School of Computer Science and Software Engineering
Monash University

Bachelor of Computer Science (Honours)
Final Draft  2002

Concept mapping for introductory programming

by

Robyn McNamara
ram@csse.monash.edu.au

Supervisor: Dr Linda McIver
lindap@csse.monash.edu.au
Abstract

How do we know what students get out of an introductory computer programming course? Computer programming skills are notoriously difficult to assess fairly and economically. This thesis explains some of the problems inherent in teaching and assessing in the discipline, and presents a method — competency mapping — which extracts a better and more detailed picture of what students actually learn, without requiring them to sit extra assessment activities. Competency mapping is applied to the set of marks from an introductory course in computer programming, and a mathematical model for interpreting assessment is introduced to explain the resulting maps.

It was hoped that competency mapping would shed some light on the way novice computer programmers mentally structure their knowledge about the discipline, but the experiment was only a partial success owing to problems with the design of the original assessment. Some insights were, however, gained into the way students learn computer programming. Guidelines for designing assessment to work better with competency mapping are also provided, as well as a design for a dedicated competency mapping tool.
Contents

1 Introduction and Background .............................................. 1
  1.1 Introduction ....................................................... 1
  1.2 Background: education ............................................. 2
    1.2.1 Constructivism .............................................. 2
    1.2.2 Requisites for learning ..................................... 2
    1.2.3 Computer science education .................................. 3
    1.2.4 The university and its stakeholders ......................... 4
    1.2.5 The challenges of a new discipline .......................... 5
  1.3 Assessment .......................................................... 6
    1.3.1 Taxonomy of assessment ...................................... 6
    1.3.2 Introductory CS Assessment methods .......................... 7
    1.3.3 Challenges of assessment ...................................... 7
  1.4 Background: Concept maps .......................................... 8

2 Aims ............................................................................. 11
  2.0.1 Competency mapping .............................................. 12
  2.1 Benefits of competency mapping ..................................... 12
    2.1.1 Benefits for staff .............................................. 13
    2.1.2 Benefits for students .......................................... 14

3 Method and Results ....................................................... 16
3.1 CSE1301 assessment tasks .................................................. 16
3.1.1 Choice of activities ......................................................... 17
3.1.2 Data validation .............................................................. 17
3.1.3 Choice of datasets ........................................................ 18
3.1.4 Factor analysis, cluster analysis and multidimensional scaling . 19
3.2 Results ............................................................................. 22
3.2.1 Competency maps ........................................................... 23
3.2.2 Set ALL ........................................................................ 24
3.2.3 Set AMAB .................................................................... 25
3.2.4 Set TMA ........................................................................ 26
3.2.5 Set TMAB ..................................................................... 27
3.2.6 Set BMAB ..................................................................... 28
3.3 Random data .................................................................... 28
3.3.1 A model for competencies ............................................... 29
3.3.2 The random datasets ...................................................... 30
3.3.3 Performance ................................................................. 31

4 Analysis and Conclusions ..................................................... 33
4.1 Factor analysis ................................................................. 33
4.2 Cluster analysis ............................................................... 34
4.3 Methodological problems .................................................. 35
4.4 A better test ...................................................................... 37
4.5 Conclusion ...................................................................... 38
# List of Figures

1.1 Sample concept map: Cornell University Health Service

3.1 Scree plot, set ALL ................................. 22
3.2 Scree plot, set AMAB ............................... 22
3.3 Scree plot, set TMA ................................. 23
3.4 Scree plot, set TMAB ............................... 23
3.5 Scree plot, set BMAB ............................... 23
3.6 Competency map, set ALL .......................... 24
3.7 Competency map, set AMAB ....................... 25
3.8 Competency map, set TMA .......................... 26
3.9 Competency map, set TMAB ....................... 27
3.10 Competency map, set BMAB ...................... 28
3.11 Orthogonal, four competencies ................... 31
3.12 Orthomodal, 0.25/0.25 modal competencies ........ 31
3.13 Orthomodal, 0.75/0.25 modal competencies ........ 32
3.14 Mixed, 4 competencies, 3 per question .......... 32

4.1 Competency mapping tool architecture ............ 37
Chapter 1

Introduction and Background

1.1 Introduction

It has always been difficult to teach computer science to beginners. Part of the problem is that there are few ways to assess a student’s programming ability that are both practical and reliable. There seems to be broad agreement that new ways to assess programming skills are needed, but many of the proposed solutions have their own problems with fairness, reliability or cost-effectiveness.

Concept mapping, a technique for representing conceptual structures as two-dimensional maps, can be used to represent students’ understanding of course material, but the use of traditional concept mapping techniques would involve a substantial amount of organisation. Depending on the method chosen, it may also require a lot of labour to administer or to interpret, rendering it unsuitable.

This thesis presents a technique for generating competency maps — structures similar to concept maps, but covering practical abilities as well as theoretical background knowledge — from the marks for the assessment tasks that students already undertake. The new technique uses well-understood statistical methods that are widely implemented in off-the-shelf software, but are not too difficult to reimplement should a dedicated tool be needed. It simply makes better use of the data universities already collect, without much overhead on staff and without any extra assessment burden on students.
1.2 Background: education

It is easy to take the practice of teaching for granted. Teaching is often misrepresented as being a simple matter of presenting information to students. Such a view underestimates what is required of both student and teacher. In reality, teaching and learning are complex social and cognitive processes, and educational practice is the subject of a thriving research literature.

To understand what is going on in this literature, we will need to look at the prevailing educational theory: constructivism.

1.2.1 Constructivism

Constructivism is one of the most important theories of learning today. Essentially, constructivism says that students start out with some kind of mental framework (which may be sketchy), and assimilate new knowledge by slotting it into place within that framework (Ben Ari, 2001). In other words, students understand new information only in relation to information that they already know. This implies in turn that the teacher must not simply transmit information, but help learners to integrate and assimilate the new information, recognise when a learner has constructed a model that is inconsistent with it, and explain and repair the resulting misconceptions.

1.2.2 Requisites for learning

There are a few obvious prerequisites that need to be in place in order for learning to be effective. Students need to feel safe to explore their environment. This is a problem in computer science, because many students enter the discipline with little previous computing experience and, like many novice computer users, may fear that misusing the computer will somehow break it.

Similarly, students need to feel socially safe: they must not feel vulnerable to loss of face. This is always a challenge; few people enjoy being wrong, and yet the experimentation and exploration that are necessary to construct a new mental model often takes students down a few blind alleys and wrong turnings before they get it right. If students feel that they will be open to ridicule or exposure, then it will be much harder for them to attain deeper levels of understanding.
1.2.3 Computer science education

Computer science is a new discipline, and has been taught as a University subject in its own right for less than the lifetime of many of its practitioners. Although there is a large literature in computer science education, especially concentrating on the teaching of programming, most of it is concerned with the choice of programming language ([Jarc, 1992; Howland, 1997]), or examines particular teaching or assessment practices (Carbone and Kaasbøll, 1998; Chamillard and Joiner, 2001; Hagan and Sheard, 1998). These authors are usually informed by extensive teaching experience, but rarely tie the practice into a theoretical framework. The authors that treat theory (Powers and Powers, 1999) are rarely specific about classroom practice. It seems that the foundation theories of computer science education are still under construction.

Constructivism in computer science education

As Ben Ari (2001) points out, the nature of introductory computer programming courses makes them highly suitable to the constructivist model. In general, students entering such courses have limited exposure to programming, and therefore lack preexisting misconceptions. In comparison, students usually enter introductory Newtonian physics handicapped by a lifetime spent under gravity and in the constant presence of friction: naive physics is Aristotelian, characterised by the belief that objects will stop moving in the absence of force, but naive computer programming is practically nonexistent. Moreover, it is usually easy for the student to explore the world of computer programming. Testing a new hypothesis about how the compiler will behave gets almost instant feedback. Again, this is not the case in physics: practical physical experiments usually require a degree of setup, measuring and calculation. Tam (2000), writing from the perspective of distance learning, also gives constructivist arguments that support the use of computers as a medium of instruction even in non-computing contexts.

Experience does, however, suggest one domain of prior experience that may interfere with the successful construction of a model: language. Most computer languages are based on human languages, at least in vocabulary, and computer programs do resemble a list of instructions that one might give another person, so it is only to be expected that students will unconsciously make the same assumptions when programming that they do in speech. And yet, computer languages are very different to human languages. When we interpret language, we rely heavily on context to disambiguate references: they are usually taken to refer to the most salient object that matches the description, either in the environment, the prior conversation, or in the body of knowledge shared by the speaker and the hearer (Clark, 1996). To date,
no computer language has been developed that can interpret the way we do — and such power of interpretation would be undesirable in any case, since that flexibility gives rise to the ambiguities in natural language! Several authors have commented on the problems that arise from novices’ inappropriate extensions of natural language (Spohrer and Soloway, 1986; du Boulay, 1989).

Therefore, the steep learning curve that is the hallmark of introductory programming can be explained as the foreseeable result of the students’ having to construct their own mental model of the discipline, with little interference from prior programming knowledge but some interference from prior natural language knowledge. Computer programming teachers can help them in this construction by helping them orient themselves in the new domain, and by helping them to identify and overcome any misconceptions they might acquire. Mayer (1989) gives evidence that a broadly constructivist educational technique — explicitly providing students with a “model”, or descriptive analogy, for the constructs they are learning — has a positive effect on deep learning.

1.2.4 The university and its stakeholders

Education does not take place in a vacuum. There are many bodies that have an interest in the teaching process, and each body needs to be satisfied that the whole business has been worthwhile. Before the outcomes of the education process can be evaluated, it is a good idea to examine the needs of the people who have a stake in it. Assessment can only be evaluated in this wider context: does it actually do what we need it to do?

Many groups have an interest in the way universities conduct their teaching, and each has a slightly different perspective on university activities. The most obvious group is the student body. Students enter university for many different reasons, but it is safe to say that they wish to be trained for a career, to be treated fairly, and not to be worked too hard. Although the student body is heterogeneous, most students do not have much experience with the discipline in which they are being trained, nor do they necessarily know much about how universities are run — either administratively or educationally.

