decrease in dropout rates and significant improvement in students' grades.
- The only example found outside the programming context was NEMESIS (Marsa-Maestre et al, 2013), a framework for generating scenarios for teaching network and security systems b. An evaluation of the framework with a first-year internet security systems course found that the students and teachers were positive about the use of the framework and the scenarios generated.

Games

Game-based learning and assessment tasks are often used to motivate and engage students in the learning process. The following research reports the use of games in introductory programming classes.

- Morazán (2010) describes an introductory functional programming course that uses learning activities based on video game development. He proposes the use of these activities in introductory programming classes to motivate and enthuse students in programming.
- Kurkovsky (2013) reviews the use of mobile game development in programming courses and describes how mobile game development is used in introductory programming courses. Studies at two institutions showed mixed results in terms of engagement and motivation.
- Eagle and Barnes (2009) describe a role playing game, Wu's Castle, that was developed to help introductory programming students learn loops, nested loops and arrays. Studies showed that the game was an effective learning tool for these programming concepts and students preferred learning with this game over traditional activities.
- Bayzick et al (2013) present ALE (AndEngine Lehigh Extension), a platform for Android game development. ALE emphasises code reading before students attempt code writing. Experiences with using the platform in an introductory programming course found that students responded positively to the tool and wrote "compelling mobile games in under 18 hours" (p.213).

Tools, technologies and resources

A range of tools and technologies have been developed or adapted for use in computing education, all but one in the context of programming:

- Anderson and Gavan (2012) report on the introduction of LEGO Mindstorms NXT into an introductory programming course. They found that students' results on assignment work and exams improved, and concluded from a student evaluation that the activities were a stimulating and engaging challenge for the students (p.143). Apiola et al (2010) also describe a programming course that uses LEGO Mindstorms robotic activities. On the basis of many positive student comments during and after the course, the authors argue that robots are a powerful tool for motivating students. These conclusions were not supported by a study by McWorter and O'Connor (2009) who used the Motivated Strategies for Learning questionnaire to assess the effect of LEGO Mindstorms robotic activities on student motivation in an introductory programming course. An experimental study showed no difference in intrinsic motivation between the students using LEGO and non-LEGO activities, although responses to qualitative questions indicated that some of the LEGO students enjoyed the activities.
- Summet et al (2009) describe an introductory programming course where each student is provided with a pre-assembled robot which is used as the teaching context. Results of a comparative study showed that the robot class students gained significantly higher results than the non-robot class students.
- Daniels (2009) reports on an application of Nintendo Wii Remote (wiimote) technology in an introductory computer engineering and problem-solving class, and the laboratory exercises designed to use the technology. Following a study of the use of the technology, the authors believe that the activities helped students achieve the core learning objectives of the course and that student engagement improved.

A common application of technology in computing education is program or algorithm visualisation, which is used in computing education to clarify and explain concepts.

- Sorva et al (2013) review visualisation systems designed to help introductory computing students understand the runtime behaviour of computer programs. Evaluations of the systems provided indicate that they are generally useful in helping students learning programming; however, the influence on learner engagement is not clear.
- Pears and Rogalli (2011) present an extension to the widely used program visualisation tool Jeliot, where students are able to receive and respond to Jeliot-generated questions on their mobile phones. They propose that this can be used interactively in a lecture, providing an alternative to clicker technology.
- Australian researchers Egan and McDonald (2014) describe systems for visualising runtime memory state and their integration into the SecC system. This system will be used initially in a first-year Operation Systems course and the C Programming Language course.

The only example of a tool or technology found that was not used in a programming context was an intelligent tutoring system for learning Rapid Application Development in a database environment. An Australian study by Risco and Reye (2012) describes the Personal Access Tutor (PAT) and an evaluation of the tool in a first-year database course, showing that students and staff found it easy to use and that it was beneficial for students' learning.

Börstler et al (2011) report a study of example programs from novice programming textbooks. The programs were reviewed and evaluated by experienced programming educators and they concluded that the quality was not as high as should be expected from textbooks.

iii. Cooperative and collaborative learning

Various teaching approaches have been developed to encourage collaborative and cooperative work behaviour in first-year computing students, often with the aim of developing and fostering learning communities.

Hamer et al (2012) provide a concise overview of current research perspectives on learning communities by exploring the concept of 'contributing student pedagogy' (CSP).

> Activities that require students to collaborate, share solutions, review each others' work, or create materials explicitly for use by other students are beneficial not only to students' learning of course material, but also to their reflective, critical, creative and inter-personal skills. Such activities provide opportunities for students to engage more deeply in subject material and develop important graduate attributes. (Hamer et al, 2012, p 315)

The learning benefits of engaging learners as active co-creators of the learning experience have been demonstrated in a number of subject domains. Collaborative learning activity has been used as one of the primary methods of implementing this objective as it requires learners to externalise their understanding in order to work with their peers.

> The idea of CSP was developed by Collis and Moonen (2005) who emphasise the process of learning by engaging students as co-creators of learning resources. CSP incorporates social constructivism in a practical manner, combining both content learning and inter-personal skills acquisition in a meaningful way. (Hamer et al, 2012, p 315)

Collaborative learning describes a range of practices where students work in groups sharing knowledge or work on a project. An example of a teaching approach that uses collaborative learning is the 'peer-led team learning' (PLTL) approach as described by Murphy et al (2011). PLTL involves a small group of students working collaboratively to solve problems. Each group is led by an undergraduate workshop leader, who has been specially trained in PLTL technique. A PLTL program consists of a set of weekly one- or two-hour workshops adjunct to an academic course. The focus of this approach is to demonstrate that computer science is a collaborative activity based on problem-solving and algorithmic thinking rather than programming. The program was highly beneficial for peer leaders who also benefit from the program as they gain confidence in themselves as computer scientists.

A couple of studies discuss collaborative learning techniques used to increase engagement in lectures. Simon et al (2010) report on an application of peer instruction (PI) using clicker technology in two introductory programming units. PI is a teaching technique that involves students answering a question on a vote-discuss-revote model. An evaluation found that students were generally very positive about this approach and that the accuracy of the responses increased after a follow-up discussion. The instructor reported value in being able to identify concepts that students had not yet mastered. Kothiyal et al (2013) describe the implementation of a similar active learning strategy, think-pair-share (TPS), in a large introductory programming class. TPS involves students working on an instructor-led activity individually, in pairs, and then as a whole class. The authors report levels of student engagement for each activity ranging from 70% to 90%.

Cooperative learning, a specific kind of collaborative learning, is a teaching strategy involving students working together to improve their understanding or to complete a task. At an Australian university, Falkner and Palmer (2009) integrated cooperative learning techniques into an introductory computer science course, resulting in increased class attendance, improved learning outcomes as determined by exam results, and increased student motivation. Beck and Chizhik (2013) report on the implementation of cooperative learning in an introductory computing course and also found an improvement in students' exam results.

Lasserre and Szostak (2011) used a team-based learning (TBL) approach, requiring students to work on exercises in teams. The approach had a positive outcome on student learning: 20% more students completed the course and 20% more students passed the final exam. Informal inspections of the final exam answers suggest that students who learnt using the TBL approach had increased confidence in writing programs. Another team-based approach, reported by Hundhausen et al (2013), involved peer-reviewing code with the help of a trained moderator. A series of studies showed that pedagogical code reviews (PCR) facilitated multi-level discussions of code practices, providing opportunities to develop soft skills in introductory computing courses. The study also showed that the online implementation of PCR was not as effective as the face-to-face PCR.

Many studies have investigated the effectiveness of pair programming as a form of cooperative learning for introductory programming students. Pair programming is a programming technique where two people work together to write a program, alternating between 'driver' and 'navigator' roles. Australian researchers Corney et al (2010) implemented pair programming in an introductory programming course at an Australian university and report that it was well received by students. Wood et al (2013) describe the use of pair programming in the early weeks of an introductory programming course. Students were paired based on the comparable levels of confidence, and it was found that students with the lowest level of confidence performed better working in a pair than individually. Staff observed increased engagement, motivation and performance. Radermacher et al (2012) investigated the formation of pairs using Dehnadi's mental model consistency (MMC) test and found evidence supporting the approach of matching students according to their mental models. Salleh et al (2010) explored the effect of the personality trait of neuroticism on pair programming and reported that students' performance is not affected by different levels of neuroticism. Zacharis (2011) and Edwards et al (2010) investigated the effectiveness of online pair programming for introductory programming students. Zacharis found that students working online using pair programming produced code of better quality and more efficiently than students working individually. However, Edwards et al found that students were less satisfied with the experience of online pair programming than when they were co-located.

O'Grady (2012) reviewed the literature on the use of problem-based learning (PBL). More than a third of the 59 cases were first-year computing courses and more than half of these were programming courses. O'Grady found that both teachers and students were largely positive about their PBL experiences. However, he found that the adoption of PBL into computing courses was largely ad hoc and random and concluded that if it is to be successfully used then "motivations, objectives, learning outcomes, and graduate outcomes must be clearly defined" (p 10). Sancho-Thomas et al (2009) present the NUCLEO e-learning framework, a PBL-based environment for teaching computing courses. From the results of three different studies on the use of this framework the authors conclude that NUCLEO had a positive influence in decreasing dropout rates, raising exam pass rates, and improving team formation.

iv. Social media

Recently, various forms of social media (web 2.0) have been used in education programs to encourage collaborative work behaviour and the formation of learning communities. Using social media for learning activities is also seen as a way to engage students in learning. A number of the implementations of contributing student pedagogy involve the use of social media (Hamer et al, 2011).

Pieterse and van Rooyen (2011) report the use of Facebook in a large first-year computer science unit. A closed Facebook group was set up as an informal online discussion forum complementing a formal discussion forum set up on the department website. Analysis of the usage of the forums showed greater use of the formal forum; however, there was more evidence of an online community on the Facebook forum. The authors' impressionistic view was that students were more engaged than in previous offerings of the course.

Two studies investigated the use of blogs to support learning communities. McDermott et al (2010) describe the use of blogs in a collaborative and professional skills unit of a first-year computing course. Students were required to use a blog for a reflective diary and to post comments on other students' blog postings. The authors report that most students used their blogs in an educationally constructive way and the postings gave valuable insights into the students' experiences. Robertson (2011) describes the use of blogs in an introductory interactive systems course. Students were required to keep a design diary as a blog and to comment on the blogs of other group members. Analysis of the blogs gave insights into students' self-directed learning strategies and the support they provided to peers.

At an Australian university, Terrell et al (2011) required students to record their reflections and learning activities on a blog. Analysis of the blogs provided indications as to how well the course objectives had been met. At another Australian university Guo and Stevens (2011) used wikis for collaborative assignment work in an introductory information systems course. From the results of a student survey they provide recommendations for instructors who are considering using web 2.0 technology in their teaching programs.

v. Summary

There is a significant body of literature devoted to the theories and models of learning, various approaches to teaching, cooperative and collaborative learning techniques, and the use of social media. A large proportion of this material was highly focused on the programming domain and only a small portion related specifically to the Australian context. Further research is needed to gain better evidence for the efficacy of specific techniques and further work is also need to conduct comparative evaluations of different approaches to how we teach. It would also be beneficial to collate examples of good practice that have been rigorously evaluated to assist in their wider dissemination.

2. Current Practice in Australia

The issues raised by interviewees and examples of good practice related to how we teach are discussed in the following sections using evidence gathered from the interviews. Analysis of the interview data on this theme found four broad topics, of which the first three topics align with topics from the literature search:

1. approaches to teaching – including teaching techniques, tools, technologies and resources, and the use of games for learning and assessment tasks
2. cooperative and collaborative learning – including teaching approaches to facilitate and encourage cooperative and collaborative learning
3. social media and learning communities – use of different forms on social media in teaching programs
4. teaching experience and practice

An underlying theme across all topics is the response of academics to the perceived lack of student engagement with traditional methods of on-campus course delivery in universities, in particular the traditional lecture model of content delivery.

i. *Approaches to teaching*

A common element in this issue was the aim to increase learner engagement through converting the learning experience from a passive activity of absorbing information to an active process whereby the learner must engage and process the content in order to construct meaning from the experience.

Lectures

The most dominant themes concerning approaches to teaching are the issues with lecture delivery and responses to the lack of student engagement with learning in this space.

Several interviewees raised the issue of lack of student attendance at lectures and were making attempts to address this. For example, interviewee U7b indicated with regard to their lectures:

> *"Deliberate change to improve engagement. … A complete change of staff, a complete change of pedagogy, a restructure of the delivery approach, etc. … Because we found that the engagement and therefore the attendance and the interest … is dropping off with this sort of generation. We've made a conscious decision to put our brightest performers, you might say, on first-year units."*

In another example interviewee U15b discussed the rationale for the introduction of clicker technology into several first-year units:

> *"The other thing that is impacting the first-years is the use of clicker technology, because we have been focused on first-year units. And that has been in part to try and improve the lecture experience and also get attendance back up. You know that lecture attendance is the first thing that kind of goes when students are under pressure so we try to be quite compelling in having them in there and them knowing why it is important and what they can get from it."*

The consensus of comments indicated that it was important to provide students with an engaging and active lecture experience in order to motivate them to attend and participate in learning.

Lecture approaches that focus on transmitting content were seen as problematic since other sources of high quality information were available online in formats that could be accessed more conveniently off campus. A number of high quality MOOCs have focused on computing and ICT and are an example of the increased availability of resources of this type. The strengths of on-campus delivery were seen as being the ability to encourage active student participation, the responsiveness of lecturers in providing quality student feedback on progress, the social learning context involving their peers, and personalised feedback to students. Therefore, lecture techniques and pedagogies have been developing to take advantage of these strengths. Interviewee U15b highlights this increased focus on learner engagement in the lecture context:

> *"It has been and it is going to continue to be initiatives about making the on-campus experience, particularly lectures but also the labs and the tuts, much more, you know the flipping the classroom thing rather than just information dissemination. Just trying to really engage them in the class."*

One example of this process is the technique of the flipped classroom (Porter et al, 2013; Simon et al, 2013) and the introduction of clicker technologies. The use of these techniques is not new in other disciples but they are increasingly being adopted in the ICT domain and the Australian higher education system. Interviewees U15a and U15b described the use of flipped classroom techniques and clicker technology specifically targeted at first-year students:

> *"Clickers ran in 3 first-year units, [with] both formal and informal evaluations. Attendance was a significant improvement, unit evaluations quantitative numbers did not change much, particularly in overall view of the unit. But the qualitative comments were promising. A handful of students did not like it but a number felt it was really of benefit for them." (U15b)*

Other interviewees indicated that they had implemented some components of the flipped classroom model but some indicated that it had proved problematic to motivate students to do the required pre-reading, so the approach was discarded.

Interviewee U15b gave a brief description of the flipped classroom process:

> *"Clickers were implemented by previous Associate Dean Education [with] a few academics trialling. Data was captured in real time to push the direction of the class. It went well. More of a flipped classroom environment. Pre-reading is expected. The way those lectures work is that there will be a quick summary and then there will be some sort of question posed to the class, they tend to discuss it in small groups, ... You get a sense for how much time you need to spend on it for the response. Students will get into small groups to discuss it and then they re-answer and then you can get a sense for how their understanding is shifting through a bit of discussion and prompting of them."*

The aim of these techniques is to get students to actively engage with the fundamental concepts through a process of discussion and responses undertaken in conjunction with their peers. This also allows the lecturer to better judge the current state of understanding demonstrated by the class through their electronically submitted responses. Interviewee U7a also indicated that they had implemented a flipped classroom methodology, although clicker technology was not implemented in this case.

Interviewees U15a and U15b went on to indicate that the Faculty concerned intended to expand the flipped classroom and clickers program further:

> *"So much so that we are continuing on and we are expanding the program. Results were a bit better. What we found was, which was actually quite good, is that it brought the tail up a bit. So we thought it might have a bit of an impact on students at risk … " (U15b)*

Interviewee U15a adds *"It encourages them to actually attend. We're starting to have more units using clickers this semester."* (Good Practice Example 5). Further research is required on the impact of these techniques and technologies in the ICT domain and in the Australian context.

A variety of other approaches are used in lectures to engage students in learning experiences. Interviewee U24 uses live code writing and demonstrations to increase the interactivity of lectures. Interviewee U12 uses online quizzes within Moodle:

> *"students can either use their phone, their computer or the tablet I provide to ensure that everyone has access. It's an online quiz so they get instant feedback as to how they've gone and I get the individualised feedback so I know who's struggling".*

Role-playing is a novel approach used by two lecturers. Interviewee U23 explains:

> *"I do a lot of role play in lectures to try to reinforce some of the concepts. So I have people acting out variables and loops and things like that. It's a bit of a giggle, but students who struggle initially to try to understand what these concepts are, seem to find that really helps".*

Lecturer (U2) describes his use of role-playing:

> *"The lecturer pretends to be a person in need of some IT assistance. The students interview the 'client' and establish what they want and come up with something, usually in the form of a diagram, and receive feedback on it later. It isn't marked, it's used as a teaching tool".*

Interviewee U23 shared his experience on having guest lectures in his course:

> *"We have guest lecturers every second week in the subject and try to mix them up across different fields so you get very engaging, inspiring people. … We're very selective about who we approach to do [the lecture] and students love it. Of course we make that examinable so they actually have to come along to the guest lectures."*

Despite efforts to improve the lecture experience, a couple of interviewees expressed strong negative views about teaching in this space. Interviewee U5 encapsulates these ideas:

> *"I think the future of the lecture is in significant danger… students get very little value from lectures. The attendance is poor, the interaction is virtually all one way and today's students really don't see it as any benefit whatsoever… and the students are far busier now than they were 20 years ago when the university may have been a priority. University isn't a priority anymore. The majority of our domestic students are working at least 20 hours a week and they see uni having to fit around them, not the other way round. I understand the challenges and there does have to be a nice balance but the changes have been quite dramatic and the universities are still teaching to the students as they were 30 years ago when students would come to class."*

Teaching techniques, tools and technologies

Although discussion of how teaching is approached was focused on the lecture environment during the interviews, a variety of other teaching techniques were mentioned that were appropriate for tutorial classes or online learning, often involving the use of specific tools and technologies. The motivation behind these was to engage students in interesting and meaningful learning experiences.

Interviewee U9 explains how she focuses on students' interest to increase engagement:

> *"Every single week we have two or three 3-minute oral presentations by students on any topic of interest to them. Other students give feedback, because we're scaffolding their learning about how to present at the end of the semester. And that's great fun. …. They don't get marked on it; it's formative".*

A similar approach is followed by interviewee U6:

> *"If they ask a question that goes down a different path … I'll happily go along with them and encourage them to look into it further and let me know what they find out ."*

Interviewee U6 argues that project work needs to be authentic to promote student engagement: *"The students engage in projects that are fascinating and do authentic tasks of real world challenges and coming up and creating something new. Not just learning by rote."* Similarly, interviewee U20 stresses the importance of providing opportunities to do meaningful and motivating work in his programming unit.

Interviewee U7b discusses the use of visual programming techniques based on a Stanford University model in which students learn to program by moving objects around a screen in a game-like environment in which the effects of the code and its successful execution are immediately apparent to the novice programmer.

> *"The ladybug is very visual. The aim is to run the code and see the ladybug move in the correct way instead of the old way of running the code and not getting an error and maybe producing a report. What you are seeing is a visual representation of your result. Quite a bit different to the old pedagogy."*

In addition, at this university students now do a programming unit in every semester in order to build an orderly progression of skills acquisition.

Interviewee U10 describes an introductory programming technique called media computation where students learn to program using the manipulation of images and sounds as the context for learning about programming. This course changed from Java to a combination of C++ and Python. As U10 explains:

> *"Media computation [is] really new. Introduced three years ago, [as a] first course for people who do not know anything about computing. People learn to program by manipulating images and sounds."*

Part of the rationale for this change was to appeal to a wider audience, including for non-ICT students. *"[This] gives them a context that is very neutral. [We] found it to be quite attractive to students outside computer science."* With regard to changes in programming language selection, *"Before, Javascript, but [it had] too many idiosyncrasies. Before that Java where you build boring bank accounts, etc."* The aim with the current set of techniques is to reduce the barriers to engaging with programming concepts. (Good Practice Example 6)

Interviewee U10 indicated that there were ongoing studies into the effectiveness of these new techniques, but where they were introduced along with collaborative learning techniques they noticed a significant increase in retention.

> *"Were able to identify that there was going to be a significant improvement in student engagement and retention from that. That was why we made the change that we did, the massive change across the whole of first year."*

There is a need to follow up to see if it was a consistent improvement rather than a one-off effect:

> *"The students are very much more engaged and they are actually practicing a lot more. One of the things that we have seen, that is really nice, is that the students are more actively doing their own projects, because they seem to be more comfortable in programming and their practical application of the ideas. They are not just doing the assignments and that is it."*

So far results have been positive:

> *"The students do seem to be more engaged, they are more enthusiastic, they are attending more classes, so we are taking that as a win enough at the moment."*

Again there is a sense is that there is not really an improvement at the higher end of student performance but more engagement at the lower end, with a possible consequence that more students are able to pass the introductory programming unit.

