Programmed to succeed?: A multi-national, multi-institutional study of introductory programming courses

Sally Fincher[1], Bob Baker[2], Ilona Box[3], Quintin Cutts[4], Michael de Raadt[5], Patricia Haden[6], John Hamer[7], Margaret Hamilton[8], Raymond Lister[9], Marian Petre[10], Simon[11], Anthony Robins[12], Ken Sutton[13], Denise Tolhurst[14], Jodi Tutty[15]

Abstract

This paper describes a multi-national, multi-institutional study that investigated introductory programming courses. Student participants were drawn from eleven institutions, mainly in Australasia, during the academic year of 2004. A number of diagnostic tasks were used to explore cognitive, behavioural, and attitudinal factors such as spatial visualisation and reasoning, the ability to articulate strategies for common-place search and design tasks, and attitudes to studying. The results indicate that: a deep approach to learning was positively correlated with mark for the course, while a surface approach was negatively correlated; spatial visualisation skills are correlated with success; a progression of map drawing styles identified in the literature has a significant effect with marks; and increasing measures of richness of articulation of a search strategy are also associated with higher marks. Finally, a qualitative analysis of short interviews identified the qualities that students themselves regarded as important to learn programming well.

1 Introduction

What factors might influence entry-level undergraduate students' success in learning programming? There is considerable practical and theoretical interest in this question. Initial efforts concentrated on occupational aptitude tests – selecting and evaluating those people most likely to have a successful and fulfilling career in the emerging computing industry (Cross, 1970); (D. Mayer & Stalnacker, 1968); (Wolfe, 1971). An alternative focus on academic success emerged during the 1980's. These studies, exploring factors that might predict success in an introductory programming "CS1" course, were driven by issues such as the rapid growth in popularity of programming courses, varying levels of student ability, and the consequent demand placed on teaching resources (Barker & Unger, 1983), (Chowdhury, Van Nelson, Fuelling, & McCormick, 1987); (Leeper & Sliver, 1982)

[1] Computing Laboratory, University of Kent, UK
[2] Ulysses Club, Bargo, NSW, Australia
[3] Department of Software Engineering, University of Technology, Sydney, Australia
[4] Department of Computer Science, University of Glasgow, UK
[5] Faculty of Sciences Maths and Computing, University of Southern Queensland, Australia
[6] School of Information Technology, Otago Polytechnic, NZ
[7] Department of Computer Science, University of Auckland, NZ
[8] School of Computer Science and Information Technology, RMIT University, Melbourne, Australia
[9] Department of Software Engineering, University of Technology, Sydney, Australia
[10] Department of Computing, Open University, UK
[11] School of Design, Communication and IT, University of Newcastle, Australia
[12] Computer Science Department, University of Otago, NZ
[13] School of Information Technology, Southern Institute of Technology, NZ
[14] School of Information Systems, Technology and Management, University of New South Wales, Sydney, Australia
[15] School of Information Technology, Charles Darwin University, Australia

We believe that learning to program is problematic, and that the results achieved by students do not correlate well with their other academic results. Understanding of this phenomenon is patchy and poorly integrated, but it does seem clear that there are many influences at play. Factors suggested in the literature include mastery of one's native language (the ability to communicate clearly and effectively both in speech and in writing), number of programming languages used or examined, spatial reasoning and mathematical ability, musical ability, logical reasoning ability, and previous academic background. Measures of general intelligence correlate well (R. E. Mayer, Dyck, & Vilberg, 1989). The best indicators of success appear to be self-predicted success, attitude, keenness and general academic motivation (Roddan, 2002); (Rountree, Rountree, & Robins, 2002). However, this does not distinguish programming from other disciplines, and has such a large effect that it may mask more subtle, discipline-specific, indicators.

The most extensive of recent studies (Wilson & Shrock, 2001) explores twelve possible predictors. These include standard factors such as mathematical background, work style preference and previous programming experience, and also a range of student self assessments. Assessments that proved to be of particular interest include "comfort level" (based on students' perceptions of course/programming difficulty and level of anxiety) and "attributions" (based on students' beliefs about their reasons for success or failure). Comfort level was found to be the most significant predictor of success, with mathematical background the second, and attribution of success to luck (which correlated negatively with success) the third in order of significance.

This report presents a study of possible influencing factors that is distinctive in a number of ways. First, it is both multi-institutional and multi-national, with participants from eleven institutions in three countries. This breadth lends support to generalisations about factors that vary across prior educational experience – and hence that are likely to be influenced by educational intervention – and factors that are invariant. Second, the data examined is particularly broad, with separate subtasks exploring attitudinal, cognitive, and behavioural factors, and a short open-ended interview. This allows many diverse research questions to be addressed using multiple methods of analysis. Third, the scale of the study, with 177 participants from eleven institutions in three countries, reduces sample bias and increases generalisability. Given the cost and challenges of carrying out empirical research at this scale, few precedents for programming related studies of this size and scope exist (Lister et al., 2004); (McCracken et al., 2001); (Petre et al., 2003); and (Fincher et al., 2004) are examples).

Section 2 describes the design of the study and its focal tasks. Using the data collected, many of the questions that shaped the study design have been explored. Sections 3 – 7 present this analysis. The final section is a summary and discussion.

2 The study

2.1 Overview

The study was based on four different diagnostic tasks in an attempt to determine or eliminate factors that might relate to early programming performance. Eleven institutions participated, using the same protocol to gather data from students in introductory programming courses that were taught during 2004. Data was then pooled and analysed. The four focal tasks were:

- a standard paper folding test, a **cognitive** task focusing on spatial visualisation and reasoning;
- map sketching, a **behavioural** task used to assess the ability to design and sketch a simple map, and to articulate decisions based on that map;
- searching a phone book, a **behavioural** task used to assess the ability to articulate a search strategy;
- a standard study process questionnaire, an **attitudinal** task focusing on approaches to learning and studying.

A subset of the researchers conducted small pilot studies to trial and refine the overall process and the specifics of the behavioural tasks. The attitudinal and cognitive tasks employed standard instruments as described below. In most cases student participants completed all four tasks, and a short open-ended interview.

The design of this study takes account of several factors. *Paradigm independence*: the use of diverse and generalised stimuli makes the tasks independent, so that comparisons can be made across paradigms, languages, and pedagogic styles. *Triangulation*: the study combines different approaches and collects both qualitative and quantitative data, in order to provide opportunities to contradict or corroborate within the study, by comparing the different factors. *Building on existing work*: part of the study replicates work for which there is standardised data available. *Scale*: the number of institutions means that the total number of participants recruited is at a scale unusual in the literature.

The difficulty of predicting programming success is compounded by the lack of an agreed, established 'core' list of essential programming concepts, let alone any robust instruments for assessing students' acquisition of programming concepts or misconceptions. Certainly there is nothing comparable to the 'Force Concepts Inventory' in Physics (Hestenes, Wells, & Swackahmer, 1992) or explorations in Signals and Systems (Nasr, Hall, & Garick, 2003). Hence, like many researchers in this field, we relate our findings from the diagnostic tasks to the grades achieved by students in an introductory programming course, leaving somewhat open the question of how accurately these grades reflect the students' programming ability.

2.2 Method

The protocol

Each researcher followed the same protocol for data collection. The researcher met with students from their institution in an individual session scheduled near the start of their introductory programming course. During this session participants completed the paper folding, map and phone book tasks, and a short open-ended interview[1]. Towards the end of

[1] In the case of institutions E and I the paper folding test was administered to participants collectively.

their course participants completed the study process questionnaire task. The specifics of the four tasks, and interview, are discussed in Sections 2.3 – 2.7.

Participants

A total of 177 volunteer participants were recruited from the introductory programming courses at eleven institutions of post-secondary education in Australia, New Zealand and Scotland: ages ranged from 17 to 50 (three quarters were 22 or younger), with 137 males and 40 females. One institution (I) contributed 32% of the total participants, but not for all tasks, the next highest contribution was 8%. It should be noted that, across all institutions, not all participants completed all tasks.

2.3 Task 1: Paper folding test

The *Paper Folding Test* (VZ-2) is taken from the ETS Kit of Referenced Tests for Cognitive Factors (Ekstrom, French, Harman, & Dermen, 1976). The test is designed to measure visualisation and spatial reasoning, based on the ability to manipulate and transform spatial patterns, and hence to recognise whether one image is a transformation of another. In this case, participants identify which pattern of holes would result in an unfolded sheet of paper after holes are punched through an arrangement of folds.

The test consists of 20 questions, in two sets of 10, with a time limit of three minutes per set. The instructions to participants, including a simple example, are shown in Appendix A. Data recorded in this task were the time taken to complete each set, and the numbers of questions answered correctly, incorrectly, and not at all.

2.4 Task 2: Map sketching

This task is drawn from classroom practice and a tradition in computer education that uses commonplace examples to convey programming concepts and make them relevant to students (Curzon, 2002). The goal is to assess participants' ability to design and sketch a simple map, and articulate decisions based on that map.

Participants were asked to sketch a map of a route between two known locations on or near their campus, a map that would be useful to a stranger. They were then asked to annotate the map with decision points, describing how to recognise each decision point and what to do at that point. The protocol used by the researchers is shown in Appendix B. Data recorded in this task included the sketch maps (with any annotations), an audio recording of the session, and researcher's notes regarding the order in which the map was drawn.

2.5 Task 3: Phonebook searching

The phonebook task was intended to use an everyday activity to assess the participant's ability to articulate a commonplace search strategy.

Participants were asked to look up a specified name in the local phone book, and then describe the process that they had used to find the name. They were then asked to look up a second name, describing that search as they conducted it. The protocol used by the researchers is shown in Appendix C. Data recorded in this task included an audio recording of the session, and researcher's notes regarding the nature of the search and the quality of the articulation.

2.6 Task 4: The study process questionnaire

The *Biggs Study Process Questionnaire* derives from the notion that students' perceptions and learning activities are central to learning. An 'approach to learning' encompasses the relationship between student, context, and task (Biggs, Kember, & Leung, 2001). The revised questionnaire assesses deep and surface approaches to learning in the context of a particular course.

Towards the end of their introductory programming course participants completed the revised questionnaire, consisting of 20 closed-response questions, scored on a 5-point Likert scale. The instructions and questions are shown in Appendix D. The only data recorded were the participants' answers to each question.

2.7 The interview questions

The final element of the study was a wholly qualitative-semi-structured interview. We had no particular expectations regarding the outcome, apart from a belief that the richness of qualitative data can highlight factors that are difficult to capture using more structured tasks, and can facilitate the exploration of a wide range of approaches and methods of analysis.

At the end of their main session (having completed tasks 1 – 3) participants were asked:

a) What do you think we were trying to find out?
b) How do you think the sketch–map task might relate to programming?
c) How do you think the phone book task might relate to programming?
d) What qualities or skills do you think are important to learn programming well, to "get it"?

An audio recording of each session was made.

3 Study process questionnaire

3.1 Focal question

Are there identifiable aspects of approaches to study that correlate to early programming performance?

3.2 Background and motivation

The emphasis in the Biggs revised two-factor Study Process Questionnaire R-SPQ-2F is upon the context-specific and the situated nature of learning. "Students' approaches to learning are conceived as forming part of the total system in which an educational event is located" (Biggs *et al.*, 2001, p. 135). This approach to learning will depend upon a number of factors that range from the personal (e.g. motivation, available time, personal perception of task demands) through environmental (e.g. classroom climate, learning activities, assessment methods) to institutional factors (course culture, curriculum design). These different factors affect how a student perceives the demands of a specific learning task and then how they choose to deal with it.

The R-SPQ-2F is designed to measure two different learning approaches, namely deep and surface. Students adopting a deep approach aim from the outset to develop a broad understanding of the task and relate it to other topics and their personal experience. Students adopting a surface approach build their view from facts and details of activities with the aim of reproducing material rather than making theoretical connections.

While individuals do have a preference for deep or surface approach that is relatively stable over time (Biggs, 1987), learning approach is not a fixed trait of the individual and can fluctuate over time and between tasks (Coffield, Moseley, Hall, & Ecclestone, 2004)[2]. In some circumstances, students will combine both learning approaches. Which approach, or combination of approaches, students use will depend both on their motivation and the strategies they adopt in reaching their goals. The deep approach is typically motivated by intrinsic interest in the material while conversely the surface approach is often associated with fear of failure. Where achievement is a major motivation both approaches are often combined. Such students become adept at organising their study time and methods, matching their approach (whether surface or deep) to the demands of the task and attending to cues given by lecturers as to what type of work will obtain good grades. Thus as the motivational mix and consequent strategy adoption vary from subject to subject and time to time, an individual student's learning approach is also likely to vary.

Not all students are able to accurately discriminate their possible motives for learning and associate them with appropriate strategies. The level of congruence between the strategies chosen and motivation of the students reflects the extent to which students are behaving metacognitively (Biggs, 1987). Metacognition refers to the reflective self-awareness that students have in consciously using their cognitive processes in learning. This ability to select appropriate strategies is important for success. For example, a deep learning approach being associated with a surface strategy resulted in a poor English performance as the wide reading produced more data than the reproducing strategy could handle (Biggs, 1987). Studies have

[2] Coffield *et al.* (2004) refers specifically to Noel Entwistle's instruments Approaches to Studying Inventory (ASI), Revised Approaches to Studying Inventory (RASI) and Approaches to Learning and Study Skills Inventory for Students (ASSIST) rather than Biggs SPQ or R-SPQ-2F model. However as the conceptual models underpinning these instruments are very similar and their development has been interrelated, comments on Entwistle's instruments are likely to be relevant to Biggs's instruments.

also reported that students unfamiliar with study and from 'non-traditional' (i.e. without entry qualifications) backgrounds may lack the skills to match their motivations with appropriate learning approaches (Coffield et al., 2004).

Although a clear theoretical case can be put for an association between learning approaches and achievement, care is needed as unexpected or contextual factors can disrupt this association (Coffield et al., 2004). The structure of the curriculum and demands of the assessment strongly influence the learning approaches taken. Surface approaches to learning (Biggs, 1987) may foster an undesirable dependency by 'spoon-feeding' (Coffield et al., 2004), Biggs *et al.* (2001) argue that where the teaching and assessment methods are not aligned to the aims of the course, a surface approach can unwittingly be encouraged. In these learning environments, adopting a surface approach can be a strategic decision and such strategic approaches are often associated with high achievement (Coffield et al., 2004). Students can be influenced to take a deep approach by being given freedom in their learning, experiencing quality teaching with a realistic pace and real-life illustrations and a supportive and lively classroom environment.

