

Progress report for confirmation of candidature

Robyn A. McNamara

November 25, 2005

1 Title

Computer Programming Skills Assessment in Introductory Computer Science Education

2 Introduction

Computer Science programs around the world seem to have problems at second-year level and later. Many authors have reported that their second-year students are unable to write even relatively simple programs (McCracken et al. 2002, Jacobson 2000, Califf & Goodwin 2002, Daly & Waldron 2004), and it is reasonable to assume that this inability could be a contributing factor not only to the low success rates mentioned in the previously cited studies but to the low morale and high attrition mentioned in studies such as Ramalingam et al. (2004) and Werner et al. (2005).

Many authors attribute these issues to deficits in the students' ability to analyse and solve problems, and several have proposed or implemented courses that explicitly address problem-solving (Allan & Kolesar 1997, Beauboeuf et al. 2001, Barros et al. 2003, de Raadt et al. 2004) or abstraction (Bucci et al. 2001). Lister et al. (2004) provide convincing evidence that many students lack the ability to read and analyse code, without which the problem-solving process cannot even begin. Ramalingam et al. (op. cit.) posit that exercises designed to bolster students' self-efficacy – confidence in their ability to produce work at a given level – may improve retention rates. Many authors blame student plagiarism for the mismatch in achievement level between first-year and second-year courses (Lorenzen 1981, Jacobson 2000), and there is evidence that plagiarism occurs at first-year level (Sheard et al. 2002).

No matter what reasons underlie the difficulty – and these reasons are likely to be complex, multifactored and variable according to individual and institutional context – if students can pass first-year programming prerequisite courses and then prove on entering second year to be unable to program at a sufficient standard, then the programming skills assessment that students undertake at first-year level is clearly not doing its job.

It seems logical that a better understanding of how to assess these skills will not only help Computer Science academic staff to evaluate each student's readiness to progress to later courses or into the workplace, but will also lead to better understanding of the pedagogy of Computer Science.

The proposed research will explore the practical and pedagogical issues surrounding the assessment of programming ability, examine some of the factors that could be interfering with its effectiveness, and point the way towards the development of more effective assessment techniques.

2.1 Research question

Why do many second-year Computer Science students have trouble with their programming tasks despite having passed their first-year prerequisite programming courses?

2.2 Subsidiary questions

1. Is the first-year assessment valid?
 - (a) to what extent do staff believe the assessment is valid, and why?
 - (b) is it reliable?
 - (c) are the assessment instruments good predictors of later performance?
2. Is first year teaching the skills required for second year?
 - (a) what skills are required?
 - (b) are the students learning these skills?
 - (c) are these skills being assessed?
3. Are students deep-learning?
 - (a) what impact does assessment have on student approaches to learning?
 - (b) do students perceive assessment practices as encouraging or discouraging particular approaches or study habits?
4. Are there organizational factors that may be interfering with assessment or learning?

3 Assessment

3.1 Terminology

Assessment is the process of making inferences about a student based on a measure of their performance at some task. The measure itself, for example an exam, is not the assessment; nor is the result. Assessment happens when we think about what it means for a student to have achieved their result. This interpretation is done in the light of what we think the assessment task is measuring, how other current and former students have performed similar task, and the likely consequences of our assessment.

Assessment serves many purposes, and it would be foolish to try to critique it by addressing only one of these purposes. It is especially useful to distinguish between

the *evaluative* role and the *educative* role. Evaluative assessment is designed to tell somebody something about how well the student is performing. The term subsumes summative assessment, which tells the institution and teaching staff how the student is doing and allows them to make administrative decisions such as whether to allow the student to progress to a higher level, and formative assessment, which tells the student how well he or she is performing and which areas of performance, if any, need improvement. This role also covers the use of student assessment to evaluate the teaching program itself. The key differentiator is that evaluative assessment is *seeking information* about student understanding.

Educative assessment, on the other hand, is a learning activity. In first-year Computer Science at Monash, pracs definitely serve a summative role – students must achieve at least 45% overall in the prac stream or they will not be allowed to progress. Students also obtain informal formative assessment from their demonstrators, who tell them where they need to improve in order to do well. Pracs must also serve an educative role – for most students, the bulk of their programming experience takes place when they are either in or preparing for their prac classes.

This distinction is loosely based on the taxonomy of (Hodson n.d.), who categorised assessment as formative, summative, evaluative, or educative. In his taxonomy, however, “evaluative assessment” refers solely to assessment for program evaluation. For the purposes of this research, the distinction between formative, summative and evaluative assessment is less important than the distinction between assessment *for* learning and assessment *of* learning, so the three noneducative categories have been rolled together.

3.2 Assessment and learning

Assessment may have a range of effects on student learning. Teaching staff will usually set assessment during semester with educative intentions, in the expectation that completing the assessment will encourage the students to familiarize themselves with parts of the subject domain. Assessment may also be set with the expectation that it will encourage students to adopt particular learning styles. For example, a lecturer wishing to encourage reflective learning may require students to submit a reflective journal along with other coursework. Such expectations may or may not be explicitly conveyed to the students, and may not have even been articulated at all, but they are conscious and deliberate on the part of the staff.

Assessment may also have unintended and unanticipated consequences for learning – the so-called “hidden curriculum” effect (Snyder 1973). For example, if tasks are set that reward rote-learning over comprehension or creativity, then students will tend to adopt rote-learning strategies regardless of the stated preferences of the lecturer. Students will also tend to adopt rote-learning and other shallow learning practices if they feel themselves to be under time pressure. This is not surprising; deep learning requires time for reflection. Moreover, this effect does not only occur when students have been set tasks that are onerous or poorly timed so that they clash, but also when students run out of time due to their personal circumstances or even through their own poor time management (Thomson & Falchikov 1998).

The hidden curriculum is conditioned not only by the teaching and assessment

practices of the current course, but also by the students' previous experience with similar courses (Sambell & McDowell 1998). In the context of the current research, this implies that students' approaches to their programming work in second year will be coloured by their experiences in first-year programming courses – not only by their developing grasp of the official curriculum, but by their developing sense of what it means to complete programming prac work and the strategies that are best to approach it. Students may therefore approach their work in ways that surprise the staff who set it, and their results may be interpreted incorrectly.

3.3 Quality of assessment

Assessment is measurement, but it is an indirect measurement. When we assess a student, we are making an inference about an abstract quality of the student – their capabilities, knowledge, motivation or attitude, for example – on the basis of an observation of the student's performance at some task. The observation itself may be a direct measurement, for example a count of the number of correct selections made in a multiple-choice test, or it may itself be problematic and subjective, such as the myriad observations that go toward a demonstrator's perception that a student comprehends the prac work they have submitted.

3.3.1 Reliability

The reliability of an assessment task refers to its *goodness as a measure*; in other words its relative freedom from errors of measurement, using the scientific sense of the word “error” rather than the colloquial – this kind of measurement error does not stem from a preventable mistake, but from the impossibility of devising an assessment task whose results will always be completely free of construct-irrelevant variance (Feldt & Brennan 1993). In practice, reliability boils down to consistency. “In validity the emphasis is on a test's agreement with the objective; in reliability, upon agreement with itself.” (Jordan 1953, pp.26–27). A test whose results are inconsistent cannot be said to be telling us much about the testee's grasp of the construct under investigation, and therefore cannot be held to be a useful way to assess it.

