Teaching in First-Year ICT Education in Australia: Research and Practice

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

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

Matthew Butler
Monash University
Australia
matthew.butler@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 details current research and teaching practice for first-year Information and Communications Technology (ICT) students at Australian universities. The project aims to record and disseminate good practice in first-year ICT teaching in Australia. The aim of the paper is to examine how academics are addressing the challenge of engaging first-year ICT students in the learning process. Two sources of data are used, a systematic survey of research literature from the last five years and detailed interviews of 30 academics involved in first-year teaching duties. Academics from 25 Australian universities represented a range of universities, including six from the Go8 group, three from the ATN group, and five from the IRU group. The paper highlights current areas of research, any gaps in the research literature, examples of current good teaching practices, and recommendations for further research.

Keywords: First Year; Student Experience; Teaching.

1 Introduction

This paper presents a survey of current research literature and current practice in Australian universities for the teaching of first-year ICT students. It is motivated by the unique challenges facing ICT educators as they design and deliver educational experiences for first-year students in the ICT domain. The challenges faced by ICT students in the transition from secondary education are evidenced by the relatively high rate of attrition in ICT courses, a reduced engagement in on-campus learning experiences and a perceived lack of relevance to some potential student groups (Sheard, Carbone, & Hurst, 2010). In a search of the literature we found few examples that addressed these issues in the ICT context and in the Australian setting. While a lot of worthwhile research is being conducted into specific teaching practices in specific contexts, there is a need to properly collate and review this research in order to drive change in practice more broadly across the Australian Higher Education sector.

To investigate current research and practices in first-year ICT courses in the Australian context, the authors investigated six broad themes that together describe the learning experience: "what we teach", "where we teach", "how we teach", "how we assess", "learning support" and "student support". Only the "how we teach" theme is presented in this paper due to space considerations. Within this theme the different aspects of teaching are discussed in relation to issues such as student engagement, student retention, learning outcomes and broadening the relevance of ICT courses to a wider range of students.

2 Research Approach

The research team (the authors of this paper) designed two phases for this project: a review of research literature from the last five years, and interviews of academics involved in the delivery of first-year programs to survey current practice. A detailed description of the methodology used in this project is reported in *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). Accordingly, a brief summary is presented below, with focus placed on the "how we teach" theme.

In phase 1 a systematic review was conducted of the literature from 2009 to 2014 in the area of computing education. Keyword searches were carried out in Google Scholar and the IEEE Xplore and ACM Digital Library databases, along with manual searches of key computing education journals and conference proceedings.

In phase 2, semi-structured phone interviews were conducted with academics from Australian universities between February and March 2014. Participants were identified as key staff involved with the design and/or delivery of ICT courses to first-year students. Thirty academics from 25 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 interviews averaged 53 minutes, with detailed notes being taken. They were audio recorded so that relevant comments could be transcribed at a later time. The interview script

focused on six key themes and all interviewees were sent the interview questions before the interview. Questions were devised to elicit responses about initiatives in teaching practice; for example, "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?". Follow up questions on specific issues were also asked where appropriate.

3 How we teach

The investigation of teaching in first-year ICT courses in Australian universities 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. We begin our investigation of teaching in first-year ICT courses with a review of the literature. This gives a broad perspective of assessment in first-year ICT courses during the past five years, highlighting Australian studies. Following this, an analysis of our interviews of academics provides insights into teaching practices in Australian courses.

3.1 Literature Perspectives in ICT Teaching Practice

The systematic literature review identified 57 papers that were considered relevant to the theme of "how we teach", grouped into four main topics:

1. theories and models of teaching and learning
2. approaches to teaching
3. cooperative and collaborative learning
4. social media and learning communities

All papers were set in the higher education sector and in the ICT discipline. Most papers were focused on teaching in first-year courses. Fifty papers (88%) dealt with teaching programming, particularly introductory programming. Eleven were Australian studies.

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 examples 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 inherently programmers or have great difficulty picking up programming fundamentals. 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 transferring their knowledge of programming concepts learnt in Alice 3 to the Java context. Using this approach they found dramatic improvement in students' final exam 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' mental 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.

Approaches to teaching

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

1. 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, Lattu & Pasanen (2012) present CSLE (Creative-Supporting Learning Environment), a theoretical framework for designing a course 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, Winikoff & Cranefield (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' performance in their programming exam.

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. 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.

2. Games

Game-based learning and assessment tasks are often used to motivate and engage students in the learning process. Eagle and Barnes (2009) and Morazán (2010) describe their use of games in introductory programming courses. They report findings of studies that show that learning activities based on games are useful tools to interest and enthuse students in programming. However a study of the use of mobile games by Kurkovsky (2013) found mixed results in terms of student engagement and motivation.

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).

3. Tools and technologies

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. Apiola, Lattu & Pasanen (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 powerful tools 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 to clarify and explain concepts.

Sorva, Karavirta & Malmi (2013) reviewed 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 Heinsen Egan and McDonald (2014) describe systems for visualising runtime memory state and their integration into the SeeC 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 outside the 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.

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). The concept 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). 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 has been used as one of the primary methods of implementing CSP as it requires learners to externalise their understanding in order to work with their peers.