The R-SPQ-2F yields the two approach scores, surface and deep respectively and their component motive and strategy score:

- surface approach (SA)
    - motive (SM): meet requirements minimally
    - strategy (SS): limit to bare essentials and reproduce through rote learning
- deep approach (DA)
    - motive (DM): study to actualize interest and competence
    - strategy (DS): is to read widely and interrelate with previous relevant knowledge.

3.3 Analysis

Given the contextual nature of the questionnaire, for this study the student responses needed to be framed by their experiences from their programming course rather than any other course. (Biggs, 1987) p.39 notes that "deep-related scores are not expected to relate to performance unless the student is intrinsically interested in the task". Where students had no preference for the subject area, the correlation is not as high. Accordingly the following sentence was added to the questionnaire:

> "If you think your answer to a question would depend on the subject being studied, please give the answer that would apply to your programming course."

Students also answered the questionnaire towards the end of the semester after they had experienced the course. Despite these precautions, and given the variations in the administration of the questionnaire across institutions discussed below, some students may have answered about their learning approach in general rather than in respect to programming. If so, some students may have recorded a deep approach to learning in general rather than programming and any correlation with mark is likely to be less strong.

Due to individual institution considerations, both the methods of administration (email, online, face to face in a lecture or interview, over the phone) and the time of delivery (last weeks of semester, before exams, after exams, after results) varied.

Given the timing of the questionnaire clearly students who withdrew early in the semester were not included in the sample. While this was unavoidable it does mean that a group of students who were not successful were not included in this task.

The results from the questionnaire were combined for all institutions and compared to the final mark. Where a complete set of data had not been collected for a student their figures were not included in the analysis. This typically occurred in two situations: either students withdrew after the questionnaire was administered but before the end of the course and so had an incomplete final mark, or students missed questions on the Biggs questionnaire form. From 129 "complete" students (i.e. those that for which we had participant data and grade greater than zero on the programming course) this left a population of 104. Once correlated the two approach scores, surface (SA) and deep (DA) and their component motive (SM and DM respectively) and strategy (SS and DS respectively) score were calculated.

3.4 Results

The scores on the Biggs questionnaire of all complete students (n=129) are summarized in Table 3.1. As a population, the students are not strongly aligned with either deep or surface learning, and their scores on the given questions are equivocal.

	DA (10-50)	DM (5-25)	DS (5-25)	SA (10-50)	SM (5-25)	SS (5-25)	Q3 (1-5)	Q6 (1-5)	Q8 (1-5)	Q13 (1-5)
Mean	28.8	14.5	14.3	23.5	10.8	12.7	2.3	2.8	2.2	3.1
Std. Dev.	7.0	4.0	3.7	7.2	3.7	4.0	1.3	1.1	1.2	1.1

Table 3.1 Aggregate statistics for complete students on the Biggs instrument

The results of the correlations between the Biggs questions and final results are presented in Table 3.2 below.

Measure	Correlation with Mark	Significance	R Square
DA	0.29	0.003	0.083
DS	0.26	0.007	0.068
DM	0.26	0.007	0.068
SA	-0.25	0.009	0.063
SS	-0.23	0.020	0.051
SM	-0.24	0.013	0.058

Table 3.2 Correlations of Biggs questions with final result.

Graphs of DA and SA against mark are shown in Figure 3.1. The correlations for the 4 highest questions are shown in Table 3.3.

Measure	Correlation with Mark	Significance	R Square
Biggs Instrument Question 6	0.31	0.001	0.10
Biggs Instrument Question 13	0.31	0.001	0.09
Biggs Instrument Question 8	-0.25	0.008	0.06
Biggs Instrument Question 14	0.24	0.014	0.06

Table 3.3 Highest correlations for individual questions

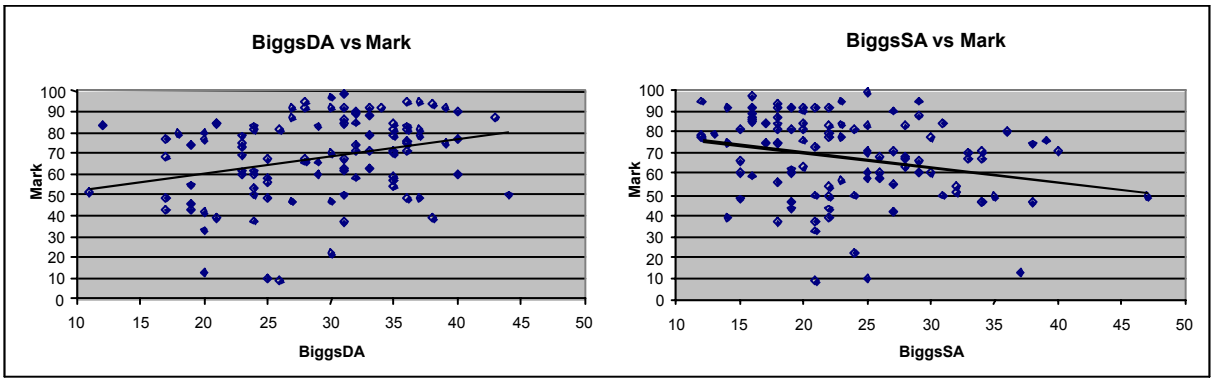


Figure 3.1 Deep learning approach against mark and shallow learning approach against mark showing trend lines

To investigate the degree of institutional difference within the complete dataset, correlations on the Biggs scales were calculated for each institution. The correlations are shown in 3.4. Each row of that Table shows, for a single participating institution, the correlation to student marks for DA and SA. For several of the institutions, the amount of data (N) is very small and no strong conclusions can be drawn. However, correlations are broadly consistent with the mark/DA correlation usually positive and the mark/SA correlation usually negative.

Institution	"Complete" Students	N	DA	SA
A	12	11	0.63	-0.58
E	13	13	0.42	-0.58
F	8	8	0.37	-0.02
H	13	12	0.43	-0.52
I	39	22	0.36	-0.29
J	12	9	0.54	-0.06
K	9	8	-0.48	-0.56
N	7	7	0.11	0.14
P	16	14	0.08	-0.22
Total	129	104		

Table 3.4 Learning approaches within the various institutions.

Pooling the "complete" student population into quartiles, depending on their grade, the average learning approach scores for each of the four quartiles are shown below in Table 3.5.

Quartile	BiggsDA	BiggsSA	BiggsDM	BiggsDS	BiggsSM	BiggsSS
1	26	26	13	13	12	14
2	29	26	15	14	12	14
3	29	22	14	15	10	12
4	31	20	16	16	9	11

Table 3.5 Learning approach scores for each of the quartiles

3.5 Discussion

As would be expected there was a positive correlation between deep learning approaches and mark, and a negative correlation between surface learning approaches and mark. Students

who engaged more deeply with the material tended to do better than those who did not. This result is consistent with results reported in (Biggs, 1987).

Biggs (Biggs, 1987) suggests students who adopt a deep learning approach are likely to succeed and those who employ a surface approach are likely to fail. A student's choice of learning style is dependant upon the complex interaction between the learning environment and that student's decisions, motivation and metacognitive ability. In a study such as this across multiple institutions with different curricula and assessments, strategic decisions made by students on the most appropriate learning approach to adopt will vary from institution to institution. Biggs *et al.* (2001, p. 137) writes that at the end of the course the scores "may describe how teaching contexts differ from each other". This variation in the strengths of the correlations was seen between the institutions.

When the average approach scores for the different quartiles are examined, the interesting feature is that the difference between the deep and surface approach scores becomes more prominent as the mark increases. In the top quartile, the deep scores are higher than the surface scores while in the bottom quartile the deep and surface scores are the same. Thus, on average, students in the bottom quartile are neither predominantly using a surface or deep approach to learning. It needs to be noted that as the surface and deep approach scores are measured by different questions, in theory it is possible for a student to score high (or low) on both, although (Biggs, 1987) claims that high scores for both learning approaches are incompatible. This has been noted as a weakness of the questionnaire as the underlying theory sees surface/deep approaches on a continuum rather than as orthogonal (Coffield et al., 2004).

One explanation for this difference is the metacognitive skills of the students. Students in the top quartile are clearer about their motivation and can discriminate the appropriate learning strategies. Conversely students in the bottom quartile have a mixture of motivations and strategies. Possibly their metacognitive skills are lower and they have more difficulty in identifying appropriate strategies for the learning environment or are unable to match their motivation to an appropriate strategy.

These results so far have been for students who completed the courses. There were a number of students who completed the Biggs questionnaire at the end of the semester and then dropped out before the exam. As their final mark was incomplete these students were not included. However, when these students were included the correlations described above were all stronger. Not surprisingly those who discontinued at this late stage were dominated by surface learners.

4 Paper folding test

4.1 Focal question

Is success in a cognitive task focusing on spatial visualisation and reasoning associated with success in early programming performance?

4.2 Background

As discussed in Section 2.3, the paper folding test is one of a series of tests for measuring cognitive abilities. It should be noted that this study is not the first to use a paper folding test as a means of assessing aptitude for computer programming. (Evans & Simkin, 1989) used the same test, but it only accounted for a small portion of their entire study.

4.3 Analysis

The performance of students on this task was a score out of 20, the total number of correct answers selected over the two sets of 10 questions.

4.4 Results

On the paper folding test, participants in the current study scored a mean of 14.1 out of 20, with a standard deviation of 3.4. This result is consistent with that of (Ekstrom et al., 1976), who reports a mean score of 13.8 and a standard deviation of 4.5 (for 46 college students). Of the 129 complete students, only 13 completed the first set of 10 paper folding questions in less than the allotted 180 seconds, taking between 110 and 165 seconds.

Overall, a small positive correlation between performance in the paper folding test and mark (correlation = 0.17, p=0.047, R^2 = 0.03) was found. Results of the correlation between paper folding and mark are presented below in Figure 4.1.

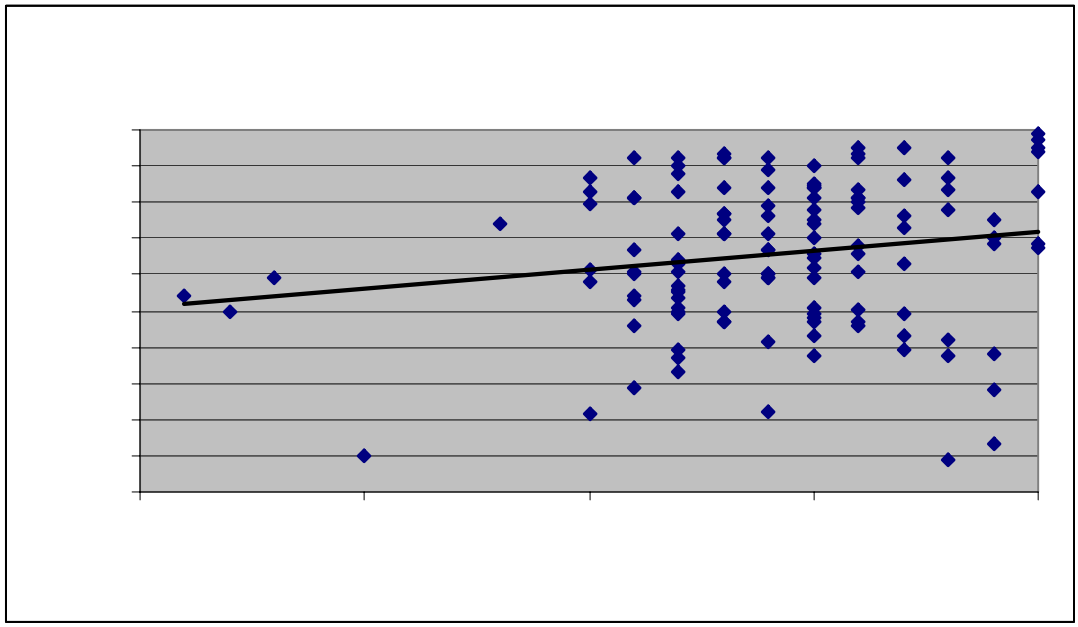


Figure 4.1 Scatter graph of paper folding test score against final mark

No clear correlation was found between paper folding and learning approach. No correlation was found between paper folding and a surface approach but a slight negative correlation was found between paper folding test score and a deep learning approach.

4.5 Discussion

The correlations between mark and paper folding score are significant while not being overly strong. The lack of correlation with learning approach may be attributed to the fact that learning approach scores are an attitudinal measure while the paper folding test is a cognitive measure. The slight negative correlation between a deep learning approach and the paper folding may also be a reflection of the time pressures involved in the paper folding test. Strategic approaches eliminating some of the suggested answers may well be more effective in the short time period than actually trying to work out what the paper will look like unfolded. Anecdotal evidence supports this with reports of students using strategies such as counting the number of holes or using the presence/absence of a particular dot to eliminate possible answers.

5 Map design

5.1 Focal questions

Is there a relationship between the style of map (landmark, route or survey) produced for a navigation task and marks achieved in a programming course? Is there a relationship between the style of descriptions of navigational decision points and the marks achieved in a programming course?

5.2 Background and motivation

There is evidence that programmers use mental imagery when designing programs (Green, 1997); (Petre & Blackwell, 1999). In a study of expert programmers, Petre and Blackwell identify a variety of spatial metaphors used by programmers to capture the way they think about emerging solutions during the programming task. All of the experts they studied reported using spatial representations such as describing a problem space as a landscape:

> "…oh, that happens over there… it's on the horizon, so I can keep an eye on it, but I don't really need to know…" (p117).

Finding one's way in information spaces has been likened to wayfinding and navigation in physical space. Metaphors of getting lost, navigational links, hyperspace, disorientation, moving up and down in text, information landscapes, and so on are now commonplace in the literature on hypertext and the World Wide Web (McKnight, Dillon, & Richardson, 1991), (Benyon & Wilmes, 2003).

In the context of programming, program code has been characterised as a virtual space – a Codespace (Cox & Fisher, 2004) and many of the problems that programmers face in navigating a Codespace have been likened to those encountered when navigating physical space. While developing, and in understanding program code, the programmer has to locate code segments and move between them. Cox and Fisher stress the importance of a structural view of a program – its layout and organisation – for being able to move around a Codespace.