In Computer Science, as is the case with other disciplines that require the use of technology in their assessment, reliability will be threatened if something goes wrong with the machines the students need to use. If the student lab network goes down while students are working on a prac or assignment, then their results will not be consistent with the marks they would have achieved had the system worked correctly. A similar effect will apply if the work environment provided is distracting in some way – too hot, too cold, or too noisy. These environmental factors are partially under institutional control.

Reliability may also be threatened by factors outside institutional control. A student's performance on an exam may be affected by illness, for example, or tiredness, or emotional state; this will mean that the student will get results that are different to the results they may have achieved if they were tested on a different day. It is very difficult to eliminate these factors, although they may be ameliorated by strategies such as continuous assessment that seek to reduce the impact of

these transient influences on the overall mark. It is unreasonable to expect perfect reliability, and better to ask whether the assessment is reliable *enough*.

Assessment may be intrinsically unreliable. For example, if students are given instructions that are unclear, inconsistent, or do not match the way that the work will be assessed, then marks are unlikely to be consistent with the students' true abilities. The assessor has no control over reliability threats that are intrinsic to the student, such as illness, and is likely to have little control over environmental threats to reliability. He or she does have control over the intrinsic quality of the assessment tasks as they are presented to students.

Not all reliable assessment is valid. The assessment may be a good measure, but measuring something other than the desired psychological construct. For example, suppose this question appeared on an introductory C programming exam:

You can store several items of the same type in an:

- a) pointer
- b) array
- c) struct
- d) variable

A student who knows C data types well will select option b on the basis of that knowledge, but it is possible for a student who has no knowledge of C to infer the correct answer from the structure of the question. The preamble ends with "an", and b is the only option that begins with a vowel. To some extent, the question is measuring English language skills rather than knowledge of C data types. To that extent, the question is reliable, but invalid.

There are many reasons that assessment may turn out to be unreliable, and therefore it is useful to distinguish between different categories of reliability.

Interrater reliability refers to consistency across assessors. If there is only one marker, or if the assessment is marked by a computer program, interrelater reliability is not an issue. If the marking is shared, care needs to be taken that all markers agree on standards.

Stability, or test-retest reliability, refers to consistency over time. A test is said to be stable if redelivery of the same test to the same cohort of students would garner the same marks.

Alternate-form reliability refers to consistency across forms. If different items that test the same concept get similar marks, then the test has alternate-form reliability.(Salvia & Ysseldyke 1995)

Reliability can be evaluated empirically, but is also subject to qualitative analysis. Interrater reliability, for example, may be evaluated by having examiners double-mark students' work and checking to see how close the marks are. It can also be evaluated qualitatively by examining the procedures that the examiners go through to ensure consistency of marking, such as rubric construction.

3.3.2 Validity

Validity is the most important consideration in choosing assessment. Loosely speaking, assessment is valid to the extent that it measures the construct it is designed to

measure. This informal formulation, although simple and comprehensible, turns under closer examination into an epistemological tar baby – how can you know whether the assessment is measuring the desired construct when all alternative measures of the construct must also answer questions about their validity?

Most recent discussion of validity draws heavily on the work of Messick (1993). Messick’s approach draws on epistemology and the philosophy of science rather than on statistics or social science, and concludes that assessment is valid to the extent that it supports the decisions that are based upon it. This operational definition sidesteps the epistemological and ontological questions altogether, returns the focus of the investigation of assessment instruments to their use in context, and provides a justification for including the social consequences of assessment in its evaluation.

All assessment is inferential in nature. Students undertake an assessment instrument such as an exam or assignment, and examiners must infer the students’ levels of ability from the work submitted. The validity of an assessment instrument is “the degree to which evidence and theory support the interpretations of test scores *entailed by proposed uses of tests*” (American Educational Research Association et al. 1999, p 9, italics mine). That means that it is futile to evaluate the validity of an assessment instrument without taking into account the uses to which it is put. In the context of first-year programming courses, the assessment is valid if the students who achieve an overall pass at the end of semester are ready to proceed to second year Computer Science courses, and the students who fail are not yet ready to proceed to second year.

In order to serve as the basis of these decisions, the assessment must measure psychological constructs – skills, abilities, knowledge or attitudes. An assessment item or instrument is said to have *construct validity* if it is reasonable to use a student’s results to make inferences about his or her mastery of the unit. Although construct validity is the basis for any reasoning about validity, as Messick (1993) convincingly argues, the fact that it can not be directly measured means that other evidence, such as measures of the subtypes of validity, must be used to support claims about it.

Evidence regarding the validity of of an assessment item can be drawn from many sources. Comparing scores against scores on related assessment items gives a measure of *criterion validity*; however, caution must still be exercised in the interpretation of this comparison as it must with any correlation study.

Criterion validity is subdivided into *concurrent validity* and *predictive validity*, in which the external criteria are the results of assessment instruments in related fields that are taught or tested over the same period of time or later, respectively. That means that correlations between two parts of an exam can be used as evidence of concurrent validity, provided these parts can be reasonably expected to be measuring the same construct, and predictive validity can be measured by examining correlations between assessment results for a unit and those for follow-on units. Predictive correlations must be interpreted with great caution, as too high a coefficient of correlation may indicate that little learning has occurred across the board!

In order to demonstrate construct validity, it is not sufficient to demonstrate that assessment results correlate strongly with other data that are assumed to relate to the construct in question. It must also be shown that assessment results correlate

less strongly with data that are assumed *not* to relate to the construct in question. This form of evidence is known as *discriminant validity*. As an example, imagine that analysis of first-year and second-year programming prac results show a strong correlation between a first-year prac on linked list construction and a second-year prac in which students are expected to use a linked list to store data for manipulation by a more sophisticated algorithm. This correlation by itself could be interpreted as criterion-related evidence for the validity of these pracs – it is reasonable to accept that the marks are strongly correlated because both tasks are testing the student’s ability to understand and use linked lists. However, if on closer examination the other pracs also correlate equally strongly, even when they are ostensibly testing quite different concepts such as the use of input/output functions to store data in a file on disk, then we must question whether the pracs are primarily testing what they are supposed to test, some more fundamental concept such as debugging skill or comprehension of programming syntax, or some construct-irrelevant factor such as the ability to use a keyboard effectively.

Because it is usually impossible to examine all aspects of the domain of a unit within the constraints imposed by budget and time, it is necessary to select assessment items that examine a representative sample of the subject domain. The *content validity* of an assessment instrument is the extent to which the items that comprise the instrument are a representative and reasonable sample of the domain (American Educational Research Association et al. 1996). Content validity is not susceptible to quantitative analysis and is usually determined by review undertaken by experts in the domain. In educational assessment, it is possible to evaluate content validity by comparing the assessment material to the list of unit aims and objectives that most institutions require as part of the development of any course. These aims are in turn routinely scrutinized by a variety of internal and external stakeholders including accrediting organizations such the Australian Computing Society.

Construct validity is also subject to a form of non-quantitative validation. The design of assessment instruments proceeds according to the designers’ theories about how students process the task and how performance on the task allows inferences about psychological constructs to be drawn. For this construct-related validation, it is not sufficient to determine a correlation, as is the case with criterion validity; it is also necessary to formulate an explanation for such a correlation in the context of the conceptual framework surrounding the construct of interest.

3.3.3 Validity and reliability

Classical validity theory held that, for a measure to be valid, it must be reliable. This viewpoint has been challenged in the literature by authors such as Moss (1994), who makes a convincing case that reliability concerns should not be the primary driver for assessment design. The standardization necessary for the application of many reliability metrics imposes constraints on assessment design that are wholly out of step with modern theories of assessment and learning, disallowing portfolio-based assessment, group assessment, and many other innovative assessment strategies. This is undeniably true, yet it is still true that reliability is a necessary condition for validity.

