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

Judy Sheard
Monash University
Australia
michael.morgan@monash.edu

Michael Morgan
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 presents an investigation of assessment in first-year Information and Communications Technology (ICT) courses with a focus on Australian universities. This study was part of a project that aimed to identify and disseminate good practices in first-year ICT teaching in Australian universities. Through a systematic review of the last five years of research literature and interviewing 30 academics who were involved in the design and delivery of the first-year learning experience in Australian universities, we have formed a comprehensive view of current assessment practices, and outlined the unique challenges faced by teachers when designing assessment for their first-year ICT students. Key findings of the literature survey and the insights gained from the academic participants have been collated to provide examples of good practice in the field and to recommend areas for further investigation.

Keywords: First Year; Student Experience; Assessment; Academic Integrity.

1 Introduction

Assessment is a key component of the learning experience of university students. Assessment is used to measure the level of knowledge and skills that students have obtained, and determines their grades and course progression. Assessment can be used during the learning process to give students feedback on their work. An important consideration is that the form of assessment influences how students approach their study, with a consequent influence on learning outcomes (Biggs, 1996).

There are a variety of ways that students may be assessed, and the form of assessment used is often discipline-specific. For example, students learning to program may be assessed by a practical task on a computer. With recent moves to blended learning and technology-enhanced learning environments there are now new imperatives and opportunities for different forms of assessment.

Considering the central role of assessment in the student learning experience, it is critical that teachers choose the form of assessment that is appropriate for the learning situation and desired learning outcomes. In first-year courses it is also important to consider that students may not have encountered some forms of assessment in their previous education. The transition from secondary to tertiary studies is a difficult process for many students, first-year courses have high rates of attrition (Sheard, Carbone, & Hurst, 2010) and it is important to consider any possible influences on this experience.

In this paper we report findings of a study that investigated assessment practices in first-year Information and Communications Technology (ICT) courses in Australia. The study comprised a review of recent literature on assessment practices in ICT courses and a survey of Australian academics involved in teaching first-year ICT courses. The aims of the study were: 1) to gain a comprehensive view of how students in first-year ICT courses are assessed; 2) to determine factors influencing choice of assessment used; and 3) to identify examples of good practice in assessment in first-year ICT courses in Australia that could be adopted and disseminated widely. This study is part of a larger project exploring teaching practices in first-year ICT courses.

2 Research Approach

This section describes the approach used to investigate research and current practices in assessment in the first year of ICT courses in Australia. The investigation was conducted by the authors as part of a project that investigated the broader topic of research and practice in teaching 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 assessment, only findings from the ‘how we assess’ theme will be reported.

Two phases were designed by the authors for this project; a systematic review of research literature from the previous five years, and interviews of academics involved in the delivery of first-year programs in

Australia. 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, only a brief summary is presented below, focusing on the 'how we assess' theme.

Phase 1 of the project consisted of a systematic review of 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 in 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), and three Regional Universities Network (RUN) universities. The interviews averaged 53 minutes. Detailed notes were taken, and the interviews 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 being interviewed. Questions asked to elicit responses about initiatives in assessment practice included: 'What kinds of assessment items are used in the first-year courses?', 'For which assessment items is feedback given to students?', 'How much of the assessment is assessed automatically?', and 'For work not done in test conditions, what techniques are used to verify that the work is the student's own work?'. Follow-up questions on specific issues related to the themes were asked where appropriate.

3 How we assess

The investigation of assessment in first-year ICT courses in Australian universities covered the areas of assessment strategies, summative and formative forms of assessment, and tools to assess student learning or to facilitate the marking process. We begin our investigation of assessment 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 the interviews of academics provides insights into assessment practices in Australian courses.

3.1 Literature Perspectives on Assessment in ICT

The systematic literature review found 38 papers that were concerned with assessment in university ICT courses during the previous five years. The literature on assessment was grouped into five topics:

- assessment design and strategies
- exam assessment
- non-exam assessment
- automated assessment
- assessment instruments and tools

All papers were set in the higher education sector. A high number of papers (27, 79%) dealt with assessment in first-year courses or assessment that was applicable to the first year. Most papers (33, 87%) dealt with issues concerning assessment of programming, and almost half (18, 47%) were Australian studies.