In a study of novice programmers, (Mosemann & Wiedenbeck, 2001) investigated whether different program navigation methods would lead to differences in program comprehension, as characterised by (Biggerstaff, Mitbander, & Webster, 1993);

> "A person understands a program when able to explain the program, its structure, its behaviour, its effects on its operational context, and its relationships to its application domain in terms that are qualitatively different from the tokens used to construct the source-code of the program."

Mosemann and Wiedenbeck hypothesized that a sequential flow strategy would lead to a fragmented bottom-up view of the program, whereas a control flow strategy would lead to a top-down, hierarchical view of a program. They found no significant difference between the two strategies on measures of program comprehension, and that novice programmers were comfortable with both styles of navigation. They note that in spite of the sequential method being most natural for novices, those that used a control flow strategy performed as well.

We were interested in finding whether these patterns of preferences for navigation strategies were reflected in map drawing tasks, and how these physical navigation strategies related to performance in a programming course.

Landmark, route, and survey knowledge

(Werner, Krieg-Bruckner, Mallot, Schweizer, & Freksa, 1997) suggest a hierarchy in acquisition of spatial knowledge, from landmark to route to survey knowledge, which they describe as follows:

> "Different forms and representations of spatial information can be identified in systems navigating in complex surroundings. One of the most common distinctions in spatial navigation research concerns the difference between landmark, route, and survey knowledge of an environment. Landmarks are unique objects at fixed locations, routes correspond to fixed sequences of locations as experienced in traversing a route; survey knowledge abstracts from specific sequences and integrates knowledge from different experiences into a single model."

A similar transitional model is suggested by (Poucet, 1993), who identifies three stages in building a survey representation:

- A representation of place with local reference frames (landmarks)
- Place representations are linked to together but retain local references (routes)
- The reference frames for different places are changed to a common reference system (survey)

We used these distinctions as a basis for coding the maps produced in the mapping task.

5.3 Analysis

5.3.1 Coding of the map-drawing data

The map-sketching exercise was designed "to assess students' ability to articulate a simple familiar search and decision strategy effectively".

Participants were asked to carry out a mapping task in two phases, firstly to draw a map of a given area so that the researcher could get from one location to another. Details of the protocols are outlined in Appendix B. Participants were then asked to annotate the map with key decision points, explaining what a person using the map would need to do at the decision points (after (Lynch, 1960) and (Passini, 1984)).

Maps constructed in this activity were photographed or scanned electronically and the resulting images printed. These maps were then analysed for evidence one of one of the three broad navigation strategies of landmark, route or survey.

Decisions about the strategy used by participants were based on identifying the following criteria in the participants' maps:

Landmark:

- the focus of the map is key visual landmarks (fountains, sculptures, notable buildings) which are given prominence
- landmarks are unique objects
- landmarks often exaggerated in size or distorted, or otherwise given prominence (e.g. labelling, bolder print)
- route moves from landmark to landmark

Route:

- the focus of the map is the actual route taken
- route takes prominence, often with a path identified by a line or written directions on the map

- features on the route are identified

Survey:

- the map focuses on an overview of the area surrounding the route and includes detail beyond the route
- there is a focus on overall structure of the area
- may include compass points
- may not have the route drawn on, or the route is not prominent
- shows integrated knowledge of the area

A sample of each style of map and key characteristics follows.

Example of a landmark map

The map in Figure 5.3.1 was characterised as using a landmark map-drawing strategy because the route is predominately identified by key landmarks on the route, for example: 'big square cube thing', 'cobblestone path', traffic lights', etc. The focus of the map is these landmarks.

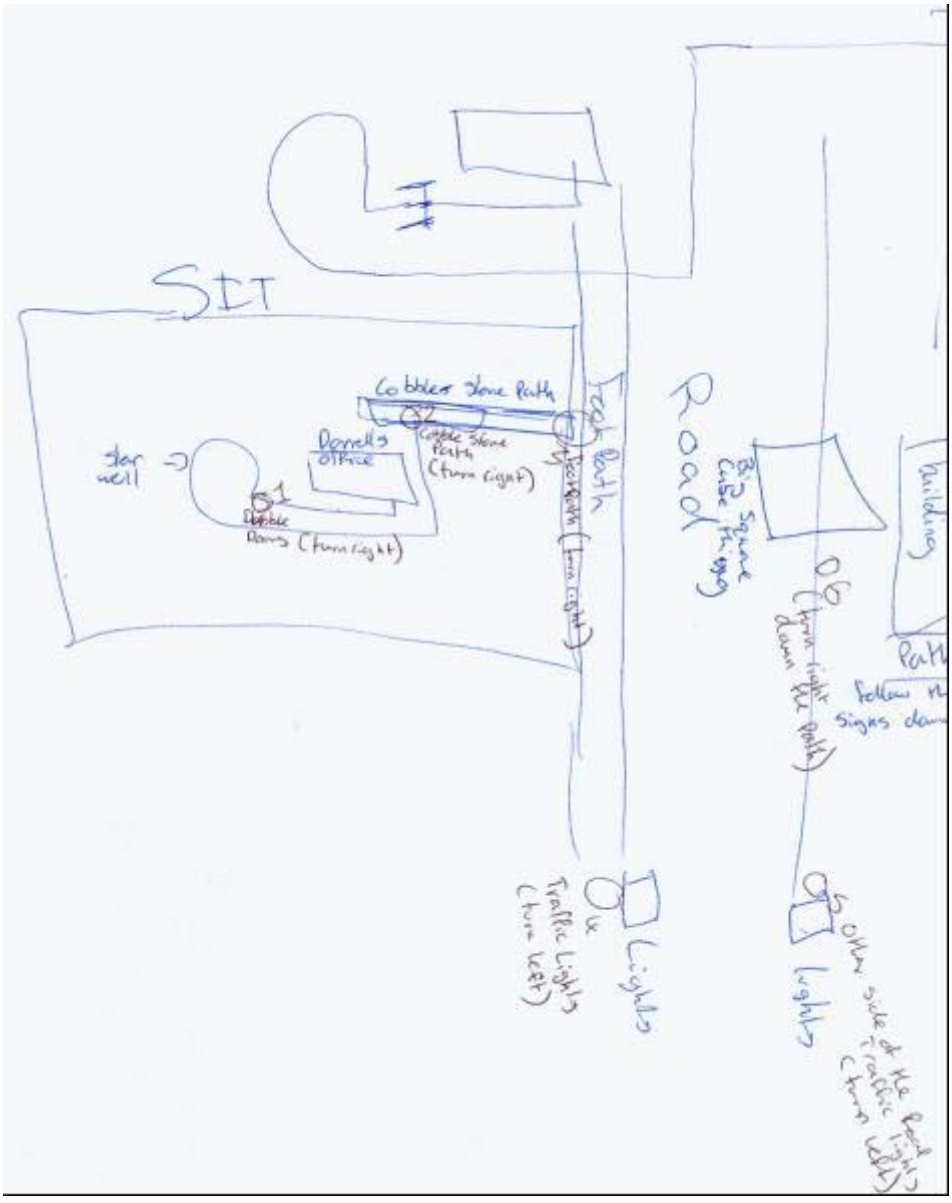


Figure 5.3.1 Sample of a student map that demonstrates the 'landmark' map-drawing strategy

Example of a route map

The map in Figure 5.3.2 was characterised as using a route map-drawing strategy because the route is predominately identified by a path from the beginning to destination. The focus is on the route, with little detail provided of the surrounding area. Some landmarks are evident (the Botanical garden for example) the focus of the map is on the path to be taken.

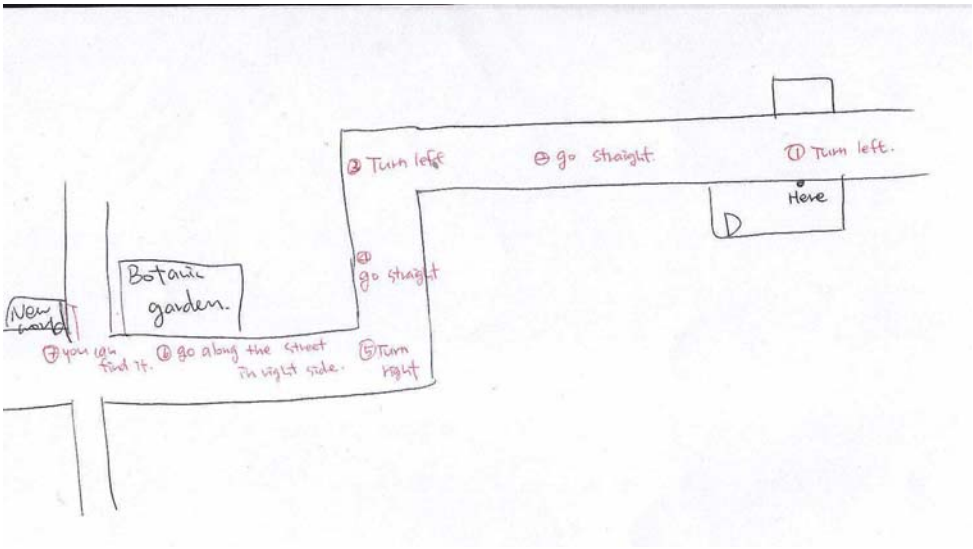


Figure 5.3.2 Sample of a student map that demonstrates the 'route' map-drawing strategy

Example of a survey map

The map in Figure 5.3.3 was characterised as using a survey map-drawing strategy because the map predominately includes a survey of the surrounding area to the route. Detail goes beyond the path for the route, providing 'survey' information for orientation. This map shows survey information around the route to get to the required building, and also survey information of the building.

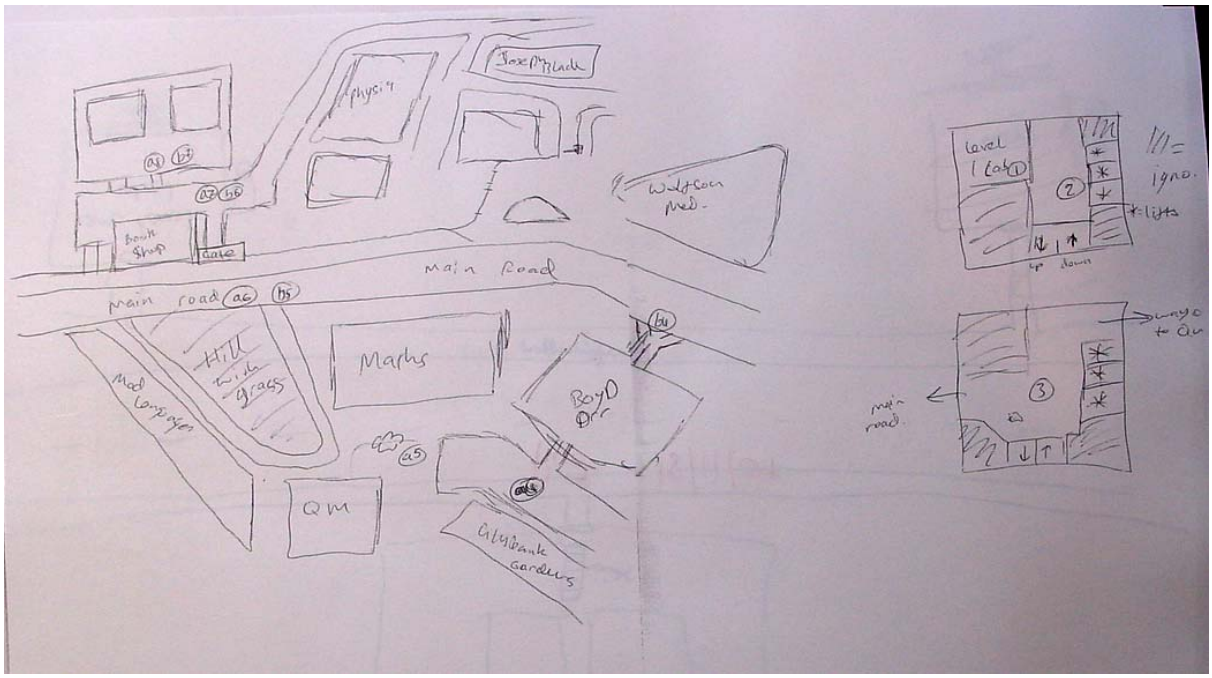


Figure 5.3.3 Sample of a student map that demonstrates the 'survey' map-drawing strategy

5.3.2 Issues in coding the maps

Coding of some participants' maps did cause some difficulties. These difficulties were: firstly, that not all maps could clearly be identified as belonging to one category; and secondly that the physical environment of the task for some institutions meant that students could not demonstrate a use of more than one strategy. Elaboration of these difficulties follows.

Strategy identification problems

While some students' maps could be clearly be identified as using either landmark, route or survey strategy, some maps were difficult to identify as using just one clear strategy. Some maps appeared to include features from two different strategies, for example: a map that provided route information, but also included traces of survey data on the edges of the route. Such survey information was minimal, but did go beyond the focus on the route. An example of a map that provided such problems is shown in Figure 5.3.4

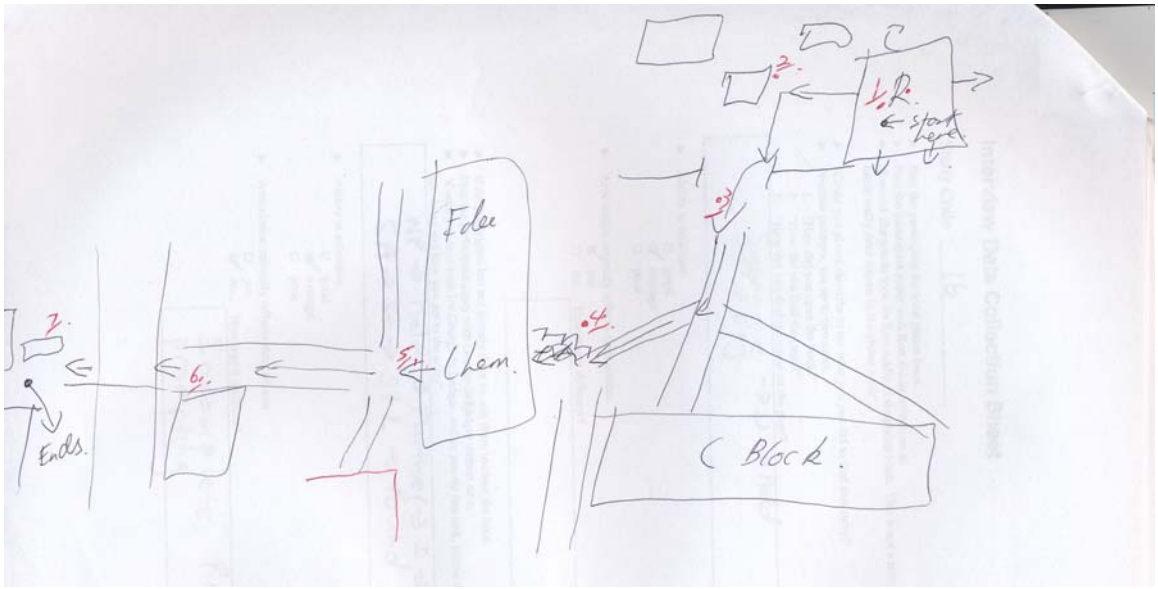


Figure 5.3.4 Sample of map that provided difficulties in coding – map-drawing strategy with features of both route and survey.