Educational resources

There were several comments in the interviews regarding the creation and use of educational resources. Interviewee U7a described an open educational resources (OER) scheme. This was a learning object repository of submitted student work that was created and maintained on a formal basis:

> *"Previous students' work can be referenced, can be extended, can be reused, and can be enhanced. That means the currently enrolled students can make use of previous students' work for improvements, for extensions and for some other kinds of extra work; however, students need to follow the OER scheme."*

The aim was to build up a rich repository of student-generated content, and participation in the scheme was voluntary.

Another interviewee, U15a, described an e-publishing initiative called Alexandria that was based on WordPress infrastructure. The aim was to create dynamic and interactive learning objects that could be distributed on a variety of platforms.

> *"We have another project taking [the] online learning repository type thing and creating kind of learning modules. Again trying to do them in a more dynamic way, so short videos with interactive applets students can experiment with and stuff."*

This is a type of e-publishing with interactive elements embedded, such as quizzes, applets and discussion forums.

ii. Cooperative and collaborative teaching

This topic is concerned with teaching approaches that involve students in collaborative or cooperative learning activities. Cooperative and collaborative learning activities were highlighted in the interviews as examples of active learning pedagogies for first-year students. Interviewee U10 explains:

> *"We do a lot of student contribution work in first year. We use all of them across the curriculum, but in first year it is very much based upon peer assessment and peer review, peers working together in collaboration. So our curriculum was restructured ... about 4 years ago now. We completely rebuilt the first-year curriculum around collaborative learning."*

The aim here is to recast learning from being an isolated and solitary activity to being an intensely social activity where the student is engaged and motivated by negotiating shared goals, responsibilities, and cooperative tasks involving their peers. The social nature of this learning experience and the intense engagement is intended to reduce the social isolation of students, which has been shown to be one of the significant risk factors for students dropping out of courses. (Good Practice Example 7)

Interviewee U10 elaborates:

> *"In the collaborative workshop sessions students do a lot of very active learning, they have little mini-lectures, that are interjected between collaborative learning activities where the students are often asked to build upon each others' work, to share each others' work and do peer review and peer assessment."*

Here the aim is to foster a range of skills related to the ability to plan solutions, negotiate roles, and evaluate progress, rather than just to absorb specific information. These social skills are deemed to be important in the context of future employment in the ICT field and tend to produce a more engaging learning experience.

According to interviewee U10, however, these collaborative learning techniques require a range of specific teaching techniques in order to ensure their successful implementation.

> *"They are very heavily guided through the workshops. They are all face-to-face, so we have quite a lot of workshop supervisors who work with the groups. So the workshop supervisors go through training every year to sort of guide them into how to work with the student groups."*

Groups in this case consist of about three students. U10 states that the learning activity will consist of *"maybe a 10-minute e-lecture and then a 10-minute independent activity and then a 20-minute collaborative activity."*

Careful planning and management is required to ensure productive collaborative learning activities within the groups, so that no one group member dominates the session.

> *"And so they will go through and get a group and change the person they are talking to every time to make sure everyone is getting a chance to contribute, and we change scribes and so on."*

In this case training is provided for the tutors implementing the collaborative learning activities.

> *"And we have a set of little techniques and practices for small group work that our tutors are taught to use to help them to get groups to work together. ... They will change the person who is at the keyboard for a different activity. They will get the two student groups to swap resources half way through so they can have a discussion about how they are interpreting resources. All sorts of little activities that they do basically to move things around and make sure that they are still talking."*

Again further research is needed to formally describe and evaluate the impact of these techniques in the Australian ICT context.

A related active learning pedagogy is focused on problem-solving skills and in setting the frame of reference for learning activities in authentic problem contexts relevant to the ICT domain. This is demonstrated by interviewee U9:

> *"We have got peer collaboration within classes and some topics (our subjects are called topics here) use partnership learning. And there is a student focus of what is going to be taught. There is a topic in which students undertake an external challenge of a real world scenario for Engineers without borders."*

The aim is to increase the relevance of the content to the students' real world experience and to demonstrate to them the rationale for learning specific skills by demonstrating how those skills will be used in the real world. (Good Practice Example 8)

The innovation in this example is that this experience is targeted at first-year students in a professional skills unit rather than being delivered in a capstone unit in the third year of the course.

> *"Now our Computer Science, Engineering, and our ICT students participate in that, where they design real world solutions for ICT problems in third-world countries. They design their own solution and it is incredible what they do in first year."*

Students are motivated to gain skills as they go to complete the current project rather than completing a series of units to gain a set of decontextualised prerequisite skills to be used at a later time.

iii. Social media

This topic is concerned with use of social media for learning activities in first-year ICT. Interviewee U24 describes the use of social networking software UCROO to support learning communities (Good Practice Example 9). UCROO is a social networking application for Australian universities only, and was developed by post-graduate students from Deakin University (which is not the university of interviewee U24). It is an educational social networking site based on Facebook.

> *"Looks a lot like Facebook, acts a lot like Facebook. The students are very familiar with it. They know how to use it immediately. It is unit specific, so you set a unit up in this. It definitely has an educational focus because you can set up assessment dates, and the like. Each unit has a wall on which you can post, do a poll, ask questions, put up a file, link to a web page. But students can, too, so you get connections like you get Facebook friends. Everyone who is your friend, you have one common wall that you can see."*

This is a rich tool set of features to promote social connections and to allow posting of news and resources. This is very different from the limited tool set available in the current generation of LMSs. According to interviewee U24:

> *"It is a lot more friendly to use as a tool than the stuff that is packaged in the learning management system. Blackboard [was] built in 1997, so is very old and clunky. Nowhere near as responsive and to have all the different type of social networking tools that the social networking software [has], that the students have access to. And so they look with such disdain on the software that is part of Blackboard for that purpose. The institution insists that we use those sorts of tools but they just don't resonate with students at all."*

With regard to the implementation of the UCROO software, interviewee U24 stated that active participation of the lecturer is required to monitor the content posted to the site to avoid potential offensive material being uploaded, although access can be restricted to invitational only. However, interviewee U24 had not found any problems with postings:

> *"The students have been wonderful and I have never had any complaints about what the students post on it but you decide what flavor it has, you keep the momentum going throughout the semester."*

Interviewee U24 highlights the benefits of social networking in connecting online students with the on-campus student group: *"So it is work but it is work that I have found to be very beneficial to the students, particularly externals."*

According to interviewee U24 the software was:

> *"Introduced 18 months, 2 years ago, to the introductory programming class, because they of course are a really quiet class because they are programmers, they tend to be quiet. They tend to be not so out there socially, and I also wanted my external students to get to know my internal students and for my internal students to be reminded that the class does not only consist of them."*

The initial results have been positive:

> *"It has been magnificent, students have loved it and I have had an enormous amount of student interactivity as in student between student on UCROO each time I have used it. … Both times it actually really surprised me how these people just took to it like ducks would take to water."*

The lecturer is also starting to build social networking tools more broadly into the unit, such as Skype for external presentations, web-based clicker systems for in-class polling, etc. Therefore the use of social networking has shown the potential to increase peer feedback, and to integrate online and on-campus students if implemented correctly.

Further research and evaluation on the impact of social networking techniques on the ICT domain is required.

Another interviewee, U4, describes issues with the use of social media:

> *"It is pretty much open slather for Facebook in the course, none of the staff are watching that space. It is difficult to encourage students to use it because they think this is just another burden on what they're required to do. But again it's something that we encourage and point out that it's for their benefit because there are theories that if they interact with each other it will enhance learning."*

To address these issues they plan to use Facebook in a more structured way to enhance interactions among the student cohort.

Several lecturers observed that because students cannot be forced to use social media, there is a need to duplicate all communication to the class in both the social media platform and the learning management system, thus increasing the work for staff. One interviewee remarked:

> *"The university is moving towards more social media but I think there are a few issues in using that extensively in teaching because students don't really distinguish between whether the social media contact is social or educational. It kind of blurs the boundaries for them." (U7b)*

iv. Teacher experience, expertise and change

This topic relates to the perceived importance of ensuring that experienced and high-quality teachers were assigned to teach first-year units in order to provide a high-quality learning experience.

There was a general consensus in the interview comments that permanent staff of high quality were needed for first-year teaching. Several comments indicated that allocating quality staff for first year was a priority; interviewee U10 stated *"the bulk of teaching is done by permanent staff, casuals supervise workshops rather than teaching"*, and interviewee U24 *"As little as possible teaching for first year is done by sessional staff."* According to U7b:

> *"We've made a conscious decision to put our brightest performers on first-year units so they're the ones that when they get in front of a class they really light up the room. Even if they don't have as much experience as another senior academic. We're actually really trying to keep the kids interested and in some instances that is a younger person."*

An interesting example of the nature of change in teaching introductory programming is provided by interviewee U7b, who says that in order to effect change in this unit a complete change of staff, a complete change of pedagogy, and a restructuring of the delivery approach were implemented.

> *"A whole heap of stuff changed, it was not an easy change for staff here because I removed almost all previous staff from the unit, we are just talking programming, and I put in a whole new teaching group. But the reaction from the students is saying that that is working."*

Therefore, in order to implement a radical restructuring of the programming unit a change in staff was required rather than existing staff adopting new pedagogies:

> *"Took over as head of program in 2012 did a full strategic review, talked to everyone, put paperwork through in 2012 for a 2013 start."*

Interviewee U7b discussed the difficulty in implementing radical change in teaching methods while retaining staff who had been teaching in a fixed manner for a long period:

"Because of the use of the very experienced programming staff, because they had not been in industry recently, they were still sort of teaching old habits and teaching in old techniques. I don't mean that Java, I mean their teaching style. So they really were not in touch with the way the current generation thinks and likes to learn and so we had quite negative comments about the lecturers before and they are improving ….. "

Therefore a complete change in staffing was implemented:

"But it is a conscious decision, and it is hard decision to make because I am using some very, very, good staff on these massive lectures but it is a real strategic move for us."

Here the aim is to allocate staff who can maintain the interest of first-year students and to whom the students can easily relate:

"And in some instances that is a younger person, I don't like to use young versus old because that is not the case that we look for, what we do look for is people who stand in front of a group and capture their imagination. Some of the people are closer to the demographic of the students, let me put it that way, but some are not."

In some ways this is an attempt to ensure that the staff allocated to the first-year teaching are aware of and in touch with the perspectives and interests of current students.

Interviewee U7b has indicated that the initial results of these changes are positive for the students concerned:

"We have definitely got great engagement, or better engagement. The staff themselves have noticed a great change in the students, particularly at the smaller campuses. They are really starting to get some great interaction with the students."

Evaluation of the changes is currently under way:

"Student feedback on the unit, student feedback on the teaching, they have definitely improved, they are much more interested, they find the lecturer really sort of gets them, you know, they can talk more easily to the lecturers."

An interesting effect is that the lecturers themselves perceive the teaching experience as more positive for them personally:

"The feedback from the actual lecturers and tutors themselves, is that they come out of the session saying wow that was really good, I really enjoyed that, whereas before it was wow that was a struggle, I really had to push them along today, that sort of thing."

In short, the nature of the staff allocated to first-year courses needs to be carefully considered and in order to achieve radical change in teaching methods sometimes a change in the staff involved needs to be considered.

v. Summary

The issues raised by the interviewees in this theme were concerned with significant shifts in teaching practice, particularly with regard to the nature of lectures and practical labs. Specific teaching techniques included the use of visual programming strategies, collaborative and cooperative learning, and problem-based learning. Increasingly, the use of social networking is being investigated as a means to support the formation of learning communities. Finally, comments by interviewees highlight the importance of the careful selection of appropriate academic staff to undertake the delivery of first-year units.

3. Future Directions and Recommendations

A number of good teaching practices in Australian ICT courses were identified in this theme. These are all concerned with initiatives to increase students' interest and engagement in learning and to improve learning outcomes. Areas of future research are suggested by gaps in the literature and by issues raised by the interviewees. The literature in this area is overwhelmingly dominated by a focus on techniques to teach introductory programming courses. Research on other areas of the first-year ICT curriculum is needed. This report has documented a number of initiatives aimed at increasing ICT student engagement in the learning process. There is a clear need for more formal evaluations of the effects of these teaching initiatives in the Australian ICT context and to collate examples of good practice for wider dissemination. While initial results in many cases are positive, more evidence is required to justify sector-wide change. Finally, the nature of the staff allocated to first-year courses needs to be carefully considered.

Recommendation 4

Research into teaching in ICT is overwhelmingly dominated by a focus on techniques to teach introductory programming courses. Research is needed on other areas of the first-year ICT curriculum.

Recommendation 5

This report documents a number of initiatives to increase ICT student engagement in the learning process. There is a clear need for more formal evaluations of the effects of these teaching initiatives in the Australian ICT context and for the collation of examples of good practice for wider dissemination.

Recommendation 6

When allocating teaching responsibilities, careful consideration should always be given to the appropriateness of the staff allocated to first-year courses.

How we assess

Overview

The ‘How we assess’ theme focuses on assessment-related issues in first-year courses in Australian universities. The theme includes assessment strategies, techniques and tools. The tools may be instruments to assess students’ learning or to facilitate the assessment marking process. Different forms of summative and formative assessment are covered and are discussed in the contexts of provision of feedback, verification of student work, and issues associated with academic integrity.

To provide a context for this theme we first explore assessment processes, forms and tools, particularly in first-year ICT programs, as reported in the literature. Findings from the interviews of ICT academics are then presented to provide insights into assessment practices in the first year of Australian ICT courses.

1. Literature Perspectives

The systematic literature review found 38 papers that were concerned with the ‘How we assess’ theme. The literature in this theme was grouped into six topics:

- assessment design and strategies
- exam assessment
- non-exam assessment
- automated assessment
- assessment instruments
- academic integrity

All papers were set in the higher education sector. A high number of papers (27, 79%) dealt with assessment in first-year courses or assessment which was applicable to the first year. All but one were set in the ICT discipline. Most papers (33, 87%) dealt with issues concerning assessment of programming, and almost half (18, 47%) were Australian studies. See Table 6 for a list of the Australian papers for this theme.

The following review is organised according to the six topics that emerged from analysis of these papers.

Topic	Australian-focused references
Assessment design and strategies	Richards, D. (2009). Designing project-based courses with a focus on group formation and assessment.
	Thomas, R. N., Cordiner, M., & Corney, D. (2010). An adaptable framework for the teaching and assessment of software development across year levels.
Exam assessment	de Raadt, M. (2012). Student created cheat-sheets in examinations : impact on student outcomes.
	Gluga, R., Kay, J., Lister, R., Kleitman, S., & Lever, T. (2012). Coming to terms with Bloom : an online tutorial for teachers of programming fundamentals.
	Harland, J., D’Souza, D., & Hamilton, M. (2013). A comparative analysis of results on programming exams.
	Simon, Sheard, J., Carbone, A., Chinn, D., Laakso, M.-J., Clear, T., de Raadt, M., D’Souza, D., Lister, R., Philpott, A., Skene, J. & Warburton, G. (2012). Introductory programming : examining the exams.

	Sheard, J., Simon, Carbone, A, Chinn, D., Laakso, M-J , Clear, T., de Raadt, M., D'Souza, D., Harland, J., Lister, R., Philpott, A., & Warburton, G. (2011). Exploring programming assessment instruments: a classification scheme for examination questions.
	Sheard, J., Simon, Carbone, A., D'Souza, D., & Hamilton, M. (2013a). Assessment of programming : pedagogical foundations of exams.
	Sheard, J., Simon, Dermoudy, J., D'Souza, D., Hu, M., & Parsons, D. (2014). Benchmarking a set of exam questions for introductory programming.
	Shuhidan, S., Hamilton, M., & D'Souza, D. (2009). A taxonomic study of novice programming summative assessment.
	Shuhidan, S., Hamilton, M., & D'Souza, D. (2010). Instructor perspectives of multiple-choice questions in summative assessment for novice programmers.
	Simon, Sheard, J., Carbone, A., Chinn, D., Laakso, M.-J., Clear, T., de Raadt, M., D'Souza, D., Lister, R., Philpott, A., Skene, J. & Warburton, G. (2012). Introductory programming: examining the exams.
Non-exam assessment	Cain, A. & Woodward, C. J. (2013). Examining student reflections from a constructively aligned introductory programming unit.
	Gray, K., Thompson, C., Sheard, J., Clerehan, R., & Hamilton, M. (2010). Students as web 2.0 authors: implications for assessment design and conduct.
	Gray, K., Waycott, J., Clerehan, R., Hamilton, M., Richardson, J., Sheard, J., & Thompson, C. (2012). Worth it? Findings from a study of how academics assess students' web 2.0 activities.
	Terrell, J., Richardson, J., & Hamilton, M. (2011). Using web 2.0 to teach web 2.0: A case study in aligning teaching, learning and assessment with professional practice.
	Waycott, J., Sheard, J., Thompson, C., & Clerehan, R. (2013). Making students' work visible on the social web: a blessing or a curse?
Academic integrity	Nguyen, T. T. Le, Carbone, A., Sheard, J., & Schuhmacher, M. (2013a). Integrating source code plagiarism into a virtual learning environment: benefits for students and staff.

Table 6: 'How we assess' literature focused in the Australian context

i. Assessment design and strategies

A couple of papers were found that focused on assessment of first-year students in university courses in general. A review by Yorke (2011) of assessment and feedback practices in the first year of university highlights the importance of early and timely feedback and a pedagogy that encourages students to reflect on their learning. A comprehensive report by O'Neill and Noonan (2011) presents a series of resources to assist in designing assessment tasks. An underlying principle is to build first-year students' confidence with low-stakes assessment before moving progressively to high-stakes assessment. Staff are encouraged to contain the amount of assessment they build into their units to allow students time and opportunity for in-depth engagement with the teaching program. This strategy is based on the idea that to be successful in learning, students need to be engaged and empowered.

A number of papers deal specifically with assessment strategies in ICT courses. Taking a holistic view of the assessment process in programming courses, Australian researchers Thomas et al (2010) propose the 'teaching and assessment of software development' framework (TASD) and give examples of its use across multiple year levels. Barros (2010) discusses the importance of assessment strategies in introductory programming and proposes a set of techniques and criteria to consider when designing programming assessment and grading. For assignment work he incorporates a plagiarism detection tool and oral assessment, and for the final practical exam, a minimum acceptable grade. Both studies report positive results in terms of student satisfaction and higher grades.

A problematic area for assessment in ICT courses is group work. An Australian researcher (Richards, 2009) discusses ways of assessing group work (including peer assessment) and the challenges of providing a fair distribution of marks to each group member. Hahn et al (2009) investigated different forms of assessment for pair programming, and propose that a combination of self, peer, and facilitator assessment can increase the amount of feedback to the students, resulting in higher levels of achievement.

ii. Exam assessment

A final written exam is a common form of summative assessment in computing courses. A number of papers reported studies of exam assessment, and these were all in the context of introductory programming. Much of this work has been conducted by Australian researchers.

Petersen et al (2011) analysed 15 introductory programming exams to determine the types of question and the topics they covered. They concluded that some questions were too difficult for introductory programming students due to the high number of concepts students were required to understand in order to answer the questions.

A corpus of work led by Australian researchers has investigated the use of formal examinations for the summative assessment of programming. The initial phase of this research investigated the structure of programming exam instruments, including an in-depth study of the types of question used. This involved development of a scheme to classify programming questions on a number of dimensions including style, course content, skill required to answer, difficulty and complexity (Sheard et al, 2011). The classification scheme was applied to questions in 20 programming exam papers from multiple institutions (Simon et al, 2012). The study found that introductory programming examinations vary greatly in the coverage of topics, question styles, skill required to answer questions, and the level of difficulty of questions. Harland et al (2013) used the same classification scheme to further explore question difficulty. The next phase extended this work to design a set of questions suitable for benchmarking in introductory programming courses (Sheard et al, 2014).

Another aspect of this work was an investigation of the pedagogical intentions of the educators who construct exam instruments (Sheard et al, 2013a). This involved interviews with programming teachers to gain an understanding of how they go about the process of writing an exam, the design decisions they make, and the pedagogical foundations for these decisions. The study found that the process of setting exams relied largely on intuition and experience rather than explicit learning theories or models. Exam formats are typically recycled, and questions are often reused. While there is variation in the approaches taken to writing exams, all of the academics take a fairly standard approach to preparing their students for the exam. Although some academics consider that written exams are not the best way to assess students, most tend to trust in the validity of their exams for summative assessment.

Another group of Australian researchers investigated summative assessment of introductory programming, focusing on the use of multiple-choice questions in exams. Most instructors in their study considered multiple-choice questions appropriate for testing questions on the lowest levels of the Bloom taxonomy, but less than half were confident that multiple-choice questions could be used to test understanding of programming concepts (Shuhidan et al, 2009; 2010). A problem faced in the investigation of exam questions is the difficulty in applying Bloom's taxonomy to classify exam questions according to their cognitive level. An Australian research team has developed an online tutorial to train researchers in the use of this taxonomy (Gluga et al, 2012).