Assessment design and strategies

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

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

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

Exam assessment

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

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

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

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

Another group of Australian researchers investigated summative assessment of introductory programming, focusing on the use of multiple-choice questions in exams (Shuhidan, Hamilton, and D'Souza, 2009; 2010). Most instructors in their study considered multiple-choice questions appropriate for testing questions on the lowest levels of the Bloom taxonomy (Bloom, 1956), but less than half were confident that multiple-choice questions could be used to test understanding of programming concepts (Shuhidan, Hamilton, and D'Souza, 2009; 2010). A problem faced in the investigation of exam questions is the difficulty in applying Bloom's taxonomy to classify exam questions according to their cognitive level. An Australian research team has developed an online tutorial to train researchers in the use of this and other taxonomies (Gluga et al, 2013).

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

Non-exam assessment

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

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

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

Portfolio-based assessment is rather less common than assignments. Australian researchers Cain and Woodward (2012) describe an introductory programming unit where students are assessed entirely on a portfolio of work produced during the semester. The design of the unit was founded on Biggs's constructive alignment (Biggs, 1996), which proposes alignment between the learning activities, assessment, and intended learning outcomes. An evaluation showed that students were positive about their learning experience. Pears (2010) reports on the use of portfolio assessment in an introductory programming unit for the purpose of implementing a continuous assessment model. He found that students who completed the unit produced code of a higher quality than typically produced by first-year students.

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

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

Automated assessment

The time-consuming tasks of collecting, marking, and giving feedback to students on their assessment work has led to the development of tools to help manage these

processes. All of the assessment tools that we found were specifically designed for use in introductory programming classes.

Law, Lee, and Yu (2010) present PASS – Programing Assignment aSsessment System. PASS provides feedback for programming assignments by executing a set of instructor-prepared test cases and then comparing the expected output with the actual output. PASS also allows the teachers to monitor the testing process of students' submissions in real time and to share with the entire class examples that demonstrate good practice. A study of PASS showed a positive impact on students' self-efficacy.

Wang et al (2011) discuss the role of automatic assessment in introductory programming and present a tool, AutoLEP, for automatic analysis and assessment of student programs. They describe their use of this tool for in-semester formative assessment and for end-of-semester exams. Students and staff were enthusiastic about the tool, with staff reporting that students showed increased interest in programming and improvement of their skills.

Llana, Martin-Martin, and Pareja-Flores (2012) present an online free laboratory of programming (FLOP), which hosts a repository of programming problems that students can attempt and have automatically assessed. Preliminary results indicate positive improvement in students' motivation, skills, and self-efficacy.

Johnson (2012) presents a tool, SpecCheck, for testing conformance of programs to the assignment specification prior to submission. A small study showed that students were willing to accept having to produce highly structured homework in return for faster grades and feedback.

Shaffer and Rossen (2013) present the Programming Learning Evaluation and Assessment System for Education (PLEASE), a code-checking and submission system. Using data collected from the system, the lecturers were able to identify parts of the course where students were experiencing difficulties and make adjustments to the teaching program. The results of a small study indicated that the tool was useful in optimising course structure.

Assessment instruments

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

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

Summary

The literature on assessment in first-year ICT courses relates predominantly to programming. Nearly half of the papers found were from the Australian context, indicating research strength in this area. Although exam assessment has attracted the most research, a number of other forms of assessment have been investigated. Underlying motivations for academics' choice of assessment were often pedagogical: to encourage student engagement, provide timely feedback, or ensure academic integrity; or they were pragmatic: to ease the burden of marking. With the trend of an increased reliance by students on online course materials, further research is suggested on methods to improve the automation of assessment and provide quality feedback on students' work, while maintaining the academic integrity of the assessment process.

3.2 Current Assessment Practice in Australia

The interview questions sought information about assessment practices in first-year ICT courses in Australia. The responses gave insights into current assessment practices and issues faced by teaching staff. Thematic analysis was used to extract and code responses and to identify the major issues raised. The responses to these questions are discussed under the main topics that were identified from the analysis of the interview data: assessment design and strategies, exam and non-exam assessment, and automated assessment. The issues of provision of feedback, verification of student work, and other issues associated with academic integrity are discussed in terms of the different forms of summative and formative assessment. In reporting the findings, representative quotes have been included to further elucidate the discussion.

Assessment design and strategies

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

A couple of interviewees mentioned their university having an overall assessment strategy. Interviewee U8 commented that at her university, "*assessment revolves around problem solving – looking at authentic situations*". An assessment guide based on Biggs's theory of constructive alignment (Biggs, 1996) had been developed at one university. Constructive alignment was also mentioned as a theoretical basis of portfolio assessment at another university.