Map-drawing strategy limited by physical environment

Some institution locations were such that the physical environment limited students to drawing maps that could make little use of landmarks and had little scope for providing survey information. These locations did not allow students to draw detailed orienting information that was indicative of a survey strategy. Such an institution was a campus based in a high-rise building, in a street of high-rise buildings that had a visually consistent streetscape. The task that students were effectively reduced to in this locations was 'maze-running', a cognitively different activity (Passini, 1984), which provides no scope for including landmark information or survey detail. An example of a map from this campus is shown in Figure 5.3.5.

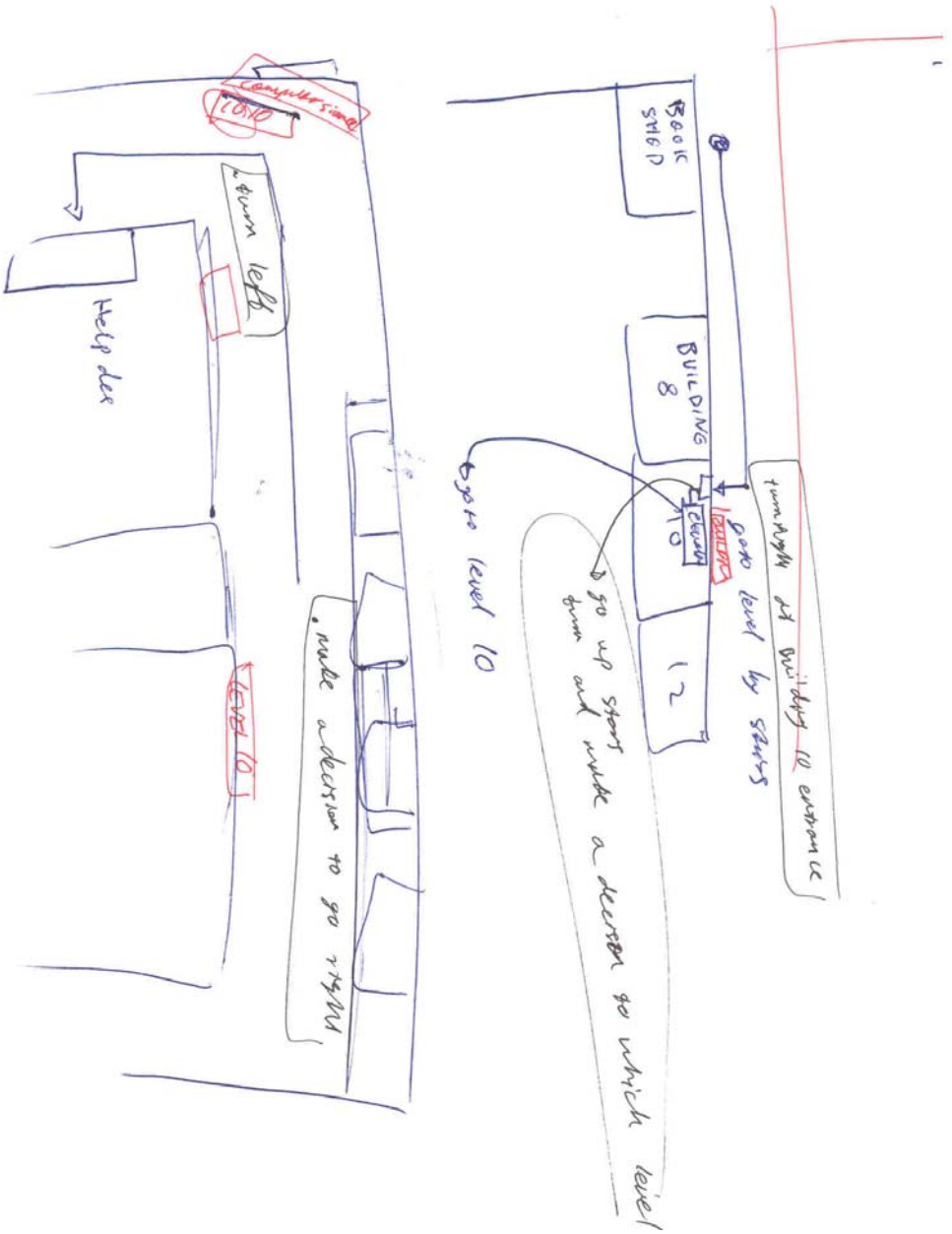


Figure 5.3.5 Sample of map that provided difficulties in coding – map-drawing strategy limited by physical environment of city streets and high-rise building.

The strategy to overcome the first issue, that of difficulty in coding some maps, is described in the following section. Analysis was carried out on the data for all institutions, with some further analysis excluding data from those institutions where participants were restricted in the map-drawing strategies they were able to use, due to the nature of the environment they were sketching.

5.3.3 Coding of maps for reliability

Two researchers worked together on one data set and developed a coding scheme to identify the mapping strategies (landmark, route and survey). This scheme was trialed on another set of data for one institution and refined to the scheme described above in the introduction to this section. The two researchers then independently coded the remaining maps as landmark, route and survey. The codes from the two researchers were compared with a reliability of

76% (with the data for the institutions whose physical environments that caused problems, as described above, removed). A third researcher then considered the codes that were anomalous. Ultimately the three researchers reached consensus on the final coding of the maps with originally anomalous coding.

5.4 Results

5.4.1 Descriptive statistics

Basic descriptive statistics were calculated using SPSS V11.0 for each institution. Table 5.4.1 shows means, Ns, and standard deviations as a total for each institution (166 of 177 participants submitted artefacts for this task), and also divided according to the map-drawing strategies of landmark, route and survey.

Institution	Mapstyles, Totals (and missing maps)	Mean	N	Standard deviation
All	Missing maps	41.0	32	25.0
	Landmark	54.7	15	27.7
	Route	64.7	84	21.0
	Survey	64.2	35	27.0
	Total	59.1	166	25.3
A	Landmark	8.0	1	-
	Route	57.7	6	34.0
	Survey	69.0	7	26.9
	Total	59.7	14	32.1
B	Landmark	-	0	-
	Route	53.8	6	13.2
	Survey	94	2	8.5
	Total	63.8	8	22.0
E	Landmark	64.0	2	24.0
	Route	64.6	5	17.4
	Survey	69.2	6	19.3
	Total	67.0	13	17.6
F	Landmark	78	1	-
	Route	60.0	8	18.2
	Survey	-	0	-
	Total	61.8	9	18.1
H	Landmark	-	0	-
	Route	76.1	11	14.2
	Survey	90.5	2	2.1
	Total	78.3	13	14.1
I	Missing maps	41.0	32	25.0
	Landmark	64.5	3	27.1
	Route	67.9	13	18.4
	Survey	56.8	4	38.8
	Total	50.31	52	27.0
J	Landmark	-	0	-
	Route	70.3	12	27.0

	Survey	83.0	2	5.7
	Total	71.4	14	25.3
K	Landmark	73.0	1	-
	Route	58.5	8	26.4
	Survey	62.0	1	-
	Total	60.0	10	23.7
M	Landmark	37.7	3	7.8
	Route	57.0	6	18.2
	Survey	56.0	2	46.7
	Total	51.6	11	21.8
N	Landmark	10.0	1	-
	Route	64.8	5	11.2
	Survey	77.0	1	-
	Total	58.7	7	23.8
P	Landmark	72.3	3	24.7
	Route	65.8	5	18.7
	Survey	41.9	8	20.6
	Total	55.1	16	23.7

Note: There are no missing maps for participants at institutions unless otherwise stated.

Table 5.4.1 Means, Number and Standard Deviations of final marks for each institution by map-drawing strategy, totals and missing maps

Trends in the map-drawing strategies adopted by students in each institution are shown in the box-plots in Figures 5.4.1 to 5.4.13. Figure 5.4.1 shows the box-plot for all institutions, and Figure 5.4.2 shows a box-plot for all institutions except the two institutions that were problematic to code (as discussed in Section 5.3.2), above.

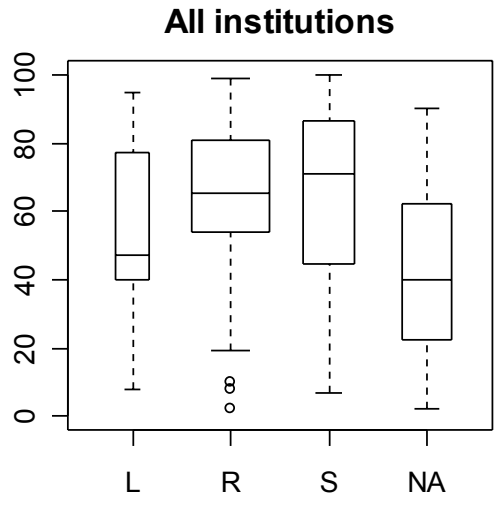


Figure 5.4.1 Box-plot of map-drawing strategies for all institutions

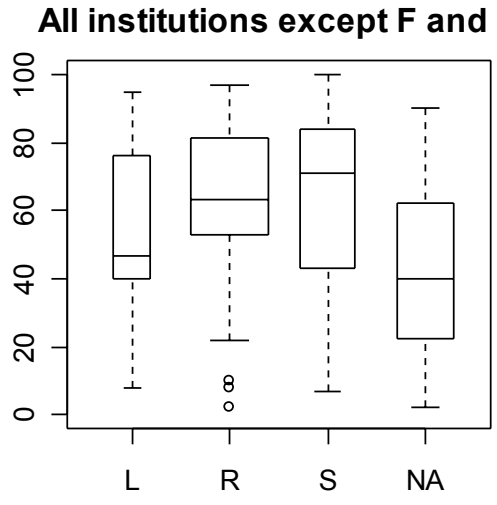


Figure 5.4.2 Box-plot of map-drawing strategies for all institutions, except F and H

Figure 5.4.3 shows the box-plot for the one United Kingdom institution involved in the study, indicating a trend in which students who draw maps focusing on landmarks gain

higher overall final marks, students who draw route maps gain the middle marks in the cohort and students who draw survey maps gain overall lower marks.

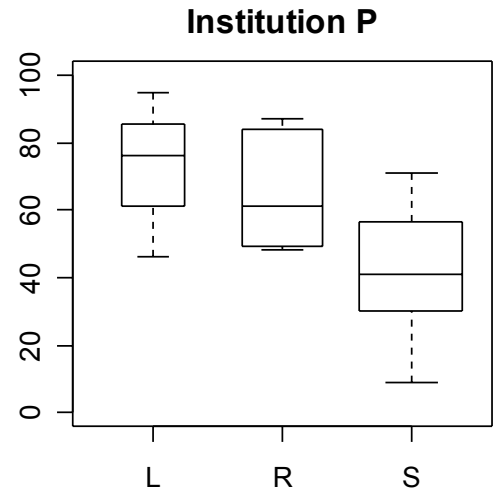


Figure 5.4.3 Box-plot of map-drawing strategies for United Kingdom institution P

Figures 5.4.4 to 5.4.7 Show the box plots for institutions in New Zealand. The box-plots for these institutions indicate a trend in the opposite direction to those found in the United Kingdom institution. Students drawing maps focusing on survey style gain highest marks, while those using a landmark style gain lowest marks.

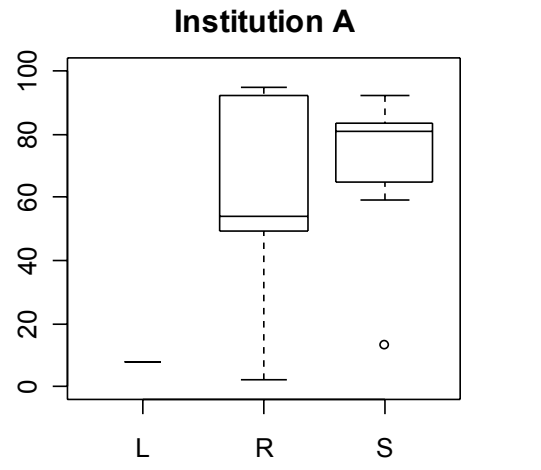


Figure 5.4.4 Box-plot of map-drawing strategies for New Zealand institution A

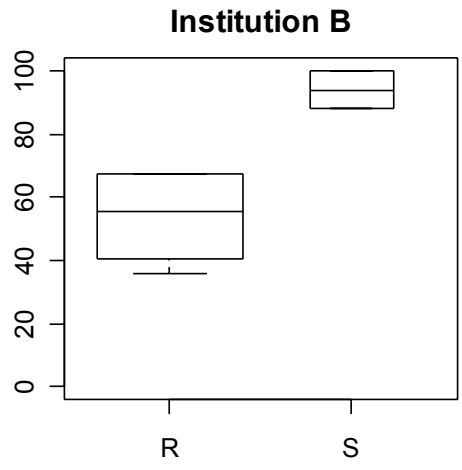


Figure 5.4.5 Box-plot of map-drawing strategies for New Zealand institution B

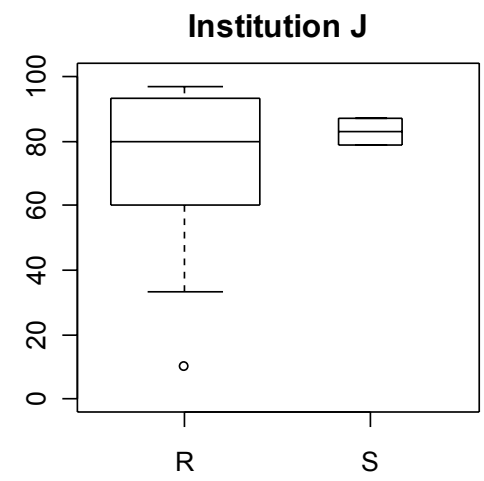


Figure 5.4.6 Box-plot of map-drawing strategies for New Zealand institution J

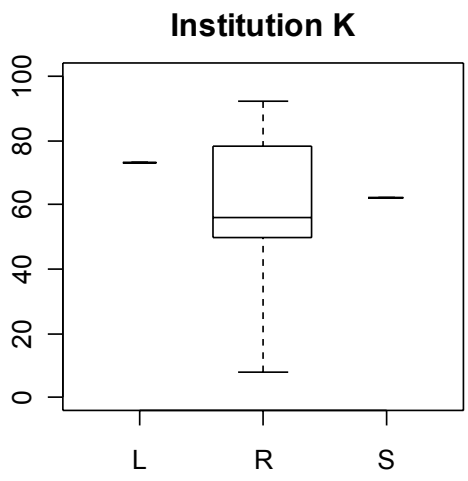


Figure 5.4.7 Box-plot of map-drawing strategies for New Zealand institution K

Box-plots for the Australian institutions, Figures 5.4.8 to 5.4.13, show a trend consistent to that found in the New Zealand institutions, but not as strong.