The university staff, both academic and general, must also be considered stakeholders. Staff who teach follow-on courses need to know what level of ability they can expect from incoming students who have completed prerequisite courses, and administrative staff need to ensure that resources are being well used. University policy-makers wish to ensure that the university’s reputation remains high, both as a research institution and as a teaching institution.
Universities are also responsible to the industry bodies — both to the future employers of their graduates and to professional associations such as the IEEE or the ACM. Employers expect a certain level of workplace competence in their graduates, and professional bodies need to scrutinise the quality of the courses taught in order to accredit them. In Australian public universities, the government is also a major stakeholder: as the university’s primary source of funds, it needs to be assured that its funds are well-spent. The wider community is also entitled to expect competence and professional standards of behaviour from university graduates.

University courses must address the concerns of all these stakeholders. It can be particularly difficult to evaluate how well a course is working: for example, student evaluations are often used, but as mentioned above, students might not know their discipline well enough to form an informed opinion on the subject. Counting the absolute financial cost of the course is not sufficient to assess its worth: the benefits of the course need to be counted as well. Even the pass rate of a subject is meaningless in itself, without information on the course content and the methods and standards of assessment.

1.2.5 The challenges of a new discipline

As well as the demands listed above, which are common to all university courses, computer science faces some challenges all its own. The youngest of the sciences, it has achieved a great deal of publicity. Computers are now found in most workplaces and many homes, and newspapers breathlessly report the achievements of computer manufacturers and software development companies. At the same time, computer science and computer programming are not well understood in the general community.

The high profile of computing in the general press is probably one of the factors pushing demand for computer programming courses. Despite the increasing number of people qualifying, demand for qualified programmers, although not as strong as it was at the height of the dot.com boom, is still much greater than the supply of graduates. Denning (1999) and McKeown and Farrell (1999) comment that this industry demand draws potential academics away from research and education. The fact that this situation was not an artifact of the Web-driven 1990s is demonstrated in Curtis (1983).

The community’s lack of understanding of computer sciences causes another problem for introductory courses. Learning computer science is not simply a matter of knowing how to program. Constructivism would lead us to expect that programming ability rests on some conceptual basis that needs to be built before the skill transfer can be effective. Furthermore, a competent programmer needs to be able to do many other
things than program: the speed with which the basic components of the programmer’s work environment — operating system, hardware, languages, applications — succeed one another implies that the ability to retrain will be most important. This means that the programmer needs to acquire sufficient conceptual background to be able to extend his or her knowledge without assistance from others, which will entail the assimilation of a lot of abstract theory. However, graduates often come into computer science courses with the expectation that programming will be the sole focus of their course. This disjunction between student expectations and the reality of the computer science syllabus is arguably a cause of student attrition (Sanders and Mueller, 2000).

1.3 Assessment

We have seen that assessment is critical to ensuring that the university meets its obligation to its stakeholders. Furthermore, computer science assessment faces extra challenges in that it is a relatively new discipline. We will now examine assessment in greater detail.

1.3.1 Taxonomy of assessment

Assessment is classified as either formative or summative, depending on its purpose and the uses to which its results are put. Formative assessment is student-centred, in that its main purpose is to give feedback to students. In other words, it helps students to form a picture of the course content and where they stand in relation to it. Students can then plan their study more efficiently, and will be able to make better use of the resources provided to help them. The verbal feedback a demonstrator gives to a student can be considered formative assessment.

Formative assessment is therefore one of the ways in which a university discharges its obligation to students.

Summative assessment is designed to summarise a student’s knowledge of the course material. It is one of the means by which universities fulfil their obligations to industry and to professional bodies, ensuring that a student who does not meet minimal standards do not pass. When “assessment” is mentioned, the examples most people call to mind from their school years are largely summative: exams, tests, essays and quizzes.

As a rule of thumb, any assessment which counts toward the final subject mark is summative, while any assessment for which the student receives significant feedback
is formative. Obviously, very few assessment activities are purely summative: that would imply that the students are never even told their mark.

1.3.2 Introductory CS Assessment methods

A wide range of assessment methods have been used in computer science courses. Exams are used almost everywhere for summative assessment, and pracs are also popular (Knox and Woltz, 1996; Chamillard and Joiner, 2001).

Multiple choice testing is controversial. Although it is widely and justly criticised\textsuperscript{1}, it is also widely used. It is one of the very few assessment strategies that can be assessed accurately by a computer, so it is ideal for courseware delivered over the World Wide Web.

Formative assessment must not be neglected. Hagan and Sheard (1998) have demonstrated that group discussion classes have a positive impact on student results. Related to group discussion is peer learning (Wills et al., 1999), one of the so-called “alternative assessment methods”, in which students work in groups and may comment on each other’s work.

1.3.3 Challenges of assessment

We have seen that students need formative assessment if they are to maximise their chances of success in the course. Yet it has been shown many times that students tend to deprioritise assessment that is not summative. This tendency, called “selective negligence”, was first described in Snyder (1973). This influential book documented the disparity between academic staff’s stated ideal of engaged, imaginative learners and the cram-and-forget strategies that students found actually worked in their weekly quizzes. Snyder saw this “hidden curriculum” as a property intrinsic to the institution, but Sambell and McDowell (1998) point out that students come to university with a great deal of experience with assessment, and that this prior experience informs their attitudes to their tertiary study.

Selective negligence may cause students to ignore activities that do not count toward the final mark, but it has been shown that making all assessment summative also has negative effects on students’ study habits. Thomson and Falchikov (1998) found that, if students feel time pressure, they are more likely to adopt surface learning strategies

\textsuperscript{1}For example, Paxton (2000) points out that a student who is solely assessed by multiple choice can achieve a qualification without ever having engaged in the discourse of their discipline
such as rote memorisation. This is true whether the time pressure was imposed by poor planning on the part of the teaching staff or by poor time management on the part of the students.

Good assessment must therefore be sufficiently attractive to induce the students to pay attention to it, while not being so cumbersome as to impose an undue burden on their time. This is particularly true of formative assessment.

Good assessment must also be equitable: students should not be disadvantaged on the basis of gender, cultural background, disability, or any other grounds not directly related to merit in the subject. Much work has been done on addressing the perceived disadvantages women face in computer science (Cohoon, 1994; Clarke and Teague, 1994; Carter and Jenkins, 1999; Brown et al., 1997), while Paxton (2000) discusses some cultural issues raised in South African universities.

Evaluating assessment also poses a problem. It is easy to show that a given task produces consistent results, but it is difficult to prove that the task is actually testing the concept that it is supposed to test. Many assessment activities do not give much feedback to the course designer, especially if the course designer is not the marker.

Despite all the hyperbole one hears about academia versus the “real world”, universities also have real financial and practical constraints on their operations. Examiners, lecturers, invigilators, demonstrators and markers all have to get paid, and therefore cost must be considered when designing an assessment task. Universities also have a limited supply of equipment and teaching space. An assessment strategy that is otherwise perfect will not be workable if it is too expensive, or if it requires resources that the university does not have and cannot readily obtain.

What is needed is a method that can determine what students know: not only its content, but its structure. This method needs to be able to be implemented without undue effort on the part of the students, and without entailing a lot of expense for the university. This thesis presents such a method.

1.4 Background: Concept maps

Concept maps were invented by Joseph Novak in the 1960s for use as a teaching tool. They are quite simple: labelled boxes represent concepts in a syllabus, and lines or arrows denote relationships between the concepts. If students develop a concept map at the start of a course, then teaching staff will have a better idea of their preexisting conceptual framework. Teachers can also present a course syllabus in the form of a concept map, showing how the ideas being taught are interrelated. Students can
also use concept maps as a notetaking tool, to represent the information in an article or to depict the structure of a novel. It is clear how these activities fit in with a constructivist view of the teaching process.

William Trochim (1986) later developed the concept map into a strategic planning tool for use in the design of organisational components. Trochim’s technique differs significantly from Novak’s original idea in that, while Novak’s maps are generated for one person as a means of communicating complex ideas to many, Trochim’s are generated by many people as a means of developing complex ideas. The sample concept map at figure 1.1 is from Trochim (1989) and was generated by stakeholders in Cornell University’s Health Services.

In Trochim’s method, a group of participants are collected that have some stake in the organisation that is planned. Initially, participants brainstorm to compile a list of concepts, which are then written on separate index cards. Each participant then
sorts the cards into piles of related concepts. It is important that no constraints are placed upon the participants’ sorting.

The results of the sorting are then tabulated, and a correlation matrix $M$ is created from those results. If concepts $i$ and $j$ were grouped together by $N$ participants, then $M_{i,j} = N$. The grouping relation is commutative, so the matrix $M$ will be symmetrical.

Cluster analysis is then carried out on the correlation matrix to group the concepts into categories, and the categories are examined by the participants to see what the concepts in each have in common. Multidimensional scaling (MDS) is then applied to generate a 2D image of the domain, in which the degree of correlation is inversely proportional to the distance between the categories. This map can then be used to design new organisational structures, to define the responsibilities of the new organisation, to communicate its purpose to clients, and for many other purposes.
Chapter 2

Aims

The aim of this project is to use concept mapping principles to develop a tool that can be used to improve formative assessment and assist in course evaluation for introductory computer programming without adding significantly to the course load on students and without undue cost to the university, either in labour or in materials. A student-derived concept map of introductory computer programming would certainly help with assessment and evaluation. By showing the degree of relatedness of parts of the course as the students see it, it would expose any "missing links": parts of the course which should be related, but which are not close together on the student-derived concept map. Such missing links may indicate that students are failing to construct part of the course correctly, because they are mentally pigeonholing concepts that should be unified. The syllabus can then be refined, to make the connections between these topics more explicit. In effect, course designers can use the results of concept mapping to provide assembly instructions for the construction of new knowledge.

Such a concept map could be generated by the students according to Trochim’s method. However, organising the brainstorming sessions that would be required would be painful: if the activity is not summative, the students are unlikely to do it; yet it is difficult to see how it can be made summative. Moreover, any activity needs to be considered carefully before being added to the course. If students feel pressured, their learning tends to suffer as they forsake deep learning strategies for surface learning (Thomson and Falchikov, 1998). Furthermore, overseeing the collation, validation and entry of several hundred sets of grouping information would be a daunting and labour-intensive task. It seems that Trochim’s method would not achieve the feasibility goal.
But universities already collect and store a great deal of information about how students conceptualise their subjects: student results. Obviously, if two exam questions are testing the same basic concept then the marks for those questions should show a positive correlation, unless something has gone drastically wrong with the delivery of the question. This brings us to my key insight: it should be possible to work backwards and infer conceptual relatedness from strength of correlation. Multidimensional scaling and cluster analysis could be applied to correlation matrix calculated from students’ marks, producing an empirically-derived concept map.