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 techniques. Murphy et al. claim that their PLTL 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, then 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 requiring students to work 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 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, Agrawal & Agrawal (2013), involved peer-reviewing code with the help of a 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, Teague & Thomas (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 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, Walia & Rummelt (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, Stewart & Ferati (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, Stewart & Ferati found that students were less

satisfied with the experience of online pair programming than when co-located.

O'Grady (2012) reviewed the literature on the use of problem-based learning (PBL). More than a third of the 59 cases reviewed 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, Fuentes-Fernández & Fernández-Manjón (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.

Social media and learning communities

Recently, various forms of social media (web 2.0) have been used in education programs to encourage collaborative work and the formation of learning communities. Using social media 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, Brindley & Eccleston (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, Richardson & Hamilton (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.

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. These were frequently discussed in terms of influences on student learning, motivation, and engagement.

A large proportion of this material was highly focused on the programming domain and only a small portion related specifically to the Australian context.

3.2 Current Practice in Australia

The interviews of Australian academics sought information about teaching practices in first-year ICT courses. The responses gave insights into current teaching practices and issues faced by teaching staff. Thematic analysis was used to extract and code the responses and to identify and define the major issues raised. The responses are discussed below under the main topics that were identified from the analysis of the interview data: "approaches to teaching", "cooperative and collaborative learning" and "social media and learning communities". These broadly align with three of the four topics from the literature search. 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.

Approaches to teaching

A common element in this topic was the aim of increasing 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. The most dominant concerns regarding teaching were the issues involving 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, ... 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. Recently, lecture techniques and pedagogies have been developing to take advantage of these strengths.

One example of this process is the technique of the flipped classroom (Porter, Bailey-Lee & Simon, 2013; Simon et al., 2013) incorporating the use of clicker technologies. Interviewees U15a and U15b described the use of flipped classroom techniques and clicker technology specifically targeted at first-year students:

"Clickers were implemented...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, ... 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."

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.

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

"What we found, 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)

"It encourages them to actually attend. We're starting to have more units using clickers this semester." (U15a)

However, other interviewees indicated that they had implemented some components of the flipped classroom model but that it had proved problematic to motivate students to do the required pre-reading, so the approach was discarded. 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 these concepts seem to find that really helps".

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 many efforts to improve the lecture experience, some interviewees expressed strong negative views about it. 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 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."

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 for these was to engage students in interesting and meaningful 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".

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."

Interviewee U10 describes the media computation introductory programming technique where students learn to program using the manipulation of images and sounds as the context for learning about programming.

"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 wider audience appeal, including for non-ICT students. 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 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.

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 was voluntary.

Another interviewee, U15a, described an e-publishing initiative called Alexandria, based on WordPress infrastructure. The aim was to create dynamic and interactive learning objects that could be distributed on a variety of platforms. This is a type of e-publishing with interactive elements embedded, such as quizzes, applets and discussion forums.

"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."

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. ... it is very much based upon peer assessment and peer review, peers working together in collaboration. 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 students are 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. 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 ... 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."

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. Interviewee U9 provides an example:

"We have got peer collaboration within classes and some topics 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 ... 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."

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. 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.

Social media and learning communities

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. 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."

UCROO has 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 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 [...] between students on UCROO each time I have used it. ... 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 and web-based clicker systems for in-class polling.

However, several interviewees cautioned against the use of social media. As U4 explains:

"It is difficult to encourage students to use it because they think this is just another burden on what they're required to do."

Interviewee U7b 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."

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 is required on the impact of social networking techniques on the ICT domain.

4 Discussion

Our analysis of recent literature shows that while there is a significant body of literature devoted to teaching in the first year of ICT courses, much of this literature is focused in the programming context. We propose that further research is needed to explore other aspects of the first-year ICT curriculum to gain a better understanding of the first-year ICT student experience.

The topics that emerged from an analysis of the interview data broadly align with those found 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, along with the integration of social networking tools to support the formation of learning communities. 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 described in this paper. 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, using 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.

5 Conclusion and future work

From this study we have documented a number of initiatives aimed at increasing ICT student engagement in the learning process. The study raises a number of key research areas that need further investigation. 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. While initial results in many cases are positive, more evidence is required to justify sector-wide change. The amount of published literature on programming education also highlights a need to conduct research in other areas of ICT curriculum, to ensure a better overall first-year experience for ICT students.

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 Dr Beth Cook who worked as a research assistant to conduct the interviews and to prepare the detailed interview notes.

7 References

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

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.

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.

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

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.

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

Collis, B. and Moonen, J. (2005). An On-Going Journey: Technology as a Learning Workbench, University of Twente, Enschede, The Netherlands.

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.

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.

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, 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.

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

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.

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., Simon, B., McCauley, R., & Zander, C. (2009). CS1 students speak: advice for students by students. *ACM SIGCSE Bulletin*, 41(1), 19-23.

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

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

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

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.

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

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.

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.

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.

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.

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

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.

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

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.

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.

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

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).

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.

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.

Sheard, J., Carbone, A., & Hurst, A. J. (2010). Student engagement in first year of an ICT degree: staff and student perceptions. *Computer Science Education*, 20(1), 1-16.

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.

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.

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.

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.

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

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.

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.