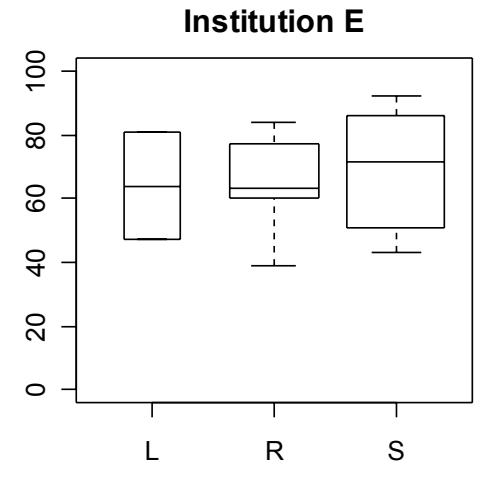


Figure 5.4.8 Box-plot of map-drawing strategies for Australian institution E

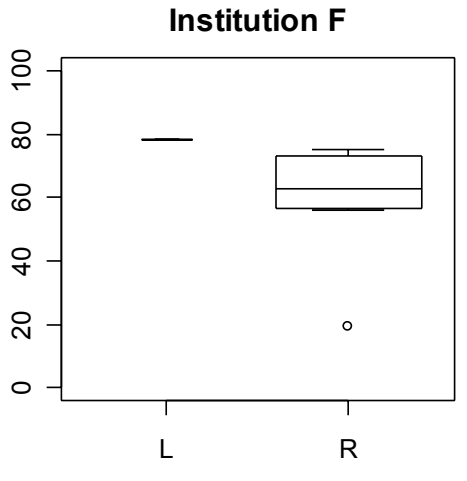


Figure 5.4.9 Box-plot of map-drawing strategies for Australian institution F

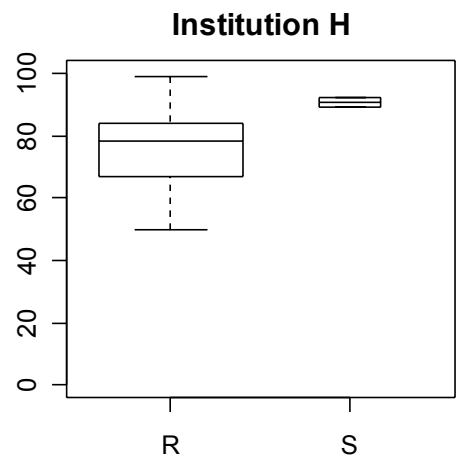


Figure 5.4.10 Box-plot of map-drawing strategies for Australian institution H

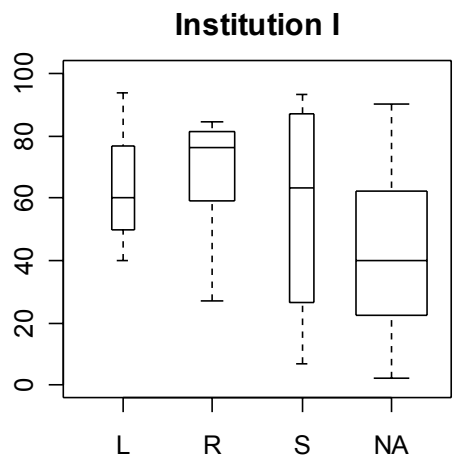


Figure 5.4.11 Box-plot of map-drawing strategies for Australian institution I

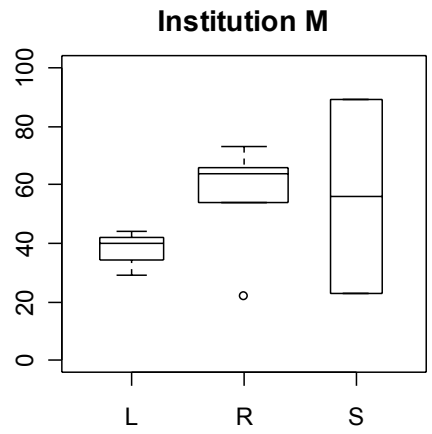


Figure 5.4.12 Box-plot of map-drawing strategies for Australian institution M

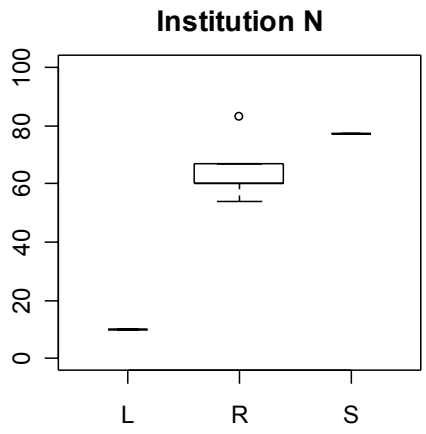


Figure 5.4.13 Box-plot of map-drawing strategies for Australian institution N

5.4.2 Analyses of variances

Analyses of variance using the software package SuperANOVA 1.11 indicate some statistically significant differences in the map drawing strategies for some variables. It should be noted however that there are uneven cell-sizes for the comparison groups (see Table 5.4.1). For this reason the following ANOVA results should be interpreted cautiously.

Two-way analysis of variance of marks based on mapstyle and country

Table 5.4.2 shows the results on a two-way analysis of variance of marks based on mapstyle and country, indicating a significant interaction (p=.02) between the two variables.

Type III Sums of Squares

Source	df	Sum of Squares	Mean Square	F-Value	P-Value
Country	2	119.340	59.670	.115	.8911
MapStyle	2	731.465	365.733	.708	.4948
Country * MapStyle	4	6205.512	1551.378	3.002	.0210
Residual	125	64608.441	516.868		

Dependent: Mark

Table 5.4.2 Results of ANOVA of marks for the variables mapstyle and country

Means for each of the groups, together with the cell-sizes for the groups are shown in Table 5.4.3. The variation in cell-sizes should be noted by the reader. The n for the analyses reported in tables 5.4.3 to 5.4.6 is 121. Missing maps and maps which include interior spaces (see section 5.3.2, above) have been excluded.

Means Table
Effect: Country * MapStyle
Dependent: Mark

	Count	Mean	Std. Dev.	Std. Error
NZ, S	12	74.917	22.781	6.576
NZ, R	31	61.798	26.463	4.753
NZ, L	2	40.500	45.962	32.500
A, S	15	67.480	27.310	7.051
A, R	48	66.401	17.123	2.471
A, L	10	52.250	25.770	8.149
UK, S	8	41.875	20.553	7.266
UK, R	5	65.800	18.727	8.375
UK, L	3	72.333	24.705	14.263

Table 5.4.3 Means, counts, standard deviations and standard errors for each cell in the ANOVA based on mapstyle and country

A plot of the interaction between the variables mapstyle and country is shown in Figure 5.4.14. This plot indicates that the trend for students in the United Kingdom institution to gain higher marks when adopting a landmark map-drawing strategy is the opposite in New Zealand and the Australia. Students from Australia and New Zealand who draw survey-based maps are more strongly linked to higher final marks. This result is consistent with the results in the previous section on descriptive statistics, and while they indicate statistical significance, the uneven cell-sizes suggest that these results should be interpreted as an interesting trend worthy of further investigation.

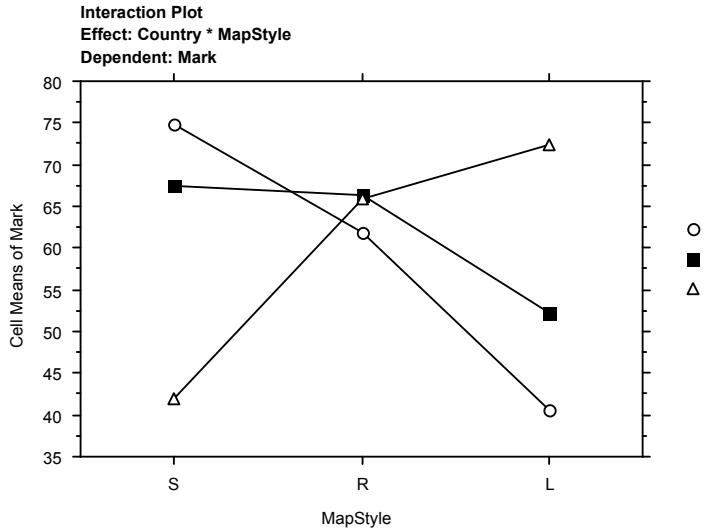


Figure 5.4.14 Plot of interaction between students' map-drawing style and country of the institution in which the students is studying.

Analyses of variance of mapstyle and student descriptions of decision points in the map-drawing exercise

Analyses of variance were undertaken to detect differences between the map-drawing styles students adopted (landmark, route, survey) and the descriptions they used for decision points in the map-drawing exercise. Student descriptions of decision points were categorised as being based on *environmental factors* (what a feature looked like e.g. large tree, big red building, etc), as *functional descriptors* (in terms of the function of a building e.g. a car-park, a coffee shop rather than the name of a feature), as a *label* (giving a name of a building, street, etc), and *other* types of description. These categorisations were then calculated as a percentage of the total number of decision points offered by a student. Tables 5.4.3 and 5.4.4 show the results of analysis of variance based on mapstyle and students' use of environmental and functional descriptors, respectively. These ANOVAs indicate statistically significant differences between mapstyle and environmental descriptions (p=.04) and between mapstyle and functional descriptors (p=.02). No significant differences were found for other decision-point descriptors (label and other).

Type III Sums of Squares

Source	df	Sum of Squares	Mean Square	F-Value	P-Value
MapStyle	2	.599	.300	3.213	.0438
Residual	118	11.001	.093		

Dependent: envp

Table 5.4.3 Results of ANOVA for the variables MapStyle and percentage of environmental descriptors of decision points

Type III Sums of Squares

Source	df	Sum of Squares	Mean Square	F-Value	P-Value
MapStyle	2	.500	.250	3.873	.0235
Residual	118	7.615	.065		

Dependent: funp

Table 5.4.4 Results of ANOVA for the variables MapStyle and percentage of student functional descriptors of decision points

Means for each of the groups, together with the cell-sizes for the groups are shown in Tables 5.4.5 and Table 5.4.6 . Variations in cell-sizes should be noted by the reader.

Means Table
Effect: MapStyle
Dependent: envp

	Count	Mean	Std. Dev.	Std. Error
S	35	.609	.286	.048
R	72	.480	.321	.038
L	14	.652	.265	.071

Table 5.4.5 Means, counts, standard deviations and standard errors for each cell in the ANOVA based on mapstyle and percentage of student environmental descriptors of decision points

Means Table
Effect: MapStyle
Dependent: funp

	Count	Mean	Std. Dev.	Std. Error
S	35	.195	.208	.035
R	72	.310	.278	.033
L	14	.149	.227	.061

Table 5.4.6 Means, counts, standard deviations and standard errors for each cell in the ANOVA based on mapstyle and percentage of student functional descriptors of decision points

Figures 5.4.15 and 5.4.16 show respectively the plots for the mean percentage of environmental and functional descriptors used by students organised according to map-drawing style adopted. In comparing the two plots, it is interesting to note that the trends are reversed.

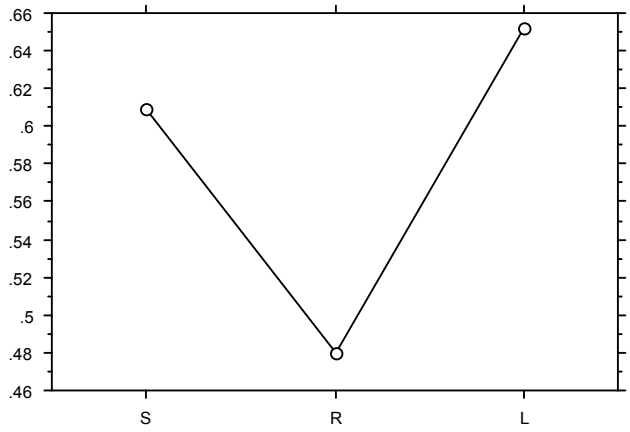


Figure 5.4.15 Plot of the percentage of environmental descriptors used by students organised according to map-drawing style.

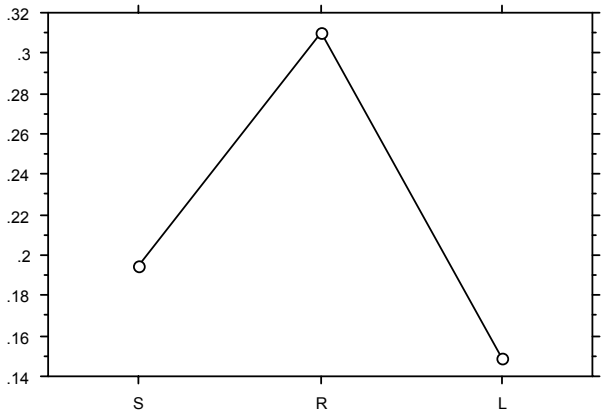


Figure 5.4.16 Plot of the percentage of functional descriptors used by students organised according to map-drawing style.

