Understanding the Teaching Context of First Year ICT Education in Australia

Matthew Butler
Monash University
Australia
matthew.butler@monash.edu

Judy Sheard
Monash University
Australia
judy.sheard@monash.edu

Michael Morgan
Monash University
Australia
michael.morgan@monash.edu

Katrina Falkner
University of Adelaide
Australia
katrina.falkner@adelaide.edu.au

Simon
University of Newcastle
Australia
simon@newcastle.edu.au

Amali Weerasinghe
University of Adelaide
Australia
amali.weerasinghe@adelaide.edu.au

Abstract

This paper reports on an investigation of the teaching context of first-year Information and Communications Technology (ICT) courses at Australian universities and the influences of this on students' learning experiences. This is part of a larger project which aimed to identify and disseminate good practices in ICT teaching at Australian universities with a specific focus on the first-year experience. We conducted a systematic review of the research literature from the previous five years and an online search of information on existing courses and content, and interviewed 30 academics concerned with design and delivery of the first-year learning experience in 25 Australian universities. From our study of teaching context we gained a comprehensive view of the current curricula, teaching models and teaching spaces and were able to outline the unique challenges that our first-year ICT students face and to recommend areas for further investigation.

Keywords: First Year; Student Experience; Curriculum; Learning Spaces.

1 Introduction

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. The volume of the literature concerned with specific ICT education 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.

In this paper we report findings of a study that investigated the teaching context in first year Information and Communications Technology (ICT) courses in Australia. The study comprised a review of recent literature on what content is taught in ICT courses, the teaching delivery models used and where the teaching takes place; a survey of Australian university websites; and interviews of Australian academics involved in teaching first year ICT courses. The aims of the study were: 1) to gain an overview of what is taught in first year ICT courses in Australia; 2) to gain understanding of the teaching delivery models used; 3) to gain understanding of where teaching is conducted; and 4) to identify examples of good practice in first year ICT courses in Australia that could be adopted and disseminated widely. This study is part of a larger project of teaching practices in first year ICT courses.

2 Research Approach

This section describes the approach used to investigate the teaching context in the first year of ICT courses. The investigation was part of a project that investigated the broader topic of research and practice in ICT courses in Australia. To conduct the project, the team developed a framework with six themes that together describe the learning experience: "what we teach", "where we teach", "how we teach", "how we assess", "learning support" and "student support". As the focus of this paper is about teaching context, only findings from the "what we teach" and "where we teach" themes will be reported.

The project was conducted in two phases:

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: Information about ICT courses in Australia was gathered from a survey of university websites. In addition, extensive interviews were conducted with 30 first-year ICT academics from 25 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.

2.1 Literature Review

In order to identify current research 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. 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 and 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.

2.2 Survey of current practice

A survey of current practice was conducted via a survey of online information on ICT courses at all Australian universities and interviews of relevant academics. 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.

Participants were selected from each participating university in Australia that delivered an ICT course. Project members nominated relevant people at various universities from their knowledge of the ICT education community. Where this could not be done, the contact details listed on faculty and degree websites were used to initiate e-mail contact. Thirty academics from twenty-five Australian Universities were interviewed. These included six Group of Eight (Go8), three Australian Technology Network (ATN), six Innovative Research (IRU) universities and three Regional University Network (RUN).

The interview script was designed by the project team using the six project themes as a framework. The script consisted of a number of semi-structured questions. The questions related to this paper can be found in the Appendix. 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. 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. All interviews were conducted by telephone 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.

Twenty-nine interviews (one interview involved two participants) 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. The notes were annotated with 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. A more detailed description of the methodology used in this project can be found at *Experiences of first year students in ICT courses: good teaching practices*: Final Report: ICT student first year experiences (http://www.acdict.edu.au/ALTA.htm).

The following section reports the results of our investigation into teaching context. We first describe the curricula and curriculum designs of first year ICT courses drawing upon the data gathered from the "what we teach" theme. Following is an investigation of teaching models and teaching spaces drawn from the "where we teach" theme. These themes cover the broad area of the teaching context.

3 What we teach

Our investigation of what we teach 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).