2.0.1 Competency mapping

When the technique is applied this way, it is not strictly correct to call it “concept mapping” The term implies that what is being measured is purely conceptual, but no assessment method captures pure conceptual understanding: there is always a practical element. Furthermore, computer programming is not in itself a pure theory subject. The ability to write useful, efficient and correct programs is more of a craft than a science. Because practical abilities such as proficiency with the compiler are likely to affect students’ marks, it is to be expected that they should show up on the map — indeed, for some tasks they might dominate the clustering pattern.

“Competency mapping” is a better phrase. A competency is a practical ability that is informed to some degree by theoretical understanding: this is a much better model for assessment than any purely conceptual model. Although the aim of the assessment may be to work out the students’ grasp of a concept, what is measured is of necessity the result of some practical process.

2.1 Benefits of competency mapping

Competency maps have many potential benefits for students and teaching staff. Of course, because staff and students share many goals, these benefits are not entirely divisible; some aspects of competency mapping will benefit both staff and students. A partial list of potential uses for competency mapping follows. It is likely that more benefits will be discovered as the technique matures.
2.1.1 Benefits for staff

If competency mapping can actually give a picture of the structure of the course as the students experience it, teaching staff will be able to use that picture as the basis for course refinement. The identification of key concepts is the first step towards designing a syllabus. The information gained can also be published to the students, for example by including it in the subject information handout that students usually receive in their first lecture, or by putting it on the courseware web page.

Of course, it is quite possible that the structure revealed by analysis of student results does not match the lecturer’s idea of the conceptual structure of the course. In this case, the revealed structure may suggest ways in which the course can be improved. For example, if two competencies that should be related (for example, C pointers and passing by reference) are not clustered together, it could indicate a need to make the connection more explicit to the students. If the competency map uses all the coursework marks as input, this will not help the students of that year; however, it may well help teaching staff to refine the coursework for the next delivery of the course. It would also be useful to staff who are teaching follow-on courses, as they would gain a better idea of which topics need revision.

A competency map using only the marks for half of the course can be produced if staff wish to refine the course on the fly, but care must be taken that the data are sufficient: if the only marks on record are the first six prac marks, it is unlikely that any useful conclusions can be drawn. It is not yet certain how many points are needed for competency mapping to be useful, but it is likely to depend on the amount and complexity of the course material.

These uses assume that competency mapping will elucidate the structure of the course. If, however, the technique does not do this, then there are still potential benefits: logically, we would expect that activities that test strongly related competencies should show correlations in their marks; if this is not the case, there must be some reason. For example, written exam questions about linked lists might not correlate strongly with practical questions about linked lists if success in pracs is more closely related to factors other than subject knowledge. This could be the case if the some students find their work environment — operating system, compiler and editor — difficult to use. In this case, prac questions will tend to cluster much more strongly with other prac questions, and much less strongly with theory questions. The competency map can show that there is a problem; it is then up to the teaching staff to investigate that problem. Of course, competency mapping over subsequent years of the course will help the staff know when they have ameliorated the problem.
In order to satisfy Ethics Committee requirements, this project only uses deidentified marks data and no demographic information is available. However, in a university setting, competency mapping can be used to compare demographic subsets of students to verify equity of access. If it is suspected that there is a systematic problem with some students’ access to education, for example if there is concern that students of non-English speaking background are finding a particular activity especially difficult because of the complex language used to explain it, then competency mapping can be applied separately to the results from students belonging to that group and the results compared to a competency map derived from the marks of the rest of the student body. In this case, a problem with English would result in a distorted cluster arrangement: written-answer questions and questions with complex requirements would tend to cluster together. The technique may also be used to determine whether female students conceptualise the subject differently to male students. Again, if a problem is found, competency mapping over subsequent years will show staff whether the remedies are working.

2.1.2 Benefits for students

The primary benefit of competency mapping for students is the increased understanding of the student viewpoint that the teaching staff will have, and the resulting likely course improvements. However, students should also benefit directly from it.

A constructivist view of the teaching process suggests that students will assimilate new knowledge and gain new skills more readily if they can be made aware of how those new competencies interrelate with knowledge and skills that are already mastered. Of course, lecturers know this; most new topics begin with an explanation of the new material in relation to material already seen. However, this explanation is almost always exclusively verbal. Information about relationships is often best presented in visual form, especially if the relationships are multidimensional: pictures are two-dimensional, but words are one-dimensional, strictly ordered in time. Therefore, having access to a two-dimensional map of the course structure may help students construct their understanding of the course material.

If it is possible to use competency mapping to break the subject down into components that are close to orthogonal, it should also be possible to design assessment on the basis of that breakdown. Once the components are known, assessment tasks can be designed that test them individually, or (since it is virtually impossible to test anything in isolation) as close to it as possible. Thus a test can be delivered to students that is quite small, but gives results that are interpretable in terms of the course’s competency map.
Because competency mapping measures correlations between task marks across students, it is obviously impossible to generate a competency map based on a single student’s data; however, numeric results can be presented alongside the group competency map — for example, by shading regions that correspond to topics that the student needs to work on. In this way, a student may be able to use her test results to determine her own weaknesses, and then consult the map to see how they relate to the rest of the course: using this map and compass, she may find it easier to navigate through the material.

If she still has trouble understanding the material, she may ask a staff member for help. In this case, if the staff member has access to her test results, it would be easier to pinpoint the misconception that is at the heart of the problem. Experience shows that determining the problem is almost always harder and more time-consuming than solving it; figuring out what needs to be explained is more difficult than developing an explanation, especially considering that teachers can develop a set of explanations that work and re-use them. This means that the student need not worry as much about coming to consultation, and (because consultation time can be used more effectively) the teaching staff are more likely to be free to help her.
Chapter 3

Method and Results

To generate a concept map, cluster analysis and multidimensional scaling are applied to proximity data generated from the number of times concepts were clustered together. Competency maps are generated in a similar way: after student marks data is collected, cluster analysis and multidimensional scaling are applied to proximity data generated from the matrix of correlations between the marks. We were able to use SPSS for the generation of the correlation matrix, the cluster analysis and the multidimensional scaling.

The first stage in the construction of the competency map was the acquisition of student marks data. Ethics committee approval was sought and gained to use the results from CSE1301 Computer Programming, semester 1, 2001. This is the first computer science subject that students undertake in a Computer Science degree at Monash University. Between three and four hundred students each year undertake this subject.

3.1 CSE1301 assessment tasks

Students sitting CSE1301 in 2001 were required to undertake 14 distinct summative assessment tasks:

- A practical component comprising 12 weekly prac, which were marked in situ by the students’ demonstrators (lab tutors). This component was worth 30% of the final mark. Six of the prac also contained some bonus questions, whose marks were recorded separately.
• A mid-semester test, worth 10% of the final mark.
• A 28-page final exam, worth 60% of the final mark.

3.1.1 Choice of activities

Activities were chosen for inclusion in the study on the basis of availability. The aim was to maximise the number of activities included, taking ease of availability and likely usefulness into account.

The prac and bonus marks were easy to get, as they were entered into a database at the time of marking.

The exam and the mid-semester test were delivered on paper, and although it would have been easy to acquire an electronic list of total marks, it is impossible to infer much conceptual structure from a single numeric mark. The finer the granularity of measurement, the better the chances of being able to infer conceptual structure. Therefore, the physical exam papers were acquired, and the page totals were transcribed from the front of each exam. This was a time-consuming task, so it was decided not to repeat the process with the mid-semester test and therefore the test results are not included in this study. This still left 12 prac marks, 6 prac bonus marks, and 28 exam pages — a total of 46 marks per student, which was felt to be ample.

The prac and exam marks were combined into a single file, one line per student and one column per activity. At this time, the data was stripped of identifying features in accordance with Ethics Committee guidelines. Therefore, no breakdown on student demographics (for example, gender or ethnicity) was possible using this data. It was, however, possible to break students down according to their final mark.

The activities that were used comprise 12 prac, ranging from an introduction to using the computer to an advanced assignment; six bonus prac questions, which were not compulsory and were not attempted by many students; and 28 exam pages. These activities are described in more detail in Appendix II.

3.1.2 Data validation

Before analysis began, the data was vetted to ensure that the marks values were sane. This entailed:
• checking that no student was recorded as achieving more than the maximum mark for any assessment task. Such records were considered to be erroneous, and the whole line was removed before analysis. This only affected two students, and all other marks were assumed (perhaps rather optimistically) to be correct.

• removal of records for students who had not sat the exam. Most of these students had dropped out during the year and had not sat at least some of the prac, and it was felt that including these students in the data would skew the results. The remaining students had sat a supplementary exam, but the questions on that exam were different.

• replacing “absent” or “sick” marks in the prac section with zeroes. This does not accurately reflect the student’s summative mark: a student who is absent from a prac does indeed score zero, but a student who is sick receives his or her average prac mark. However, for the purposes of this study, the summative mark is less important than the student’s personal experience: regardless of earned mark, a student who is absent from a prac has not experienced it. This replacement affected 205 of the 454 students who were recorded in the prac database. It is worth noting that only 362 students sat the exam: many of the students who had been absent from praccs during the year had in fact dropped out of the course and had therefore been recorded as absent from praccs, and had not sat the exam. Most of the students who did sit the exam were only absent once or twice.

After this modification, there were still 350 student records left in the main dataset, each comprising 46 separate activity marks: a total of 15,916 marks.

3.1.3 Choice of datasets

Several subsets of the data were chosen for analysis. The first dataset chosen for analysis was, of course, the full set ALL: all students and all activities. However, this dataset produced a graph with so many points on it that it was difficult to interpret it, so ways were investigated to reduce its visual complexity.

A correlation matrix was calculated for the data, using Pearson’s r. The first look at the matrix showed some surprising results: correlation coefficients between exam questions were relatively high across the board, mostly in the neighbourhood of 0.7, whereas correlation coefficients between praccs tended to be around 0.35. Prac bonus questions correlated poorly with almost everything. Examining the raw data showed
that very few students had attempted them, and they were eventually removed from consideration from most datasets, leaving 40 activities per student\(^1\).

When the exam paper was analysed, it was found that the first ten pages consisted of multiple-choice questions, four to a page. Another four pages contained short-answer questions, six to a page. If the questions on each page had been related, this would not have been a problem; however, the questions were heterogeneous. So many factors contribute to the marks for each page that it is unlikely that cluster analysis would be able to do much to draw them out, so datasets were generated with totals, rather than page marks, for the multiple-choice and short-answer questions. This left 28 unique activities per student\(^2\).