5.5 Discussion

There is a general trend for students who drew a survey map to gain higher marks than those who drew a route map, who in turn do better than those who drew landmark maps. We refer to this as the "L-R-S" trend. This trend mirrors the hierarchy of spatial knowledge described above (Werner et al., 1997); (Poucet, 1993). These parallels suggest that our participants do have preferred navigational strategies, and that these strategies are relevant to success in an introductory programming course.

How these are relevant is a matter for conjecture at this stage, but the literature reviewed briefly above suggests that different navigational strategies may effect the way in which programmers are able to navigate programming code and to form a conceptualisation of the major features of this code in so doing.

The grouped results for UK, NZ and Aus (see Figure 5.4.14) show an interaction effect between country and mapping style. The differences in cell sizes means that the interpretation of this result must be treated with caution. If we accept that there is an effect, we cannot tell whether this is due to general educational effects, the nature of the courses in each country or the components used to determine the final mark for each course. The reasons for differences in the results for these countries requires further investigation.

A confounding factor, which may have contributed to the lack of significance in some of the results reported, was the differences in the physical environments for which the maps were drawn. We speculate that the difficulty with these maps is due to the "maze-like" nature of the route sketched, principally through building corridors. Navigation in mazes is known to involve different navigational approaches (Werner et al., 1997) (Passini, 1984).

Although a significant relationship was found between descriptions of decision points given by participants and their map style (see Tables 5.4.3 and 5.4.4), these decision point descriptors were not related to final marks. This suggests that students' ability to represent navigational knowledge visually may not be related to their ability to describe it. There are some potential implications here for the use of written examinations over practical examinations, in capturing students' comprehension and creation of program structures.

In conclusion, this study identified trends, relating map sketching styles to performance in introductory programming courses. We speculate that a student's ability to move through abstract information spaces, recognising the major features or landmarks in those spaces is related to success in program comprehension and creation. These are the same skills used in physical navigation tasks.

We feel that further investigation in this area is warranted.

6 Articulating a search strategy

6.1 Focal question

One of the main questions that shaped the design of the study was: is there a correlation between students' ability to *articulate their strategies* for commonplace search tasks and their performance in a first programming course?

6.2 Background and motivation

Anecdotal evidence suggested that programmers are better at describing search processes than non-programmers. If true, this would imply that the metacognitive ability to describe strategy might be relevant. In order to explore this possibility the phone book task was chosen as a representative example of a commonplace search activity. This task is drawn from classroom practice, stemming in turn from a tradition in computer education of using commonplace examples to convey programming concepts and make them relevant to students (Curzon, 2002).

A phone book search task has previously been used in a study of naïve, novice / beginning, and experienced programmers (Onorato & Schvaneveldt, 1986). This study required participants to write instructions for looking up a specified name in a phone directory. It was found, among other things, that experienced programmers were more likely than novices to incorporate programming constructs in their descriptions, and that they were more likely to use terms referring to the phone book as opposed to its contents. To the extent that the task of writing instructions can be compared with that of articulating one's own actions in speech, this study produced one further finding that is pertinent to out work. With regard to novice programmers (equivalent to our participants), the study found no significant difference between these students before and after their first semester of programming. This is useful because it suggests that we have not compromised our own data by collecting it at somewhat different stages of the semester.

In designing this task we sought to explore two aspects of the ability to articulate strategies, accuracy and richness. Accuracy was interpreted as how well a given description matched the experimenter's observation of how the participant undertook the task. In the event, the accuracy ratings collected during the task sessions were later perceived by the experimenters as being unreliable. Hence the measures explored in this report are all attempts to capture the richness of the articulation. These include: a count of the number of alternative search strategies elicited from each student at the end of the task; an assessment of the richness of the student's articulations made by individual experimenters during the task; and a similar but more focused assessment of all transcribed articulations made collectively by three of the experimenters.

The collective assessment of the transcripts attempted to capture a measure of richness specifically based on the degree to which the participant articulated their search strategy. This focus was determined after much discussion of the transcripts, and an initial attempt at analysis based on distinctions similar to those drawn by (Onorato & Schvaneveldt, 1986) who compared the use of terms relating to the contents of the phone book and terms relating to the book itself. It was apparent from the transcripts that, as expected, almost all participants described factors relating to surface content such as specific names and letters, and many referred to aspects of the book such as page numbers, columns and so on. While the interview protocol did not explicitly ask for a strategy or algorithm, we found however that many participants had also described (to varying extents) factors relating to the strategy

of their search, for example noting intentions, giving reasons for particular actions, or describing conditional or repeated aspects of the process.

We hypothesised that those participants who naturally articulated their strategy were demonstrating the ability to situate their actions in a preconceived algorithm or plan, and that this ability might constitute a useful predictor of success in learning to program. Combined with the results of (Onorato & Schvaneveldt, 1986) an interesting developmental picture could be developed. Onorato and Schvaneveldt found that experienced programmers are more likely than novices to incorporate programming terms in their descriptions of a phone book search:

> "For example, the experienced programmers were more likely to make use of loops and to consider more alternatives en route to their solution goal. Hence, they have an edge in both conciseness and preciseness. Unfortunately, however, this experiment could not determine if this edge was a direct result of programming experience or, if in contrast, programmers were equipped with such abilities to begin with." Onorato, p. 369.

Our analysis of novices during the very early stages of their first programming course directly addresses the question posed by Onorato and Schvaneveldt: do those who go on to become successful programmers have pre-existing abilities or tendencies to describe processes in a strategic / algorithmic manner?

6.3 Analysis

Articulations

Each participant completed two searches of the phone book (see the protocol in Appendix C). They were asked to describe the first search after completing it, we refer to this as Articulation 1. They were asked to describe the second search while working on it, we refer to this as Articulation 2.

Number of alternative strategies

After completing the search articulations, participants were asked to describe any alternative strategies they could think of for conducting such searches. From the interview transcripts we extracted the number of distinct alternatives proposed by each participant. We did not count methods that used external resources (such as using the internet or asking a friend to help), or the suggestion to "look in the index if there is one". Most other suggestions were accepted as valid, and distinct strategies were counted individually even if they fall within the same broad approach (for example, a linear search forward from the start of the directory and a linear search backward from the end). We examined the relationship between the number of alternatives articulated and the students' results in their courses.

Interviewer's rating

During each experiment session the interviewer noted their perceptions of the participant's search articulations as Poor, Average or Good. Subsequent discussion elicited the potential for ambiguity in what had actually being coded. Were researchers coding the quality of expression, the accuracy of the articulation, the length of the utterance, the amount of detail in the articulation, or some combination of these and perhaps other measures? While conscious of the variance that this ambiguity might impose on our results, we examined the relationship between the ratings and the students' results in their courses.

Collective rating

In discussing the potential difficulties of the interviewer's rating measure it was decided that a collective analysis should be attempted by a rating team of three experimenters. As discussed above the focus of this analysis was the distinction between the articulation of a *search*, which can be purely descriptive (based on the surface features of the book's contents such as such as specific names and letters), and the articulation of a search *strategy*, which gives some impression of applying a particular plan or an algorithm.

Transcripts were assessed by the rating team using the same agreed criteria. Raters looked for evidence of strategic elements such as:

- statements of reasons (**why** a task is performed in addition to **what** it does);
- statements of intentions;
- descriptions of tests or conditionals assessed during the process;
- descriptions or hints of repeated tasks;
- particularly detailed descriptions of aspects of the process.

These general criteria were made more specific by looking for particular linguistic markers such as:

- those relating to reasons:
    - *because, so, assuming*
- those relating to intentions:
    - *have to, need to, want to, trying to, look for, find out, make sure*
- those relating to general search:
    - *find, found, refined, narrowed, checking, until, repeat, test, backtrack, again, continued, before, after, forward, back, too far, closer*

Transcripts from 104 interviews from 9 institutions were examined. As each interview contains articulations for two searches (as described above) there were a total of 208 articulations. Using the criteria outlined above, the rating team individually scored each articulation as Poor, Average, or Good. Each experimenter scored articulations independently in batches of about 10. At the end of each batch scores were discussed and a collective score agreed, thus each articulation received an agreed collective score. A typical example of a Poor, an Average and a Good articulation is shown in Table 6.1. Agreement between the three experimenter's individual scores was high, and consensus was reached in all cases. Once again, we examined the relationship between the ratings and the students' results in their courses.

POOR: Open the book. Like, M. C, D, E, M. I want M-C. A, B, C. M-A, M-C, looking over the page. M-B. Go back. M-E, M-A, A, B, C. D. M, A, C. M-C. M-C-L-A. Ok, M-C-L-A. And M&M. There.
AVERAGE: I'm going to go to [city name] again as it is more likely it is going to be in there. I'm going through the alphabet backwards as I started with R towards M. I've got to M but I'm still too far through the alphabet. I've found Mc, found McL so am going to go back further, I've found McD, McDo. I'm still too far, I'm going back further, I found McDonald, I still at McDonald M, so go back further, McDonald G, McDonald [first name].
GOOD: Now, I know from previous experience that telephone directories are difficult with Mc names, because sometimes they're Mac, sometimes they're Mc, and often they're listed in not alphabetically strictly speaking. So, I'll start off my searching finding Ms and I'll just go just take it at face value, so then I'll look to see Mc, L. Looking at the name on the top of the page, trying to find Mc, Mc will be before that, seems to be jumping from Ma to Me quite quickly, which leads me to believe that it is probably not listed under Mc, no it doesn't look as if it is, so maybe it's listed under Mac so I'll just flip back to Mac ... yes, start looking for the 3rd letter, so I'm looking for Mac, and it is listing the Mc's there to, so I know I'm looking in the right place, so I'm looking for Mc-L or Mac-L, ..., Ls ... with a La ... there's a [first name], ok so that's not the correct spelling of McLaughlin ... and there it is.

Table 6.1 A typical example for each rating (Articulation 2)

6.4 Results

6.4.1 Overview

We found a number of significant and marginally significant effects (but nothing that would constitute the philosopher's stone of success prediction!). Note that all results described in this section consider only students who completed the programming course (i.e. students with a final mark of 0 are excluded from the analysis). Including the incomplete students strengthens the significance of all the results somewhat, but a mark of 0 can not be seen as a statistically valid measure of programming ability.

6.4.2 Number of alternative strategies

This section reports on the number of alternatives articulated (where participants had a mark greater than zero): a population of 111. The analysis based on number of alternative strategies articulated shows no significant results. While the trend is in the expected direction, i.e. students who articulated more strategies have higher mean marks, neither ANOVA ($F_{3,107}$ = 2.030; p = .114) nor correlation (r = .176; p = .065) are significant.

Descriptive Statistics

Group	Mean	Std Dev.	Std Err	N
0	55.048	25.146	4.445	32
1	67.968	24.785	4.131	36
2	66.233	22.389	4.021	31
3	68.175	21.653	6.251	12

Analysis of Variance for Y=Mark

Source	Type III SS	Df	Mean Sq.	F	Prob.
Model	3489.213	3	1163.071	2.030	0.114
Error	61297.358	107	572.873		
Total	64786.571	110			

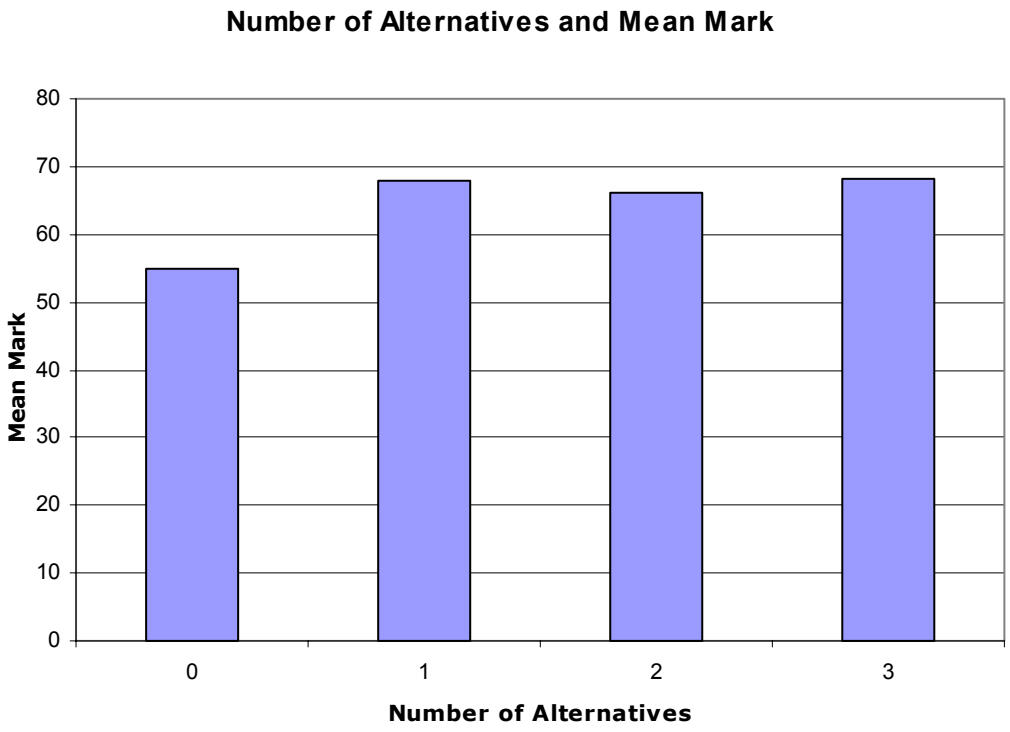


Figure 6.1 Main effect of grouping by number of alternatives articulated on mean mark

If we consider a coarser grain of analysis participants can be divided into just two groups, those who gave alternatives, and those who did not. At this level there is a clear effect. Participants who gave 0 alternatives had lower marks, on average, than those who were able to generate alternative strategies ($F_{1, 109} = 6.091$; $p = .015$).