Another Australian researcher (de Raadt, 2012), investigated the use of 'cheat sheets' in introductory programming exams and found that students who took hand-written notes into their exam performed better than students who did not have notes.

iii. Non-exam assessment

Research studies on forms of assessment other than exam assessment focused mainly on assessment of programming. Studies of summative and formative were found and some reporting innovative practices.

A common form of in-semester assessment is the programming assignment. A grounded theory study by Kinnunen and Simon (2010; 2012) explored introductory programming students' experience of their assignments, and found that students' self-efficacy is not necessarily related to their experiences of success in programming.

A novel approach by Lee et al (2013) embedded assessment into an educational computer game designed to teach programming. A study of students' use of this game showed that incorporating assessment increased students' use of the game, the levels they achieved, and the speed at which they played the game.

Assessment by portfolio assessment is rather less common than asignments. Australian researchers Cain and Woodward (2012) describe an introductory programming unit where students are assessed entirely on a portfolio of work produced during the semester. The design of the unit was founded on Biggs's constructive alignment, which proposes alignment between the learning activities, assessment, and intended learning outcomes. An evaluation showed that students were positive about their learning experience. Pears (2010) reports on the use of portfolio assessment in an introductory programming unit for the purpose of implementing a continuous assessment model. He found that students who completed the unit produced code of a higher quality than typically produced by first-year students.

Peer review is a form of assessment used for both formative and summative assessment. Assessing the work of peers can encourage student engagement and deeper learning (Carter et al, 2011). Peerwise, a collaborative web-based tool, enables students to create and share multiple-choice questions and allows students to peer-review questions submitted by others. Evaluation of the use of Peerwise has shown that it can foster student engagement and have a positive impact on learning (Denny et al, 2010; Purchase et al, 2010).

The use of social media (web 2.0) in education has led to new forms of assessment where students demonstrate their learning through online tasks which are often co-created and visible to their peers, and, in some cases, to wider audiences. These new forms have brought challenges for students and teachers in using unfamiliar authoring tools and applying appropriate citation and referencing to their work. Studies by Australian researchers Gray et al (2010) investigated examples of assessment using different web authoring tools and showed how principles of good assessment practice were reflected in each case. Further studies investigated the affordances of web 2.0 technologies for assessment, along with issues of ownership, privacy, and visibility of work (Gray et al, 2012; Waycott et al, 2013). A case study by Terrell et al (2011) describes assessment of a web 2.0 task in an introductory information management course under the framework of constructive alignment.

iv. Automated assessment

The time-consuming tasks of collecting, marking, and giving feedback to students on their assessment work has led to the development of tools to help manage these processes. All of the assessment tools that we found were specifically designed for use in introductory programming classes.

- Law et al (2010) present PASS – Programing Assignment aSsessment System. PASS provides feedback for programming assignments by executing a set of instructor-prepared test cases and then comparing the expected output with the actual output. PASS also allows the teachers to monitor the testing process of students' submissions in real time and to share with the entire class examples that demonstrate good practice. A study of PASS showed a positive impact on students' self-efficacy.
- Wang et al (2011) discuss the role of automatic assessment in introductory programming and present a tool, AutoLEP, for automatic analysis and assessment of student programs. They describe their use of this tool for in-semester formative assessment and for end-of-semester exams. Students and staff were enthusiastic about the tool, with staff reporting that students showed increased interest in programming and improvement of their skills.
- Llana et al (2012) present an online free laboratory of programming (FLOP), which hosts a repository of programming problems that students can attempt and have automatically assessed. Preliminary results indicate positive improvement in students' motivation, skills, and self-efficacy.
- Johnson (2012) presents a tool, SpecCheck, for testing conformance of programs to the assignment specification prior to submission. A small study showed that students were willing to accept having to produce highly structured homework in return for faster grades and feedback.
- Shaffer and Rossen (2013) present the Programming Learning Evaluation and Assessment System for Education (PLEASE), a code-checking and submission system. Using data collected from the system, the lecturers were able to identify parts of the course where students were experiencing difficulties and make adjustments to the teaching program. The results of a small study indicated that the tool was useful in optimising course structure.

v. Assessment instruments

A few studies report the development of specialised assessment instruments. Ford and Venema (2010) trialed the use of short objective tests to test students' knowledge of fundamental programming concepts after their introductory programming course. Gouws et al (2013) designed a test to determine students' computational thinking ability prior to entering their computer science course. Elliott Tew and Guzdial (2010) propose a method for developing a language-independent assessment instrument for introductory programming.

vi. Academic integrity

The apparent prevalence of plagiarism and collusion is a topic of concern in the assessment of introductory programming. Australian researchers Nguyen et al (2013a) present a source code similarity reporting tool developed as a Moodle plugin. Studies of staff and student reaction to the tool showed its usefulness in deterring and detecting plagiarism and its potential as an educative tool.

vii. Summary

The literature on assessment in first-year ICT courses relates predominantly to programming. Nearly half of the papers found were from the Australian context, indicating a research strength in this area. Important issues included assessment design and strategies, exams, and other forms of assessment. Also of note are the areas of automated assessment, assessment instruments, and academic integrity. With the trend of an increased reliance by students on online course materials, methods to improve the automation of assessment and provide quality feedback on students' work, while maintaining the academic integrity of the assessment process, may require further research.

2. Current Practice in Australia

The interview questions related to the theme of 'How we assess' sought information about assessment strategies, forms of assessment, feedback, automated assessment, and academic integrity in first-year ICT courses. The responses gave insights into current assessment practices and issues faced by teaching staff. The responses to these questions are discussed under the main topics that were identified from the analysis of the interview data.

i. Assessment design and strategies

Students in first-year ICT courses are typically assessed via an end-of-semester written examination following in-semester tasks that may include assignments, portfolios, tests, tutorial exercises or presentations. The most common assessment models used are assignment work and a final exam combined with either a mid-semester test or tutorial assessment.

A couple of interviewees mentioned their university having an overall assessment strategy. Interviewee U8 commented that at her university, "*assessment revolves around problem solving – looking at authentic situations*". An assessment guide based on Biggs's theory of constructive alignment had been developed at one university. Constructive alignment was also mentioned as a theoretical basis of portfolio assessment at another university.

A number of interviewees had designed assessment strategies to address the issue of lack of student engagement. Interviewee U7a explains:

> *"Previously, I have implemented some unit rules for encouraging student engagement. For example, the tutorial attendance is no lower than 85%. That will be recorded. Secondly, students' tutorial attendance is marked and also we have some in-class quizzes."*

Most interviewees mentioned assessment policies at their university. It is common practice to set thresholds for exams that students must reach in order to pass a unit. Most often this threshold 50%, but 40% is also used. Several interviewees mentioned mandated percentages of supervised work. In order to avoid over-assessment, some universities limit the number of assessment tasks per semester. In a couple of cases, a maximum of four assessment items was allowed; and in another case two major assignments and an exam were recommended. At one university it was a policy to provide feedback on an assessment task within 2 weeks, and to have an assessment task within the first 5-6 weeks of the semester in order to give early feedback to students.

ii. Forms of assessment

An end-of-semester written exam is the typical form of summative assessment in first-year ICT courses. Exams are seen as necessary to verify that it is the student's own work that is being assessed; however, some interviewees expressed concerns that a written exam is not necessarily a good method for establishing what the students have learned. One interviewee mentioned a move away from exams at her institution but not for first-year courses. Most exams are weighted between 40% and 60% of the overall mark for a unit, with 50% the most common weighting. The lowest weighting was 20% and the highest was 70% of the overall mark.

In combination with an end-of-semester exam there are a variety of other forms of summative assessment. The most common is assignment work, done individually or sometimes in a group. Often more than one assignment is set during the semester. Some interviewees mentioned checkpoints for assignments where students must show their tutor their progress. Checkpoints are incorporated to encourage students to start work early and to give them feedback. However, they are also used to monitor their work, which can help determine whether the work finally submitted has been done by the student.

Tests held during semester are a common form of assessment. These may be mid-semester tests worth from 10% to 20% or a series of smaller tests often conducted online using the LMS or another tool, such as ViLLE. Some interviewees stated a preference for continuous assessment with smaller tests rather than one larger test. One interviewee commented that he does not hold a mid-semester test as the semester is only 11 weeks long.

Another common form of assessment is tutorial work. This involves assessment of tasks performed in the tutorial, often on a weekly or fortnightly basis. Typically this is low-stakes assessment with a few marks (1-2%) allocated for each assessment. Interviewees mentioned that assessment in tutorials is a strategy for encouraging students to come to class and to work in class. An additional benefit was that tutors could observe students working and alert them to possible cases of plagiarism. However, interviewee U18, while acknowledging the benefits of lab assessment, found that it was "*more trouble than it was worth*".

Some universities use portfolio assessment. At one university portfolio assessment is embedded into each year level, and students are given training in their first year to help them understand the expectations of this form of assessment.

At another university portfolio assessment has been used for the past 5 years in an introductory programming unit (Good Practice Example 10). The portfolio assessment has been designed using Biggs's constructive alignment. Interviewee U1 explains:

> *"This has been one of the changes that I think had a big impact as well on the pass rates for the introductory programming unit, … a large change, moving away from assignments and exams to submitting a portfolio of assessments".*

Interviewee U1 describes the process:

> *"Each week the student will develop pieces of work that demonstrate how they've met one or all of the unit learning outcomes and each week we have a formative feedback process. With the portfolio assessment it has weekly feedback. It's 100% portfolio assessed so they don't get a grade until the end of the semester".*

Interviewee U1 goes on to explain the grading process at the end of semester:

> *"each student has to submit a portfolio that demonstrates how they have met all of the unit learning outcomes. Then there is a scale by which they can meet [the learning objectives]. To meet them to an adequate level there are criteria. To meet them to a credit level there are separate criteria, and so on for distinction and high distinction. This allows students to work to their expectations. Some students only want to pass the unit and they're not interested in doing really well and so they'll make sure that they've met all of the pass criteria. They don't*

have to worry about meeting the distinction or high distinction criteria. That's not what their goal is in life".

At this university the portfolio assessment was a big change in the way the introductory programming is taught and students are assessed:

> *"Each week the students submit work to get feedback so that they can improve that work and thereby improve their understanding. There's no punishment for doing that. Previously if students did an assessment at the beginning of semester and did poorly they lost those marks and they can never get them back. That doesn't encourage them to want to learn about what they did wrong, they start focusing on what they've got to do for next week so that they don't lose more marks."*
>
> *"With this what we can do is go back and really focus on those very first things they didn't understand and make sure they understand those before they move on to the next thing. Some people might take a few weeks to get through the first few tasks they have to complete whereas others might get them done very quickly."*

Other less common forms of assessment mentioned were presentations and submitted homework tasks; one interviewee gave students a mark if they visited the lecturer to ask a question.

There were indications of a growing use of social media for assessment tasks. For example, interviewee U7a allowed students to use social media to deliver an e-learning information resource that they developed as an assignment task. Interviewee U24 discussed how he uses blogs and UCROO, an educational social-networking site based on Facebook. However, another interviewee raised a concern related to plagiarism when using social media: *"We've told them not to talk about the assignment but it's hard to police so I discourage it because of the plagiarism issue"*.

iii. Feedback

The comments by interviewees indicate that feedback is an important part of the assessment process. At most institutions feedback is given on all forms of in-semester assessment. Formative feedback on assignments is often given verbally during tutorials or consultation times. Portfolio assessment allows for continuous formative feedback throughout the semester. Feedback on summative assessment is typically given verbally for tutorial tasks and is written on assignment work. In the case of class tests, feedback is usually just a score.

A number of interviewees described detailed critiques for summative assessment of assignment work involving comments and scores for individual components. Assignment work is often assessed using rubrics. A couple of interviewees stated that they give feedback on assignment work as a summary at a lecture. In one case feedback on assignment work is given only in this open forum; however, students are also given the opportunity to discuss their work individually with their lecturer.

Some interviewees mentioned particular approaches to giving feedback for assignments submitted online. The GradeMark tool from Turnitin was mentioned by some as facilitating provision of feedback through dragging and dropping of comments. Interviewee U9 details a university-wide policy of e-assessment (Good Practice Example 11):

> *"all student work must be submitted online and returned online, and that was trialed last year and has gone live this year. So we have been embedding feedback in online assessment."*

All assignment submission times are recorded and therefore the timeliness of the feedback provided to students is also recorded. A permanent record of all feedback is also stored, in case an issue arises. This university-mandated policy has the potential effect of allowing an audit of the quality and promptness of the feedback provided to all students in every course. Therefore a systematic process may be implemented to improve the standard and responsiveness of the feedback delivered to students.

Some assessment tasks enable instant feedback on performance. Examples are online quizzes and programming assignments with automated assessment. One interviewee commented that the instant feedback was very popular with the students.

The only feedback on exams is through viewing the exam script. Most interviewees indicated that very few students do this. Interviewee U16 stated that at his university comments are written on the exam scripts with the expectation that at least some students will come and look at them.

iv. Automated assessment

Automated assessment is not used to any great extent in most universities. The most common use is for quizzes and multiple-choice question components of tests and exams. There were some examples of automatic testing of programming assignments. Interviewee U18 said that automatic assessment was used for: *"80% of the marks – none of it is automatic, but all of it has automated support."*

The use of multiple-choice assessment varies, and appears to be controversial. One interviewee sets a mid-semester test and most of the exam as multiple-choice questions due to a large enrolment (250 students). Another uses multiple-choice questions in exams but says that more than 50% of assessment using multiple-choice questions would be frowned upon at his university. Interviewee U17 sets an exam of multiple-choice questions, arguing that: "*the only other option I can think of is to have programming problems on the exam paper but the exam is not the place where you can do any thinking*".

v. Academic integrity

Verification of work

In trying to determine whether a submitted assessment task is the work of the student submitting it, the interviewees use a range of strategies including interviewing, monitoring and observing.

Most agreed that interviewing students about their submitted assignment work was an effective way of verifying that the work was their own and identifying possible cases of plagiarism or collusion. A couple of interviewees described thorough interview processes (Good Practice Example 12). For example, interviewee U18 commented "*At the interview they are expected to discuss the code they've written and make changes to it*". Interviewee U15b proposed that an interview does not have to be long to be effective:

> *"You can [ask] just a few pointed questions about their motivation for the design they made, why they did it that way, and you can start to poke them a bit and say 'if we change this what would happen?'; 'if you wanted to do this feature how would you do it?'. I've used the interview and they tend to be pretty good at picking up where it might not be all the student's own work".*

Despite its acknowledged effectiveness, interviewing every student as part of the assessment process is used in only a few institutions, typically in programming units. Many interviewees have too many students and too few resources to conduct interviews. Interviewing had recently been abandoned at a couple of universities. As interviewee U16 explained, interviewing was "*extremely effective but very time-consuming, so we just couldn't keep it up*." A number of interviewees said that they interviewed students only if they were suspicious of the work. Interviewee U12 said that interviews are not used in her university because the previous Head of School was concerned that "*it could mean asking different questions of different students and could cause issues*".

Sometimes there are opportunities for less formal verification approaches where students can be questioned in their tutorials during the formative stages of an assignment. Some interviewees are alerted to possible cases of plagiarism through monitoring students' work and observing patterns of participation. Interviewee U24 incorporates a tutorial participation mark as part of the assignment mark, stating that "*it's actually a way of encouraging students to work every week and it's also a way of controlling plagiarism*".

Tools are sometimes used in verification of student work. The plagiarism detection tool Turnitin is frequently used for text-based assignments; however, the use of plagiarism detection tools for programming assignments appears less common. Tools such as MOSS (Measure of Software Similarity), JPlag, and ESP were mentioned for detection of code plagiarism; however, one interviewee suggested that plagiarism detection tools were not suitable for first-year programming as there is usually too much similar code. Interviewee U2 only follows up on obvious plagiarism, seeing the assignments "*as learning opportunities as much as assessment*".

However, plagiarism detection tools are not useful in detecting cases where students have commissioned their assignment work. Some interviewees rely on the assignment markers noticing disparities either within the submitted work or between the submitted work and the student's normal work. As interviewee U6 explained:

> *"you get a pretty good eye for it once you've marked a few things and you know the standard or the hallmark of the student's work and if something significantly deviates from that you can start looking into that. I'll always keep an eye out for phrases or chunks of text that look like they've been written in a different style".*

However, this becomes more difficult in large classes with multiple markers and does not always cover the cases where someone else has done the work. A couple of interviewees mentioned that they had found their assignments advertised on a code purchasing site. A strategy used by interviewee U22 is to give each assignment with a unique name to make it easy to do a Google search to find any plagiarised code. Another interviewee mentioned a network of universities that monitored code purchasing sites to pick up on cases where assignments had been commissioned.

Discouraging cheating

A number of strategies were used to discourage cheating. All universities had invigilated assessment in at least the exam component. As interviewee U20 noted, "*the only thing you can absolutely guarantee are the moderated parts, which are the exams*". In a number of universities, students were required to gain a minimum exam mark, typically 40% or 50%, to pass a unit. A couple of interviewees commented that they used exams to pick up on students who had not done their own assignment work. However, Interviewee U4 noted that his university has a policy that "*exams are not to be for the purpose of ensuring that people haven't cheated*".

Interviewees suggested a number of strategies to encourage students to do their assignment work. These were seen as preferable to punitive approaches. Some stress to their students that writing code on their own will help them with their exam. One interviewee uses careful assessment design where assignments are not just taken from the textbook; a couple of others set assignments tailored to individual students, allowing students to negotiate their own assignment. There was no consensus about whether students should work individually or with others on their assignments. Interviewee U19 requires students to work their assignments in pairs as he considered that "*this makes it* *much less likely that they will seek outside help*"; whereas at another university all first-year assignments are individual.

Two interviewees explained how they use email messages to discourage plagiarism, either sent from the lecturer …

> *"I would send an email to students normally around that the time the assignment is due because I think most plagiarism occurs when students get behind and the assignment is due and they quickly find a friend to copy from. I tell them that if they have fallen behind to ask me, not their mate." (U13)*

… or sent from the Head of the School every semester:

> *"…every semester the HOS sends an email to all students saying there were X number of students found guilty of plagiarism this semester and you should all be taking this seriously. So he also gives feedback to students about what students have been caught plagiarising to show them that we're actually catching them and doing something about it." (U17)*

Two interviewees also mentioned how Turnitin is used to discourage plagiarism through detection. Interviewee U25 mentioned: *"We advise the students that their assignments would be put through Turnitin."* and interviewee U5 mentioned: *"They're all very well aware of Turnitin because when they put their assignment in they get a report back."*

Penalties for breaches of academic integrity

Every university has a standard procedure to deal with academic breaches. Most universities have a designated officer to ensure that standard penalties are imposed across the school, the faculty, or the university. Substantial breaches are dealt with at the higher levels of management outside the particular school. For example, a Dean's review was required to deal with substantial breaches in one university. Many universities maintain details of academic breaches in a central register or in the individual student's file.

The penalties imposed depend on the severity of the breach, the weighting of the assignment, and whether it is a repeat offence. Penalties range from zero marks for the specific assessment, to failing the unit, all the way through to being excluded from the university. Interviewee U23 said that for repeat offenders "*it could go all the way to a student having their enrolment terminated which would be a very rare thing but it has happened in the past*".

Interviewee U12 discussed the importance of understanding the overall situation when an academic breach occurs:

> *"However, it's not just 'OK, you've plagiarised, you're going to get this penalty'. It's looking at the circumstances around it and what's happened; whether they've understood what plagiarism is. And whether they've acknowledged what's happened. "*

When asked what would happen to a student who had copied something from the Internet and it was their first offence, interviewee U9 explained:

> *"They would be educated and make sure that they do the quiz [students are expected to complete an academic integrity quiz which is 5% of their overall grade]. They would be told about proper paraphrasing and citing sources etc."*

Educating students about academic integrity is discussed in the 'How we strengthen the learning environment' chapter.

vi. Summary

In contrast to the shifting landscape found in the 'How we teach' theme, most interviewees discussed traditional forms of assessment. The few innovative assessment practices found were designed to encourage attendance (e.g. tutorial assessment), engage students in learning activities (e.g. social media), or encourage good work habits (e.g. portfolios). Interviewees' comments indicated the high importance they place on giving feedback on work during semester. Finally, academic misconduct is a problematic area and there are a range of techniques used to verify students' work and discourage plagiarism and collusion.