3.1.4 Factor analysis, cluster analysis and multidimensional scaling

It had originally been planned to use William Trochim's Concept System software but this turned out not to be possible. Concept System is a dedicated system for Trochim's own version of concept mapping and will not accept data at the correlation matrix stage. It would have been possible to write a dedicated competency mapping system, but then there would not have been time to analyse the data in any depth. It was therefore decided to use SPSS for factor analysis, multidimensional scaling and cluster analysis.

The raw numeric marks were converted to percentages before the analysis to ensure that they were commensurate. Factor analysis was performed on this normalized dataset to try to determine the appropriate number of clusters to use. In factor analysis, components of variation are successively extracted in such a way as to maximize the amount of variance captured at each step, until all the variance in the original data has been accounted for. The amount of variance extracted at each step is called the *eigenvalue*, due to its method of computation.

When the eigenvalue extracted at each step is plotted, the resulting graph is called a *scree plot*. It is usually shaped like a cliff with a scree slope at the bottom: an initially sharp descending gradient, as the most important factors are extracted; followed by a gentler gradient, corresponding to minor factors; and finally the graph flattens out

---

\(^1\)Note that the sets ALL and TMA include the bonus questions.

\(^2\)Note that set ALL is the only dataset that includes separate marks for each page of multiple-choice and short-answer questions.
completely as the point of diminishing returns is reached. Further factors contribute little to the model, and may be considered to be artifacts.

The “true” number of factors may be estimated in two ways: the Kaiser criterion, which counts factors with eigenvalues greater than 1; or the scree test, which tries to find the point of diminishing returns on a scree plot. Both of these tests only give a rule of thumb: the Kaiser test tends to overestimate, while the scree test tends to underestimate the true number of factors. It was therefore necessary to go back to the data and, using the estimates gleaned from factor analysis, choose the clustering that made the most sense.

Hierarchical cluster analysis with number of clusters ranging from 2 to 12 was performed on each dataset, and multidimensional scaling (MDS) was used to map the data into two dimensions. Multidimensional scaling is a method for assigning variables to points in two or more dimensions, in such a way that variables that are have a higher measured similarity — in this case, correlations between marks — are closer together. It is essentially an optimisation problem: the stress, which is a numerical measure of “badness-of-fit”, is minimised over several iterations.

It is important to note that the clustering is not done on the points generated by multidimensional scaling. Because multidimensional scaling rarely produces a perfect fit to the input data, it is possible for points to cluster together on the basis of correlation that do not look particularly close on the multidimensional scaling plot. If this actually happens, it indicates that the final stress, or “badness-of-fit”, was too high and that the dimensionality of the scaling should be increased.

Finally, the points generated by multidimensional scaling were grouped into separate files according to the cluster they had been assigned, and gnuplot was used to plot them to a graph.

This process was performed for the following sets of data:

- **ALL** — All students, all questions, including bonus marks
- **AMAB** — All students, single totals for multiple choice and short answer, excluding bonus marks
- **TMA** — Top 115 students by mark (33%), single totals for multiple choice and short answer, including bonus prac marks
- **TMAB** — Top 115 students by mark (33%), single totals for multiple choice and short answer, excluding bonus prac marks
- **BMAB** — Bottom 115 students by mark (33%), single totals for multiple choice and short answer, excluding bonus prac marks

20
Very few of the bottom 115 students had any non-zero bonus marks, so attempting to use them would have been futile. In fact, all of these students had zero for bonus question B8: factor analysis cannot be performed if any of the variables are zero in all cases.
3.2 Results

The results of the factor analysis are shown in figures 3.1, 3.2, 3.3, 3.4 and 3.5. The point of diminishing returns is clear in the scree plots for the whole-group data (figures 3.1 and 3.2). These plots show a marked drop in significance beyond factor three, but the Kaiser criterion shows eight factors with eigenvalues greater than 1 for set ALL and four such factors for set AMAB. The correct number of significant factors is probably between these two criteria, so set ALL was plotted (see figure 3.6) with six clusters and set AMAB was plotted (see figure 3.7) with five: a dotted circle shows the two clusters that would be grouped together in a four-factor plot.

In general, when fewer students are included in the dataset, the factorial scree slope is gentler. This can be seen on all three of the student-subset plots, and was also observed when analysing randomly-generated data as described in Section 3.3: in a small sample, random fluctuations seem more significant. This effect also exaggerates the disparities between the scree test and the Kaiser criterion, making it difficult to assess the correct number of clusters to use. It was often necessary to examine the clustering for different numbers of clusters and select the number that seemed to make the most sense or be the most informative.

Including bonus prac questions did not appear to make much difference to the factor analysis for the top students. This could arise from the fact that very few students, even among the top 33%, attempted the bonus questions. These datasets, TMA and TMAB, both had Kaiser criteria of 12 and were plotted at 10 and 11 clusters respectively (see figures 3.8 and 3.9). Regardless of the number of clusters used, the data for the top students tends to form one large cluster and a number of singletons. These two datasets were plotted at a level that shows a little clustering beyond the single large cluster.
The scree plot for the bottom 33% of students — set BMAB — is unique, in that it shows two dominating factors of almost equal magnitude. All the other plots generated from the student data show a single dominating factor. This set had Kaiser criterion of eight, and was plotted at the seven-cluster level in 3.10. Dotted circles show the superclustering that would occur at the six-cluster and five-cluster levels.

### 3.2.1 Competency maps

The competency maps are shown at figures 3.6, 3.7, 3.8, 3.9, and 3.10. These maps have been produced from the data included in Appendix III. The points on the map have not been labelled with activity names as to do so would obscure the shape of the data; however, the point coordinates of all activities are listed in Appendix III.
3.2.2 Set ALL

Figure 3.6 shows the map derived from all students’ marks. Cluster 1, marked by red diamonds, contains all exam questions except for the seventh and tenth pages of multiple-choice questions, the question on digital logic, the question on linked lists, the question on generating test data, both of the sorting questions and the largest programming question; and prac 7, 8 and 9. Cluster 2, marked by blue squares, contains the seventh and tenth pages of multiple-choice questions, the questions on digital logic and generating test data, and prac 1–6, 10 and 11. Cluster 3 contains the exam questions on linked lists and the bubblesort algorithm, the largest exam programming question, all the prac bonus questions except those for prac 3 and 5, and prac 12. The remaining tasks — the exam question on selection sort and the bonus prac questions from prac 3 and 5 — are singletons.
3.2.3 Set AMAB

The competency map for the set of data derived from all students’ marks, excluding bonus marks and rolling the marks from multiple-choice and short-answer questions into single marks, is shown at figure 3.7. The clustering for the exam questions is similar to that seen for set ALL: cluster 1 comprises all the exam questions except for the ones on digital logic, linked lists, test data, both of the sorting questions, and the largest programming question. Cluster 2 contains the exam questions on digital logic and test data, and prac 1-4, 10 and 11. Cluster 3 contains the exam questions on linked lists and bubblesort, the largest programming question on the exam, and prac 12. Cluster 4 is a singleton, containing only the exam question on selection sort; while the remaining pracs — 5-9 — are all in cluster 5.

Clusters 2 and 5 are closely related, and at the four-cluster level they form a single cluster. The dotted ellipse shows this superclustering.
3.2.4 Set TMA

This map is derived from the marks of the top third of students ranked by total mark, including prac bonus questions. It comprises a single large cluster, a smaller cluster, and a lot of singletons. In the large cluster are all exam questions except for those on linked lists, binary search, both sorting questions and the largest programming question, and all the prac other than the bonus questions and prac 12. The secondary cluster, labelled cluster 10, comprises prac 12 and the bonus question to prac 11. All other tasks are singletons.
3.2.5 Set TMAB

This map was derived from the marks of the top third of students ranked by total mark, but does not include marks from prac bonus questions. It is similar to figure 3.8, in that it comprises a large main cluster, a smaller cluster, and many singletons. In this case the smaller cluster is cluster 1, and contains the multiple-choice and short-answer exam questions, and the exam question on debugging. The large cluster is cluster 2 and contains all the pracs except prac 12, and the exam questions on digital logic, testing and data structures, and the two smaller programming questions. All other tasks are singletons.

Note that the two singletons in the top right, which are the exam questions on finding the next largest square and calculating reciprocal by reference, are not clustered together even though multidimensional scaling places them close together. This is possible because the clustering is done on the raw data, rather than the scaled points.
3.2.6 Set BMAB

The last competency map shows the sharpest division between prac and exam tasks. Cluster 1 contains all the exam questions apart from those on digital logic, test data and selection sort, which are singletons; it also contains prac 12. Cluster 2 contains prac 1-5, 10 and 11; cluster 3 contains prac 6 and 7, while cluster 4 contains prac 8 and 9. At the five-cluster level, clusters 2, 3 and 4 combine to form a supercluster, which shown by the dotted ellipse. Note that this supercluster contains all prac tasks except prac 12.

3.3 Random data

How can these graphs be interpreted? Exactly what can be inferred from them? It is a circular problem: the accuracy of the representation of the map cannot be checked
against the underlying conceptual structure of the subject, because the underlying conceptual structure (as modified through the students' learning experiences) is not known with certainty. It is useful to generate dummy data with known properties and see how well competency mapping performs. This exercise will also shed light on the best ways to design assessment for use as input to competency mapping.

In order to generate this dummy data, it was first necessary to design a mathematical model for assessment marks.

### 3.3.1 A model for competencies

There are two entities that we must model: students and questions. Each question will test one or more competencies, and each competency will contribute a proportion of the total mark for the question between 0 (completely irrelevant) and 1 (marks for that question depend exclusively on this competency). Furthermore, each student will have a certain degree of mastery of each competency, between 0 (no idea at all) and 1 (complete mastery).

Assume that the domain comprises \( N \) competencies (roughly equivalent to the clusters on the map). Assume, optimistically, that the clusters are independent and disjoint: they do not overlap.

The set of capabilities of each student can then be represented as a vector of positive numbers \( S \) between 0 and 1, where \( S_i \) represents the proportion of competency \( i \) that the student has mastered. Assessment tasks can be modelled as a stochastic vector of positive numbers \( Q \) between 0 and 1, where \( Q_i \) represents the contribution of competency \( i \) to the question. Now, the probability of a given student getting a given question right is the product of the student's capability vector and the question's capability vector. To put it another way, the expected mark is \( (Q,S)M \), where \( M \) is the total marks available for the question.