Assessment is valid to the extent that it is feasible to base decisions on its results.

Usually, these decisions involve an inference from the assessment results to one or more constructs of interest, and a further inference from the putative construct to a course of action. In order to be able to draw an inference from result to construct, there must be a degree of consistency in these results. Without this consistency, the result is essentially meaningless – the construct-relevant information it contains, if any, is swamped by construct-irrelevant noise. This consistency is what is meant here by reliability. If the assessment is project-based, its reliability might not be measurable by the metrics that are commonly applied to standardized tests, but it must be present for the assessment to be useful at all.

For this reason, it is necessary to gather evidence for the reliability of the assessment. The standard statistical tests may be used for assessment that is amenable to such measurement, but it is also possible to gather qualitative evidence that measures have been taken to support interrater and intrarater consistency.

3.3.4 Validity in CS Ed literature

Validity is not usually explicitly addressed in the Computer Science Education literature. Indeed, it is rarely addressed in discipline-specific tertiary education literature at all. This is not because the concept of validity is not applicable, but because addressing such concerns requires both an unusually broad skill set in both Education and the target discipline and more resources than are usually devoted to journal articles. In particular, criterion-based arguments for validity may require result sets from units across the course. Such arguments are also likely to take a paper beyond the length limit for SIGCSE (the Association of Computing Machinery's Special Interest Group for Computer Science Education) or ITiCSE (Information Technology in Computer Science) conference papers.

Authors in Computer Science Education usually support changes in assessment by arguments from pedagogy or student surveys. A review of the ITiCSE Proceedings for 2004 shows that, of five papers that dealt with the introduction or evaluation of assessment, one evaluated the assessment against a criterion, one supported the style of the assessment with an argument from pedagogical theory, and all five used student surveys. A larger regional conference in the same year, Australasian Computing Education (ACE) 2004, had sixteen papers dealing with assessment, of which ten supported their evaluation with an argument from theory, nine used student surveys, three used arguments relating to organizational factors such as instructor workload, and five compared against a criterion. Of the five who used a criterion, three used the final mark in the unit under evaluation. Because that final mark was partly dependent on the assessment mark, those criteria must be considered flawed.

Of the validation mechanisms used, the criterion-related evidence is strongest. Arguments from pedagogical theory are also evidence regarding validity, relating as they do to the reasons for believing the assessment to have construct validity, but they do need to be supported by other evidence. The pedagogy of computing education is still developing, and so the theory itself should not be taken for granted. Student surveys must be considered weak evidence for validity. They must be conducted while the students are still available as a body, and so are often run before the final assessment has been delivered and almost always before the students have received their results. If the course has not prepared them well for the next stage in

their development, they will not yet be in a position to know. Such surveys can give valuable information about the student experience, levels of confidence and comfort, but their utility as tools for construct validation is questionable.

There are several other papers that use criterion-based arguments. For example, Daly & Waldron (2004) measured the correlation between first-year prac exam marks and third-year capstone project mark, and Chamillard & Braun (2000) explored the correlations between a range of introductory programming assessment instruments. It should be noted, of course, that studies like Daly and Waldron's require a two-year gap between the administration of the assessment at first-year level and the collection of the final project mark. By the time the analysis has been completed, the first-year course may well have changed to keep up with new technologies. Moreover, most students in Daly and Waldron's study did their capstone project in pairs. This presumably lowered the correlation coefficient, but when the data was reanalysed to exclude results from students who worked in pairs the cohort was reduced to only 26 students. Daly and Waldron do not give the significance of the correlations they found.

Despite the paucity of discipline-specific literature on assessment validity, it is still a viable and necessary field for investigation. Equity considerations alone are sufficient to prompt investigation into validity. Construct validity implies that the mark a student gets for his or her work accurately reflects the student's level of mastery. Assessment that does not have construct validity must have its marks determined by some factor other than mastery, and this is unfair. Conversely, if there are good reasons to believe that the assessment *is* valid and yet its marks seem to show a strong correlation with a factor such as gender or cultural background, then this correlation may point out an unsuspected bias elsewhere in the course.

4 Preliminary work

4.1 Honours project

My Honours project was based on a statistical analysis on data from CSE1301 Computer Programming, the first unit in the core Bachelor of Computer Science and Bachelor of Software Engineering degrees, for first semester 2001. It was finished in 2002 and formed the basis for a conference paper presented at ACE2004. (McNamara 2002, McNamara 2004)

The project was an attempt to discover the conceptual structure of introductory computer science based on patterns of correlation in the marks students obtained for different tasks. The marks students got for different tasks were assumed to depend on different basic competencies (abilities informed by knowledge); the primary determinant of the mark should be the student's mastery of the competencies that the task was designed to assess. If this was the case, and if there were not too many of these competencies involved in the assessment, it should have been able to infer something about these competencies from examining marks.

I was unable to determine anything about underlying conceptual structure from my analysis of these marks. Rather than the domain of the assessment, the mode of assessment – prac or exam – turned out to have a much stronger correlation to the

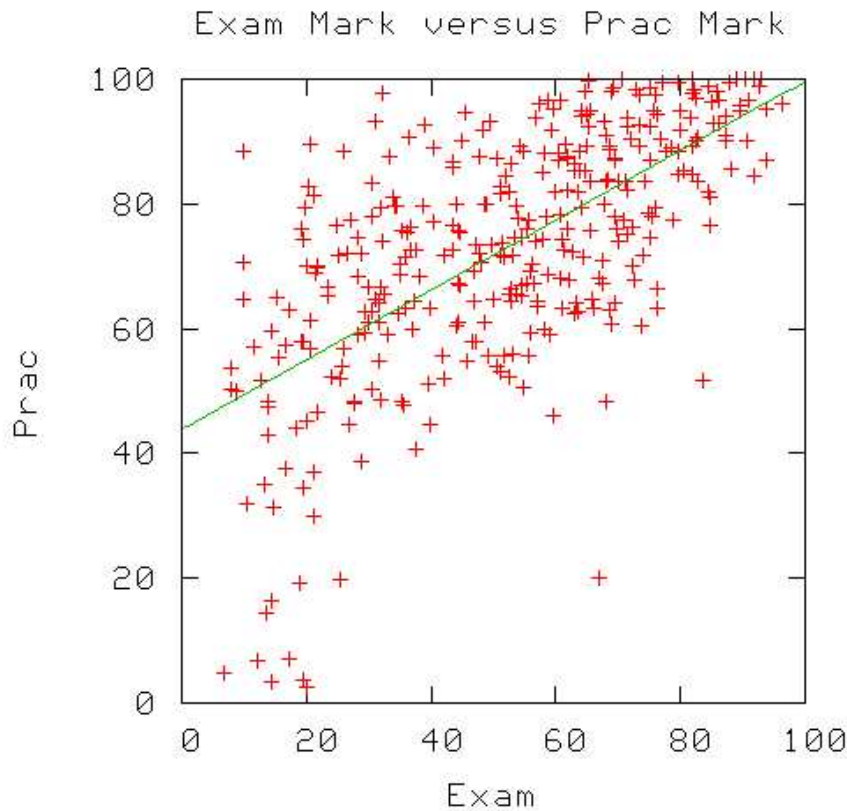


Figure 1: Scatterplot of prac vs exam mark, from seminar

mark than the relatedness of the topic. That is to say, marks for a prac on pointers were more similar to marks for a prac on I/O than they were to an exam question on pointers. Correlation coefficients between different pracs were around 0.7 regardless of the content of the prac. Correlation coefficients between prac questions and exam questions were as low as 0.3.