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

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

Most interviewees mentioned assessment policies at their university. It is common practice to set thresholds

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

Exam assessment

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

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

Non-exam assessment

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

Tests held during semester are a common form of assessment. These may be mid-semester tests worth from 10% to 20% or a series of smaller tests often conducted online using the LMS or another tool, such as ViLLE (Rajala et al, 2007). Some interviewees expressed a preference for continuous assessment, with smaller tests rather than one larger test. One interviewee commented that he does not hold a mid-semester test as the semester is only 11 weeks long.

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

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

At another university portfolio assessment has been used for the past five years in an introductory programming unit. The portfolio assessment has been designed using Biggs's constructive alignment. Interviewee U1 explains:

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

Interviewee U1 describes the process:

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

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

"Each student has to submit a portfolio that demonstrates how they have met all of the unit learning outcomes. Then there is a scale by which they can meet [the learning objectives]. To meet them to an adequate level there are criteria. To meet them to a credit level there are separate criteria, and so on for distinction and high distinction. This allows students to work to their expectations. Some students only want to pass the unit and they're not interested in doing really well … That's not what their goal is in life."

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

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

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

There were indications of a growing use of social media for assessment tasks. For example, interviewee

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

Automated assessment

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

Feedback

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

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

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

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

At interviewee U9's university all assignment submission times are recorded and therefore the timeliness of the feedback provided to students is also recorded. A permanent record of all feedback is also stored, in case an issue arises. This university-mandated policy has the potential effect of allowing an audit of the quality and promptness of the feedback provided to all students in every course. Therefore a systematic process may be implemented to improve the standard and responsiveness of the feedback delivered to students.

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

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

Academic integrity

Three subthemes emerged from analysis of the academic integrity theme.

Verification of work

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

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

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

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

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

Tools are sometimes used in verification of student work. The plagiarism detection tool Turnitin is frequently used for text-based assignments; however, the use of plagiarism detection tools for programming assignments appears less common. Tools such as MOSS (Measure of

Software Similarity), JPlag, and ESP were mentioned for detection of code plagiarism; however, one interviewee suggested that plagiarism detection tools were not suitable for first-year programming as there is usually too much similar code. Interviewee U2 only follows up on obvious plagiarism, seeing the assignments "*as learning opportunities as much as assessment.*"

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

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

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

Discouraging cheating

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

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

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

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

… or sent from the head of the school every semester:

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

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

Penalties for breaches of academic integrity

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

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

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

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

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

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

4 Discussion and Recommendations

Although a variety of forms of assessment were identified in the literature, most interviewees mainly discussed traditional forms of assessment. The few innovative

assessment practices found were designed to encourage attendance (e.g. tutorial assessment), engage students in learning activities (e.g. social media), or encourage good work habits (e.g. portfolios). Interviewees' comments indicated the high importance they place on giving feedback on work during semester. Academic misconduct is a problematic area and there are a range of techniques used to verify students' work and discourage plagiarism and collusion.

A number of areas identified concerning assessment practice warrant further investigation. Overwhelmingly, the context for research and discussion in assessment was in the context of programming. There were a variety of techniques and tools for assessment of programming, but very few in other areas of study. We suggest that research on assessment techniques for other areas of the first-year ICT curriculum might be appropriate. The recent adoption of social media has led to innovative forms of assessment and there were reports of its use in a number of universities; however, few studies were found that evaluated the use of this assessment form in first-year ICT courses. This is an area that could be further investigated.

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

5 Conclusions and Future Work

Our investigation of assessment in the first year of ICT courses found that most of the literature is related to assessment in programming courses. Assessment of programming is an active area of research in Australia, although most of the work is focused on exam assessment. In contrast, the good practices in assessment identified in Australian ICT courses are concerned with portfolio assessment, interviewing students to verify assignment work, and using appropriate tools to facilitate and expedite provision of feedback for in-semester tasks and assignments.

Assessment is a key part of the total learning experience of our ICT students and has a major impact on their educational outcomes. This study contributes to our knowledge of assessment practices in first-year ICT courses and motivations and impediments to their use.

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 acknowledges 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

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

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

Bloom, B.S. (1956). *Taxonomy of Educational Objectives: Handbook I: Cognitive Domain*. Longmans, Green and Company.

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.

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

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

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

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

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

Gluga, R., Kay, J., Lister, R., Simon, & Kleitman, S. (2013). Mastering cognitive development theory in computer science education. *Computer Science Education*, 23(1), 24-57.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Rajala, T., Laakso, M.-J., Kaila, E., & Salakoski, T. (2007). VILLE: a language-independent program visualization tool. *Seventh Baltic Sea Conference on Computing Education Research*, 151-159.

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

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

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

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

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

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

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

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

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

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

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

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

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

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