3.1 ICT Courses in Australia

A survey of ICT courses in Australian universities found that 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. 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.

3.2 Literature Perspectives

In the literature search 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, Teague & Thomas, 2010; Mason, Cooper & de Raadt, 2012; Mason & Cooper, 2014) and all three of these papers relate specifically to programming.

Approximately half the papers found discuss high-level curriculum design issues. 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 on curriculum 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, Jones & Miller, 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 most represented context, with 11 papers relating specifically to curriculum issues within this area of study. Mason, Cooper & de Raadt (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, Teague & Thomas, 2010; Hu, Winikoff & Cranefield, 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, Cordiner & Corney, 2010). Other sub-themes that were found in the literature relating to curriculum include investigation of gender issues (Koppi, Roberts & Naghdy, 2012) and career progression and its implications for curriculum design (von Konsky, Jones & Miller, 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.

3.3 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.

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 the internationalisation of the teaching curriculum, not only because of these demographic estimates, but also because of the international nature of ICT.

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.

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 Java 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. While a number of interviewees discussed their shift to Python, 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.

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. In terms of research, a formal evaluation and comparison of the range of approaches currently being trialled 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)

3.4 Summary

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.

The study of curricula and curriculum design provides a background for our investigation of teaching context in terms of teaching models and teaching spaces.

4 Teaching Context

Our investigation of teaching context was drawn from the 'where we teach' theme which focused on the teaching models and 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.

4.1 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. 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, Meltzoff & Kim (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, Carbone & Sheard (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, Hamilton & D'Souza (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 De 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, Davis & Anderson (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 compared 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.

4.2 Current Teaching Context in Australia

The interview questions related to the teaching context 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.

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.

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*."

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.

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 Echo360; 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.

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).

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.

4.3 Discussion

The aims of the study were: 1) to gain an overview of what is taught in first year ICT courses in Australia; 2) to gain understanding of the teaching delivery models used; 3) to gain understanding of where teaching is conducted; and 4) to identify examples of good practice in first year ICT courses in Australia that could be adopted and disseminated widely. A key finding from our investigation of what is taught in first years ICT courses was that there is 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 universities of the Australian academics 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.

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.

Further scope for 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 of exactly how course curriculums are developed with these frameworks in mind. 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.

Our investigation of the literature on teaching context found 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 and the changing nature of the layout of computing laboratories. A prominent topic raised by interviewees was the design and use of teaching spaces to engage students in active learning experiences. The layout of physical teaching spaces was reported to be increasingly diverse and flexible. Various new physical and virtual learning environments are tailored to the needs of first-year ICT students. Further research is needed to assess the impact of these changes to the teaching environment on student performance and on the student experience.

There were strong indications from the interviewees that 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.

5 Conclusion

Our investigation of teaching context in first-year ICT courses in Australia has highlighted many new initiatives in teaching delivery models and the design of teaching spaces, driven largely by a desire to provide interesting learning environments and active learning experiences. The research has identified the need to undertake further research investigating such areas as curriculum design, development of graduate attributes, and understanding the needs of the ICT industry. An imperative now is also to assess the effectiveness of the innovations identified in engaging students and enhancing their learning. Evidence from such evaluations is essential for promotion of these innovations and driving change in the ICT teaching sector.

6 Acknowledgements

This project was undertaken with the support of the Australian Council of Deans of Information and Communication Technology through the ALTA Good Practice Reports Commissioned for 2013–2014 grant scheme (http://www.acdict.edu.au/ALTA.htm).

The project team would like to acknowledge the work of Beth Cook who worked as a research assistant to conduct the interviews and to prepare the detailed interview notes.

7 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.

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

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.

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

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

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.

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

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

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.

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

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.

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

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.

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.

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.

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.

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

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.

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.

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.

8 Appendix

Below are the indicative interview questions used to capture current practice regarding student demographics, curriculum, and teaching spaces:

Demographics

- What undergraduate computing degree(s) do you offer?
- In which faculty? Or are they multi-faculty?
- How big is the first-year cohort? (We agreed that we were talking principally here about Australian campuses, though some respondents with overseas offerings might also mention those.)
- 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)?