If the content of a task is not the prime determiner of mark, is it really measuring programming ability, or is it measuring something else? If both pracs and exams are measuring the same quantity, why does the mark seem to depend so strongly on the kind of assessment rather than the student's ability? The fact that this effect seemed stronger for the bottom third of students raises many more questions.

4.2 Seminar

In early 2005, I reanalyzed the same data, seeking possible reasons for the disparity in marks for tasks that ostensibly measure the same quantity.

It seems reasonable to assume that the more like computer programming a task is, the more similar the mark will be to a criterion computer programming mark. We are led to the conclusion that prac marks are as close as we are going to get to a programming criterion.

This does not imply anything about the absolute strength of the correlation be-

tween prac marks and components of the exam mark. It does imply that exam questions that are more like programming should correlate more strongly with the prac mark than questions that are less like programming. We can, therefore, make the following prediction: Programming questions on the exam will be a better predictor of prac mark than short-answer questions, which will in turn be a better predictor than multiple-choice questions.

I took the task-by-task marks I had for my Honours data, categorized each exam task as multiple-choice, short-answer or coding, and calculated a percentage mark for each category for each student. I also calculated a percentage mark for the exam overall, and for the prac component. I drew a scatterplot of mark versus mark for each pair of categories, and interpolated a best-fit line on each scatterplot. One such scatterplot is shown at Figure 4.2.

I found that programming questions and short-answer questions predict prac mark equally well, with multiple-choice questions not far behind. This is disturbing because the marking scheme did not penalize incorrect answers, so a student could expect to achieve a score of 20% on this section through sheer guesswork.

A notable subsidiary result is that it seems very easy to get a high mark in the prac component. The scatterplot shows most points in the top corner, above the best-fit line, and indeed the facilities for the pracs are very high – up to 0.88 in one case. There are many different potential causes for this finding. Plagiarism is one possible reason, although pracs are supervised; or demonstrators could be providing more assistance to students than they are supposed to. Perhaps the open-book nature of the labs is more important in determining success than examiners assume. It could be that demonstrator feedback is allowing students to learn faster or in some way “better” than assessors assumed when setting marking schemes. This would explain the high facilities, but not the poor correlation to exam mark, unless the in-class learning was not transferable to the exam situation – perhaps it is not deep learning, although it would be reasonable to expect that the higher level of engagement in a lab situation compared to a theory study session should lead to deeper learning, or perhaps the pen-and-paper environment is simply so different to the online environment that the exam is testing a different skillset. In the latter case, of course, we must assure ourselves that both tested skillsets are construct-relevant.

The facility of the prac work needs to be interpreted with great caution. The educative role of this assessment is of paramount importance. It is the only chance most students have to practice their programming in the presence of a mentor, with immediate or near-immediate feedback on their performance. This group practice also allows students to develop their confidence, emotional engagement and sense of community. The apparent ease of the introductory programming prac stream could turn out to be very important to the morale and motivation of the student body. Even if the results as currently interpreted turn out to be of questionable validity, scrapping the pracs would probably be a retrograde step. A better idea would be to reinterpret prac results, possibly as criterion-referenced competency assessment in the manner of Lister & Leaney (2003) and Box (2004).

These results are unpublished, but were presented at a seminar in March, 2005 at Monash and repeated in July at RMIT.

5 The BCompSci at Monash University

The context of this research is the Bachelor of Computer Science at Monash University, so it is appropriate to describe the degree structure and assessment methods that are used. This section may be omitted by readers who are already familiar with the Monash BCompSci and its assessment.

5.1 Computer Science Assessment Methods

The assessment methods used in the core units for the Bachelor of Computer Science at Monash include the following:

- **Pracs.** These are held in labs that are equipped with computers, and may be connected to the Internet. Students are allowed to bring in any books or papers they wish. A demonstrator is present to help students with problems, mark their work, and ensure that the work presented is the student's own.
- **Exams.** These are closed-book, invigilated, pen-and-paper exams, and are invigilated. They are held at the end of every semester. All core CS units have a large closed-book final exam component to their assessment.
- **Assignments.** These are take-home written exercises that may or may not include a substantial programming component. In general, students work on assignments without supervision, although staff are available for consultation should a student wish extra help or need the requirements clarified.
- **Class tests.** These are low-weight, secret-task, pen and paper “mini- exams” that are usually held during the regular lecture timeslot.

Other assessment techniques used in CS but not in the Monash BCompSci core:

- **Prac exams.** These are held in a computer lab, but are held under exam conditions. As with exams, the tasks are not usually given to the students in advance. Prac exams may or may not be open-book or open-Web. They are more authentic assessments of programming ability than paper exams, but are not used in core subjects because they require a substantial investment in organization and infrastructure. Prac exams are used in CSE2/3295 Perl Programming and CSE2/3291 UNIX Tools
- **Group assessment.** Because most graduate destinations in industry or research will require Computer Science majors to work collaboratively, group assessment is taking off in popularity in many institutions around the world.

5.2 Structure of the Monash BCompSci

The BCompSci course at Monash comprises eleven core Computer Science units, two units of Mathematics, and a number of electives. For the purposes of this research, we will only consider the core Computer Science units. These core units and their prerequisites are shown in Figure 5.2.