3. Future Directions and Recommendations

The good practices in assessment identified in Australian ICT courses are concerned with portfolio assessment, interviewing students to verify work, and using appropriate tools to facilitate and expedite provision of feedback. A number of areas identified within this theme warrant further investigation. Overwhelmingly, the context for research and discussion in assessment was in the context of programming. There were a variety of techniques and tools for assessment of programming, but very few in other areas of study. Research on assessment techniques for other areas of the first-year ICT curriculum might be appropriate. The recent adoption of social media has led to innovative forms of assessment and there were reports of its use in a number of universities; however, few studies were found of the use of this assessment form in first-year ICT courses. This is an area that could be further investigated. A key issue raised by interviewees was that the trend for increased online delivery had placed demands on academics to create appropriate assessment tasks for this context and to verify the identity of the student undertaking the assessment. There is a clear need for work in this area. Related to this, there was a perceived need for more tools to automate assessment for large groups and to facilitate feedback. We propose that these issues require further research in order to ensure valid and fair assessment for our first-year students.

Recommendation 7

We found a variety of techniques and tools for assessment of programming but very few in other areas of ICT study. Research is needed on assessment techniques for other areas of the first-year ICT curriculum.

Recommendation 8

The recent adoption of social media has led to innovative forms of assessment; however, there were few studies found of the use of such forms of assessment in first-year ICT courses. This is an area that should be further investigated.

Recommendation 9

The trend for increased online delivery has placed demands on academics to create appropriate assessment tasks for this context and to verify the identity of the student undertaking the assessment. There is a clear need for work in this area.

Recommendation 10

There is a need for more tools to automate assessment for large groups and to facilitate provision of feedback to students.

How we strengthen the learning environment – learning support

Overview

The 'How we strengthen our learning environment' theme focuses on approaches to help first-year students in ICT courses in Australian universities become effective learners. This includes methods to facilitate the development of learning communities, along with programs to assist students with study skills, teamwork, language and communication skills, and to educate students about academic integrity.

1. Literature Perspectives

The systematic literature review found 15 papers that were concerned with the 'How we strengthen the learning environment' theme. The literature in this theme was grouped into four topics:

- learning communities
- teamwork skills
- study skills
- academic integrity

All papers were set in the Higher Education sector and in the ICT discipline. Most were concerned with teaching in first-year courses. Ten papers dealt with the teaching of programming and introductory programming in particular. Eight were Australian studies. See Table 7 for a list of the Australian papers for this theme. The following review is organised according to the key sub-themes that emerged from analysis of these papers.

Topic	Australian-focused references
Learning communities	Jenkins, G., Lyons, K., Bridgstock, R., & Carr, L. (2012). Like our page – using Facebook to support first year students in their transition to higher education.
	Maleko, M., Hamilton, M., & D'Souza, D. (2012). Novices' perceptions and experiences of a mobile social learning environment for learning of programming.
Teamwork skills	Egea, K., Kim, S., Andrews, T., & Behrens, K. (2010). Approaches used by cross-cultural and cross-discipline students in teamwork for a first-year course in web design.
	Richards, D. (2009). Designing Project-based Courses with a Focus on Group Formation and Assessment.
Study skills	Sheard, J., Carbone, A., Chinn, D., & Laakso, M.-J. (2013b). Study habits of CS1 students: what do they say they do?
	Teague, D. (2009). A people-first approach to programming.
	Carbone, A. Hurst, J. Mitchell, I., & Gunstone, D. (2009). An exploration of internal factors influencing student learning of programming
Academic integrity	Nguyen, T. T. L, Carbone, A., Sheard, J., Schuhmacher, M., de Raadt, M., & Johnson, C. (2013b). Educating computer programming students about plagiarism through use of a code similarity detection tool.

Table 7: 'Learning support' literature focused in the Australian context

i. *Learning communities*

A prominent theme in the literature is the potential benefit of learning communities for supporting students in their learning. Most work in this area is focused on online communities that can form through the use of platforms such as social media, discussion forums, blogs and wikis. The use of social media in teaching is also discussed in 'How we teach'.

Several studies were found of the use of social networking sites to support learning communities. Two examples of Australian studies are Jenkins et al (2012) and Maleko et al (2012). The main proposition of these papers is that social networking sites such as Facebook can provide greater support for creating and maintaining social networks than the tools built into current learning management systems such as Blackboard. A number of key aspects include the familiarity of students with the Facebook interface, the 'always on' nature of the Facebook application, and the accessibility of the application on mobile devices.

Online communities may be formed within a single student cohort or across multiple cohorts. Smet et al (2010) report a peer support scheme in which senior students are used as online peer-tutors to facilitate online collaboration and knowledge construction among first-year students. The study concluded that this approach was valuable for supporting students' personal development and for improving online tutoring behavior. Another study found that Facebook could be used to strengthen the communication between first-year and senior students, thus helping to establish a stronger community (Smith et al, 2012).

ii. *Teamwork skills*

The ability to work in a team is an essential skill for ICT students and for their future employment, but teamwork is often a problematic area for students. A couple of papers reported approaches to assisting students develop their teamwork skills. An Australian study by Egea et al (2010) evaluates an intervention designed to increase students' awareness of cross-cultural and cross-disciplinary differences in teams within a first-year undergraduate web-design course. Findings indicated that raising awareness on cross-disciplinary and cross-cultural differences and having strategies to address these differences has a positive effect on group dynamics.

A literature survey by Richards (2009), focusing on the key design choices of project-based courses, concludes that it is vital for the lecturer to stay in touch with groups in order to identify issues and deal with them expediently. The survey also finds that a key factor affecting students' experience of the project-based courses is whether conflicts among group members can be resolved.

iii. *Study skills*

Assistance with study skills can be a valuable way to support students in their learning. Several studies have investigated student study habits and learning behavior in order to develop teaching approaches and strategies to help students with their learning. All the studies found were in the context of programming.

In order to understand the impact of student behaviors on the outcome of the programming assignments, Edwards et al (2009) analysed data from the first three programming courses over a period of five years. They found that students who start early and finish early on assignments receive higher grades that those who start and finish later. Furthermore, they do not appear to spend more time on the assignments. Edwards et al (2009) suggest that starting earlier allows students more access to help and they have more time to think of alternative solutions without having the pressure to finish on the day.

An Australian study (Sheard et al, 2013b) explored the study habits of introductory program students in order to understand the motivating factors for their behaviour. A key finding of the study was that students are turning away from traditional teacher-provided and teacher-directed resources such as text books and moving towards online resources, relying heavily on the internet to get help for their studies. Another study (Zander et al, 2009) focused on understanding students' preferred learning styles to learn mathematics and introductory programming. According to students' responses, a reflective style is used in learning mathematics, whereas an active style is used in learning programming. In addition, students appear to regard learning programming as mainly visual while mathematics has a strong verbal component.

A couple of Australian studies have explored affective issues including motivation, determination and confidence within the context of learning introductory programming. Teague (2009) argues that students' final grades do not relate strongly to their confidence, motivation, determination, or ability. She proposes that educators need to take a holistic view and explore beyond the assessment grades in order to understand the barriers that affect the learning of programming. The study reports that both confidence and determination are likely to be key factors contributing to a successful learning experience, but that confidence alone is unlikely to be a reliable predictor of a student's success. Another Australian study (Carbone et al, 2009) explored the internal factors that influence first-year ICT students' learning of programming. One finding was that the lack of skills to complete the task at hand had a negative impact on motivation whereas the presence of skills had a positive impact. On the other hand, lack of motivation affects one's ability to persevere to obtain the skills.

Egan et al (2011) report on a workshop that provided a student-focused proactive intervention called Academic Enhancement Program (AEP). The workshop aimed to help students develop academic self-regulation skills. Analysis of student responses shows that students appreciated the assistance provided. Furthermore, students reported changing their academic strategies based on the knowledge gained from the program.

iv. Academic integrity

The issue of academic integrity was the focus of a couple of studies. One study (Yang et al, 2014) analysed the reasons behind code similarity to promote software reuse and avoid plagiarism. They required students to discuss the advantages and disadvantages of each code-similarity aspect (class structure, class relation, class content, and logic flow) that they identified in assignment solutions. The study reports that the majority of students achieved the learning outcome of this activity. An Australian study (Nguyen et al, 2013b) evaluated the effectiveness of a plugin that integrated two well-known plagiarism detectors, JPlag and MOSS, into a learning management system, Moodle. The academics who use the plugin consider that it will assist in detecting and discouraging plagiarism. Both academics and students thought that making the code similarity report public would help increase the awareness of plagiarism, but there were some concerns about negative effects on student anxiety.

2. Current Practice in Australia

The interviewee responses to questions related to 'How we strengthen the learning environment' gave specific information about their use of social media to develop learning communities and about strategies to help educate students about study skills and academic integrity. The responses to these questions are discussed under the main topics that were identified from the analysis.

i. Social media and learning communities

Many interviewees discussed their use of forums to facilitate communication within their units, and a few mentioned social media for this purpose. Social media were also seen as a tool to develop and support learning communities. The most frequently mentioned social media forms were social networking sites such as Facebook, but wikis, blogs, Twitter, and discussion forums were also mentioned.

Interviewee U11b discussed his plans for using social media:

> *"I would like to get much more into it, providing students with questions by Twitter or having a course-based social media site. I encourage students to have their own social media site to talk to each other without the course leader's involvement. It's better to encourage them than to just have them do it themselves anyway which tends to happen at the moment".*

Interviewee U15b explained how he differentiated between different social media forms: *"Twitter more for the instant feedback, instant connection. Facebook is more for building a bit of a community".* In 'How we teach' there is a detailed description of the use of discussion forums, blogs, and UCROO, an educational social networking site based on Facebook, and of their benefits for connecting on-campus and off-campus students and developing learning communities.

Interviewee U9 mentioned how students are encouraged to use LinkedIn to *"be professional from the start. So there's collegiality with their lecturers and higher grade students in LinkedIn as well".*

In addition to these approaches a few lecturers discussed how students themselves, independently of the lecturers, have used Facebook groups to undertake their group work collaboratively. Finally, some responses focused on using social media to increase social support via social events.

ii. Study Skills

A variety of approaches are used to help first-year students with study skills. Interviewees described programs and materials developed at their institutions to help students with skills including researching material, time management, exam preparation, and communication skills.

Many universities offer study skills programs run by the library or the central learning and teaching unit. These may be information sessions, workshops, or individual consultations. In some universities these programs are compulsory. When they are not compulsory, lecturers are usually expected to identify students in need and direct them to these programs. Study skills programs are normally offered during orientation week, early in first semester, or throughout the first year of a course.

At one university an innovative study skills program called MeetUp involves workshops and incorporates a peer mentoring scheme (Good Practice Example 13). Interviewee U11a explains how the MeetUp program works:

> *"the majority of education about study skills is done by MeetUp leaders … we think students respond better to someone around their own age [explaining] what they have found works and what to do and what not to do, rather than us lecturing them. The MeetUp leaders play a fundamental role in this type of education".*

Interviewee U11b further explains:

"the workshops that we run for the on campus students … go for three hours and we have a MeetUp session for half an hour each week [with] higher level students who have been trained to deliver information to students on time management, essay writing, and study skills. These are all recorded and put up on the study desk for the external students … MeetUp leaders have a lot of training and weekly meetings to talk about issues. ... they have support from our learning management team who provide them with encouragement and help."

Interviewee U10 described a peer mentoring approach at her university:

"we hook all our students up with a student mentor in the very first week of university, the peer mentor will meet with them a few times in the first few weeks and also email them, so they also tell them about the program."

A common approach to teaching study skills is to integrate study skills development into the courses. This is often done with assistance from a central teaching unit. As interviewee U8 explained:

"in some units we bring the study skills people to the class where it's appropriate in the curriculum. For example … in the first year, library [staff] will come and do a presentation about how to use the library."

Interviewee U5 described similar assistance:

"We're working with the library to develop online resources in Moodle, with the lecturers drawing resources from there and building them into their units. They could have information and activities for practical classes and online quizzes for students to do as well."

Some interviewees emphasised a need to integrate study skills throughout a degree program. For example, interviewee U9 explained:

"we scaffold their learning so that [the soft skills] would appear in first year and there is also something in second year and then there are capstone topics and they are specifically bringing in those soft skills. Particularly we've looked at the need to be able to express themselves adequately in writing and in sentences and orally, and what they're doing, and talk about it."

Interviewee U15b described his attempt to assess study skills in an introductory programming course:

"The T&L people came in and gave a bit of an overview of formal writing for design … and then it was made part of the assessment item. So the assignment was the solution to the problem and also a document that described motivation, etc. There's also oral communication stuff that will be done in a similar way. Research is a tough one to deal with for first year and we'll probably do that from an informal perspective – not looking up journals or things like that – but how you go and find out more about it, how you know the information is trustworthy. We're trying to build those sorts of things in to teaching activities and the assessment".

Interviewee U20 described how assessment of study skills is built into the early assessments of the first year at his university:

"All units have early assessment tasks … particularly in the first first-year subject, they're the type of thing that will ease people in. We'll graduate the complexity and the way that the early assessments are structured so that the early ones are fairly easy but illustrate the sorts of work practices that they should be doing".

In order to improve essay-writing skills, one university introduced an English competency rubric that is expected to be applied for every written assignment. This rubric also provides basic feedback on sentence structure and word usage to students.

The staff member noted that the feedback should prompt students, if required, to seek the workshops on study skills.

Time management was mentioned by a number of interviewees as an important skill for students to develop. Interviewee U7b described her approach to helping students with time management:

"I try and encourage them to think of university as their first major job and we do 4 units a semester, so if you do one unit a day you're working 4 days a week and that still gives you three days to party. Encourage them to consider it to be an 8 hour a week deal. … I'm seeing some students addressing that but we have many who still work too long hours and expect to be able to fit everything in and still have a social life and that doesn't work."

Interviewee U25 explained his innovative approach of using flowcharts to help students with their scheduling of tasks (Good Practice Example 14):

"We give the student a flowchart of how to utilise their week. Say for example if a student comes to the class without preparation it is very difficult for them to grab the information. So we give them detailed information of what to do – you need to study this, you need to read this example before you come to the class. Now attend the class and what they need to do next is solve the questions. They need to go to the modules of the interact site and solve a particular part. Then they have to attend the online meeting and if they have questions or didn't understand anything in the lecture they can ask the question online. They can also try some sample online quizzes to try which are not marked but students can try them and then they have to do the online quiz which is marked. This is a strategy of how to cover a topic. The flowchart provides guidance on the sequence and time required to prepare for and complete work. Why I did it was because a student sent me an email asking how to manage the subject because he said there are many things to do but he wasn't sure which I need to do first and then what. So I sent the student the flowchart and then when I was preparing the subject outline the next session I decided to include the flowchart and give it to every student."

iii. Academic integrity

Interviewees discussed their approaches to educating students about academic integrity. Most mentioned links to university plagiarism policies in course websites and in documents such as unit outlines and assignment specifications. Often, when submitting an assignment, either physically or electronically, students are reminded about plagiarism and are required to submit a declaration stating that the work is their own. One interviewee mentioned that they expect students to agree to terms and conditions related to the academic policy:

"In our course, in the first introductory lecture we talk about it and then whenever they go to login to the environment, first they had to read about this academic honesty policy and agree to those terms and conditions" (U21).

In some universities academic integrity is taught as a part of a centralised unit, while in others it is covered within individual units, especially first-year units.

"[academic integrity] is something that I discuss with them because the unit that I do is generally one of the first units they are likely to do. I talk about some of the specific traps that people could fall into; working together or ending up with someone's code and submitting that as your own and that sort of thing. So I discuss it specifically in the lecture as to what can happen and how they can protect themselves against getting into that situation." (U3a)

A particular concern was helping first-year students distinguish between acceptable collaboration and collusion:

"Often in first year, because they are working together, it's a case that we have to have a chat with them about collaboration and when it's appropriate and when it's not. It comes up at those points. We tell them it's OK to work in collaboration in the sessions but when it comes to the practical we want them to work independently and tell them what that means. So we have that conversation as we need to with the different types of classes that we have." (U10)

Many universities have created online resources to educate students about academic integrity. At one university students are expected to complete a quiz on academic integrity in their first year of study. This quiz is worth 5% of a student's grade. Another university teaches academic integrity by way of an online module that can also be used as a contract with the student:

"Essentially it's an online module that takes students maybe an hour and a half and they have to go through a series of exercises in academic integrity. My suspicion is that it's partly used to educate the students in academic integrity and also partly as a contract so that students can't say they didn't know something was plagiarism. If they say that you can point to that and say 'you completed this, you did know.' That is compulsory. They can't enrol in semester 2 unless they've done it. It's not a formal unit, it's an online module." (U3b)

Common techniques used to help students learn issues related to academic integrity are scenarios or examples (Good Practice Example 15). Interviewee U10 explained how she uses examples to teach students about general issues related to academic integrity and to prompt discussion about expectations within their learning environment:

"We go through examples in the beginning of the course to try and get them to be aware of what the issues are and then as students might have an issue, we discuss it more specifically with them as we go through the practicals and the workshops."

Another interviewee explained how he used scenarios successfully to engage students in a discussion about academic integrity:

"I used it within that communication technology course, used the scenarios as tutorial-based problems … We made sure that we included things such as getting someone else's code base and modifying it; finding an assignment that was almost complete, and not copying it, but reading it. We made sure that the scenarios were very related to the discipline. For some scenarios there was no consensus, even among staff. We actually got legal involved as well, so the legal people also said that there were some scenarios where it was quite hazy as to whether it was plagiarism or not. " (U12)

Another interviewee described how online forums are used to educate students about academic integrity.

"Forums are used to discuss academic integrity and what is acceptable. Students are advised not to post solutions to formulas on the study desk, just use an example or talk about it in general terms. When we respond to questions we give examples that are quite different to what is in the assignment. There is a link to library resources on AI as well." (U11a)

An important aspect of academic integrity in computing course is teaching students about how to appropriately reference code:

"Most of the code they need to reference is mine. At first-year level they're given a class skeleton and they write some methods in it. I say that I wrote the skeleton code so they have to leave my name under the author field and add their own name. Some of them cite Google. There is a huge amount of code that you can ask people to write or is just available. They usually take some from the text book and then modify it and they're not expected to reference that. There's a template from the text book. If they are using a library – students doing extensions may use a library or use code and build on it – they have to say what the base code was and what they added to it." (U18)

A couple of interviewees were concerned at the different views about academic integrity held by their colleagues:

> *"We have more problems with the staff who enforce it who don't understand that. Staff who say, 'the student only copied 200 lines of code; how can it be copying?'. Well it's like writing 300 words of an assignment and getting it right word for word. It's not just students who have that problem" (U5)*

3. Future Directions and Recommendations

The good practices identified in this theme are concerned with programs to assist students with study skills and time management. Based on the literature identified and interviews conducted, a number of avenues for future research arise into ways to provide effective learning support for the first-year ICT students. As social media plays a major role in the lives of the current student cohort, there is a clear need to explore how such media can be used effectively to strengthen support for learning. There is also a clear need to understand how educators can help to develop the communication skills of first-year ICT students; very little research has been done in this area. Finally, a variety of approaches are used to educate students about academic integrity, and there is a clear need for work on understanding the effectiveness of these.

Recommendation 11

Social media play a major role in the lives of the current student cohort. There is a clear need to investigate how these media can be used effectively to strengthen learning support.

Recommendation 12

There is a clear need to understand how educators can develop the communication skills of first-year ICT students, as very little research has been done in this area.

Recommendation 13

A variety of approaches are used to educate students about academic integrity; there is a clear need for work on understanding the effectiveness of these.

How we support our students – student support

Overview

In order to examine 'How we support our students' in the context of Australian the first-year ICT students, it is important to understand the nature of the student experience and the factors that shape that experience. Areas of focus will include:

- transition support
- social support
- equity programs
- at-risk behaviour analytics

In general terms, this theme is concerned with programs to support students in their social integration into university. More specifically, this includes programs designed to assist students in their transition from school to university, programs designed to increase social support structures, and specific programs designed to address equity issues, in particular increasing participation and support for female students and for indigenous students.

1. Literature Perspectives

The systematic literature review found 77 papers that were concerned with the 'How we support our students' theme. While broadly relevant to the theme, 21 papers turned out to have no specific relevance to ICT, leaving 56 papers for the detailed review. Within the literature, there is an equal focus between exploring and understanding motivational factors, such as factors leading to success or failure, and describing interventions designed to increase student support. Table 8 shows the list of Australian papers for this theme.