Descriptive Statistics

Group	Mean	Std Dev.	Std Err	N
0	55.048	25.146	4.445	32
1	67.318	23.137	2.603	79

Analysis of Variance for Y=Mark

Source	Type III SS	Df	Mean Sq.	F	Prob.
Model	3428.691	1	3428.691	6.091	0.015
Error	61357.880	109	562.916		
Total	64786.571	110			

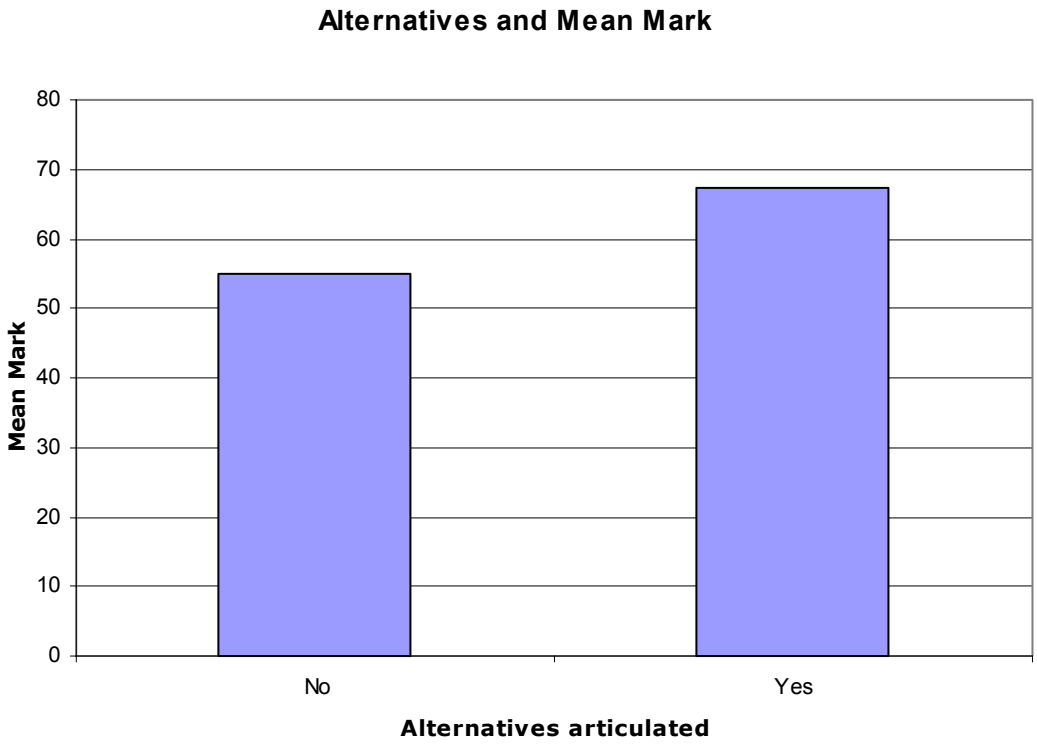


Figure 6.2 Main effect of grouping by any alternatives articulated on mean mark

6.4.3 Interviewer's rating

This section reports on the ratings interviewers gave participants' articulations (where participants had a mark greater than zero): a population of 134. Here we consider the interviewers' ratings for Articulation 1 and Articulation 2.

Main effect of Articulation 1 group on mark

For the first articulation the trend is in the expected direction, participants rated Poor and Average in general have lower marks than do those rated Good. However, the effect is only marginal (F = 2.185; p=.117).

Descriptive Statistics

Group	Mean	Std Dev.	Std Err	N
Poor	59.896	21.428	2.972	52
Average	61.570	25.259	3.994	40
Good	69.555	23.460	3.620	42

Analysis of Variance for Y=Mark

Source	Type III SS	Df	Mean Sq.	F	Prob.
Model	2363.555	2	1181.777	2.185	0.117
Error	70864.738	131	540.952		
Total	73228.292	133			

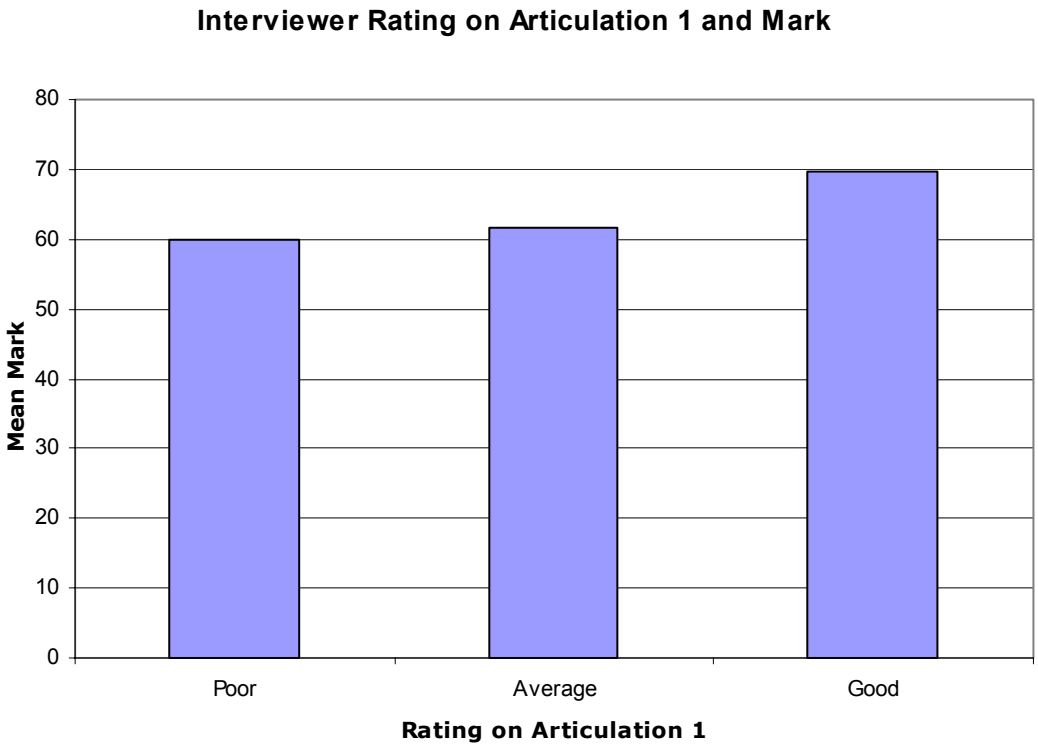


Figure 6.3 Main effect of rating on Articulation 1 on mean mark

Main effect of Articulation 2 group on mark

The same analysis for the second articulation is shown below.

Descriptive Statistics

Group	Mean	Std Dev.	Std Err	N
Poor	58.453	20.837	3.337	39
Average	63.083	23.143	2.963	61
Good	69.735	25.977	4.455	34

Analysis of Variance for Y=Mark

Source	Type III SS	Df	Mean Sq.	F	Prob.
Model	2325.303	2	1162.652	2.148	0.121
Error	70902.989	131	541.244		
Total	73228.292	133			

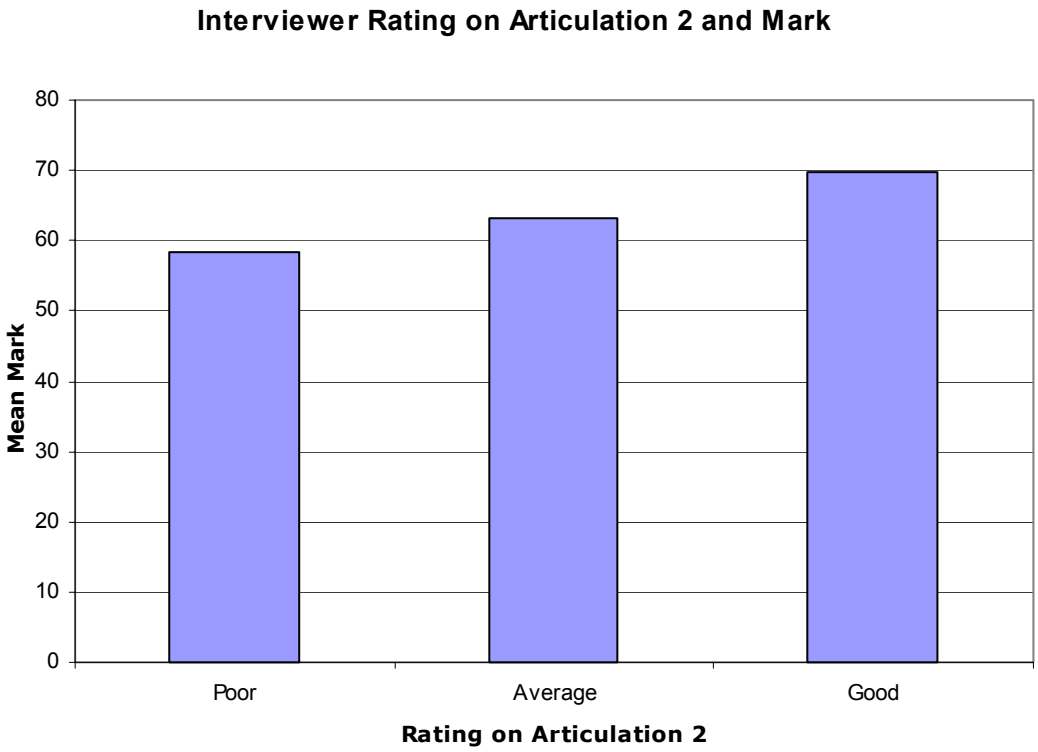


Figure 6.4 Main effect of rating on Articulation 2 on mean mark

Again, subjects whose articulation was rated Good had higher mean marks than those rated Poor or Average, but the effect is not significant. However, the pattern when grouping by Articulation 1 (articulate after performing the search) is noticeably different from the pattern when grouping by Articulation 2 (articulate while performing the search). In the former analysis, Poor and Average had approximately equal mean marks; in the later there appears to be a linear trend from Poor, through Average, to Good. There is also a large change in the distribution of the ratings themselves, as shown in the table below:

Group	N for Articulation 1	N for Articulation 2
Poor	52	39
Average	40	61
Good	42	34

Table 6.2 Change in the distribution of ratings

Many more participants were rated as Poor for Articulation 1 than for Articulation 2. Many fewer participants were rated as Average for Articulation 1 than for Articulation 2. Presumably participants moving into the Average group for Articulation 2 raised the mean mark of that group. For example, 14 participants from the high rating Good group of Articulation 1 moved into the Average group for Articulation 2. The complete summary of participants in different rating combinations is shown below:

Descriptive Statistics

Group	Mean	Std Dev.	N
Poor Poor	58.978	17.703	25
Poor Average	57.870	25.712	22
Poor Good	70.750	16.820	4
Average Poor	52.689	28.676	9
Average Average	64.624	23.767	25
Average Good	62.167	27.578	6
Good Poor	61.750	21.577	4
Good Average	68.522	16.697	14
Good Good	71.458	27.345	24

Table 6.3 Summary of participants in different rating combinations

The group title indicates the subject's scores for Articulations 1 and 2. For example, the "Good Average" group was rated Good on Articulation 1 and Average on Articulation 2.

From Table 6.3, we can see that the majority of participants (74 of 133) received the same rating on both tasks. A smaller group of participants (N = 51) moved one level between tasks, and these were divided approximately evenly between students who performed better on Articulation 1 and those who performed better on Articulation 2 (N = 23 and 28 respectively). Very few participants moved the two levels from Poor to Good (N = 4) or from Good to Poor (N = 4). Thus we see that performance is relatively consistent on the two articulation tasks. While there is no main effect of the above grouping ($F = .995$; $p = .443$), there is a perplexing anomaly in the performance on the 4 participants who rated Poor when articulating after the task (Articulation 1), and Good when articulating during the task (Articulation 2). These four participants had a mean mark of 70.75, well above the overall mean for participants rated Poor on Articulation 1 (59.89). While no conclusions can be

drawn from such a small sample, it would be interesting to explore further the reasons that strong students might perform poorly on Articulation 1.

6.4.4 Collective rating

This section reports the collective ratings for the articulations (where participants had a mark greater than zero): a population of 96. Here we consider the collective ratings of search strategy for Articulation 1 and Articulation 2.

Main effect of Articulation 1 group on mark

The observed trend in this analysis is identical to that in the previous ones: participants who are better at articulating their strategies in the phone book task, tend to earn higher marks. Although the main effect is not statistically significant, the extremely low N in the Good group severely limits the power of the analysis.