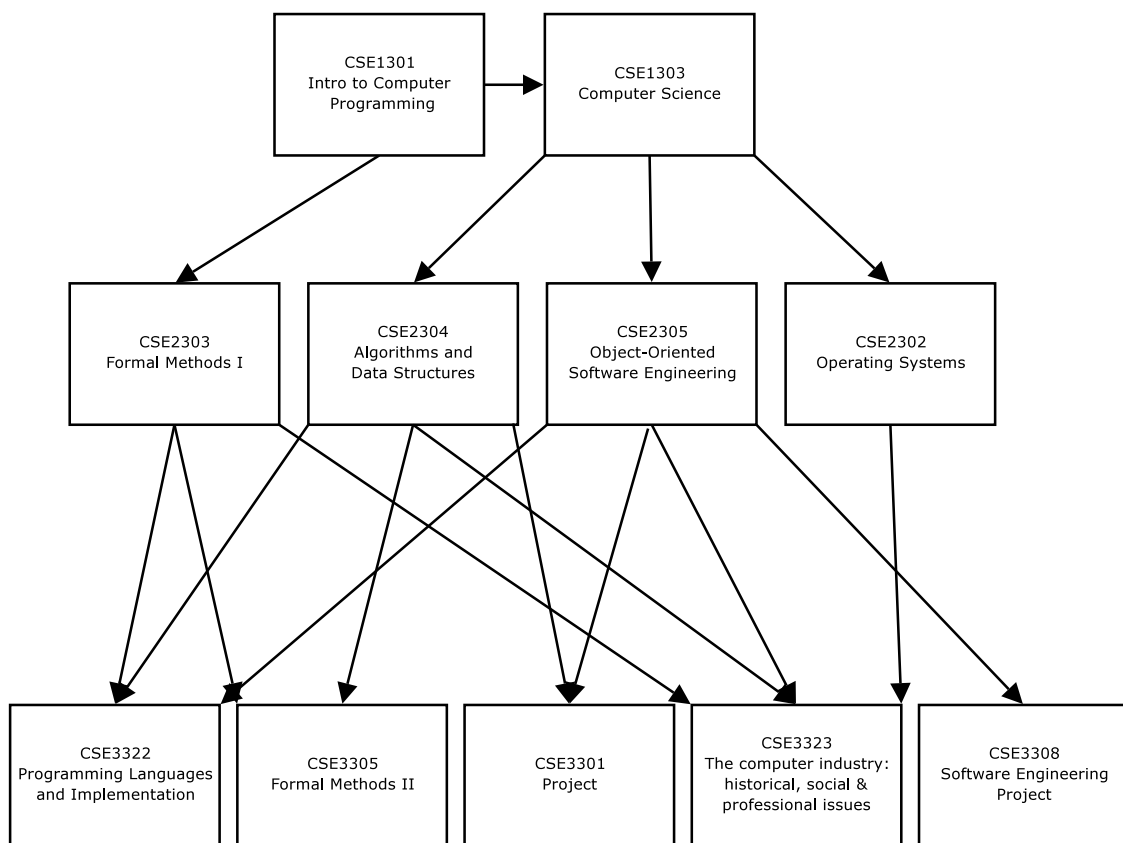


Figure 2: Prerequisites among the BCompSci core

5.2.1 First-year core

CSE1301 and CSE1303 should both be considered introductory programming units. CSE1303 introduces some computer science topics but most students are still fledglings at this stage, and programming issues dominate in the minds of many if not most students. The subjects are structured similarly in terms of assessment: weekly pracs, a mid-semester test, and a 60% closed-book, written final exam.

5.2.2 Second-year core

CSE2303 Formal Methods I is a theory subject, but introduced illustrative pracs to give students hands-on experience with concepts covered in lectures. Pracs comprise 30% of the final mark; the closed-book, written exam fills the remaining 70%.

CSE2304 Algorithms and Data Structures has both strong theory and practical components, as students are expected to be able to design, analyse and implement their own algorithms as well as learn well-known algorithms. Again, this subject is 30% prac and 70% closed-book, written exam. Pracs evaluate how well students can implement algorithms, and so are closely related to the unit's domain.

CSE2302 Operating Systems covers the theory of operating systems, and also has a series of pracs that allow the students to gain practice in both using system calls in programming tasks and implementing small sections of an operating system. The theory component is assessed by closed-book written exam (70%) and assignment (30%). The in-class programming is a hurdle requirement. In recent years, the hurdle has taken the form of a mark deduction for students who achieve less than a criterion, with the amount of penalty depending on their distance from the criterion. In 2005, this deduction ranged from zero to 40 marks, with the maximum penalty only applying to students who submitted no prac work at all.

CSE2305 Object-Oriented Software Engineering teaches the theory of object-oriented design and exemplifies the actuality of object-oriented programming with a series of programming and design assignments using C++. Students have fortnightly prac classes, during which they can get help with their programming tasks, but the assignments are marked outside class. Assessment is 80% closed-book, written exam, 20% assignment.

6 Research methodology

A mixed research methodology (Johnson & Christensen 2004) is the most appropriate way to conduct this study. A purely quantitative approach might be able to demonstrate that a problem exists in assessment. However, discerning the causes underlying the problem, finding means of remediation, and discovering what students and staff consider important is going to require a qualitative approach. Conversely, a purely qualitative study will allow some insight into the problems staff and students face, but will be limited in what it can tell us about the severity and prevalence of these problems.

Validity is subject to empirical evaluation in the form of correlation studies for criterion validation. It is also subject to qualitative evaluation. The lecturer who

sets an assessment task presumably has some basis for believing that the task is a good way to assess students' abilities, whether that belief is derived from experience, pedagogical theory, or some other source. If these beliefs are elicited, they can be examined to see how well they hold together. This combination of approaches to form an overarching argument about the validity of assessment resembles the validation methodology suggested by Kane (2001)

Most of the research subquestions can be approached by both qualitative and quantitative methods.

- Is the assessment valid?
 - Quantitative: measure correlations as evidence of predictive validity
 - Quantitative: apply empirical measures of reliability
 - Qualitative: investigate staff reasons for applying assessment
 - Qualitative: investigate the measures taken to ensure interrater reliability such as multiple marking or joint rubric development
- Is first year teaching the skills required for second year?
 - Quantitative: seek patterns in the correlations between marks for first year and second year programming tasks
 - Qualitative: check that the skills second-year lecturers believe will be required match the skills first-year lecturers are trying to teach
 - Qualitative: examine first-year assessment tasks to see whether they do, in fact, require the desired skills
 - Qualitative: ask students whether they feel that their experience in first-year has left them well-prepared for second year
- Are students deep-learning?
 - Qualitative: ask students about their study strategies and how they approach their assessment
 - Quantitative: apply an instrument such as the Approaches to Studying Inventory (Entwistle et al. 1994)
- Are there organizational factors that may be interfering with assessment or learning?
 - Qualitative: ask staff what, if anything, is preventing them from applying their ideal assessment techniques
 - Qualitative: ask staff and students what they think about assessment practices
 - Qualitative: ask staff what they consider to be good assessment
 - Quantitative: if any such factors are identified in interviews, see whether they are reflected in systematic patterns in student marks

In each case, the quantitative and qualitative methods support one another.

6.1 Empirical tests for reliability

There are a number of ways to evaluate the reliability of an assessment task based on test scores alone.

- test-retest: the same test is given to the same students twice. The reliability is the correlation between test scores. Although this is the easiest reliability measure to justify from a statistical point of view, it requires the imposition of an extra piece of assessment with no new content whatsoever and is therefore difficult to justify pedagogically or practically. Furthermore, in order to avoid a significant practice effect, students will need to have been “taught the test” – and concentrating on form rather than content is not usually a hallmark of pedagogical soundness. (Jordan 1953, p.27)
- split-test: the test is split into equivalent subtests, e.g. odd items and even items. Reliability is the correlation between marks on the subtests. This is hard to do properly, because the subtests need to be commensurate – that is, they must have similar facility and coverage. (Jordan 1953, p.28) It can be hard to calibrate this before the test has been administered to students. Moreover, not all assessment tasks in Computer Science lend themselves to this kind of treatment. It would probably be feasible to use the split-test method on the exam for CSE1303 Computer Science, which usually comprises a number of small questions, but not for the pracs in CSE2304 Algorithms and Data Structures, which are marked holistically. Because pracs are fortnightly and students are likely to improve in their programming ability, albeit at different rates, it is unlikely that the split-test method is appropriate across the board for evaluating reliability in Computer Science assessment.
- parallel forms: Similar to the test-retest method, but this time the second test is of a different form to the first. As with the test-retest method, applying this method gives an accurate value for reliability but is difficult to justify in practice. Because the two forms need to be commensurate, they are likely to be difficult to write for same reason as split-test forms are. (Stanley 1971, Jordan 1953, p.27)
- internal consistency measures. These are statistical measures such as Cronbach’s Alpha and the Kuder-Richardson 20. These measures are applied to student results after they have been collected, so no extra testing is necessary, but they are harder to calculate than other reliability measures. The Kuder-Richardson 20 is used when items are scored with one for a correct answer and zero otherwise, and so is more useful for selected-response tests (Stanley 1971, p.397). Although final exams in Computer Science may have a selected-response component, they also usually have a significant constructed-response component and so Cronbach’s Alpha is more appropriate here.

Cronbach’s Alpha is defined as: $\frac{N}{N-1} \times \frac{1 - \sum_i \sigma_i^2}{\sigma^2}$ where N = number of items, σ_i^2 = variance on i th item, σ^2 = test score variance. (Stanley 1971, p.399)

6.2 Other empirical quality metrics

Besides the reliability metrics above, there are two other metrics that can be applied to raw student marks for quality assurance purposes.