Topic	Australian-focused references
Transition support	Crosthwaite, C., & Kavanagh, L. (2012). Supporting transition, engagement and retention in first year engineering.
	Gale, T., & Parker, S. (2012). Navigating Change: A typology of student transition in higher education.
	Kift, S. (2009b). Articulating a transition pedagogy to scaffold and to enhance the first year student learning experience in Australian higher education. Final Report for ALTC Senior Fellowship Programme, 2009b.
	Nelson, K. J., Smith, J. E., & Clarke, J. A. (2012). Enhancing the transition of commencing students into University: an institution-wide approach.
	Wilson, K. (2009). The impact of institutional, programmatic and personal interventions on an effective and sustainable first-year student experience.
Social support	Carbone, A., Wong, J., & Ceddia, J. (2011). A scheme for improving ICT units with critically low student satisfaction.
	Falkner, K., & Munro, D. S. (2009). Easing the transition: a collaborative learning approach.
	McCarthy, J. (2010). Blended learning environments: using social networking sites to enhance the first year experience.
Equity support	Grant, S., Dyson, L. E., & Robertson, T. (2010). A participatory approach to the inclusion of indigenous Australians in information technology.
	Lasen, M. (2010). Education and career pathways in information communications technology: what are schoolgirls saying?
	McLachlan, C., Craig, A., & Coldwell, J. (2010). Student perceptions of ICT: a gendered analysis.
	Roberts, M. R., McGill, T. J., & Hyland, P. N. (2012). Attrition from Australian ICT degrees: why women leave.
	Chinn, D., Sheard, J., Carbone, A., & Laakso, M. J. (2010) Study habits of CS1 students: what do they do outside the classroom?
At-risk behaviours and analytics	Falkner, N. J., & Falkner, K. E. (2012). A fast measure for identifying at-risk students in computer science.
	Grebennikov, L., & Shah, M. (2012). Investigating attrition trends in order to improve student retention.
	Nelson, K. J., Duncan, M. E., & Clarke, J. A. (2009). Student success: the identification and support of first year university students at risk of attrition.
	Purnell, K., McCarthy, R., & McLeod, M. (2010). Student success at university: using early profiling and interventions to support learning.
	Quinn, C., Bennett, J., Clarke, J. A., & Nelson, K. J. (2012). The evolution of QUT's student success program: 20,000 students later.
	Sheard, J., Carbone, A., Markham, S., Hurst, A. J., Casey, D., & Avram, C. (2008). Performance and progression of first year ICT students.
	Hoda, R., & Andreae, P. (2014). It's not them, it's us! Why computer science fails to impress many first years.

Table 8: 'Student support' literature focused in the Australian context

The literature in this section largely presents the view that student success is not associate uniquely with academic matters, and that social, personal and cultural issues can be just as crucial as technical competence or study skill in determining overall academic success. Nikula et al (2011) identify an aspect of this in their comprehensive holistic model of 'course well-being', identifying both intrinsic and extrinsic motivational factors that affect student success, as well as 'course hygiene', which is associated with factors such as ease of access to staff and ready availability of resources and support. These factors are complex and multi-faceted; for example, the inter-related issues of culture, societal expectations, and community support indicated by Margolis and Fisher (2003) in their discussion of equity and gender issues within ICT programs. Because of this complexity, it is frequently difficult to determine causality from specific interventions and programs, with many interventions incorporating a combination of social, transitional, or equity measures on one hand and academic support structures and changes in pedagogy on the other.

Much of the literature that focuses on understanding and exploring non-academic reasons for success and failure identifies that perceptions of success and confidence have a significant impact on actual success. When students perceive that they are succeeding in their studies, they are more likely to achieve higher grades, regardless of the accuracy of their initial assessment. Similarly, when students are lacking in confidence or assess their performance negatively, they are more likely to withdraw or to struggle with future studies in the discipline (Fink, 2013). Accordingly, many of the identified interventions and programs include building self-confidence as a critical objective (Howles, 2009). Common across much of the analysis is the view that successful interventions are best structured within a curriculum framework (Bedford & O'Brien, 2012; Kift & Field, 2009) incorporating both generic and discipline-specific aspects that increase relevance and authenticity (Grant et al, 2010).

2. Current Practice:

The remainder of the analysis of the related literature will be explored relevant to the broad categories identified:

- transition support, including analysis of transition concerns according to both academic staff and student perceptions, and transition programs and interventions designed to reduce attrition
- social support, including the effects of social support structures on student engagement and success, and programs and interventions designed to introduce or enhance social support structures
- equity programs, including analysis of equity issues, particular concerns held by equity groups, and programs and interventions designed to target specific equity groups and increase their participation
- systems and processes designed to identify at-risk behaviour either in individuals or in groups

iv. Transition support

Transition support has received significant attention over recent years, both in Australia (Kift, 2009a; Bedford & O'Brien, 2012) and more broadly (Penn-Edwards & Donnison, 2011; Jamelske, 2008), moving from short orientation programs with an academic focus to extended, more holistic, programs addressing the notion that transition "has as much to do with changes in a person's social life and living arrangements as it has to do with leaving the former educational setting of small classes and managed learning…" (Bedford & O'Brien, 2012, p53). While transition is widely acknowledged, it is still of concern to students, who "know that university is going to be different from high school but they do not expect a difference" (Crisp et al, 2009).

One significant issue remaining is how to engage students in transition and social support programs once they are made available. While most universities provide extensive support for students, the majority of students are often unaware of the support available, or when aware, do not avail themselves of the opportunities. Further, Buglear (2009) has identified in his study of attrition behaviours that the majority of students who withdraw early in their enrolment have not engaged with any online resources.

Penn-Edwards and Donnison (2011) report on a small-scale study designed to explore critical stages of engagement with support structures, identifying five critical points in the structure of a transition program: acceptance of place at university; orientation week; return of first assignment; end of semester; and end of year. In their study, they identify a need for a secondary transition stage at the start of the second semester of study, and the need to balance assessment load during the first semester to avoid overloaded work patterns. These additional needs requiring additional transition support, but at a level that is difficult to determine. Students in this case study indicated that information about support programs was not difficult to find when searched for, but that as course materials were typically the only resource accessed, students would benefit from direct links from curricular material to support programs.

Nelson et al (2012) describe an institution-wide transition program based at QUT, explicitly linking transition support and activities with curriculum development and development of teacher and assessment skills. When designing such programs there appears to be a delicate balance between embedding support programs within the curriculum, achieving the degree of specialisation required to achieve authenticity and relevance within the support structures, and avoiding replication of support programs and activities. Brinkworth et al (2009) observe that students report common issues and concerns regardless of discipline groupings. Anagnostopoulou and Parmar (2010) report on several UK institutional interventions designed to provide online support for transition programs.

Grebennikov and Shah (2012) describe an extensive study of attrition within a large multi-campus Australian University, identifying that low-SES students and first-in-family students are generally at an elevated risk of withdrawal. In their analysis, they identify several key factors for successful retention and transition programs:

- quality of student orientation
- accuracy and speed of enrolments and fee invoicing
- provision of contact for students to promptly resolve their administrative problems
- first-year student engagement in learning
- ensuring student clarity of expectations
- more active promotion and communication of support services and facilities

In their extensive analysis of literature within the area of transition, Gale and Parker (2012) identify three types of transition: 'induction', typically the immediate focus of the majority of transition programs; 'development', such as mentoring and career development programs; and 'becoming', the longer-term adjustment. Developing a sense of belonging has been established as crucial to academic success, and is equally a crucial component of social support programs that are designed to complement and extend strategic transition and orientation programs.

v. Social support

Social factors, such as isolation and a lack of belonging, are commonly identified in studies on motivational factors impacting student success. Studies indicate that students feel “part of an anonymous mass” (Rosh White, 2006) and that forming of bonds within a peer group at the start of tertiary studies addresses a common cause of transition distress (Cameron, 2007). Social support structures such as student learning communities provide both academic learning support and increasing opportunities for interaction with peers, which have been shown to increase student confidence, sense of belonging, and retention (Falkner & Munro, 2009; Howles, 2009; Fink, 2013). In contrast, and demonstrating the complexity of this area, Kinnunen and Simon (2010) demonstrate that self-efficacy concerns can be exacerbated by exposure to peers (in their case via pair programming exercises), where hostile pairings can negatively affect students’ perceptions of their own abilities.

Chopin (Kift, 2009a, p15) introduced a collaborative marking assignment designed to raise self-confidence and create social support structures. McCarthy (2010) identifies the importance of social networks in assisting transition concerns, exploring the use of online learning communities to establish social support structures. Gray (2013) identifies relationships between positive engagement in online social learning communities and measures of social adjustment, and between social adjustment and persistence at university.

Devey and Carbone (2011) describe the introduction of a peer-assisted study scheme (PASS) within a first-year introductory programming unit, with their program designed to target at-risk units rather than at-risk students. Studied more broadly (University of Wollongong, 2010; O’Brien, 2006), with related programs such as supplemental instruction (Martin & Blanc, 1981), PASS programs have been shown to have positive impact both upon students’ social development and upon their success through the development of academic skills.

vi. Equity support

Women have long been under-represented in ICT programs, with numerous studies identifying this issue (Roberts et al, 2012; Margolis & Fisher, 2003; Anderson et al, 2005) attempting to understand the reasons behind it, and seeking possible strategies to increase participation. Some studies have also identified a higher attrition rate for female students (Barker et al, 2009), although more broadly in Australia, DEEWR figures identify similar attrition rates for males and females. Indigenous students are also significantly under-represented in ICT courses; community-based intervention programs designed to increase indigenous enrolments had not previously identified ICT as a priority area (Grant et al, 2010).

Grant et al (2010) explore indigenous student participation in ICT courses through the analysis over five years of an equity intervention program. The program combines a pre-IT enrolment program, designed to increase awareness of ICT and confidence in potential students, with an indigenous student support program specialised to the ICT programs on offer.

The trend for low female participation rates in ICT is not isolated to Australia; extensive studies have identified low participation rates across a number of post-industrialised nations (Anderson et al, 2005; Lasen, 2010), despite significant national and international investments to address this gender imbalance (Lasen, 2010). At the national level, this has included efforts to increase the accuracy of ICT career reporting and statistical analysis, along with expanding access to ICT career information (SkillsMatch).

Prior studies indicate that previous experience in ICT is an indicator of success in first-year ICT units, although this falls away in later years (Tafliovich et al, 2013). Female students tend not to recognise their own previous experience, tend to underrate their abilities in comparison to their male peers (Roberts et al, 2012), and are more dependent on reassurance from other people. Self-confidence, based in part upon an assessment of previous experience and abilities, is a critical factor in student success. Building confidence in female students, helping these students recognise their abilities, and establishing a sense of belonging for female students are therefore common factors in programs designed to help support female students in ICT (Roberts et al, 2012; Margolis & Fisher, 2003). Carter et al (2011) hypothesise that increasing interest by making the curriculum more engaging for female students will also increase self-confidence, enabling female students to reach their potential. In their discussion, Carter et al point out that curriculum tends to bias towards the majority population, and explicit efforts to redress this would benefit female students and other minority groups within ICT.

There is considerable contention between the introduction of female-only support programs and programs designed to support equity in support. Carter et al (2011) summarise this debate well, identifying the tension between the benefit of providing female role models and mentors and the benefit of creating mixed groups, which tend to improve overall team performance.

Doerschuk and Mann (2009) describe their INSPIRED program, a transition program targeted at female computing students with the aim of increasing participation and retention. While the INSPIRED program addresses recruitment, it also explores retention beyond the first year of study to encourage female students to continue their ICT careers and explore further study options. The study by Doerschukan and Mann (2009) tracked student progress across two INSPIRED cohorts, identifying a significant increase in course completion rate and retention, although participants in the INSPIRED program were few in number and self-selected.

Lasen (2010) reports on an extensive qualitative study of potential female ICT students, investigating the reasons and motivations behind their choice of whether to study ICT in secondary school. Lasen identifies a clear lack of awareness of what ICT is, for both takers and non-takers of ICT courses, and a lack of understanding of the level of creativity involved and the broader relationship between ICT, society, and innovation. Peters and Pears (2013) report on a study designed to improve understanding of how ICT students develop identity in relation to the field, promoting good practices in how we discuss our discipline, and suggesting that the way students learn programming in the beginning of their education "presumably addresses a rather narrow perspective on computer (applications)".

Johnston et al (2009) explore the impact of gender on entrepreneurship, analysing correlations between personality types and potential for entrepreneurship. Although their statistical analysis was inconclusive, they suggest that while the ICT industry has been a male-dominated industry, in the developing world local business startup using ICT is an attractive prospect for many women,.

Ballantyne et al (2009) report on a study of low-SES students within an Australian university, further identifying a high proportion of students within their sample as coming from non-traditional demographics, including mature age entry, part-time study, and first in family. In their study they explore student perceptions of their study environment, identifying strong positive associations with study environment and progress.

vii. At-risk behaviour analytics

Some studies explore the detection of students who are exhibiting at-risk behaviours, typically representing a blend of academic and social behaviours that are categorised as potentially leading to failure or attrition. Because this section is deals with non-academic learning support, we will focus on examples from the literature that address analytics related to social support structures.

Analysing at-risk learning behaviours associated with timely submission of assignments, Falkner and Falkner (2013) identify a correlation between behaviours developed in first year and subsequent activities. Garcia-Solorzano et al (2012) introduce a learning analytics system designed to assist teachers in identifying at-risk students through the creation of 'data portraits' for individual students. Data portraits are used to visually represent students' social behaviours, such as forum activity, and can be used to identify common patterns that lead to student attrition. While the approach appears promising, detailed studies are required to determine whether representative data portraits for at-risk students can be clearly defined for the purpose of automated analysis and detection. Haig et al (2013) provide a similar set of analysis tools that can provide early detection of at-risk behavior by examining resource access and social access patterns within individual units. Again, this work needs to be explored in more depth before automated detection can be assured. These works indicate just a small amount of the growing work in learning analytics, designed to assist both teachers and students in identifying at-risk behaviours, and to assist students to develop self-regulated learning skills.

Johnston et al (2009) explore the use of analytics with a different aim, using personality type indicators and other analysis to identify potential for ICT entrepreneurship. One of their goals is to determine whether there is a correlation between potential ICT entrepreneurs and gender – this is discussed further under that specific category.

3. Current Practice in Australia

The interview questions related to the theme of 'How we support our students' queried the type and extent of support systems across first-year ICT courses in Australian universities, specifically addressing support systems for equity groups.

i. At-Risk Support Programs

All universities have programs to help support at-risk students, primarily through mentoring, first-year advisers, and drop in centres that support, and in some cases initiate, contact with first-year students. Students are typically identified as at-risk via analysis of their attendance and/or performance in early assessment opportunities or class activities, although some universities contact all new students. Some universities emphasise the social aspects, integrating social clubs to create peer support.

Several interviewees discussed the approaches used to increase student engagement in online environments. The following examples describe the use of follow-up emails.

> *"What I've found with first-year students is that there's a mountain of information when they enrol and there's emails going left right and centre. One of the approaches I use is to tell them that they can ignore many of my emails but they shouldn't ignore one of them and I call them weekly housekeeping emails. So I do give them a bit of a fright – tell them they could miss a deadline that's been changed or some assessment that's been modified. On a Friday for the following week I send the email saying 'these are all the things you should worry about next week' and that is quite successful. They heed that message and they look out for that email so I connect with them in that way outside of classes." (U4)*

> *"We normally ring around after about 4 weeks if they haven't engaged in the unit and say 'what's happening, do you have a problem?' The university has actually finally decided that it's a university problem and has established a student experience team to look at that problem and do the ringing themselves rather than relying on academics to do it … They can get the figures out of Blackboard, so if a student hasn't downloaded any material or hasn't logged into Blackboard they can get a report and contact the students. Usually it's students who haven't worked out how to unenrol or didn't realise they hadn't unenrolled." (U22)*

Interviewee U24 uses a combination of both weekly emails and follow-up emails:

> *"We also use weekly emails to all students saying what's happening, what they're going to be doing each week. This was something that we instituted just for external students initially but we find that the internal students use it in a similar way. We've had internal students comment about how useful those weekly emails are because it gives them some indication of what they're doing. The other thing that I do is that I use the quickest and easiest way that the LMS provides of monitoring whether students have actually been into the system or not. It's not usually an issue with the face-to-face students because they have to go into it in class and do stuff, so they're all there. What I do for the external students is to, at least once a week and sometimes twice a week in the first two weeks, go in and see if students have been in and actually checked out anything. There is a very fast way of doing that in the system. If there's nothing there in the first week I mark it on my spreadsheet and if there's nothing there half-way through the second week I email students and if they still haven't been in by the third week, I'll call them."*

Some interviewees reported support for specific at-risk groupings, such as maintaining contact lists for at-risk groups, including first-in-family and equity groups within the discipline (e.g. female students within ICT or Engineering). Interviewee U21 describes the program:

> *"At-risk students are identified in the first few weeks and they are contacted by email, and if no response a telephone call, and at the end sending a letter. So we try to identify them: we use a matrix to identify at-risk students – it includes attendance; we put a very low stake assessment item in the first three weeks of the course e.g. online quiz at the end of week 2 which is 2% of the final grade. We can use that to identify students who are not engaging. Whenever a student enrols in a subject the Interact system can tell us which student accesses which item. So we can identify which student did not access that subject. The lecturer can generate a report on when students access the material in Interact."*

However, it is more common for programs targeting female students to be aimed at recruitment rather than retention of current students, despite expressed concerns over participation by female students:

"We do have interventions that we run outside of the school, for example, in K-12 to try to encourage more girls to take up IT. Within IT itself we have tried, for the early years to have a girls' club but we can't get them to come. Also to some extent it's enforcing that discrepancy. That's operating in the postgraduate and final year but not at the lower level." (U9)

In contrast, one interviewee indicates the combined purposing of an outreach program targeted at recruiting female students to facilitate interaction among current female students:

"For women we have our Women of SITE [Science, IT, and Engineering]. We've got a Facebook page, we've got our workpage coming out this year. We're also incorporating into that Robogals which is a group that's run by students for secondary school. University students go out to secondary schools to try and engage new students into that. We use that for women to try and get them to talk to one another. You may only have half a dozen females in your group. Trying to get them to speak and get together and to give them a sense of being part of this. We have very low student numbers in STEM." (U14)

Several interviewees indicated the presence, either past or current, of student social club environments designed to support female students:

"We have our group called Women in Engineering and Technology and that's run by our Faculty office. It sets up large social events for women within the Faculty to get together and have a bit of a chat but also they talk about problems they might be having in their courses. As part of that, they also have mentoring and industry events so female students can talk to female industry members." (U12)

With the introduction of government HEPP funding to support the participation of low-SES students in tertiary study, several interviewees reported on the introduction of recent programs targeting recruitment of low-SES students.

"In terms of having something specifically for certain demographics, our funding for that HEPP and PASS stuff is based on low-SES students. But it's still voluntary. We don't ask them to put up their hand based on their parent's income or whether they're first in family. We recognise that out cohort has quite a high percentage of first in family and low SES based on where we are. We do have programs that are based on that but are not restricted to that." (U15)

Interestingly, one interviewee reported on the tensions introduced when developing support programs for specific equity groups:

"The School has been fortunate in gaining some funding in the past few years through HEPP funding; funding targeting low-SES students. The interventions that we have developed through that funding have been driving the people who manage the HEPP funding because we have been applying it to the whole School. They want us to tell them how it helps low-SES students. From the school's perspective, whatever we do is for everybody." (U9)

Few interviewees reported results or review of their support programs for at-risk students, but one interviewee did provide some evidence for the effectiveness of these programs. This may indicate a need to discuss further and share knowledge on these types of programs.

"We used to interview the students. or those students who would have been on the list to contact – that would also include students who were currently on restrictions in their study – I would say about 10% of the students took up the offer of having a discussion with

a member of staff. Of that 10% I would say the majority of them had already recognised that there was a problem and were taking steps to rectify it. Of the 90%, we just never saw them. If they're not engaging with the university it's really difficult to get in touch with them." (U9)

ii. Transition

Many interviewees indicated the presence of a transition program, mostly combining centrally-run and locally-run activities. One interviewee described the transition program in detail, indicating many elements identified in the literature as good practice, such as the activation of direct contact based on indications of poor engagement in online learning systems (Good Practice Example 16).

"At the university level the main program is the student success project (SSP) that I mentioned before. It is interventionist and this is one reason why the uni encourages or dictates that 15% of the assessment must be completed before week 5. I think that is mandated now across the university. That is so that SSP can intervene in the way I described earlier if there are problems.

I would have to inform the coordinators of SSP of when the assessments are due in the first 4-5 weeks. I would then advise them of submissions and non-submissions and they would follow it up.