Note that, under this model, assessment is an attempt to infer each student's vector by getting the student to sit a number of tasks with (it is hoped!) known \( Q \) vectors. The aim of teaching, of course, is to get all the \( S_i \) as close to 1 as possible; more realistically, to get the \( S_i \) over some predetermined minimum value. This project is an attempt to infer the \( Q \) vectors.

Of course, this model is an oversimplification. It is unlikely that real domains have independent disjoint competencies. However, it is good enough to be used as a basis for analysis: it is simple, and not hard to calculate.
This model may have uses outside competency mapping, and can provide a basis for any competency-based analysis of student results.

### 3.3.2 The random datasets

Data was generated according to several different models:

- **orthogonal** — precisely one element in each Q vector is non-zero. In this model, each question tests a single competency.

- **mixed** — multiple competencies may contribute to each question. This was modelled by beginning with a zero vector and adding $1/N$ to $N$ randomly-chosen elements, for various values of $N$. The example mixed-model dataset uses $N=3$.

- **orthomodal** — this is an attempt to model the case in which questions are delivered in two modes which have different cognitive overheads. The Q vectors consist of a mode vector appended to a base competency vector. The mode vector has two elements, of which exactly one will be non-zero; this models the mode of delivery. For this dataset, the base competency vector was orthogonal.

The data was generated using python scripts, three of which are attached at Appendix IV.

Student competency vectors were generated according to a normal distribution, with identical mean and standard deviation for each student and competency. Note that this does not imply that the student competency vectors are identical; only that they were randomly generated by the same function. This simple model does not take into account the possibility of a mixed student population; nor does it take into account the intuition that some topics are inherently harder than others. Because this project is seeking to draw inferences about the subject matter of the course rather than about the student population, this was felt to be adequate.

Fourteen datasets were randomly generated, and the following were chosen for inclusion in this thesis:

- **O4-40** — orthogonal, four competencies, 40 questions
- **OM4-40** — orthomodal, four base competencies, two modal competencies at 0.25, 40 questions
- **OMA4-40** — orthomodal, four base competencies, one modal competency at 0.25 and one at 0.75, 40 questions
Figure 3.11: Orthogonal, four competencies

Figure 3.12: Orthomodal, 0.25/0.25 modal competencies

- M4-3 — mixed, four competencies, three competencies per question

3.3.3 Performance

Competency mapping performed predictably well on the orthogonal datasets. In these datasets, each question assessed only one competency. As can be seen from the example, at figure 3.11, they produced tight, well-separated clusters that are easy to distinguish. They do not resemble the results of competency mapping on actual student data, but this is unsurprising: student assessment is far from orthogonal. However, it does imply that questions that are as close as practical to orthogonal might produce competency maps that are easier to read and interpret.

The mixed datasets produced the plots that looked most like those obtained from real data. (See figure 3.14.) Competency mapping did a good job of clustering the questions according to the strongest base competency, but the clusters tend to overlap more than the clusters drawn from real student data do. This is at least partly due to the higher stresses in the multidimensional scaling: when an 80-question mixed dataset with eight base competencies was plotted, the stress in the scaling was over 0.4. In comparison, the stresses for multidimensional scaling in the real student data were in the neighbourhood of 0.1. The reason for this poor fit is not known at this stage. The example shown at figure 3.14 had a stress value of roughly 0.18.

The competency map for the orthomodal dataset with both modal competencies set to 0.25, shown at figure 3.12, looks very similar to the map for the orthogonal dataset.
The main difference is that the clusters are less tight. However, when one modal competency is set at 0.75 — modelling a mode in which delivery and interface issues dominate student performance — and the other remains at 0.25, the results, shown at figure 3.13, are quite different. While the tasks for the lighter mode still cluster by base competency, the tasks for the heavier mode cluster together most strongly. These “mode A” tasks, which were assigned to cluster 5 in the example figure, form a single, relatively diffuse cluster. This resembles the clustering of prac tasks in the real data, which opens up the possibility that issues associated with the delivery of prac, rather than familiarity with essential concepts or mastery of basic skills, may be the dominating factor in determining prac marks.

Other interpretations of these data are possible, of course. For example, it is possible that the prac are measuring programming ability, but that the exam is not: in other words, the ability to program may be acting as a modal competency in that it is a determiner of marks in one mode but not in the other. But the exam contained programming questions as well as theory questions! If this is the reason for the divergence in correlations, then it must take a substantially different skill set to write a good program under exam conditions rather than in a lab class.
Chapter 4

Analysis and Conclusions

Frustratingly little structure was visible within the exam, but it is interesting to see a large divide between theory questions and pracs and even more interesting to see that this divide does not hold true for the top students.

4.1 Factor analysis

For most datasets, there was a single dominating factor with a large eigenvalue. Identifying that factor cannot really be done with the data available in this project, but several candidates come to mind.

There are many factors other than academic ability and mastery of the domain that are likely to affect performance on an exam: memory, especially if the students were trying a cram-and-forget strategy; ability to work quickly; ability to handle pressure; ability to understand and follow written English instructions.

It is possible to draw up a similar list of non-academic factors that are likely to affect performance in pracs. The most obvious factors are those that stem from the task itself: the ability to cope with the interface provided to the student in pracs. This usually includes a text editor, a compiler, and some mechanism for running programs. It may also include the ability to perform basic operating system functions, such as creating a directory or copying a file, and theoretical background knowledge about how the filesystem works. A student who does not understand what a “directory” or “folder” is may suffer as the semester progresses if it means that she cannot organise her prac work efficiently.
It would be a mistake to overlook the other differences between prac and exams. While taking an exam is an inherently solitary task, prac are often social. Students may learn through interaction with peers; on the other hand, a shy student may feel exposed and uncomfortable. Of course, all students need to be able to interact with their prac demonstrator; a student who is too shy to ask for help is at a definite disadvantage. In this way, social skills may be considered to be modal competencies. Another modal factor — it surely could not be called a competency — is cheating. In general, it is much easier for a student to cheat without detection in a prac than it is in an exam. It is to be hoped that such activity is not sufficiently prevalent to make a large difference in the competency map for the subject, but the possibility that it has affected some marks cannot be ignored, and it must surely affect prac and exam marks differentially.

4.2 Cluster analysis

It was hoped that the clustering would provide an insight into the structure of the course, ideally a competency-based decomposition that would be able to be used as the basis for assessment design. Unfortunately, such decomposition can not be read from the results that we obtained. However, some valuable and unexpected insights were gained in the process of the analysis.

The most obvious result is that prac tasks are clustered with prac tasks and exam tasks with exam tasks. This tendency seems more marked among the less able students and less marked among the better-performing students. Compare the clustering in figures 3.10, 3.7 and 3.9. In the first of these, which shows the results from the bottom third of students, all but one of the prac questions cluster together. In the second, showing the results for the whole student body, the prac cluster with exam tasks about digital logic and test data. In the third, representing the top third of students, the prac (excluding prac 11) cluster with exam tasks on digital logic, data structures, test data, and the two smaller programming questions. Clearly, the better a student performs at computer science, the more strongly correlated her prac and exam results are. This could be interpreted as simply showing that a student in the top third of the class gets good marks at everything, but the reality is more complex than that: if the observed tendency is simply the result of selecting students on the basis of ability, then the bottom third of students (which is likely to be as homogeneous in terms of ability as the top third) should also show stronger correlations. In fact, the observed correlations are weaker for the bottom third of students.
This may indicate that the weaker students are not applying theory to practical situations, or are not allowing lessons learnt in practice to illuminate their understanding of theory, or it could indicate some overriding factor that affects one kind of task but not the other: for example, difficulty using the computer system or difficulty with English. This second possibility mirrors the idea of the “modal competency” that was introduced in Section 3.3.

It is significant that the exam task for which students were asked to develop sets of data to test a function is clustered with most of the prac questions for the whole-group and top-group data. The correlation between the test-data task and the prac marks could indicate that the ability to test code fully is a marker for the ability to program; alternatively, it might only mean that only students who finish the pracs get practice at testing. The former hypothesis is easy to test: give intensive lessons in software testing to a group of students, and see whether their programming ability improves as a result.

It is rather surprising that the programming questions from the exam do not cluster with the prac questions for the whole-group data. It is informative that the two smaller programming questions are clustered with the prac questions for the top third of students. This is further evidence that the majority of students do not apply theory to practice. Students’ programming practices need to be investigated. The program design principles that students are told to apply would serve them well enough in an exam situation, but if they are actually applying a rapid-prototyping code-test-debug cycle in pracs, in which the compiler rather than the designer forms the first line of defence against bugs, then they will not perform well in any context where a computer is not present.

In most cases, the harder questions cluster together. For example, for set AMAB (figure 3.7), the exam questions about linked lists and bubblesort, the largest programming question on the exam, and prac 12 cluster together. For set ALL (figure 3.6), the equivalent cluster contains the exam questions on linked lists and bubblesort, the largest programming question on the exam, and four of the prac bonus questions. None of these tasks had an average mark of more than 33%, and for prac bonus 8 and prac 12 the average mark was under 7%.

4.3 Methodological problems

This research has not been without problems. The most obvious problem is the sample size: although many students were included in the study, only one semester’s
data was used. Competency mapping cannot be considered to have been thoroughly tested until the study has been repeated.

When the tasks were examined closely, more problems became apparent. The exam was not well laid out for the purposes of competency mapping. As demonstrated in section 3.3, competency mapping works best with orthogonal data; however, the exam had up to six questions to a page. Even if each question only assessed two competencies, the mark for that page would reflect up to twelve competencies: this is too many to be able to extract reliably. This effect would have been ameliorated if we had used the marks for each question, rather than the page mark, but the page marks were written on the front of the exam and the per-question marks were much less accessible.

A similar problem applied to the prac. There was a three-week project carried out across prac 10–12, marked by demonstrators during the prac class. If a student finished an assessable component of this project a week early, it is possible that some demonstrators marked it and entered it in the column for the previous prac. That would have distorted that student’s contribution to the correlation matrix. Furthermore, it is not clear what effects, if any, the groupwork had on the final mark.

Prac 12 combined searching, sorting and recursion with the final assessment for the group project. The low average mark of 5.33% for P12 could indicate some irregularity in entering that mark — perhaps not all demonstrators keyed the mark for the assignment in the column for that week, or perhaps students simply did not have time to finish. Because Prac 12 was the last for the semester, it is also possible that not all demonstrators had entered their marks at the time the database was copied. It may be wisest to consider P12 an outlier.