The *facility* of an assessment item, also called *difficulty*, is the percentage of the group that get the item right. For marks that are composed from the results of multiple items, facility is the average mark expressed as a percentage. (Athanasou 1997)

For a norm-referenced test, facility should be around 50% for each item. Criterion-referenced items may have much higher facilities if they refer to criteria that all students are expected to master. Mastery learning based computing units such as those described by Lister & Leaney (2003) and Box (2004) should have high facilities. Lister and Leaney set a minimum pass mark of 70% for their multiple-choice exams, and this was achieved by 71% of students at their first attempt. Box, applying the same design philosophy to a second-year course, found that only 17% of her students achieved that mark and had to drastically lower the pass mark to 53%. The students did achieve a 90% pass rate in the lab exam on their first attempt.

If the facility of an activity that is ostensibly norm-referenced turns out to be very high, it is not necessary to make the activity harder. There may be very good reasons for it: it may be placed after a particularly difficult task as a confidence booster, or it may be testing skills that all students really ought to have. More to the point, assessable tasks are often the main learning activity for Computer Science units. There may be good pedagogical reasons for the activity. It may, however, be appropriate to reframe the task as a criterion-referenced exercise in the model of Lister and Leaney.

The second metric that should be applied is *discrimination*. This can be approximated by the facility measured for the top 25% - facility measured for the bottom 25%, but is actually defined as the correlation coefficient between the mark on the item and the overall test score. Discrimination measures the item's ability to discriminate between high-achieving students and low-achieving students. It is therefore of more interest in norm-referenced assessment than criterion-referenced assessment. A discrimination over 0.4 is considered reasonable for norm-referenced assessment, but once again, low discrimination does not mean that an assessment task should be scrapped if it is needed for student learning.

6.3 Data collection and empirical analysis

I will gather the prac, exam, and mid-semester test marks for the first-year and first-semester second-year core subjects. It would be preferable to have a full question-by-question breakdown of the exam marks at least, but this would require access to the exam papers themselves.

This can be done using current or historical data, but I would prefer to use historical data. Firstly, it will be easier to obtain ethics approval for historical data. Secondly, I will be interviewing second-year students for the qualitative analysis and their first-year results will be historical. If the interviews are carried out in second semester, then the second-year results will be historical as well. If time permits, I can also request the results for the second-semester second-year core, but following the

cohort across three semesters is likely to narrow the field as the inevitable attrition occurs.

- **Reliability.** Validity is impossible without reliability; a measure that is unreliable cannot be said to be assessing its construct. Well-known empirical measures of reliability exist, so I will apply them to the data. If question-by-question marks are available, I can apply the split-test and possibly the Kuder-Richardson metrics. Cronbach's Alpha will be suitable for marks with less fine granularity. For all assessment, including any that is not suited to empirical measurement, I will seek qualitative evidence concerning the consistency of marking from everyone concerned. Lecturers and demonstrators will presumably both be actively involved and concerned with consistency, especially interrater consistency in programming labs, and students are likely to have interesting perceptions of the consistency of marking.
- **Facility and discrimination.** Activities with very high or very low facility may be poor assessment, although activities with high facility may simply be learning exercises with marks attached for motivational reasons. It will be necessary to ask the lecturers who set the task what their purposes are and what the balance between evaluative and educative assessment is. Some activities may turn out to be more naturally construed as criterion-referenced tasks.
- **Predictive validity.** How well do programming marks in first year predict programming marks at second year? I will repeat the process I used to generate the data for my seminar. Are some tasks better predictors than others? If so, then why? Are they more accurate assessors or simply more similar tasks (in which case perhaps more attention needs to be paid to training students to cope in the assessment environment). I will need to ensure discriminant validity: strength of correlation alone does not demonstrate validity. I will need to examine the tasks in the light of the skills the lecturers intend to assess and the interrelationships reported by staff and students, predict which tasks should correlate most strongly on the basis of the constructs they are believed to measure, and then determine whether the observed pattern of correlation matches the prediction.

6.4 Interviews

Although I can gather and analyse information using only historical marks data, I will need to speak to students and staff in order to work out the best interpretation of the analysis. These interviews will need to be open-ended. It is not possible to enumerate all the ways in which lecturers might feel frustrated in the assessment they can set, or the reasons students might misconstrue what learning strategies will help them the most. It is also not desirable to phrase questions too narrowly, or they will lead.

In the interview phase, I will investigate the consistency of approach between lecturers at first and second year level, the tutors and demonstrators who are re-

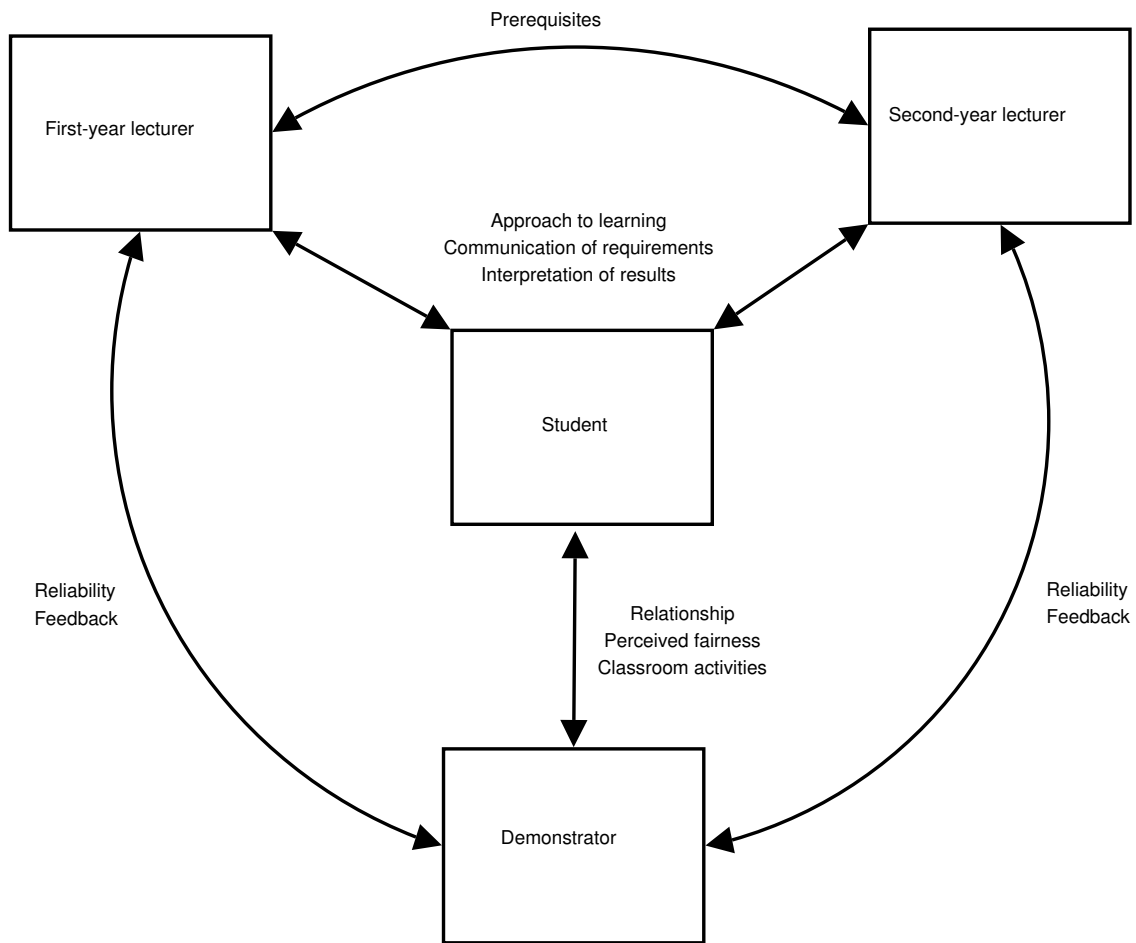


Figure 3: Relationships to explore in interviews

sponsible for a great deal of the student contact and the administration of the pracs, and the students themselves. Figure 6.4 shows the relationships of interest.