We have a T&L advisor but I'm not sure whether they have separate programs. The SSP is one of the major intervention programs for at-risk students. We used to have several T&L advisors and they were very good and very qualified at being able to deal with the different groups of students and would encourage them but unfortunately we've lost those resources and we only have one T&L advisor who doesn't have a lot of time. They only have time to refer students rather than counsel them." (U4)

The same interviewee reports on a complementary mentoring system that provides continuing support beyond the transition period (Good Practice Example 17):

"It's a mentoring service that operates on a one-on-one basis between 9 and 5 Monday to Friday for the first 10 weeks. They can go drop in any time and have a one-on-one session. This is provided on a voluntary basis by second- or third-year students and they can discuss all aspects of the course with the first-year students who use the service, including the assignments. They guide them without giving the answers and try to make sure that the students don't leave feeling frustrated. If students have other problems they can be refer them to a teaching and learning advisor." (U4)

iii. Social Support Structures

Several interviewees discussed how increasing social support can have a positive impact on student engagement. Interviewee U10 stressed the importance of having a social cohort (Good Practice Example 18):

"The whole curriculum restructure that we did was pretty much based around that. What we found is that students tend to be much more engaged when they have the social structures around them to support their learning. And that was one of the main reasons we wanted to bring collaborative learning in a much stronger way into first year. We wanted our students to have those social bonds, to have a group and to feel like they were actually contributing and having ownership of what they were doing. Our first-year

restructure has been based around that, but with the social support structures around it; the learning centre and the peer mentors etc. They are all designed to buffer and support students in those first few weeks but also to make them feel that there is a social cohort for them." (U10)

Interviewee U12 discussed how extra-curricular activities can be used to help students understand that staff are approachable:

"We encourage them to take part in things such as gaming nights. In the past we've had things such as pool competitions between staff and students, cricket, social events, to try to make them see that we're not people that aren't approachable. Particularly for students that are struggling, that they can approach us and talk about any issues; talk about the problems that they're experiencing. So it's more about showing them that we're people that can help." (U12)

Several interviewees mentioned the importance of remembering the students' names:

"What I do for the face-to-face class is that I know all the students' names by the second week. I think that's incredibly important that students see that they're not a face in the crowd, that we know who they are. And the number of students who comment on that is just amazing." (U24)

Others discussed other aspects of personal communication:

"They are most likely to do with personal contact. If students are in trouble they send them a personal email about progress. If they've been failing they get personalised online feedback fairly quickly." (U18)

One interviewee described a variety of social support structures, including curriculum-based structures to encourage the development of group work skills and social bonds, and more typical transition-based support programs including peer mentoring and a peer-led drop-in help centre (Good Practice Example 19):

"We've got peer collaboration within classes and some topics (our subjects are called topics) use partnership learning and there's a student focus on what's going to be taught. There's a topic in which students undertake an external challenge of a real-world scenario for Engineers without Borders. Our computer science, engineering, and our IT students participate in that. They design real-world solutions of ICT problems in third world countries. That's very group-led. They design their own solution and it's incredible what they do in first year. Students make it to the finals most years – the state finals if not the Australia-wide ones. That's done in our topic which is school-wide, called Professional Skills. They do it either first or second semester of their first year. Every single student does that, even our maths degree ones. They work in groups.

We have something called a help desk and study skills. We've got peer mentors. That's a big new initiative. We've implemented a university-wide peer mentoring system. Each first-year student is allocated to a peer mentor. The groups of 20 students have a higher-year or postgraduate students to assist them with their course. I've just implemented it. The DVC Academic, late last year, said that he wanted this to happen.

Previously we had something called a peer mentoring system where our mentors did not necessarily have a group of students but they were available to the student body every day. We've got a very good science innovation learning centre (SILC) where we have a room, and we've got a room there and a room in the computer science ICT building so I'm running both [the old program and the new peer mentoring program]. We've got the peer mentors and the drop-in centre for our computing students. They can get assistance throughout the week. It' mostly targeted at first-years." (U11)

iv. Retention and Attrition Programs

Many interviewees indicated formal processes associated with academic or assessment boards, either through the analysis of course results to indicate lack of engagement throughout the semester, or post-course analysis to determine students with a history of poor achievement.

One interviewee reported on an analysis program and its link to a student mentoring system:

> *"I guess the other thing that we would look at as a discipline is the marks meeting at the end of semester. People would look for the number of DNS (did not submit any work) or the number of people who withdrew after a certain period of time. That is something that people consciously look for. The issue of first-year retention does seem to be on everybody's mind.*
>
> *People would have been flagged as at risk by the staff members and first-year advisor contact. If the student is struggling and saying they don't understand but are trying, the student advisor would try to team them up with a third-year student to work through some problems together. The way the system works is that we put some nice remarks on the third-year student' transcript and they do it as a service to the university. I don't believe they get paid for it." (U5)*

Other interviewees indicated a more formal assessment process, where students who have consistently failed courses are entered into an unsatisfactory academic progress process, and are required to identify a plan for assessing and dealing with problems before being allowed to continue their studies.

> *"The other process is an unsatisfactory academic progress process which is very formal. Essentially it's students who have failed three semesters in a row. It's a simplistic definition but it's reasonably accurate. Those students are called in to talk to members of the faculty to see what particular issues they have." (U12)*

Some interviewees indicated approaches to improving retention rates that targeted the analysis and/or improvement of specific courses, rather than focusing on student support only:

> *"At the examiners' meeting staff with poor retention are identified. They then look at the course more closely and those staff members are offered support to improve performance, e.g. new teaching techniques or better resources to help with retention and attrition." (U13)*

3. Future Directions and Recommendations

The good practices identified and discussed within this theme are concerned with programs to provide support systems for students, from transition programs designed to aid all students through to procedures and programs that target at-risk groups, either through their identification from academic progress or based on equity factors. Few interviewees reported results or evaluations of their support programs for at-risk students. This indicates a need to further discuss and share knowledge on programs of these types. Many programs hinged upon analysis of academic progress, either by examining engagement in online learning tasks during a period of study, or by examining overall student results following completion of the period of study. There is a clear need for the development of learning analytics tools to better assist academic staff in early identification of at-risk students. The trend in equity-based support programs for female students in ICT programs has been towards recruitment rather than retention. There is a clear need to identify good practice retention policies and programs within an Australian context.

Recommendation 14

A number of institutions offer support programs for at-risk students, but there is little evidence of results or evaluation of these programs. This indicates a need to further discuss and share knowledge on these types of programs.

Recommendation 15

There is a clear need for the development of learning analytics tools to better assist academic staff in early identification of at-risk students.

Recommendation 16

The trend in equity-based support programs for female students in ICT programs has been towards recruitment rather than retention. There is a clear need to identify good practice retention policies and programs within an Australian context.

Summary of examples of good practice

Within each of the six themes we identified examples of good practice from the interview data. A total of 19 practices were selected for inclusion in the report. Our selection criteria were the following:

- application of an innovative practice, or of an existing practice to a new context or problem; and
- the practice addresses a current teaching or learning issue in a first-year ICT course.

Our original intention was to include examples of practices only where there was evidence from an evaluation to support any claim that the practice was achieving its stated purpose. However, we found very few examples that had such evidence. We therefore widened the scope to include practices where the interviewee gave an experiential report of successful outcomes.

The following is a list of the good practices identified. Where it was found, information of any published evaluative studies has been included. Note that in a number of cases the practice was found at more than one institution.

What we teach

Good practice example 1: a move from Java to Scribble, a visual programming language, in order to encourage introductory programming students to focus on problem solving rather than coding.

Good practice example 2: the use of programming languages for mobile development platforms to teach introductory programming. The aim is to capture students' interest by using an environment that they engage with on a regular basis.

Where we teach

Good practice example 3: the reduction of lecture time in order to increase the time spent in practical lab sessions. The aim is to increase student engagement and retention through increasing opportunities for active learning.

Good practice example 4: the design of teaching spaces to facilitate collaboration and group work.

How we teach

Good practice example 5: the introduction of the flipped classroom technique and clicker technology in lectures to engage students in active learning. The aim is to make the lecture a more interesting and valuable learning experience for students.

Good practice example 6: the use of media computation techniques in an introductory programming unit to reduce the barriers to engaging with programming concepts and to broaden the appeal of a programming unit to non-ICT students.

Good practice example 7: the redesign of the first-year curriculum around collaborative learning to engage and motivate students and increase retention through reducing social isolation.

Good practice example 8: the use of authentic problem contexts for the ICT domain. The aim is to relate learning activities to the students' real-world experiences and to demonstrate the rationale for learning specific skills by showing how those skills will be used in the real world.

Good practice example 9: using the social networking software UCROO to develop and support a learning community of introductory programming students.

How we assess

Good practice example 10: the move from assignments and exams to portfolio assessment in an introductory programming unit.

Good practice example 11: a university-wide policy of e-assessment where all assessment is submitted and returned online with feedback. This forms a permanent record of assessments and allows for a systematic process to improve the standard and responsiveness of feedback.

Good practice example 12: assessing programming assignments via interviews to determine student's understanding of the work they have submitted and enable verification that the student is the author of the work.

How we strengthen the learning environment

Good practice example 13: a program (MeetUp) to assist both on-campus and off-campus students with study skills. The program involves workshops and a peer mentoring scheme.

Good practice example 14: use of a flowchart to help students with scheduling of tasks and time management.

Good practice example 15: online resources to help educate students about academic integrity.

How we support students

Good practice example 16: student success project, a transition program for first-year students incorporating generic services and services that are integrated into the curriculum.

Good practice example 17: mentoring and social support for first-year students during the first 10 weeks of their first semester. This involves second- and third-year students providing one-to-one support for students.

Good practice example 18: a social support structure for first-year females studying ICT, involving social functions and talks from more experienced female students.

Good practice example 19: a social support program combining multiple opportunities for group and collaborative learning within courses, and peer-led support structures to provide peer-based assistance and mentorship.

Summary of Australian research studies

The numbers of research studies with an Australian context within each theme and topic are summarised in Table 9. This summary serves to indicate the focus of research in Australia.

There are a few topics that have been the recent focus of research in Australia. In the 'What we teach' theme a focus of research has been curriculum design, particularly using SFIA. In the 'How we teach' theme there has been a focus on theories and models of learning programming.

The theme with the most publications is 'How we assess'. A number of researchers from multiple institutions have been investigating the summative assessment of programming using exams and analysing the types of questions used. Several researchers have explored other forms of assessment, including the use of social media.

The summary table in combination with the theme recommendations also serves to show where further research could be focused. Three areas are suggested:

- teaching spaces
- social media
- academic integrity

Theme	Topic	Number of papers
What we teach	curriculum design	5
	first-year curriculum	3
	programming languages	2
Where we teach	virtual teaching spaces	2
How we teach	theories and models of learning	5
	tools, technologies, resources	2
	cooperative and collaborative learning	2
	social media	2
How we assess	assessment design and strategies	2
	exam assessment	10
	non-exam forms of assessment	5
	academic integrity	1
Learning support	learning communities	2
	teamwork skills	2
	study skills	3
	academic integrity	1
Student support	transition support	6
	social support	3
	equity support	5
	at-risk behaviours	7

Table 9: 'Learning support' literature focused in the Australian context

Conclusion

We have conducted a wide-ranging review of the recent literature pertaining to the first-year student experience in ICT courses, both in Australia and overseas. We have interviewed ICT academics involved in the students' first-year experience at 25 universities in Australia. Based on these two complementary approaches we have put together a comprehensive picture of the first-year experience for ICT students in Australian universities, and have made a number of recommendations addressing perceived opportunities either to improve students' first-year experience or to examine whether current practices are in fact improving that experience. Finally we have identified a number of good practices in first-year ICT education in the hope that others will consider adopting them.

References

Adegbehingbe, O. D., & Eyono Obono, S. D. E. (2012). A framework for designing information technology programmes using ACM/IEEE curriculum guidelines. *World Congress on Engineering and Computer Science 2012*.

Alammary, A., Carbone, A., & Sheard, J. (2012). Implementation of a smart lab for teachers of novice programmers. *14th Australasian Computing Education Conference,* 121-130.

Anagnostopoulou, K., & Parmar, D. (2010). *Supporting the first year student experience through the use of learning technologies.* Middlesex University and the Higher Education Academy, London.

Anderson, M., & Gavan, C. (2012). Engaging undergraduate programming students: experiences using LEGO Mindstorms NXT. *13th Conference on Information Technology Education*, 139-144.

Anderson, N., Klein, M., & Lankshear, C. (2005). Redressing the gender imbalance in ICT professions: toward state-level strategic approaches. *Australian Educational Computing, 20*(2), 3-10.

Apiola, M., Lattu, M., & Pasanen, T. A. (2010). Creativity and intrinsic motivation in computer science education : experimenting with robots. *15th Conference on Innovation and Technology in Computer Science Education*, 199-203.

Apiola, M., Lattu, M., & Pasanen, T. A. (2012). Creativity-supporting learning environment – CSLE. *ACM Transactions on Computing Education*, 12(3), 11.

Ballantyne, J., Madden, T., & Todd, N. (2009). Gauging the attitudes of non-traditional students at a new campus: an Australian case study. *Journal of Higher Education Policy and Management*, *31*(4), 301-313.

Barker, L.J., McDowell, C. and Kalahar, K. (2009): Exploring factors that influence computer science introductory course students to persist in the major. *ACM SIGCSE Bulletin,* 41(1), 153-157.

Barros, J. P. (2010). Assessment and grading for CS1 : towards a complete toolbox of criteria and techniques. *10th Koli Calling International Conference on Computing Education Research*, 106-111.

Bayzick, J., Askins, B., Kalafut, S., & Spear, M. (2013). Reading mobile games throughout the curriculum. *44th ACM Technical Symposium on Computer Science Education*, 209-214.

Beck, L., & Chizhik, A. (2013). Cooperative learning instructional methods for CS1 : design, implementation, and evaluation. *ACM Transactions on Computing Education*, 13(3), 10.

Bedford, S. B., & O'Brien, G. (2012). A view of first year transition from Down Under. *New Directions*, 8, 53-57.

Benkrid, K., & Clayton, T. (2012). Digital hardware design teaching: an alternative approach. *ACM Transactions on Computing Education*, *12*(4), 13.

Biggs, J. (1996). Enhancing teaching through constructive alignment, *Higher Education,* 32(3), 347-364*.*

Börstler, J., Nordström, M., & Paterson, J. H. (2011). On the quality of examples in introductory Java textbooks. *ACM Transactions on Computing Education*, 11(1).

Brinkworth, R., McCann, B., Matthews, C., & Nordström, K. (2009). First year expectations and experiences: student and teacher perspectives. *Higher Education*, 58(2), 157-173.

Buglear, J. (2009). Logging in and dropping out: exploring student non-completion in higher education using electronic footprint analysis. *Journal of Further and Higher Education*, 33(4), 381-393.

Cain, A., & Woodward, C. J. (2012). Toward constructive alignment with portfolio assessment for introductory programming. *IEEE International Conference on Teaching, Assessment, and Learning for Engineering 2012*, H1B-11.

Cain, A. & Woodward, C. J. (2013). Examining student reflections from a constructively aligned introductory programming unit. *15th Australasian Computing Education Conference*, 127-136.

Cameron, H. (2007), Assessment and feedback in higher education: Key links in the learning chain. *2nd Education Research Group of Adelaide Conference (ERGA 2007).*

Carbone., Hurst, J., Mitchell, I., & Gunstone, D. (2009). An exploration of internal factors influencing student learning of programming. *11th Australasian Computing Education Conference*, 25-34.

Carbone, A., Wong, J., & Ceddia, J. (2011). A scheme for improving ICT units with critically low student satisfaction. *16th Conference on Innovation and Technology in Computer Science Education*, 253-257.

Carter, J., Bouvier, D., Cardell-Oliver, R., Hamilton, M., Kurkovsky, S., Markham, S., McClung, O. W., McDermott, R., Riedesel, C., Shi, J & White, S. (2011). ITiCSE 2010 working group report: motivating our top students. *16th Conference on Innovation and Technology in Computer Science Education – Working Group Reports*, 1-18.

Caspersen, M. E., & Kolling, M. (2009). STREAM: A first programming process. *ACM Transactions on Computing Education*, *9*(1), 4.

Cheryan, S., Meltzoff, A. N., & Kim, S. (2011). Classrooms matter: the design of virtual classrooms influences gender disparities in computer science classes. *Computers & Education*, 57(2), 1825-1835.

Chinn, D., Sheard, J., Carbone, A., & Laakso, M. J. (2010). Study habits of CS1 students: what do they do outside the classroom? *12th Australasian Computing Education Conference*, 53-62.

Cooper, S. (2010). The design of Alice. *ACM Transactions on Computing Education*, *10*(4), 15.

Corney, M., Teague, D., Ahadi, A., & Lister, R. (2012). Some empirical results for neo-Piagetian reasoning in novice programmers and the relationship to code explanation questions. *14th Australasian Computing Education Conference*, 77-86.

Corney, M., Teague, D., & Thomas, R. N. (2010). Engaging students in programming. *12th Australasian Computing Education Conference*, 63-72.

Crisp, G., Palmer, E., Turnbull, D., Nettelbeck, T., Ward, L., LeCouteur, A., Sarris, A., Strelan, P. & Schneider, L. (2009). First year student expectations: results from a university-wide student survey. *Journal of University Teaching and Learning Practice*, 6(1).

Crosthwaite, C., & Kavanagh, L. (2012). Supporting transition, engagement and retention in first year engineering. *International Conference on Innovation, Practice and Research in Engineering Education*.

Cutts, Q., Cutts, E., Draper, S., O'Donnell, P., & Saffrey, P. (2010). Manipulating mindset to positively influence introductory programming performance. *41st ACM Technical Symposium on Computer Science Education*, 431-435.

Daniels, T. E. (2009). Integrating engagement and first year problem solving using game controller technology. *39th IEEE Frontiers in Education Conference, 2009*, 1-6).

Dann, W., Cosgrove, D., Slater, D., Culyba, D., & Cooper, S. (2012). Mediated transfer: Alice 3 to Java. *43rd ACM Technical Symposium on Computer Science Education*, 141-146.

Denny, P., Hanks, B., & Simon, B. (2010). Peerwise: replication study of a student-collaborative self-testing web service in a US setting. *41st ACM Technical Symposium on Computer Science Education*, 421-425.

de Raadt, M. (2012). Student created cheat-sheets in examinations: impact on student outcomes. *14th Australasian Computing Education Conference*, 71-76.

Devey, A., & Carbone, A. (2011). Helping first year novice programming students PASS. *13th Australasian Computing Education Conference*, 135-144.

Doerschuk, P., Liu, J., & Mann, J. (2009). INSPIRED broadening participation: first year experience and lessons learned. *ACM SIGCSE Bulletin*, 41(3), 238-242.

Dweck, C. S. (2000). *Self-theories: Their role in motivation, personality, and development*. Psychology Press.

Eagle, M., & Barnes, T. (2009). Evaluation of a game-based lab assignment. *4th International Conference on Foundations of Digital Games*, 64-70.

Edwards, S. H., Snyder, J., Pérez-Quiñones, M. A., Allevato, A., Kim, D., & Tretola, B. (2009). Comparing effective and ineffective behaviors of student programmers. *5th International Computing Education Research Conference*, 3-14.

Edwards, R. L., Stewart, J. K., & Ferati, M. (2010). Assessing the effectiveness of distributed pair programming for an online informatics curriculum. *ACM Inroads*, 1(1), 48-54.

Egan, R., Cukierman, D., & McGee Thompson, D. (2011). The academic enhancement program in introductory CS: a workshop framework description and evaluation. *16th Conference on Innovation and Technology in Computer Science Education*, 278-282.

Egan, M. H., & Mcdonald, C. (2014). Program visualization and explanation for novice C programmers. *16th Australasian Computing Education Conference*, 51-57.

Egea, K., Kim, S., Andrews, T., & Behrens, K. (2010). Approaches used by cross-cultural and cross-discipline students in teamwork for a first-year course in web design. *12th Australasian Computing Education Conference*, 87-96.

Elliott Tew, A., & Guzdial, M. (2010). Developing a validated assessment of fundamental CS1 concepts. *41st ACM Technical Symposium on Computer Science Education*, 97-101.

Falkner, N. J., & Falkner, K. E. (2012). A fast measure for identifying at-risk students in computer science. *9th International Computing Education Research Conference*, 55-62.

Falkner, K., & Falkner, N. J. G. (2013). Designing and supporting collaborative learning activities. *44th ACM Technical Symposium on Computer Science Education*, 537-538.

Falkner, K., & Munro, D. S. (2009). Easing the transition: A collaborative learning approach. *11th Australasian Computing Education Conference*, 65-74.

Falkner, K., & Palmer, E. (2009). Developing authentic problem solving skills in introductory computing classes. *ACM SIGCSE Bulletin*, 41(1), 4-8.

Fincher, S., Cooper, S., Kölling, M., & Maloney, J. (2010). Comparing Alice, Greenfoot & Scratch. *41st ACM Technical Symposium on Computer Science Education*, 192-193.

Fincher, S., & Utting, I. (2010). Machines for thinking. *ACM Transactions on Computing Education*, 10(4), 13.

Fink, L.D. (2013). *Creating significant learning experiences: an integrated approach to designing college courses*. John Wiley & Sons.

Ford, M., & Venema, S. (2010). Assessing the success of an introductory programming course. *Journal of Information Technology Education*, 9, 135-145.

Gale, T., & Parker, S. (2012). Navigating change: a typology of student transition in higher education. *Studies in Higher Education*, 1-20.

García-Solórzano, D., Cobo, G., Santamaria, E., Poblenou, R., Morán, J. A., Monzo, C., & Melenchón, J. (2012). Educational monitoring tool based on faceted browsing and data portraits. *2nd International Conference on Learning Analytics and Knowledge*, 170-178.

Garlick, R., & Cankaya, E. (2010). Using Alice in CS1: A quantitative experiment. *15th Conference on Innovation and Technology in Computer Science Education*, 165-168.

Gluga, R., Kay, J., Lister, R., Kleitman, S., & Lever, T. (2012). Coming to terms with Bloom: an online tutorial for teachers of programming fundamentals. *14th Australasian Computing Education Conference*, 147-156.