The methods used for data analysis need some improving. Neither the scree plot nor the Kaiser criterion seem to be adequate to give the right number of clusters. In some cases these two criteria differed by up to 10! A better method for assessing the number of clusters is needed. It is possible that minimum message length encoding (MML), which rewards goodness of fit while penalising complex models, might give a better estimate.

Although this test of competency mapping was not as successful as had been hoped, the technique should not be discarded until these methodological problems have been addressed and the test has been repeated.
4.4 A better test

Most of the problems listed in the previous section can be addressed by the development of an integrated assessment tool that can interface to a competency mapping module. A possible architecture for such a tool is shown in figure 4.1.

The requirement for small, orthogonal questions necessitates the development of an online assessment tool: the collection of per-question data from traditional written exams is too labour-intensive to be practical. It does not, however, necessitate machine marking: what can be marked by hand using paper exams can still be human-marked in a computerized assessment system. Provision can still be made for the marker to annotate their mark if necessary. It is likely that online marking would be more efficient than paper-based marking, even if all the marking is still done by humans: there are no papers to count, sort or lose, many markers can work on the same student’s exam simultaneously, and markers’ work can be checked immediately if the need arises. Students’ work can still be stored; in fact, if archived to compact disc, it will take up less space and be less subject to deterioration.

Because the questions need to be small, there must be a lot of them. The need to generate a large question bank is the primary reason that such an online assessment system could not be written as part of this project. Generating questions will be the hardest and most time-consuming part of implementing the online assessment.

Figure 4.1: Competency mapping tool architecture
The test needs to be carried out with as little overhead as possible for the students. This means that it should replace an existing form of assessment, such as the mid-semester test, rather than adding to the students' assessment burden. Alternatively, it could be administered in the final prac class for the semester, as very timely formative assessment in advance of the final exam.

The results of the online test needs to be stored in a database, which can be integrated with the prac marks database. This database needs to have data entry capability, so that as other marks are made available they can also be entered into the database.

The multidimensional scaling and cluster analysis tools can be implemented as separate modules. This will make it easier to test different algorithms.

4.5 Conclusion

This thesis describes research that, although it did not achieve its initial goal of determining the conceptual and practical structure of introductory computer programming, has nevertheless provided a lot of food for thought for anyone interested in the design and evaluation of assessment. Relationships between assessment tasks have been discovered that can point the way to many new research projects. Will teaching students how to test their code improve their programming skills, or is the ability to test a natural extension of their ability to program? Can we manufacture excellent students by explicitly drawing connections between theory-based tasks and practical tasks?

The usefulness of competency mapping itself must be considered unproven, rather than disproven, owing to the inappropriate shape of the data that was available to test it. It would not be hard to develop assessment tasks that are more appropriate to the technique, and to do so would give the opportunity of settling the question. If competency mapping is to be further explored, many questions need to be answered: how many students need to be included, and how many assessment tasks should they do? What is an acceptable stress if the results of multidimensional scaling are to make sense? What is the effect of using different algorithms for multidimensional scaling and cluster analysis? As research progresses, more such questions will arise.

Given the problems that were experienced with the input data, competency mapping looks like a promising way to analyse student marks data. It produces output that is easy to read, and its implementation is neither expensive nor difficult. In the final analysis, however, the acid test for competency mapping will not be its formal validity, nor its cost, but its usefulness to the university community.
Appendices

Appendix I: Datasets

- **ALL** — All students, all questions, including bonus marks
- **AMAB** — All students, single totals for multiple choice and short answer, excluding bonus marks
- **TMA** — Top 115 students by mark (33%), single totals for multiple choice and short answer, including bonus prac marks
- **TMAB** — Top 115 students by mark (33%), single totals for multiple choice and short answer, excluding bonus prac marks
- **BMAB** — Bottom 115 students by mark (33%), single totals for multiple choice and short answer, excluding bonus prac marks
- **O4-40** — orthogonal, four competencies, 40 questions
- **OM4-40** — orthomodal, four base competencies, two modal competencies at 0.25, 40 questions
- **OMA4-40** — orthomodal, four base competencies, one modal competency at 0.25 and one at 0.75, 40 questions
- **M4-3** — mixed, four competencies, three competencies per question

39
Appendix II: Activity descriptions

- PRAC1, introduction to Windows NT, files, folders; WWW, mail, FTP; working with .ZIP files
- PRAC2, using Borland C++
- PRAC3, C programs, numeric types, operator precedence, variables
- PRAC4, selection, testing
- PRAC5, I/O and iteration
- PRAC6, functions, pointers
- PRAC7, arrays of pointers, debugging
- PRAC8, strings, files
- PRAC9, arrays, structs
- PRAC10, design component of three-week assignment
- PRAC11, implementation of three-week assignment
- PRAC12, final assignment writeup; searching, sorting, recursion
- MCQ1-10, ten pages of multiple choice questions, three or four to a page
- SA1-4, four pages of short-answer questions, six to a page
- DL, digital logic, truth table
- REC, fill in blanks in function to calculate reciprocal, parameter passed by reference
- LISTS, fill in blanks in linked list code
- CODE1, two small functions (modulus, arrays)
- CODE2, main() to call the code in CODE1
- TEST, design test data
- DEBUG, find bugs in a program
- SQR, smallest square less than or equal to the parameter
- SEARCH, fill in blanks for binary search algorithm (in pseudocode)
- PALIN, detect whether an input string is a palindrome
- SSORT, follow selection sort algorithm
- BSORT, follow bubblesort algorithm
- DS, design data structures for use in CODE3
- CODE3, file I/O and arrays
Appendix III: Multidimensional Scaling Coordinates
<table>
<thead>
<tr>
<th>Task</th>
<th>X</th>
<th>Y</th>
<th>Cluster</th>
</tr>
</thead>
<tbody>
<tr>
<td>MCQ1</td>
<td>0.2846</td>
<td>0.0731</td>
<td>1</td>
</tr>
<tr>
<td>MCQ2</td>
<td>0.3403</td>
<td>0.3839</td>
<td>1</td>
</tr>
<tr>
<td>MCQ3</td>
<td>0.1259</td>
<td>0.4556</td>
<td>1</td>
</tr>
<tr>
<td>MCQ4</td>
<td>0.0257</td>
<td>-0.1651</td>
<td>1</td>
</tr>
<tr>
<td>MCQ5</td>
<td>0.2222</td>
<td>0.1289</td>
<td>1</td>
</tr>
<tr>
<td>MCQ6</td>
<td>0.7182</td>
<td>0.1215</td>
<td>1</td>
</tr>
<tr>
<td>MCQ7</td>
<td>1.5132</td>
<td>0.0516</td>
<td>2</td>
</tr>
<tr>
<td>MCQ8</td>
<td>-0.3146</td>
<td>-0.6810</td>
<td>1</td>
</tr>
<tr>
<td>MCQ9</td>
<td>1.0038</td>
<td>0.3732</td>
<td>1</td>
</tr>
<tr>
<td>MCQ10</td>
<td>0.9839</td>
<td>0.7816</td>
<td>2</td>
</tr>
<tr>
<td>DL</td>
<td>1.8085</td>
<td>0.0779</td>
<td>2</td>
</tr>
<tr>
<td>SA1</td>
<td>-0.4249</td>
<td>-0.0374</td>
<td>1</td>
</tr>
<tr>
<td>SA2</td>
<td>-0.0462</td>
<td>0.0690</td>
<td>1</td>
</tr>
<tr>
<td>SA3</td>
<td>-0.4201</td>
<td>0.4858</td>
<td>1</td>
</tr>
<tr>
<td>SA4</td>
<td>-0.9132</td>
<td>0.2826</td>
<td>1</td>
</tr>
<tr>
<td>REC</td>
<td>-0.5179</td>
<td>1.0154</td>
<td>1</td>
</tr>
<tr>
<td>LISTS</td>
<td>-2.0012</td>
<td>0.1191</td>
<td>3</td>
</tr>
<tr>
<td>CODE1</td>
<td>0.6348</td>
<td>0.4815</td>
<td>1</td>
</tr>
<tr>
<td>CODE2</td>
<td>-0.2479</td>
<td>0.8134</td>
<td>1</td>
</tr>
<tr>
<td>TEST</td>
<td>1.2744</td>
<td>0.1716</td>
<td>2</td>
</tr>
<tr>
<td>DEBUG</td>
<td>0.0756</td>
<td>0.0073</td>
<td>1</td>
</tr>
<tr>
<td>SQR</td>
<td>-0.5252</td>
<td>0.8427</td>
<td>1</td>
</tr>
<tr>
<td>SRCH</td>
<td>-0.6209</td>
<td>-0.0531</td>
<td>1</td>
</tr>
<tr>
<td>PALIN</td>
<td>-0.9890</td>
<td>0.4245</td>
<td>1</td>
</tr>
<tr>
<td>SSORT</td>
<td>-0.3894</td>
<td>1.2264</td>
<td>4</td>
</tr>
<tr>
<td>BSORT</td>
<td>-1.7552</td>
<td>0.2987</td>
<td>3</td>
</tr>
<tr>
<td>DS</td>
<td>0.3737</td>
<td>0.4538</td>
<td>1</td>
</tr>
<tr>
<td>CODE3</td>
<td>-1.3990</td>
<td>0.1014</td>
<td>3</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Task</th>
<th>X</th>
<th>Y</th>
<th>Cluster</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1</td>
<td>2.1050</td>
<td>-0.2547</td>
<td>2</td>
</tr>
<tr>
<td>P2</td>
<td>2.0576</td>
<td>-0.0587</td>
<td>2</td>
</tr>
<tr>
<td>P3</td>
<td>1.8777</td>
<td>-0.0510</td>
<td>2</td>
</tr>
<tr>
<td>B3</td>
<td>0.6863</td>
<td>-1.4942</td>
<td>5</td>
</tr>
<tr>
<td>P4</td>
<td>1.4913</td>
<td>-0.1400</td>
<td>2</td>
</tr>
<tr>
<td>B4</td>
<td>-2.2166</td>
<td>-0.8069</td>
<td>3</td>
</tr>
<tr>
<td>P5</td>
<td>1.2946</td>
<td>-0.4248</td>
<td>2</td>
</tr>
<tr>
<td>B5</td>
<td>-1.6303</td>
<td>-1.2632</td>
<td>6</td>
</tr>
<tr>
<td>P6</td>
<td>1.0639</td>
<td>-0.4221</td>
<td>2</td>
</tr>
<tr>
<td>P7</td>
<td>0.8482</td>
<td>-0.8482</td>
<td>1</td>
</tr>
<tr>
<td>B7</td>
<td>-2.3569</td>
<td>-0.5460</td>
<td>3</td>
</tr>
<tr>
<td>P8</td>
<td>0.4650</td>
<td>-0.3217</td>
<td>1</td>
</tr>
<tr>
<td>B8</td>
<td>-2.6571</td>
<td>-0.7330</td>
<td>3</td>
</tr>
<tr>
<td>P9</td>
<td>0.6804</td>
<td>-0.2545</td>
<td>1</td>
</tr>
<tr>
<td>P10</td>
<td>1.4685</td>
<td>-0.4523</td>
<td>2</td>
</tr>
<tr>
<td>P11</td>
<td>1.1288</td>
<td>-0.1418</td>
<td>2</td>
</tr>
<tr>
<td>B11</td>
<td>-2.5602</td>
<td>-0.4807</td>
<td>3</td>
</tr>
<tr>
<td>P12</td>
<td>-2.5672</td>
<td>-0.1638</td>
<td>3</td>
</tr>
</tbody>
</table>
Table 2: Set AMAB: MCQs and SAs compressed, all students, excluding bonuses