First-year lecturers teach introductory programming. Second-year lecturers make use of the knowledge and skills that incoming students are assumed to have gained in designing their course and its assessment. Do the assessment goals of the first-year course designers match the perceptions of the second-year course designers? This relates to the concept of validity as fitness for purpose.

Both first-year and second-year lecturers employ tutors and demonstrators, who do the bulk of the direct student contact. Demonstrators are responsible for marking pracs. As such, an evaluation of reliability must explore the mechanisms that are put in place to ensure interrater reliability between demonstrators. This will require asking the lecturers about the mechanisms and then asking the demonstrators how they are put into practice. Sometimes lecturers ask tutors to explain a particular point in class. Is this always carried out consistently?

The relationship between student and demonstrator is very important. Students get most of their feedback about the quality of their code from their demonstrator. Do students, lecturers and demonstrators agree on the role of the demonstrator in class? Demonstrators also have teaching responsibilities. What is the balance

between evaluation and explanation in class, and does it match the lecturer's understanding?

Most students undertake their class in good faith, and want to comply with what the lecturer wishes. Do students and lecturers agree on students' approach to learning? Are pracs rewarding the behaviours that lecturers wish to encourage, or are they rewarding other behaviours that may not lead to successful learning? Example: a task may be perceived by students as preferentially rewarding shallow-learning strategies, either through miscommunication or by naïve folk theories of learning. Whether this perception is accurate or not, it will probably lead to an impairment of learning if it leads students to adopt the wrong strategy.

6.5 Interview design

The interview questions must reflect the research questions, so it is appropriate to reprint them here:

1. Is the first-year assessment valid?
 - (a) to what extent do staff believe the assessment is valid, and why?
 - (b) is it reliable?
 - (c) are the assessment instruments good predictors of later performance?
2. Is first year teaching the skills required for second year?
 - (a) what skills are required?
 - (b) are the students learning these skills?
 - (c) are these skills being assessed?
3. Are students deep-learning?
 - (a) what impact does assessment have on student approaches to learning?
 - (b) do students perceive assessment practices as encouraging or discouraging particular approaches or study habits?
4. Are there organizational factors that may be interfering with assessment or learning?

The interviews must not only address relevant points, but must also interlink with the analysis of the numerical data. It will only be possible to explain patterns in the data by interpreting them in the light of the intended or perceived relationships between the assessment tasks and the units themselves.

6.5.1 Staff interviews

1. Assessment and skills
 - What assessment is given as part of the course? [**1a**; **1b**; **4**]

- What factors went into the selection of this assessment? [1a; 1b; 4]
- What assessment would you set in an ideal world? [1a]
- What skills does this assessment test? [1a; 1b; 4]
- What skills do you think are most important for students to take away from the course? What skills are required in subsequent courses? [2a]
- How are these skills assessed? [2c]
- Do you think the marks students achieve are a good reflection of their abilities? [1a]

2. Relationships with prerequisite courses

- What prior knowledge, skills and experience do students need in order to take the course? [2a]
- Do you think students are coming into the course with these skills? [2b]
- If not, how easy is it for them to catch up? [2b]
- What difficulties do students face who do not have this knowledge? [2a; N]
- Which tasks will they find most difficult? [2a; N]

3. Roles and coordination

- What do you consider to be the role of the demonstrator in prac classes? [4]
- What are demonstrators supposed to do in class?
- What do they actually do? If this is not what they are supposed to do, why do they do this?
- What should happen when a student seems close to failing – what should staff and students do? Do they usually do this? 4
- How do you communicate your expectations about how the class should operate to demonstrators? [4]
- How do you communicate them to students? [4]
- How do you help demonstrators be consistent in their marking? [1b; 4]

4. Student learning

- How much time would you prefer students to spend preparing for pracs and other assessments? [2b; 2c; 3a]
- How much time do you think they actually spend?
- How would you like students to approach learning in your subject?
- How do you think students actually approach learning in your subject? If this is not the way you would prefer, why do you think they do this? Does it have adverse effects on their understanding?

- What resources are available to students who are having difficulty with course material? Do students make use of these resources?
- What do you consider to be good study habits? Do you do anything to encourage students to study effectively? [3a]

6.5.2 Tutor/demonstrator interviews

1. Assessment and skills

- What assessment is given as part of the course? [1a; 1b; 4]
- What skills does this assessment test? [1a; 1b; 4]
- What factors do you think went into the selection of this assessment? [1a; 1b; 4]
- What skills do you think are most important for students to take away from the course? What skills will they need in subsequent courses? [2a]
- How are these skills assessed? [2c]
- Do you think the marks students achieve are a good reflection of their abilities?

2. Relationships with prerequisite courses

- What prior knowledge, skills and experience do students need in order to take the course? [2a]
- What difficulties do students face who do not have this knowledge? [2a; N]
- Which tasks will they find most difficult? [2a; N]
- Do you think students are coming into the course with these skills? [2b]
- If not, how easy is it for them to catch up? [2b]

3. Roles and coordination

- What do you consider to be the role of the demonstrator in prac classes? [4]
- What are demonstrators supposed to do in class?
- What do you actually do? What do other demonstrators do? If this is not what they are supposed to do, why do they do this?
- What should happen when a student seems close to failing – what should staff and students do? Do they usually do this?4
- How do you know what lecturers expect of you in class? [4]
- How do students know what is expected of them in class? [4]
- How do you ensure you are consistent in your marking, both with yourself and with other demonstrators? Do you get any support in this? [1b; 4]

4. Student learning

- How much time should students spend preparing for pracs? What should they do during this time? [**2b**; **2c**; **3a**]
- How much time do you think they actually spend, and what do they actually do? Why is this?
- What approaches to learning do you think will help students most in this class? What approaches do you think your students have?
- Do you do anything to encourage good study habits? If so, what? [**3a**]

6.5.3 Student interviews

1. Assessment and skills

- What assessment were you given as part of the course? [**1a**; **1b**; **4**]
- What skills do you think this assessment was supposed to test? [**1a**; **1b**; **4**]
- Why do you think your lecturer chose to give you this assessment? [**1a**; **1b**; **4**]
- What skills do you think are most important for students to take away from the course? What skills are required in subsequent courses? [**2a**]
- How are these skills assessed? [**2c**]
- Do you think the marks students achieve are a good reflection of their abilities?

2. Relationships with prerequisite courses

- What prior knowledge, skills and experience do students need in order to take the course? [**2a**]
- What difficulties do students face who do not have this knowledge? [**2a**; **N**]
- Which tasks will they find most difficult? [**2a**; **N**]
- Do you think most of your classmates came into the course with these skills? [**2b**]
- If not, how easy was it for them to catch up? [**2b**]

3. Roles and coordination

- What do you consider to be the role of the demonstrator in prac classes? [**4**]
- What are demonstrators supposed to do in class?
- What do they actually do? If this is not what they are supposed to do, why do they do this?
- What are students supposed to do in class?