Gouws, L., Bradshaw, K., & Wentworth, P. (2013). First year student performance in a test for computational thinking. *South African Institute for Computer Scientists and Information Technologists Conference*, 271-277.

Govender, I. (2009). The learning context: Influence on learning to program. *Computers & Education*, 53(4), 1218-1230.

Grant, S., Dyson, L. E., & Robertson, T. (2010). A participatory approach to the inclusion of indigenous Australians in information technology. *11th Biennial Participatory Design Conference*, 207-210.

Gray, M. (2013). When digital native meets analogue reality: a case study of ICT skills in first year university students. *PICMET'13, Technology Management in the IT-Driven Services*, 2479-2487.

Gray, K., Thompson, C., Sheard, J., Clerehan, R., & Hamilton, M. (2010). Students as web 2.0 authors : implications for assessment design and conduct. *Australasian Journal of Educational Technology*, 26(1), 105-122.

Gray, K., Waycott, J., Clerehan, R., Hamilton, M., Richardson, J., Sheard, J., & Thompson, C. (2012). Worth it? Findings from a study of how academics assess students' web 2.0 activities. *Research in Learning Technology,* 20(1), 1-15.

Grebennikov, L., & Shah, M. (2012). Investigating attrition trends in order to improve student retention. *Quality Assurance in Education*, 20(3), 223-236.

Guo, Z., & Stevens, K. J. (2011). Factors influencing perceived usefulness of wikis for group collaborative learning by first year students. *Australasian Journal of Educational Technology*, 27(2), 221-242.

Hahn, J. H., Mentz, E., & Meyer, L. (2009). Assessment strategies for pair programming. *Journal of Information Technology Education*, 8.

Haig, T., Falkner, K., & Falkner, N. (2013). Visualisation of learning management system usage for detecting student behaviour patterns. *15th Australasian Computing Education Conference*, 107-115.

Hamer, J., Luxton-Reilly, A., Purchase, H. C., & Sheard, J. (2011). Tools for contributing student learning. *ACM Inroads*, 2(2), 78-91.

Hamer, J., Sheard, J., Purchase, H., & Luxton-Reilly, A. (2012). Contributing student pedagogy. *Computer Science Education*, 22(4), 315-318.

Hanks, B., Murphy, L., Sci, C., Eng, C., Simon, B., & Zander, C. (2009). CS1 students speak: advice for students by students. *ACM SIGCSE Bulletin*, 41(1), 19-23.

Harland, J., D'Souza, D., & Hamilton, M. (2013). A comparative analysis of results on programming exams. *15th Australasian Computing Education Conference*, 117-126.

Herbert, N., Dermoudy, J., Ellis, L., Cameron-Jones, M., Chinthammit, W., Lewis, I., de Salas, K. L. & Springer, M. (2013a). Stakeholder-led curriculum redesign. *15th Australasian Computing Education Conference*, 51-59.

Herbert, N., Lewis, I., & Salas, K. De. (2013b). Career outcomes and SFIA as tools to design ICT curriculum. *24th Australasian Conference on Information Systems*, 1-10.

Herbert, N., Salas, K. De, Lewis, I., Cameron-Jones, M., Chinthammit, W., Dermoudy, J., Ellis, L. & Springer, M. (2013c). Identifying career outcomes as the first step in ICT curricula development. *15th Australasian Computing Education Conference*, 31-40.

Herbert, N., Salas, K. De, Lewis, I., Dermoudy, J., & Ellis, L. (2014). ICT curriculum and course structure : the great balancing act. *16th Australasian Computing Education Conference*, 21-30.

Hertz, M., & Jump, M. (2013). Trace-based teaching in early programming courses. *44th ACM Technical Symposium on Computer Science Education*, 561-566.

Hoda, R., & Andreae, P. (2014). It's not them, it's us! Why computer science fails to impress many first years. *16th Australasian Computing Education Conference*, 159-162.

Howles, T. (2009). A study of attrition and the use of student learning communities in the computer science introductory programming sequence. *Computer Science Education*, 19(1), 1-13.

Hu, M., Winikoff, M., & Cranefield, S. (2012). Teaching novice programming using goals and plans in a visual notation. *14th Australasian Computing Education Conference*, 43-52.

Hu, M., Winikoff, M., & Cranefield, S. (2013). A process for novice programming using goals and plans. *15th Conference on Innovation and Technology in Computer Science Education*, 3-12.

Hundhausen, C. D., Agrawal, A., & Agarwal, P. (2013). Talking about code: integrating pedagogical code reviews into early computing courses. *ACM Transactions on Computing Education*, 13(3), 14.

Jamelske, E. (2008). Measuring the impact of a university first-year experience program on student GPA and retention. *Higher Education*, 57(3), 373-391.

Jenkins, G., Lyons, K., Bridgstock, R., & Carr, L. (2012). Like our page – using Facebook to support first year students in their transition to higher education. A practice report. *International Journal of the First Year in Higher Education*, 3(2), 65-72.

Johnson, C. (2012). SpecCheck: automated generation of tests for interface conformance. *17th Conference on Innovation and Technology in Computer Science Education*, 186-191.

Johnston, K. A., Andersen, B. K., Davidge-Pitts, J., & Ostensen-Saunders, M. (2009). Identifying student potential for ICT entrepreneurship using Myers-Briggs personality type indicators. *Journal of Information Technology Education*, 8, 29-43.

Kift, S. (2009a). *ALTC First Year Experience Curriculum Design Symposium 2009, FYE Showcase Abstracts,* Strawberry Hills, NSW: Australian Learning and Teaching Council.

Kift, S. (2009b). *Articulating a transition pedagogy to scaffold and to enhance the first year student learning experience in Australian higher education: final report for ALTC senior fellowship program*. Strawberry Hills, NSW: Australian Learning and Teaching Council.

Kift, S., & Field, R. (2009). Intentional first year curriculum design as a means of facilitating student engagement: some exemplars. *12th Pacific Rim First Year in Higher Education Conference*, 1-10.

Kinnunen, P., & Simon, B. (2010). Experiencing programming assignments in CS1: the emotional toll. *6th International Computing Education Research Conference*, 77-86.

Kinnunen, P., & Simon, B. (2012). My program is ok – am I? Computing freshmen's experiences of doing programming assignments. *Computer Science Education*, 22(1), 1-28.

Kölling, M. K. (2010). The Greenfoot programming environment. *ACM Transactions on Computing Education*, 10(4), 14.

Koohang, A., Riley, L., Smith, T., & Floyd, K. (2010). Design of an information technology undergraduate program to produce IT versatilists. *Journal of Information Technology Education*, 9, 99-113.

Koppi, T., Roberts, M., & Naghdy, G. (2012). Perceptions of a gender-inclusive curriculum amongst Australian information and communications technology academics. *14th Australasian Computing Education Conference*, 7-14.

Kothiyal, A., Majumdar, R., Murthy, S., & Iyer, S. (2013). Effect of think-pair-share in a large CS1 class: 83% sustained engagement. *9th International Computing Education Research Conference*, 137-144.

Kurkovsky, S. (2013). Mobile game development: improving student engagement and motivation in introductory computing courses. *Computer Science Education*, 23(2), 138-157.

Lasen, M. (2010). Education and career pathways in information communication technology: what are schoolgirls saying? *Computers & Education*, 54(4), 1117-1126.

Lasserre, P., & Szostak, C. (2011). Effects of team-based learning on a CS1 course. *16th Conference on Innovation and Technology in Computer Science Education*, 133-137.

Law, K. M. Y., Lee, V. C. S., & Yu, Y. T. (2010). Learning motivation in e-learning facilitated computer programming courses. *Computers & Education*, 55(1), 218-228.

Lee, M. J., Ko, A. J., & Kwan, I. (2013). In-game assessments increase novice programmers' engagement and level completion speed. *9th International Computing Education Research Conference*, 153-160.

Lister, R. (2011). Concrete and other neo-Piagetian forms of reasoning in the novice programmer. *13th Australasian Computing Education Conference*, 9-18.

Llana, L., Martin-Martin, E., & Pareja-Flores, C. (2012). FLOP, a free laboratory of programming. *12th Koli Calling International Conference on Computing Education Research*, 93-99.

Ma, L., Ferguson, J., Roper, M., & Wood, M. (2011). Investigating and improving the models of programming concepts held by novice programmers. *Computer Science Education*, 21(1), 57-80.

Maleko, M., Hamilton, M., & D'Souza, D. (2012). Novices' perceptions and experiences of a mobile social learning environment for learning of programming. *17th Conference on Innovation and Technology in Computer Science Education*, 285-290.

Maloney, J., Resnick, M., Rusk, N., Silverman, B., & Eastmond, E. (2010). The Scratch programming language and environment. *ACM Transactions on Computing Education*, 10(4), 16.

Margolis, J. & Fisher, A. (2003). *Unlocking the clubhouse: Women in computing*. MIT Press.

Marsa-Maestre, I., De La Hoz, E., Gimenez-Guzman, J. M., & Lopez-Carmona, M. A. (2013). Design and evaluation of a learning environment to effectively provide network security skills. *Computers & Education*, 69, 225-236.

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).

Mason, R., & Cooper, G. (2012). Why the bottom 10 % just can't do it – mental effort measures and implications for introductory programming courses. *14th Australasian Computing Education Conference*, 187-196.

Mason, R., & Cooper, G. (2014). Introductory programming courses in Australia and New Zealand in 2013 – trends and reasons. *16th Australasian Computing Education Conference*, 139-147.

Mason, R., Cooper, G., & de Raadt, M. (2012). Trends in introductory programming courses in Australian universities: languages, environments and pedagogy. *14th Australasian Computing Education Conference*, 33-42.

McCarthy, J. (2010). Blended learning environments : using social networking sites to enhance the first year experience. *Australasian Journal of Educational Technology*, 26(6), 729-740.

McDermott, R., Brindley, G., & Eccleston, G. (2010). Developing tools to encourage reflection in first year students blogs. *15th Conference on Innovation and Technology in Computer Science Education*, 147-151.

McLachlan, C., Craig, A., & Coldwell, J. (2010). Student perceptions of ICT: a gendered analysis. *12th Australasian Computing Education Conference*, 127-136.

McWhorter, W. I., & O'Connor, B. C. (2009). Do LEGO® Mindstorms® motivate students in CS1? *ACM SIGCSE Bulletin*, 41(1), 438-442.

Moons, J., & De Backer, C. (2013). The design and pilot evaluation of an interactive learning environment for introductory programming influenced by cognitive load theory and constructivism. *Computers & Education*, 60(1), 368-384.

Morazán, M. T. (2010). Functional video games in the CS1 classroom. *Trends in Functional Programming*, 166-183. Springer Berlin Heidelberg.

Murphy, C., Powell, R., Parton, K., & Cannon, A. (2011). Lessons learned from a PLTL-CS program. *42nd ACM Technical Symposium on Computer Science Education*, 207-212.

Nelson, K. J., Duncan, M. E., & Clarke, J. A. (2009). Student success: The identification and support of first year university students at risk of attrition. *Studies in Learning, Evaluation, Innovation and Development*, 6(1), 1-15.

Nelson, K. J., Smith, J. E., & Clarke, J. A. (2012). Enhancing the transition of commencing students into university: an institution-wide approach. *Higher Education Research & Development*, 31(2), 185-199.

Nikula, U., Gotel, O., & Kasurinen, J. (2011). A motivation guided holistic rehabilitation of the first programming course. *ACM Transactions on Computing Education*, 11(4), 24.

Nguyen, T. T. L, Carbone, A., Sheard, J., & Schuhmacher, M. (2013a). Integrating source code plagiarism into a virtual learning environment : benefits for students and staff. *15th Australasian Computing Education Conference*, 155-164.

Nguyen, T. T. L, Carbone, A., Sheard, J., Schuhmacher, M., de Raadt, M., & Johnson, C. (2013b). Educating computer programming students about plagiarism through use of a code similarity detection tool. *Learning and Teaching in Computing and Engineering*, 98-105.

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: The University of Wollongong.

O'Grady, M. J. (2012). Practical problem-based learning in computing education. *ACM Transactions on Computing Education*, 12(3), 10.

O'Neill, G., & Noonan, E., (2011). *Designing First Year Assessment Strategically*, 1-46. http://www.ucd.ie/t4cms/designifyassess.pdf, accessed 24 Jun 2014.

Patitsas, E., Voll, K., Crowley, M., & Wolfman, S. (2010). Circuits and logic in the lab: toward a coherent picture of computation. *15th Western Canadian Conference on Computing Education*, 7.

Pears, A. (2010). Conveying conceptions of quality through instruction. *7th International Conference on the Quality of Information and Communications Technology*, 7-14.

Pears, A., & Rogalli, M. (2011). mJeliot: A tool for enhanced interactivity in programming instruction. *11th Koli Calling International Conference on Computing Education Research*, 16-22.

Penn-Edwards, S., & Donnison, S. (2011). Engaging with higher education academic support: a first year student teacher transition model. *European Journal of Education*, 46(4), 566-580.

Petersen, A., Craig, M., & Zingaro, D. (2011). Reviewing CS1 exam question content. *42nd ACM Technical Symposium on Computer Science Education*, 631-636.

Peters, A. K., & Pears, A. (2013). Engagement in computer science and IT – what! A matter of identity? *Learning and Teaching in Computing and Engineering 2013*, 114-121.

Pieterse, V., & van Rooyen, I. J. (2011). Student discussion forums: what is in it for them? *Computer Science Education Research Conference*, 59-70. Open Universiteit, Heerlen.

Porter, L., Bailey-Lee, C., & Simon, B. (2013). Halving fail rates using peer instruction: a study of four computer science courses. *44th ACM Technical Symposium on Computer Science Education*, 177-182.

Purchase, H., Hamer, J., Denny, P., & Luxton-Reilly, A. (2010). The quality of a PeerWise MCQ repository. *12th Australasian Computing Education Conference*, 137-146.

Purnell, K., McCarthy, R., & McLeod, M. (2010). Student success at university: using early profiling and interventions to support learning. *Studies in Learning, Evaluation, Innovation & Development*, *7*(3).

Quinn, C., Bennett, J., Clarke, J. A., & Nelson, K. J. (2012). The evolution of QUT's Student Success Program: 20,000 students later. *International First Year in Higher Education Conference.*

Radermacher, A., Walia, G., & Rummelt, R. (2012). Improving student learning outcomes with pair programming. *9th International Computing Education Research Conference*, 87-92.

Richards, D. (2009). Designing project-based courses with a focus on group formation and assessment. *ACM Transactions on Computing Education*, *9*(1), 2.

Risco, S., & Reye, J. (2012). Evaluation of an intelligent tutoring system used for teaching RAD in a database environment. *14th Australasian Computing Education Conference*, 131-140).

Roberts, M. R., McGill, T. J., & Hyland, P. N. (2012). Attrition from Australian ICT degrees: why women leave. *14th Australasian Computing Education Conference*, 15-24.

Robertson, J. (2011). The educational affordances of blogs for self-directed learning. *Computers & Education*, 57 (2), 1628-1644.

Robins, A. (2010). Learning edge momentum: a new account of outcomes in CS1. *Computer Science Education*, 20(1), 37-71.

Rosh White, N. (2006). Tertiary education in the noughties: the student perspective. *Higher Education Research & Development*, 25(3), 231-246.

Salleh, N., Mendes, E., Grundy, J., & Burch, G. S. J. (2010). The effects of neuroticism on pair programming: an empirical study in the higher education context. *2010 ACM-IEEE International Symposium on Empirical Software Engineering and Measurement*, 22.

Salomon, G., & Perkins, D. (1988). Teaching for transfer. *Educational leadership*, 46(1), 22-32.

Sancho-Thomas, P., Fuentes-Fernández, R., & Fernández-Manjón, B. (2009). Learning teamwork skills in university programming courses. *Computers & Education*, *53*, 517-531.

Shaffer, S. C., & Rossen, M. B. (2013). Increasing student success by modifying course delivery based on student submission data. *ACM Inroads*, 4(4), 81-86.

Sheard, J., Carbone, A., Chinn, D., & Laakso, M. J. (2013b). Study habits of CS 1 students: what do they say they do? *Learning and Teaching in Computing and Engineering 2013*, 122-129.

Sheard, J., Carbone, A., Markham, S., Hurst, A. J., Casey, D., & Avram, C. (2008). Performance and progression of first year ICT students. *10th Australasian Computing Education Conference*, 119-127.

Sheard, J., Simon, Carbone, A., D'Souza, D., & Hamilton, M. (2013a). Assessment of programming: pedagogical foundations of exams. *18th Conference on Innovation and Technology in Computer Science Education*, 141-146.

Sheard, J., Simon, Carbone, A, Chinn, D., Laakso, M-J , Clear, T., de Raadt, M., D'Souza, D., Harland, J., Lister, R., Philpott, A., & Warburton, G. (2011). Exploring programming assessment instruments: a classification scheme for examination questions. *7th International Computing Education Research Conference*, 33-38.

Sheard, J., Simon, Dermoudy, J., D'Souza, D., Hu, M., & Parsons, D. (2014). Benchmarking a set of exam questions for introductory programming. *16th Australasian Computing Education Conference*, 113-121.

Sheard, J., Simon, Carbone, A., Chinn, D., Clear, T., Corney, M., D'Souza, D., Fenwick, J., Harland, J., Laakso, M. & Teague, D. (2013). How difficult are exams? A framework for assessing the complexity of introductory programming exams. *15th Australasian Computing Education Conference*, 145-154.

Shuhidan, S., Hamilton, M., & D'Souza, D. (2009). A taxonomic study of novice programming summative assessment. *11th Australasian Computing Education Conference*, 147-156.

Shuhidan, S., Hamilton, M., & D'Souza, D. (2010). Instructor perspectives of multiple-choice questions in summative assessment for novice programmers. *Computer Science Education*, 20(3), 229-259.

Simon, Sheard, J., Carbone, A., Chinn, D., Laakso, M.-J., Clear, T., de Raadt, M., D'Souza, D., Lister, R., Philpott, A., Skene, J. & Warburton, G. (2012). Introductory programming: examining the exams. *14th Australasian Computing Education Conference*, 61-70.

Simon, B., Esper, S., Porter, L., & Cutts, Q. (2013). Student experience in a student-centered peer instruction classroom. *9th International Computing Education Research Conference*, 129-136.

Simon, B., Kohanfars, M., Lee, J., Tamayo, K., & Cutts, Q. (2010). Experience report: peer instruction in introductory computing. *41st ACM Technical Symposium on Computer Science Education*, 341-345.

Skudder, B., & Luxton-Reilly, A. (2014). Worked examples in computer science. *16th Australasian Computing Education Conference*, 59-64.

Smet, M. De, Keer, H. Van, Wever, B. De, & Valcke, M. (2010). Cross-age peer tutors in asynchronous discussion groups: Exploring the impact of three types of tutor training on patterns in tutor support and on tutor characteristics. *Computers & Education*, 54, 1167-1181.

Smith, L., Haden, P., & Mann, S. (2012). Building student communities with social media. *3rd Computing and Information Technology Research and Education New Zealand Conference*, 93-100.

Sorva, J. (2013). Notional machines and introductory programming education. *ACM Transactions on Computing Education*, 13(2), 8.

Sorva, J., Karavirta, V., & Malmi, L. (2013). A review of generic program visualization systems for introductory programming education. *ACM Transactions on Computing Education*, 13(4), 15.

Stefik, A., & Siebert, S. (2013). An empirical investigation into programming language syntax. *ACM Transactions on Computing Education*, 13(4), 19.

Summet, J., Kumar, D., Hara, K. O., Walker, D., Ni, L., Blank, D., & Balch, T. (2009). Personalizing CS1 with robots. *ACM SIGCSE Bulletin*, 41(1), 433-437.

Sweller, J. (1999). *Instructional Design in Technical Areas.* Melbourne, Australia, ACER Press.

Tafliovich, A., Campbell, J., & Petersen, A. (2013). A student perspective on prior experience in CS1. *44th ACM Technical Symposium on Computer Science Education*, 239-244.

Teague, D. (2009). A people-first approach to programming. *11th Australasian Computing Education Conference*, 171-180.

Teague, D., & Lister, R. (2014). Longitudinal think aloud study of a novice programmer. *16th Australasian Computing Education Conference*, 41-50).

Terrell, J., Richardson, J., & Hamilton, M. (2011). Using web 2.0 to teach web 2.0: a case study in aligning teaching, learning and assessment with professional practice. *Australasian Journal of Educational Technology*, 27(5), 846-862.

Thomas, R. N., Cordiner, M., & Corney, D. (2010). An adaptable framework for the teaching and assessment of software development across year levels. *12th Australasian Computing Education Conference*, 165-172.

Thota, N., & Whitfield, R. (2010). Holistic approach to learning and teaching introductory object-oriented programming. *Computer Science Education*, 20(2), 103-127.