<table>
<thead>
<tr>
<th>Task</th>
<th>X</th>
<th>Y</th>
<th>Cluster</th>
</tr>
</thead>
<tbody>
<tr>
<td>MCQ</td>
<td>0.1253</td>
<td>0.0277</td>
<td>1</td>
</tr>
<tr>
<td>DL</td>
<td>1.6444</td>
<td>0.2330</td>
<td>2</td>
</tr>
<tr>
<td>SA</td>
<td>-0.5864</td>
<td>-0.0066</td>
<td>1</td>
</tr>
<tr>
<td>RBC</td>
<td>-0.9277</td>
<td>1.0024</td>
<td>1</td>
</tr>
<tr>
<td>LISTS</td>
<td>-2.3765</td>
<td>-0.3155</td>
<td>3</td>
</tr>
<tr>
<td>CODE1</td>
<td>0.2532</td>
<td>0.6350</td>
<td>1</td>
</tr>
<tr>
<td>CODE2</td>
<td>-0.6687</td>
<td>0.8572</td>
<td>1</td>
</tr>
<tr>
<td>TEST</td>
<td>-0.0483</td>
<td>0.3700</td>
<td>2</td>
</tr>
<tr>
<td>DEBUG</td>
<td>-0.1744</td>
<td>0.0330</td>
<td>1</td>
</tr>
<tr>
<td>SQR</td>
<td>-0.9183</td>
<td>0.8447</td>
<td>1</td>
</tr>
<tr>
<td>SRCH</td>
<td>-0.9514</td>
<td>0.0436</td>
<td>1</td>
</tr>
<tr>
<td>PALIN</td>
<td>-1.3563</td>
<td>0.4373</td>
<td>1</td>
</tr>
<tr>
<td>SSORT</td>
<td>-0.7350</td>
<td>-1.2559</td>
<td>4</td>
</tr>
<tr>
<td>BSORT</td>
<td>-2.1289</td>
<td>-0.3632</td>
<td>3</td>
</tr>
<tr>
<td>DS</td>
<td>0.0100</td>
<td>0.6154</td>
<td>1</td>
</tr>
<tr>
<td>CODE3</td>
<td>-1.7708</td>
<td>0.0184</td>
<td>3</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Task</th>
<th>X</th>
<th>Y</th>
<th>Cluster</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1</td>
<td>1.9674</td>
<td>-0.1968</td>
<td>2</td>
</tr>
<tr>
<td>P2</td>
<td>1.8963</td>
<td>0.0263</td>
<td>2</td>
</tr>
<tr>
<td>P3</td>
<td>1.6940</td>
<td>0.0499</td>
<td>2</td>
</tr>
<tr>
<td>P4</td>
<td>1.3046</td>
<td>0.0161</td>
<td>2</td>
</tr>
<tr>
<td>P5</td>
<td>1.1332</td>
<td>-0.3091</td>
<td>5</td>
</tr>
<tr>
<td>P6</td>
<td>0.8774</td>
<td>-0.0327</td>
<td>5</td>
</tr>
<tr>
<td>P7</td>
<td>0.6639</td>
<td>-0.5851</td>
<td>5</td>
</tr>
<tr>
<td>P8</td>
<td>0.2422</td>
<td>-0.7551</td>
<td>5</td>
</tr>
<tr>
<td>P9</td>
<td>0.4596</td>
<td>-0.3470</td>
<td>5</td>
</tr>
<tr>
<td>P10</td>
<td>1.2792</td>
<td>-0.2033</td>
<td>2</td>
</tr>
<tr>
<td>P11</td>
<td>0.9119</td>
<td>-0.0444</td>
<td>2</td>
</tr>
<tr>
<td>P12</td>
<td>-2.9264</td>
<td>-0.7081</td>
<td>3</td>
</tr>
</tbody>
</table>
Table 3: Set TMA; MCQs and SAs compressed, top third, including bonuses

<table>
<thead>
<tr>
<th>Task</th>
<th>$X$</th>
<th>$Y$</th>
<th>Cluster</th>
</tr>
</thead>
<tbody>
<tr>
<td>MCQ</td>
<td>0.3891</td>
<td>-0.0209</td>
<td>1</td>
</tr>
<tr>
<td>DL</td>
<td>1.0979</td>
<td>-0.0643</td>
<td>1</td>
</tr>
<tr>
<td>SA</td>
<td>-0.0287</td>
<td>0.0327</td>
<td>1</td>
</tr>
<tr>
<td>REC</td>
<td>0.5453</td>
<td>-0.6069</td>
<td>1</td>
</tr>
<tr>
<td>LISTS</td>
<td>-1.2869</td>
<td>1.2650</td>
<td>2</td>
</tr>
<tr>
<td>CODE1</td>
<td>1.0297</td>
<td>-0.0328</td>
<td>1</td>
</tr>
<tr>
<td>CODE2</td>
<td>0.8374</td>
<td>-0.0709</td>
<td>1</td>
</tr>
<tr>
<td>TEST</td>
<td>0.8757</td>
<td>0.0596</td>
<td>1</td>
</tr>
<tr>
<td>DEBUG</td>
<td>0.2084</td>
<td>0.0143</td>
<td>1</td>
</tr>
<tr>
<td>SQR</td>
<td>0.4757</td>
<td>0.4749</td>
<td>1</td>
</tr>
<tr>
<td>SRCH</td>
<td>0.2715</td>
<td>0.8948</td>
<td>3</td>
</tr>
<tr>
<td>PALIN</td>
<td>0.0250</td>
<td>0.4197</td>
<td>1</td>
</tr>
<tr>
<td>SSORT</td>
<td>-0.1100</td>
<td>1.2598</td>
<td>4</td>
</tr>
<tr>
<td>BSORT</td>
<td>-1.5341</td>
<td>1.1701</td>
<td>5</td>
</tr>
<tr>
<td>DS</td>
<td>0.6811</td>
<td>-0.0170</td>
<td>1</td>
</tr>
<tr>
<td>CODE3</td>
<td>-0.8466</td>
<td>0.4779</td>
<td>6</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Task</th>
<th>$X$</th>
<th>$Y$</th>
<th>Cluster</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1</td>
<td>1.2062</td>
<td>-0.0161</td>
<td>1</td>
</tr>
<tr>
<td>P2</td>
<td>1.1527</td>
<td>-0.0839</td>
<td>1</td>
</tr>
<tr>
<td>P3</td>
<td>1.1443</td>
<td>-0.0861</td>
<td>1</td>
</tr>
<tr>
<td>P4</td>
<td>0.9725</td>
<td>0.1207</td>
<td>1</td>
</tr>
<tr>
<td>P5</td>
<td>0.9847</td>
<td>-0.1232</td>
<td>1</td>
</tr>
<tr>
<td>P6</td>
<td>1.0421</td>
<td>-0.0946</td>
<td>1</td>
</tr>
<tr>
<td>P7</td>
<td>0.8874</td>
<td>-0.1161</td>
<td>1</td>
</tr>
<tr>
<td>B7</td>
<td>-2.2464</td>
<td>-0.8836</td>
<td>9</td>
</tr>
<tr>
<td>P8</td>
<td>0.7183</td>
<td>-0.2409</td>
<td>1</td>
</tr>
<tr>
<td>B8</td>
<td>-2.8335</td>
<td>-0.0425</td>
<td>10</td>
</tr>
<tr>
<td>P9</td>
<td>0.8637</td>
<td>-0.1311</td>
<td>1</td>
</tr>
<tr>
<td>P10</td>
<td>0.8871</td>
<td>-0.0889</td>
<td>1</td>
</tr>
<tr>
<td>P11</td>
<td>0.9278</td>
<td>-0.0759</td>
<td>1</td>
</tr>
<tr>
<td>B11</td>
<td>-0.2885</td>
<td>0.1146</td>
<td>11</td>
</tr>
<tr>
<td>P12</td>
<td>-2.8237</td>
<td>0.5820</td>
<td>10</td>
</tr>
</tbody>
</table>
Table 4: Set TMAB: MCQs and SAs compressed, top third, excluding bonuses