- What do they actually do? If this is not what they are supposed to do, why do they do this?
- What should happen when a student seems close to failing – what should staff and students do? Do they usually do this? **4**
- How do demonstrators make sure they are consistent in their marking? Do you think these strategies work? [**1b**; **4**]
- What resources do you use when you are having trouble understanding something in class? Who do you ask for help?

4. Student learning

- How much time should students spend preparing for pracs and other assessments? [**2b**; **2c**; **3a**]
- How much time do you actually spend? Why is this?
- What do you think is the best way to study, and why?
- How do you handle it when you have a lot of assignments that are due at the same time?

7 Timetable

December 2005: refine interview plans and prepare Human Ethics application

4 January, 2006: submit Human Ethics application to SCERH

31 January, 2006: SCERH meets

February – March 2006: contact academic staff; arrange and conduct interviews. (Depending on availability, this process may need to be extended.) Obtain and analyse available data for the 2004-5-6 student cohort.

Semester 1, 2006: recruit students and tutor/demonstrators; conduct interviews. Commence writeup of literature review chapter (ongoing).

Semester 2, 2006: Analyse and categorise interview data. Commence writeup of methodology and results chapters.

Summer break, 2006: Prepare conclusions and recommendations. Finalize lit review, methodology and results. Rewrite introduction in the light of findings.

Semester 1, 2007: first draft to supervisors. Revision cycle to proceed until thesis is acceptable.

Note: This timetable is tentative. If the interview process or the analysis of student marks uncovers a new line of investigation, then the timetable will have to be extended so that this research can be pursued. The first draft deadline has been set conservatively for this reason.

References

Allan, V. & Kolesar, M. V. (1997), 'Teaching computer science: a problem solving approach that works', *SigCUE Outlook* **25**(1-2), 2–10.

- American Educational Research Association, American Psychological Association & National Council on Measurement in Education (1996), *Standards for educational and psychological testing*, American Psychological Association.
- American Educational Research Association, American Psychological Association & National Council on Measurement in Education (1999), *Standards for educational and psychological testing*, American Educational Research Association.
- Athanasou, J. A. (1997), *Introduction to educational testing*, Social Science Press.
- Barros, J. a., Estevens, L., Dias, R., Pais, R. & Soeiro, E. (2003), ‘Using lab exams to ensure programming practice in an introductory programming course’, *ITiCSE 2003* pp. 16–20.
- Beauboeuf, T., Lucas, R. & Howatt, J. (2001), ‘The UNLOCK system: Enhancing problem solving skills in CS1 students’, *SIGCSE* **33**(2), 43–46.
- Box, I. (2004), Object-oriented analysis, criterion referencing, and Bloom, in ‘ACE2004’, Vol. 26 of *Australian Computer Science Communications*, pp. 1–8.
- Bucci, P., Long, T. J. & Weide, B. W. (2001), ‘Do we really teach abstraction?’, *SIGCSE* pp. 26–30.
- Califf, M. E. & Goodwin, M. (2002), ‘Testing skills and knowledge: introducing a laboratory exam in CS1’, *SIGCSE 2002* pp. 217–221.
- Chamillard, A. & Braun, K. A. (2000), ‘Evaluating programming ability in an introductory computer science course’, *SIGCSE 2000* pp. 212–216.
- Daly, C. & Waldron, J. (2004), ‘Assessing the assessment of programming ability’, *SIGCSE 2004* pp. 210–213.
- de Raadt, M., Toleman, M. & Watson, R. (2004), ‘Training strategic problem solvers’, *SIGCSE* **36**(2), 48–51.
- Entwistle, N., Tait, H. & Speth, C. (1994), *Approaches to Studying Inventory* (revised). Centre for Research on Learning and Instruction, The University of Edinburgh.
- Feldt, L. S. & Brennan, R. L. (1993), *Educational Measurement*, American Council on Education Series on Higher Education, Oryx Press, chapter Reliability, pp. 105–146.
- Hodson, D. (n.d.), ‘Assessment of practical work’, *Science and Education* **1**(2), 115–144.
- Jacobson, N. (2000), ‘Using on-computer exams to ensure beginning students’ programming competency’, *SIGCSE 2000* **32**(4), 53–56.
- Johnson, B. & Christensen, L. (2004), *Educational Research: Quantitative, Qualitative and Mixed Approaches*, second edn, Pearson. Excerpted in “Embarking on Research: EDF6001 Reader Part 1”, Monash University, 2005.

- Jordan, A. M. (1953), *Measurement in Education: An Introduction*, McGraw-Hill Series in Education, McGraw-Hill.
- Kane, M. T. (2001), 'Current concerns in validity theory', *Journal of Educational Measurement* **38**(4), 319–342.
- Lister, R., Adams, E. S., Fitzgerald, S., Fone, W., Hamer, J., Lindholm, M., McCartney, R., Moström, J. E., Sanders, K., Seppälä, O., Simon, B. & Thomas, L. (2004), A multi-national study of reading and tracing skills in novice programmers, in 'ITiCSE 2004 Working Group Report', Vol. 36, pp. 119–150.
- Lister, R. & Leaney, J. (2003), Introductory programming, criterion referencing, and Bloom, in 'Proc. SIGCSE '03', pp. 143–147.
- Lorenzen, T. (1981), 'The case for in class programming tests', *SIGCSE* **13**(3), 35–37.
- McCracken, M., Almstrum, V., Diaz, D., Guzdial, M., Hagan, D., Kolikant, Y. B.-D., Laxer, C., Thomas, L., Utting, I. & Wilusz, T. (2002), 'A multi-national, multi-institutional study of assessment of programming skills of first-year CS students.', *SIGCSE 2002* pp. 125–140.
- McNamara, R. (2002), Concept mapping for introductory programming.
- McNamara, R. (2004), Evaluating assessment with competency mapping, in R. Lister & A. L. Young, eds, 'Proc. Australasian Computing Education 2004', Vol. 30 of *CRPIT*, pp. 193–199.
- Messick, S. (1993), *Educational Measurement*, 3rd edn, Oryx Press, chapter Validity, pp. 13–103.
- Moss, P. A. (1994), 'Can there be validity without reliability?', *Educational Researcher* **23**(2), 5–12.
- Ramalingam, V., LaBelle, D. & Wiedenbeck, S. (2004), 'Self-efficacy and mental models in learning to program', *ITiCSE 2004* pp. 171–175.
- Salvia, J. & Ysseldyke, J. E. (1995), *Assessment*, 6th edn, Houghton Mifflin.
- Sambell, R. & McDowell, L. (1998), 'The construction of the hidden curriculum: Messages and meanings in the assessment of student learning', *Journal of Assessment and Evaluation in Higher Education* **23**(4), 391–402.
- Sheard, J., Dick, M., Markham, S., Macdonald, I. & Walsh, M. (2002), 'Cheating and plagiarism: perceptions and practices of first year IT students', *ITiCSE 2002* pp. 183–187.
- Snyder, B. R. (1973), *The Hidden Curriculum*, MIT Press.
- Stanley, J. C. (1971), Reliability, in R. Thorndike, ed., 'Educational Measurement', second edn, American Council on Education, chapter 13, pp. 456–442.

- Thomson, K. & Falchikov, N. (1998), “Full on until the sun comes out”: the effects of assessment on student approaches to study’, *Journal of Assessment and Evaluation in Higher Education* **23**(4), 379–390.
- Werner, L. L., Hanks, B., McDowell, C., Bullock, H. & Fernald, J. (2005), ‘Want to increase retention of your female students?’, *Computing Research News* **17**(2).