University of Wollongong (2010). *PASS Program Results,* http://www.uow.edu.au/student/services/pass/evaluation, accessed 24 Jun 2014.

Verginis, I., Gogoulou, A., Gouli, E., Boubouka, M., & Grigoriadou, M. (2011). Enhancing learning in introductory computer science courses through SCALE: an empirical study. *IEEE Transactions on Education*, 54(1), 1-13.

von Konsky, B. R., Jones, A., & Miller, C. (2014). Visualising career progression for ICT professionals and the implications for ICT curriculum design in higher education. *16th Australasian Computing Education Conference*, 13-20.

Wang, T., Su, X., Ma, P., Wang, Y., & Wang, K. (2011). Ability-training-oriented automated assessment in introductory programming course. *Computers & Education*, 56(1), 220-226.

Waycott, J., Sheard, J., Thompson, C., & Clerehan, R. (2013). Making students' work visible on the social web: A blessing or a curse? *Computers & Education, 68*, 86-95.

Wellman, B. L., Davis, J., & Anderson, M. (2009). Alice and robotics in introductory CS courses. *5th Richard Tapia Celebration of Diversity in Computing Conference: Intellect, Initiatives, Insight, and Innovations*, 98-102.

Wilson, K. (2009). The impact of institutional, programmatic and personal interventions on an effective and sustainable first-year student experience. *12th Pacific Rim First Year in Higher Education Conference*.

Wood, K., Parsons, D., Gasson, J., & Haden, P. (2013). It's never too early : pair programming in CS1. *15th Australasian Computing Education Conference*, 13-21.

Yang, F.P., Jiau, H. C., & Ssu, K.F. (2014). Beyond plagiarism: an active learning method to analyze causes behind code-similarity. *Computers & Education*, 70, 161-172.

Yorke, M. (2011). Assessment and feedback in the first year : the professional and the personal. *14th Pacific Rim First Year in Higher Education Conference*, 1-31.

Zacharis, N. Z. (2011). Measuring the effects of virtual pair programming in an introductory programming Java course. *IEEE Transactions on Education*, 54(1), 168-170.

Zander, C., Thomas, L., Simon, B., Murphy, L., McCauley, R., Hanks, B., & Fitzgerald, S. (2009, July). Learning styles: novices decide. *ACM SIGCSE Bulletin*, 41(3), 223-227.

Appendices

Appendix A – ICT Courses in Australia

Degree List Jan 2014

University	Faculty/Academic College	Degree	Campus
Australian Catholic University	Business	Bachelor of Information Technology	Brisbane, Melbourne, North Sydney
Australian National University	Faculty of Engineering and Information Technology	Bachelor of Information Technology	
		Bachelor of Software Engineering	
		Bachelor of Advanced Computing (Hons)	
		Bachelor of Advanced Computing (Research and Development) (Hons)	
Bond University	Society & Design	Bachelor of Interactive Media and Design	
Central Queensland University		Bachelor of Information Technology (Majors: Application Development; Business Analysis; Network Security)	Rockhampton, Brisbane, Melbourne, Sydney, Distance Education
Charles Darwin University	Faculty of Engineering, Health, Science and the Environment	Bachelor of Information Systems	Casuarina, Distance Education
		Bachelor of Information Technology	Casuarina, Distance Education
		Bachelor of Software Engineering	Casuarina, Distance Education
Charles Sturt University	Faculty of Business	Bachelor of Computer Science (Majors: Game Programming; Computer Graphics)	Bathurst
		Bachelor of Computing Studies (Business)	Albury-Wodonga; Bathurst; Wagga-Wagga;
		Bachelor of Information Technology (Majors: Software Design & Development; Network Engineering; Systems Administration; Online Systems; IT Management; Systems Analyst	Albury-Wodonga; Bathurst; CSU Study Centre Melbourne; CSU Study Centre Sydney; Wagga-Wagga; DE Albury-Wodonga
Curtin University of	Science and Engineering	Bachelor of Science (Computing) (Streams: Information Technology; Software	Bentley

Technology		Engineering; Computer Science; Cyber Security)	
		Bachelor of Engineering (Computer Systems Engineering), Bachelor of Science (Computer Science)	Bentley Campus
		Bachelor of Engineering (Software Engineering)	Bentley, Sarawak
		Bachelor of Technology (Computing Systems and Networking)	Bentley Campus, Miri Sarawak, Sri Lanka Institute of Technology
	Curtin Business School	Bachelor of Commerce (Business Information Technology)	Bentley, Mauritius
Deakin University	Faculty of Science, Engineering and Built Environment (School of Information Technology)	Bachelor of Information Systems/Bachelor of Information Technology	Melbourne Burwood Campus
		Bachelor of Information Technology (Majors: Computer Science; Interactive Media Design; Game Development; Networking; Security; Software Development; Mathematical Modelling)	Melbourne Burwood Campus, Geelong Waurn Ponds Campus, Off Campus
		Bachelor of Information Technology (Computer Science and Software Development)	Melbourne Burwood Campus, Geelong Waurn Ponds Campus
		Bachelor of Information Technology (Games Design and Development)	Melbourne Burwood Campus, Geelong Waurn Ponds Campus
		Bachelor of Information Technology (Honours)	Melbourne Burwood Campus, Geelong Waurn Ponds Campus
		Bachelor of Information Technology (IT Security)	Melbourne Burwood Campus, Geelong Waurn Ponds Campus
		Bachelor of Information Technology (Mobile and Apps Development)	Melbourne Burwood Campus, Geelong Waurn Ponds Campus
		Bachelor of Information Technology (Professional Practice)	Melbourne Burwood Campus, Geelong Waurn Ponds Campus
		Bachelor of Information Systems	Melbourne Burwood Campus, Geelong Waurn Ponds Campus
Edith Cowan University	Faculty of Health, Engineering and Science	Bachelor of Computer Science (Majors: Computer Science; Games Programming: Software Engineering; Smart Software Systems; Mobile Application Development)	Joondalup; Mt Lawley
		Bachelor of Engineering (Computer Systems)	Joondalup
		Bachelor of Engineering (Computer Science) Bachelor of Computer Science	Joondalup
		Bachelor of Science (Cyber Security)	Joondalup; Mt Lawley
Flinders University	Faculty of Science and Engineering	Bachelor of Computer Science	Adelaide
		Bachelor of Information Technology	Adelaide
		Bachelor of Information Technology (Digital Media)	Adelaide

		Bachelor of Engineering (Computer Systems)	Adelaide
		Bachelor of Engineering (Software)	Adelaide
Griffith University	Science, Environment, Engineering and Technology (Faculties are called Groups)	Bachelor of Information Technology	Gold Coast, Nathan, Logan
		Bachelor of Multimedia	Gold Coast, Nathan
		Bachelor of Engineering (Advanced Studies)-Software Engineering Major	Nathan
		Bachelor of Engineering - Software Engineering Major	Nathan
James Cook University	Law, Business and Creative Arts	Bachelor of Information Technology (Majors: Computing and Networking; Interactive Technologies and Games Design)	Beijing, Brisbane, Cairns, Singapore, Townsville
		Computing and Networking Major	
		Interactive Technologies and Games Design Major	
	Science and Engineering	Bachelor of Engineering /Bachelor of Information Technology	Townsville, Cairns
		Bachelor of Engineering (Computer Systems Engineering)	Townsville, Cairns
La Trobe University	Science, Technology and Engineering	Bachelor of Computer Systems Engineering	Melbourne
		Bachelor of Computer Science	Melbourne
		Bachelor of Information Technology	Bendigo
		Bachelor of Computer Science in Games Technology	Melbourne
		Bachelor of Business Information Systems	Melbourne
Macquarie University	Faculty of Science	Bachelor of Information Technology (Majors: Business Information Systems; Software Technology; Web Design and Development)	North Ryde
		Bachelor of Information Technology -Games Design and Development	North Ryde
		Bachelor of Engineering (Major: Computer Engineering; Software engineering)	North Ryde
Monash University	Information Technology	Bachelor of Business Information Systems	Clayton
		Bachelor of Computer Science	Clayton
		Bachelor of Informatics and Computation Advanced (Honours)	Clayton
		Bachelor of Information Technology and Systems (Majors: Applications development; Enterprise information management; Games development; Information and communication technologies; Multimedia development)	Caulfield
		Bachelor of Software Engineering	Clayton
	Engineering	Bachelor of Computer Systems Engineering (Honours)	Clayton
Murdoch University	School of Engineering and Information Technology	Industrial Computer Systems Engineering (B Eng)	Murdoch campus (internal)
		Engineering Technology (B Eng)	Murdoch campus (internal)
		Computer Science (BSc)	Murdoch campus (internal)
		Business Information Systems (BSc)	Murdoch campus (internal and external).
		Internet Software Development (BSc)	Murdoch campus (internal and external)
		Games Software Design and Production (BSc)	Murdoch campus (internal and external)
		Games Technology (BSc)	Murdoch campus (internal and external)
		Internetworking (BSc)	Murdoch campus (internal)

		Cyber Forensics and Information Security (BSc)	Murdoch campus (internal and external).
Queensland University of Technology	Science and Engineering	Bachelor of Engineering (Computer and Software Systems)	Gardens Point
		Bachelor of Games and Interactive Entertainment (Majors: Animation; Game Design; Software Technologies)	Gardens Point
		Bachelor of Information Technology (Computer Science	Gardens Point
		Bachelor of Information Technology (Information Systems)	Gardens Point
RMIT University	Engineering	Bachelor of Engineering (Computer and Network Engineering (Honours)	City Campus
	Computing and Information Technology	Bachelor of Computer Science	City Campus
		Bachelor of Computer Science (Majors: Embedded Systems; Security; Web Systems; Computational Mathematics; Application Programming; Games, Graphics and Digital Media)	City Campus
		Bachelor of Technology (Computing Studies)	City Campus
		Bachelor of Technology (Games and Graphics Programming)	City Campus
		Bachelor of Information Technology (Majors: Application Programming; Business applications; Multimedia design; Network Programming; System Administration; Web Systems)	City Campus
		Bachelor of Software Engineering	City Campus
Southern Cross University	School of Environment, Science and Engineering)	Bachelor of Science (Major: Information Technology)	Information Technology Units available by Distance Education only
	Southern Cross Business School	Bachelor of Applied Computing	Lismore, Coffs Harbour, Gold Coast, Distance Education
		Bachelor of Information Technology (Majors: Information Systems; Software Development; Interactive Multimedia)	Lismore, Coffs Harbour, Gold Coast, Distance Education
Swinburne University of Technology	Faculty of Information and Communication Technologies	Bachelor of Business Information Systems	Hawthorn
		Bachelor of Information Technology	Hawthorn
		Bachelor of Information and Communication Technology (Majors: Software Technology; Business Systems; Business Analysis; Network Technology; Games Technology)	Hawthorn
		Bachelor of Information and Communication Technology (Network Design and Security)	Hawthorn

		Bachelor of Applied Information and Communication Technology	Hawthorn
		Bachelor of Science (Games Development)	Hawthorn
		Bachelor of Computer Science	Hawthorn
		Bachelor of Science (Computer Science and Software Engineering)	Hawthorn
University of Adelaide	Faculty of Engineering, Computer and Mathematical Sciences	Bachelor of Computer Science	North Terrace Campus
		Bachelor of Engineering (Software)	North Terrace Campus
		Bachelor of Mathematical and Computer Sciences	North Terrace Campus
Federation University Australia	School of Information Technology (Gippsland) / School of Science, Information Technology and Engineering (Ballarat)	Bachelor of Information Technology	Mt Helen Campus (Ballarat), Online Learning, Gippsland, MIT (Sydney), MIT (Melbourne)
		Bachelor of Information Technology (Business Systems)	Mt Helen Campus (Ballarat), Online Learning, Gippsland, MIT (Melbourne)
	School of Science, Information Technology and Engineering (Ballarat)	Bachelor of Information Technology (Computer Games and Digital Media)	Mt Helen Campus (Ballarat),MIT (Sydney),MIT (Melbourne)
		Bachelor of Information Technology (Professional Practice)	Mt Helen Campus (Ballarat)
University of Canberra	Faculty of Education, Science, Technology & Maths	Bachelor of Information Technology in Mainframe Computing	UC Bruce Campus
		Bachelor of Software Engineering	UC Bruce Campus
		Bachelor of Engineering in Network and Software Engineering	UC Bruce Campus
University of Melbourne	Melbourne School of Information	Bachelor of Science (Majors: Informatics, Computing and Software Systems)	Parkville campus
University of New England	School of Science and Technology	Bachelor of Computer Science	Armidale
University of New South Wales	Faculty of Engineering	Bachelor of Engineering (Majors: Bioinformatics, Computer Engineering, Software Engineering)	Kensington Campus
		Bachelor of Science (Computer Science)	Kensington Campus
University of Newcastle	Faculty of Engineering and Built Environment / Faculty of Science and Information Technology	Bachelor of Computer Science	Newcastle (Callaghan)
	Faculty of Engineering and Built Environment	Bachelor of Engineering (Computer)	Newcastle (Callaghan)
	Faculty of Science and Information Technology	Bachelor of Information Technology	Newcastle (Callaghan), Singapore, Central Coast (Ourimbah)
University of Notre Dame		*No computing/Information Technology found*	
University of Queensland	Engineering, Architecture & Information Technology	Bachelor of Information Technology	St Lucia

University of South Australia	School of Information Technology & Mathematical Sciences	Bachelor of Business (Management of Information Technology)	
		Bachelor of Computing (Multimedia)	Magill
		Bachelor of Information Technology (Majors: Business Systems; Cloud Computing; Games and Entertainment Design; Networking and Security; Software Development; Systems Administration)	Mawson Lakes
		Bachelor of Software Engineering	Mawson Lakes
University of Southern Queensland	School of Sciences	Bachelor of Science (Majors: Computing; Information Technology)	Toowoomba
		Bachelor of Information Technology	Toowoomba
University of the Sunshine Coast	Faculty of Arts and Business	Bachelor of Information and Communications Technology	Sippy Downs
University of Sydney	Faculty of Engineering and Information Technologies	Bachelor of Computer Science and Technology	Darlington Campus
		Bachelor of Information Technology	Darlington Campus
University of Tasmania	Faculty of Science, Engineering & Technology	Bachelor of Computing	Hobart, Launceston
		Bachelor of Information Systems	Hobart, Launceston
		Bachelor of Information and Communication Technology	Hobart, Launceston
University of Technology Sydney	Faculty of Engineering and Information Technology	Bachelor of Science in Games Development	City campus
		Bachelor of Science in Information Technology	City campus
University of Western Australia	Faculty of Engineering, Computing and Mathematics	Bachelor of Science (Majors: Computer Science; Applied Computing)	Crawley
University of Western Sydney	Engineering, Information and Communications Technology (ICT)	Bachelor of Computer Science	Penrith
		Bachelor of Information and Communications Technology (ICT)	Penrith, Campbelltown, Parramatta
		Bachelor of Information Systems	Parramatta
University of Wollongong	School of Computer Science and Software Engineering	Bachelor of Computer Science	Wollongong
Victoria University	College of Engineering and Science	Bachelor of Information Technology (Network and Systems Computing)	Footscray Park

Appendix B – First-year ICT Course Structures and Units

Summary

- Faculties that offer ICT degrees are predominantly Information Technology, Science, Engineering, and Business
- There are very few dedicated ICT faculties
- Degrees are typically one of the following:
    - general ICT
    - ICT with a major or specialization, including
        - game programming
        - software/application development (including mobile)
        - security
        - networks
        - web design and development
        - multimedia
    - software engineering
    - computer science
    - business information systems
- All but two Australian universities offer an ICT or related degree
- While most are located in capital cities, there are a significant number of courses offered in more rural locations. A number are also offered in off-campus mode.
- General ICT courses, most with majors, make up the majority of courses. Computer science ranks second, software engineering third, and information systems / business information systems fourth. There are also a number of miscellaneous ICT courses that focus on other specialist areas such as multimedia, game development, cybersecurity and engineering.

First-year units

- Units studied in first year depend on the type of course being taken
- Common units include:
    - programming
    - database
    - systems analysis
    - computing fundamentals
    - mathematics (predominantly in computer science)

Appendix C – Summary of ICT Courses from Interviews

Uni	Faculty	Students	Demographic	Software	Curriculum
ACU	Law & Business	600	Not sure	?	IEEE based ACS accred
Adelaide	Tech	70 in courses, 150 in units	20% OS No DE	Python, C++	ACM ACS accred
Charles Darwin	Tech	100	60% OS	HTML, Java	ACS and ACM
Charles Sturt	Business	150	50-50 local/os	Python	Not sure
Deakin	Tech	?	30% OS 10% DE	OO	ACS accred
Edith Cowan	Tech	200	Most full time 25 off campus Significant OS	Java, C	ACM/IEEE ACS accred
Fed U	Science	100?	Mostly local full time OS at Gipps	Java MySQL	ACS / IEEE
Flinders	Tech	250	Mostly local	HTML, Java, Assembly	ACS accred Eng Aus accred ACM/IEEE
James Cook	Bus/ Humanities	90	Mostly local full time	Python, Java	ACS accred
LaTrobe	Tech	30 (Bendigo)	Vast majority full time and local	Java, VB	ACM mostly
Macquarie	Science	400 (inc outside degrees)	Not sure	Processing, Java	ACM/IEEE informed
Melbourne	No faculty	300	50/50 local/OS	Python, HTML, C	ACM
Monash	IT	400	40% OS Low DE	Scribble, Java, C++, Flash, Python	ACS accred Eng Aus accred
Murdoch	Tech	160	Most local, some OS 10% DE	C, Java	ACM/IEEE CISCO ACS Accred
Newcastle	Science/IT	160	No DE Often about 30% OS	Python, Java, Gamemaker	ACM/IEE informed ACS accred
QUT	Tech	450	5% OS No DE	Python, C#	ACS accred
RMIT	Science	250	50-60% OS No DE	Java, Python	ACM/IEEE
Swinburne	Bus & Tech	300	Mostly full time Combo of local and OS	IS – VB & SQL Eng – C, Pascal CS – Pascal, C, C#, Java, Obj C, C++	Influenced by ACM/IEEE/ACS ACS accred

Southern Cross	Business	100	60% DE 30$ OS	VB, C#, HTML, Javascript	ACS accred
Sydney	Eng/IT	400	30% OS	Java, Python, R	ACS / ACM Eng Aus accres
UQ	Eng/IT & Science	150	Mostly local No DE	Python, HTML, Javascript, SQL	ACM/IEE ACS accred
Uni SA	IT & Math	250	70% local, 30% OS 15-20% DE 40% part time	Python, Java	ACM ACS accred
USQ	Bus/Arts	200+	Large cohort DE 80% local	None in 1st yr Java in 2nd	ACS accred
UWS	Tech	100+ 300?	87% local 25% DE	Java IDEs	ACM ACS accred
UWA	?	100	No DE	Java, Python, SQL	ACM informed ACS accred Eng Aus accred

Table 10: Summary of Interview Findings.

Appendix D – Interview Questions

Demographics

- What undergraduate computing degree(s) do you offer?
- In which faculty? Or are they multi-faculty?
- How big is the first-year cohort? (Australian campuses)
- What's the demographic profile of the students (overseas / domestic / distance / full-time / part time)?

What we teach

- What ICT courses/subjects/units are offered to first-year students? Briefly describe the content of each course.
- What programming languages are taught? What other software packages are taught?
- Is the content of these courses based on some external curriculum, such as the ACM/IEEE curriculum, or more on your group's own design?

Where we teach

- Describe your teaching spaces.
- In addition to physical teaching spaces, what teaching do you do in blended or online environments?
- Have you made any changes recently (in the past 5 years)? What? Why? Has it worked? How do you know (evaluation)?

How we teach

- Do you use a Learning Management System such as Blackboard or Moodle? Which system do you use?
- Do you use any 'novel' teaching practices, such as peer instruction, flipped classroom, students contributing to the learning of others, e.g. through Peerwise, student seminars, etc?
- Have teaching practices changed in the past few years?
- Have you started using new tools, technologies, programming languages, or approaches?
- How much of your first-year teaching is done by casual staff?
- Do you have programs for casual staff to enhance their contribution to student learning?
- What measures are taken to ensure the quality of the student experience?

How we assess

- What kinds of assessment items are used in the first-year courses?
- For which assessment items is feedback given to students?
- How much of the assessment is more or less guaranteed to be the work of the individual (eg moderated tests and exams?)
- How much of the assessment is assessed automatically?
- For work not done in test conditions, what techniques are used to verify that the work is the student's own work? E.g interviews for assignments.

How we strengthen the learning environment

- What use is made of social media in first-year teaching?
- How are the students educated about study skills?

- How are the students educated about academic integrity in the context of ICT courses?
- Are students ever penalised for breaches of academic integrity? What would a typical penalty be?
- Do you take any special measures to try to encourage student engagement?

How we support our students

- Do you have any special intervention programs for at-risk students? For women? For international students? For other identifiable groups?

In case we've missed anything

- What else is happening at your institution that affects the first-year experience in ICT?