<table>
<thead>
<tr>
<th>Task</th>
<th>X</th>
<th>Y</th>
<th>Cluster</th>
</tr>
</thead>
<tbody>
<tr>
<td>MCQ</td>
<td>0.0838</td>
<td>0.0556</td>
<td>1</td>
</tr>
<tr>
<td>DL</td>
<td>1.0164</td>
<td>0.0873</td>
<td>2</td>
</tr>
<tr>
<td>SA</td>
<td>-0.4447</td>
<td>0.0681</td>
<td>1</td>
</tr>
<tr>
<td>RBC</td>
<td>0.2575</td>
<td>1.0266</td>
<td>3</td>
</tr>
<tr>
<td>LISTS</td>
<td>-2.2880</td>
<td>-1.3650</td>
<td>4</td>
</tr>
<tr>
<td>CODE1</td>
<td>0.8997</td>
<td>0.0451</td>
<td>2</td>
</tr>
<tr>
<td>CODE2</td>
<td>0.6615</td>
<td>0.0490</td>
<td>2</td>
</tr>
<tr>
<td>TEST</td>
<td>0.6650</td>
<td>0.0910</td>
<td>2</td>
</tr>
<tr>
<td>DEBUG</td>
<td>-0.1855</td>
<td>0.4023</td>
<td>1</td>
</tr>
<tr>
<td>SQR</td>
<td>0.0848</td>
<td>1.0308</td>
<td>5</td>
</tr>
<tr>
<td>SRCH</td>
<td>-0.9708</td>
<td>1.1746</td>
<td>6</td>
</tr>
<tr>
<td>PALIN</td>
<td>-0.5113</td>
<td>-0.7705</td>
<td>7</td>
</tr>
<tr>
<td>SSORT</td>
<td>-0.3096</td>
<td>-1.6260</td>
<td>8</td>
</tr>
<tr>
<td>BSORT</td>
<td>-2.7994</td>
<td>0.1487</td>
<td>9</td>
</tr>
<tr>
<td>DS</td>
<td>0.4777</td>
<td>0.0408</td>
<td>2</td>
</tr>
<tr>
<td>CODE3</td>
<td>-1.6563</td>
<td>0.0824</td>
<td>10</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Task</th>
<th>X</th>
<th>Y</th>
<th>Cluster</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1</td>
<td>1.1430</td>
<td>-0.0722</td>
<td>2</td>
</tr>
<tr>
<td>P2</td>
<td>1.0743</td>
<td>-0.0097</td>
<td>2</td>
</tr>
<tr>
<td>P3</td>
<td>1.0634</td>
<td>0.0148</td>
<td>2</td>
</tr>
<tr>
<td>P4</td>
<td>0.8675</td>
<td>-0.0395</td>
<td>2</td>
</tr>
<tr>
<td>P5</td>
<td>0.8834</td>
<td>-0.0368</td>
<td>2</td>
</tr>
<tr>
<td>P6</td>
<td>0.9468</td>
<td>0.0009</td>
<td>2</td>
</tr>
<tr>
<td>P7</td>
<td>0.7433</td>
<td>-0.4203</td>
<td>2</td>
</tr>
<tr>
<td>P8</td>
<td>0.5440</td>
<td>-0.5120</td>
<td>2</td>
</tr>
<tr>
<td>P9</td>
<td>0.7395</td>
<td>-0.0485</td>
<td>2</td>
</tr>
<tr>
<td>P10</td>
<td>0.7751</td>
<td>-0.0716</td>
<td>2</td>
</tr>
<tr>
<td>P11</td>
<td>0.8149</td>
<td>0.0142</td>
<td>2</td>
</tr>
<tr>
<td>P12</td>
<td>-4.0860</td>
<td>0.6395</td>
<td>11</td>
</tr>
</tbody>
</table>
Table 5: Set BMAB: MCQs and SAs compressed, bottom third, excluding bonuses

<table>
<thead>
<tr>
<th>Task</th>
<th>X</th>
<th>Y</th>
<th>Cluster</th>
</tr>
</thead>
<tbody>
<tr>
<td>MCQ</td>
<td>0.1652</td>
<td>-0.2870</td>
<td>1</td>
</tr>
<tr>
<td>DL</td>
<td>1.5062</td>
<td>-1.2692</td>
<td>2</td>
</tr>
<tr>
<td>SA</td>
<td>-0.7536</td>
<td>-0.0821</td>
<td>1</td>
</tr>
<tr>
<td>RBC</td>
<td>-1.2984</td>
<td>-0.2640</td>
<td>1</td>
</tr>
<tr>
<td>LISTS</td>
<td>-1.6186</td>
<td>0.2791</td>
<td>1</td>
</tr>
<tr>
<td>CODE1</td>
<td>-0.7309</td>
<td>-0.5462</td>
<td>1</td>
</tr>
<tr>
<td>CODE2</td>
<td>-1.4650</td>
<td>-0.0554</td>
<td>1</td>
</tr>
<tr>
<td>TEST</td>
<td>1.0233</td>
<td>-1.1488</td>
<td>3</td>
</tr>
<tr>
<td>DEBUG</td>
<td>-0.3388</td>
<td>-0.4112</td>
<td>1</td>
</tr>
<tr>
<td>SQR</td>
<td>-1.3705</td>
<td>0.0131</td>
<td>1</td>
</tr>
<tr>
<td>SRCH</td>
<td>-0.8707</td>
<td>0.0115</td>
<td>1</td>
</tr>
<tr>
<td>PALIN</td>
<td>-1.5211</td>
<td>0.1183</td>
<td>1</td>
</tr>
<tr>
<td>SSORT</td>
<td>-0.6094</td>
<td>-1.0211</td>
<td>4</td>
</tr>
<tr>
<td>BSORT</td>
<td>-1.4506</td>
<td>0.3104</td>
<td>1</td>
</tr>
<tr>
<td>DS</td>
<td>-0.6467</td>
<td>-0.7121</td>
<td>1</td>
</tr>
<tr>
<td>CODE3</td>
<td>-1.5655</td>
<td>0.1972</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Task</th>
<th>X</th>
<th>Y</th>
<th>Cluster</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1</td>
<td>2.2899</td>
<td>0.0195</td>
<td>5</td>
</tr>
<tr>
<td>P2</td>
<td>2.1791</td>
<td>-0.5003</td>
<td>5</td>
</tr>
<tr>
<td>P3</td>
<td>1.9183</td>
<td>-0.1034</td>
<td>5</td>
</tr>
<tr>
<td>P4</td>
<td>1.5055</td>
<td>-0.0719</td>
<td>5</td>
</tr>
<tr>
<td>P5</td>
<td>1.3494</td>
<td>0.7461</td>
<td>5</td>
</tr>
<tr>
<td>P6</td>
<td>0.7403</td>
<td>0.9369</td>
<td>6</td>
</tr>
<tr>
<td>P7</td>
<td>0.3814</td>
<td>1.1013</td>
<td>6</td>
</tr>
<tr>
<td>P8</td>
<td>0.1474</td>
<td>0.9620</td>
<td>7</td>
</tr>
<tr>
<td>P9</td>
<td>0.3317</td>
<td>0.7656</td>
<td>7</td>
</tr>
<tr>
<td>P10</td>
<td>1.4815</td>
<td>0.0551</td>
<td>5</td>
</tr>
<tr>
<td>P11</td>
<td>0.9065</td>
<td>0.6886</td>
<td>5</td>
</tr>
<tr>
<td>P12</td>
<td>-1.5656</td>
<td>0.2680</td>
<td>1</td>
</tr>
</tbody>
</table>
Appendix IV: Random data generation scripts

# orthogonal.py: generate orthogonal sets of test data
# Robyn A. McNamara November 2002

from random import *

N = 4 # number of competencies
population = 400 # student population
students = []
questions = []

def vmult(v, w):
    if len(v) != len(w):
        print "Error: vectors of different lengths"
        print "v =", v,"; w =", w
        return

    p = 0.0 # accumulates vector product
    for i in range(len(v)):
        p = p + float(v[i]) * float(w[i])

    return p

# generate student vectors
for i in range(population):
    s = []
    for j in range(N): # generate student capabilities
        p = gauss(0.5, 0.25) + 0.25
        p = min(max(p,0),1) # ensure between 0 and 1
s.append(p)
students.append(s)

# generate question vectors
for i in range(10):
    questions.append([1, 0, 0, 0])
for i in range(10):
    questions.append([0, 1, 0, 0])
for i in range(10):
    questions.append([0, 0, 1, 0])
for i in range(10):
    questions.append([0, 0, 0, 1])

# work out student marks
for s in students:
    for q in questions:
        p = vmult(s, q)
        mark = 0
        for m in range(10):
            if random() > p:
                mark = mark + 1
        print "%d\t" % mark,
print
# orthomodal.py: generate orthogonal-modal sets of test data
# Robyn A. McNamara  November 2002

from random import *

N = 4  # number of base competencies
population = 400  # student population
students = []
questions = []

def vmult(v, w):
    if len(v) != len(w):
        print "Error: vectors of different lengths"
        print "v =", v,"; w =", w
        return

    p = 0.0  # accumulates vector product
    for i in range(len(v)):
        p = p + float(v[i]) * float(w[i])

    return p

# generate student vectors
for i in range(population):
    s = []
    for j in range(N + 2):
        p = gauss(0.5, 0.25) + 0.25
        p = min(max(p,0),1)
        s.append(p)
    students.append(s)
# generate question vectors
for i in range(5):
    questions.append([0.25, 0, 0.75, 0, 0, 0])
for i in range(5):
    questions.append([0.25, 0, 0, 0.75, 0, 0])
for i in range(5):
    questions.append([0.25, 0, 0, 0, 0.75, 0])
for i in range(5):
    questions.append([0.25, 0, 0, 0, 0, 0.75])
for i in range(5):
    questions.append([0, 0.75, 0.25, 0, 0, 0])
for i in range(5):
    questions.append([0, 0.75, 0, 0.25, 0, 0])
for i in range(5):
    questions.append([0, 0.75, 0, 0, 0.25, 0])
for i in range(5):
    questions.append([0, 0.75, 0, 0, 0, 0.25])

# work out student marks
for s in students:
    for q in questions:
        p = vmult(s, q)
        mark = 0
        for m in range(10):
            if random() > p:
                mark = mark + 1
        print "%d\t" % mark,
        print
# mixture.py: generate mixed sets of test data
# Robyn A. McNamara  November 2002

from random import *

N = 4  # number of competencies
population = 400  # student population
K=3  # factors per question
students = []
questions = []

def vmult(v, w):
    if len(v) != len(w):
        print "Error: vectors of different lengths"
        print "v =", v,"; w =", w
        return

    p = 0.0  # accumulates vector product
    for i in range(len(v)):
        p = p + float(v[i]) * float(w[i])

    return p

# generate student vectors
for i in range(population):
    s = []
    for j in range(N):
        p = gauss(0.5, 0.25) + 0.25
        p = min(max(p,0),1)
        s.append(p)
    students.append(s)
# generate question vectors
for i in range(40):
    q = [0.0] * N
for j in range(K):
    factor = randrange(N)
    q[factor] = q[factor] + 1.0
for j in range(N):
    q[j] = q[j] / float(K)
questions.append(q)

# work out student marks
for s in students:
    for q in questions:
        p = vmult(s, q)
        mark = 0
        for m in range(10):
            if random() > p:
                mark = mark + 1
        print "%d\t" % mark,
print

print "=" * 75
for i in range(len(questions)):
    print "Q[%d] = " % i, questions[i]
Bibliography


54