Descriptive Statistics

Group	Mean	Std Dev.	Std Err	N
Poor	58.853	24.505	3.613	46
Average	64.337	23.420	3.453	46
Good	69.000	18.921	9.460	4

Analysis of Variance for Y=Mark

Source	Type III SS	Df	Mean Sq.	F	Prob.
Model	901.822	2	450.911	0.795	0.455
Error	52778.099	93	567.506		
Total	53679.921	95			

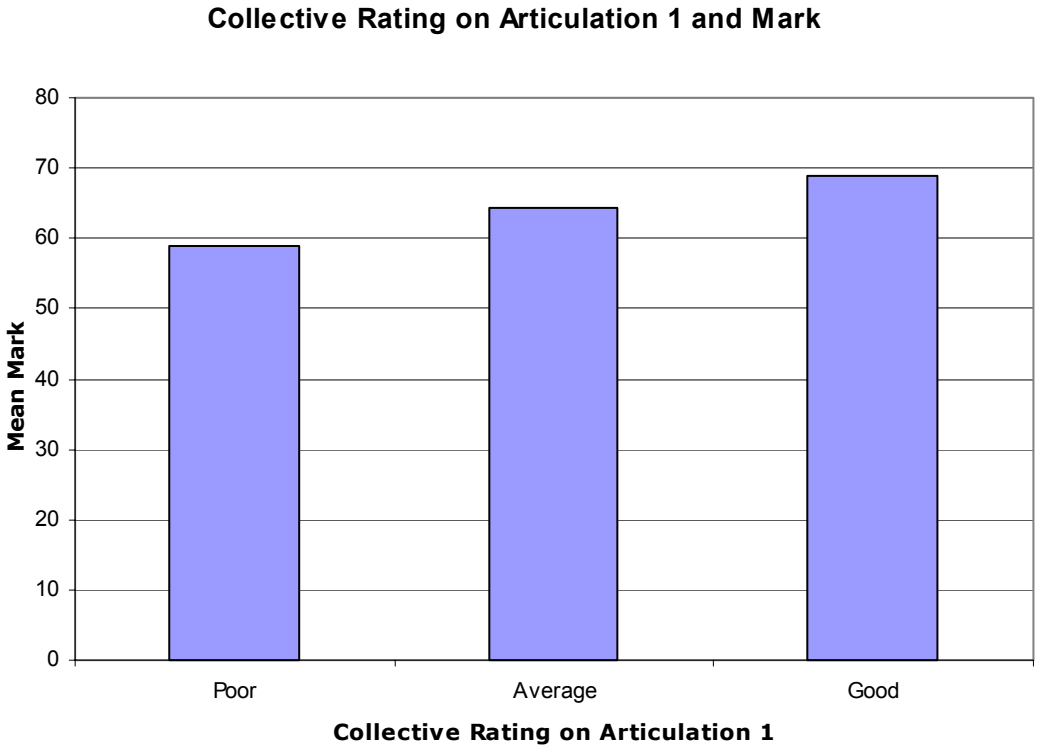


Figure 6.5 Main effect of collective rating on Articulation 1 on mean mark

Main effect of Articulation 2 group on mark

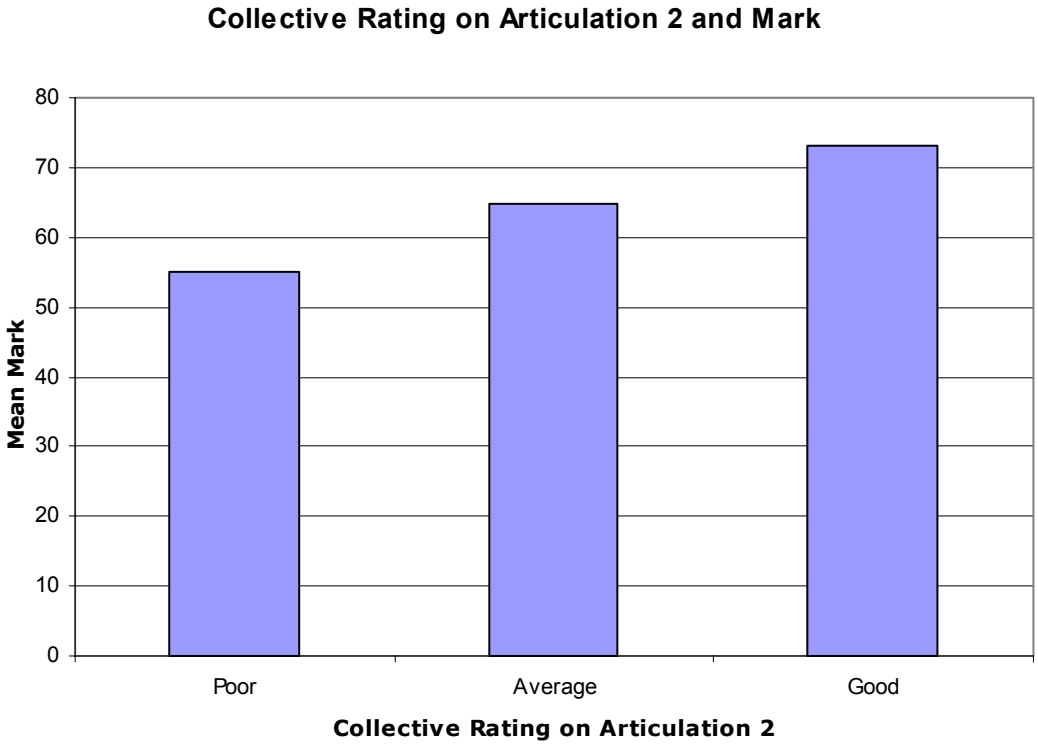
Once again we see the trend of increasing mean mark with increasing articulation performance, but in this case the effect is marginally significant.

Descriptive Statistics

Group	Mean	Std Dev.	Std Err	N
Poor	55.174	26.598	4.433	36
Average	64.817	21.993	3.050	52
Good	73.250	12.815	4.531	8

Analysis of Variance for Y=Mark

Source	Type III SS	Df	Mean Sq.	F	Prob.
Model	3101.929	2	1550.965	2.852	0.063
Error	50577.992	93	543.849		



Total	53679.921	95			

Figure 6.6 Main effect of collective rating on Articulation 1 on mean mark

These results also show a repeat of the migration toward "average" from Articulation 1 to Articulation 2. The appearance of this pattern in both rating metrics (interviewer's rating and collective rating) supports the validity and reliability of these scores as measures of the quality of the subject's articulation. The phenomenon is also of interest in its own right. It cannot be a simple practice effect, nor can it indicate that the second articulation task is simply easier, as approximately equal numbers of participants moved up to Average from Poor as moved down to Average from Good (see discussion above). Rather, performance on

the Articulation 2 task seems subject to less extreme variation than performance on the Articulation 1 task. If we wish to use tasks like these to explore the determinants of programming ability, we must be careful to have a clear understanding of their inherent biases.

6.5 Discussion

Considering the richness of the phone book search task articulations, the observed trends are all in the expected direction. All measures show that increasing richness is associated with increasing mean marks. The measures are not all statistically significant, although most do become so if students with a final mark of 0 are included in the analyses (the 0-mark students come exclusively or predominantly from the "weaker" groups). Although individual measures are weak, taken together they are forming a reliable picture.

In summary, this study provides some initial evidence that the question raised by Onorato (see Section 6.2) can be answered affirmatively: students who carry on to be successful programmers have pre-existing strengths in a strategic / algorithmic style of articulation. Some issues remain to be explored in future work however, such as the variation in performance over the different measures used in this study, the patterns of change in ratings for the same participant over different tasks, and particularly the variations in ratings between describing a search in retrospect (Articulation 1) or while actually performing the search (Articulation 2). It would also be useful to explore the extent to which the richness of articulation is independent of, or perhaps simply a reflection of, general or verbal IQ measures.

7 A qualitative analysis of exit interviews

7.1 Focal question

What qualities or skills do entry-level undergraduate students express as important to learn programming well?

7.2 Background and motivation

Studies targeted at finding the qualities, knowledge, skills, or abilities students perceive as important to learning programming well were not found. Such information may be implied from other studies such as those mentioned earlier about the factors influencing performance. However, no conclusive evidence was available.

A study of industry perceptions of the knowledge, skills, and abilities (KSAs) needed by entry level computer programmers (Bailey & Stefaniak, 2001) reports results of a survey. The survey questions were developed from focus groups of a few individuals from five companies. The list of 85 KSAs needed were divided into 53 technology skills and 20 soft skills and 12 business concepts.

There are non-discipline specific studies about students preparation for and perception of learning at university such as the work by Biggs. While these studies give a general picture of students it does not provide the insight specifically student perceptions of what it takes to learn programming well.

7.3 Analysis

Of the 177 participants, 74 provided a response to the question: "What qualities or skills do you think are important to learn programming well, to "get it"? The 74 were from 8 institutions: the 3 institutions omitted from this analysis either did not ask the question or the transcript was not available.

The transcripts of the interviews used for analysis varied from a verbatim record of the participant's response to a summary of a few words, provided by the transcriber, usually the same person who conducted the interview.

The responses, in whatever form they appeared in the transcripts, were extracted by one researcher. This first pass of the data identified and extracted the responses to all of the short open ended interview questions without further analysis, i.e. if the response was a section in the transcripts that followed the phonebook searching it was coded as the exit data.

The second pass was over the exit data only. It was an analysis of the four questions in the transcripts:

a) What do you think we were trying to find out?
b) How do you think the sketch–map task might relate to programming?
c) How do you think the phone book task might relate to programming?
d) What qualities or skills do you think are important to learn programming well, to "get it"?

In the first pass, the data from institution A only was used. The analysis looked for concepts or terms considered of interest relevant to the four questions as constituted in the data. The concepts or terms were not decided prior to analysis. 44 words or phrases were found.

The third pass reviewed the 44 words or phrases to only those that were of interest and part of the response to the qualities and skills question. The list of terms was shortened to 25 terms.

The fourth pass considered transcripts from institutions B, E, F, K, M, N and P. New terms were added to the list only when a suitable term was not in the list already. A conscious effort was made to limit the list. At the end of this pass there were 38 words or phrases in the list.

The fifth pass examined the quotes matched to the 38 words or phrases, from the least frequent to most frequent. The pass checked for overlapping or redundant words or phrases. At the end of this pass there were 34 words or phrases identified as constituted in the responses.

The 34 words and phrases were coded when a significant and separate occurrence was found, e.g. “Logic. I think it seems to me that programming is the most logical thing.” From A15 was coded once. However, “has to be able to very analytical” and “also be able to logical, logically analyse situation and be able to draw out the step before he actually program” from B10 were coded as two passages.

The frequency of passages versus word/phrase overall, for each participant and with other word/phrases were tabulated.

7.4 Results

The 34 words/phrases constituted in the 74 responses varied from 25 to 1 in frequency and 22 to 1 in count (Table 7.1) where *frequency* is the number of times the word/phrase occurred (including duplicates) for a participant and *count* is the number of participants who used the word/phrase. The highest counts are logical (22/74), an ordered or structured way of thinking or behaving, and problem solving (21/74), this is likely to mean to reach a solution in a systematic way: the ambiguity of what problem solving means is discussed in section 7.5.

The frequency and count for each word/phrase for each participant is shown in Appendix E. The maximum count for a participant is 8 and the minimum 1.

The word/phrases can each be characterised by the utterances coded in the transcripts. The set of word/phrases and quotes are shown in Appendix F.

The occurrence of one word/phrase with another is shown in Appendix G. There is no clustering of one word/phrase with one or more others.

Word/Phrase	Frequency of Passages	Count
logical	25	22
problem solving	24	21
attention to detail	14	11
consideration of alternatives	12	10
mathematics	12	10
knowledge of programming	11	10
ability to learn	10	7
knowledge of computers	10	8
modularising	8	7
planning	8	8
memorising and recall	7	7
persistence	7	6
visualise	7	7
enthusiasm	5	5
enjoyment	4	4
smart	4	4
decision making	3	3
obsessive	3	3
patience	3	3
practice	3	3
quick	3	3
ability to follow instructions	2	2
competent	2	2
creativity	2	2
documentation	2	2
efficiency	2	2
time	2	2
competitive	1	1
dedicated	1	1
don't know	1	1
knowledge of languages	1	1
nothing in particular	1	1
organised	1	1
team player	1	1
Count	**34**	**74**

Table 7.1 Frequency and Count of Word/Phrase Occurrences in Responses (n=74)

7.5 Discussion

The qualities and skills identified most often are logical and problem solving. The next top 8 are: attention to detail, consideration of alternatives, mathematics, knowledge of programming, ability to learn, knowledge of computers, modularizing, and planning. The top 10 for the industry study (Bailey & Stefaniak, 2001) are: ability to read, understand and modify programs written by others, ability to code programs, ability to debug software, listening skills, problem-solving process (decision tree, problem identification and analysis), team work skills (long term), knowledge of structured programming fundamentals, ability to implement programs, knowledge of multiple programming languages, ability to visualize/conceptualize. Taking a liberty with the interpretation of the meaning of the participants utterances there is a fair alignment between co-occurrences in the top 10:

This study	Bailey and Stefaniak
problem solving	problem solving process
• knowledge of programming	• ability to read, understand and modify programs written by others • ability to code programs • ability to debug software, knowledge of structured programming fundamentals • ability to implement programs • knowledge of multiple programming language

Listening skills was not mentioned by participants, though being able to visualize had a count of 7 in this study.

Exactly what each participant meant by an utterance is not clear as probing questions were not used. For instance, what problem solving means to participants is ambiguous. The utterances vary from “problem solving” to “Problem solving and being able to identify what the problem is and being able to solve that, sort of finding, making sure you understand how the problem works and what you want to achieve, what your goals are and then trying to develop a method that solves that particular problem”.

There may be a bias in the interpretive analysis of the data as only one researcher performed the analysis. Though during the fifth pass the utterances about which the researcher was not sure were clarified with others. The credibility and trustworthiness of the results is reasonable. The researcher had prior experience with the coding of transcripts based on what is constituted in the data, putting aside personal experience. The word/phrases credibility could be improved by using interjudge reliability (Cope, 2002) in addition to the single researcher interpretive approach based on the word/phrases abstracted from what is constituted in the data.

The reliability of the data source is weak. The confounds being the participant’s programming and life experience and age, the institution environment, when the participants were interviewed, the integrity of the transcripts especially those that are not verbatim and the experience of the interviewer.

The results suggest that the students lack an awareness of what it takes to learn programming well given the low counts, how few of the 85 KSAs (Bailey & Stefaniak, 2001) were mentioned, and accepting that the 85 KSAs are what is required.

4. Discussion

This study has explored a number of issues that may influence success in learning to program. Experimenters at eleven participating institutions used the same protocol to gather data from students in introductory programming courses that were taught during 2004. The study was based on four different diagnostic tasks: a spatial visualisation task (a standard paper folding test); a behavioural task used to assess the ability to design and sketch a simple map; a second behavioural task used to assess the ability to articulate a search strategy; and an attitudinal task focusing on approaches to learning and studying (a standard study process questionnaire). Most participants also completed a short exit interview.

The results show trends of varying strengths, in general in accordance with expectations and predictions in the literature. A deep approach to learning is positively correlated with marks in introductory programming courses, while a surface approach is negatively correlated. Interestingly, the difference between deep and surface learners' scores becomes more prominent for higher marks. This study found only a small positive correlation between scores in the spatial visualisation (paper folding) task and programming marks. This suggests that components of "IQ" other than spatial skills may account for most of the effect of IQ on programming success (R. E. Mayer et al., 1989) However, in the map sketching task a progression of map drawing styles identified in the literature, from landmark to route to survey, has a significant effect on marks. For this effect there appears to be interaction with the institution's country. However, as there was only a single example from one country, this effect remains to be explored. In a simple search task increasing measures of richness of articulation of a search strategy are generally associated with higher marks, but none of the effects are strong. There is some variation in performance for different kinds of articulation (after performing the search or while performing it) which remains to be explored. Finally, a qualitative analysis of the exit interviews identified the qualities that students themselves regarded as important to learn programming well. As might be expected these self-reported qualities cover only part of a much wider range of attributes specified in an industry survey (Bailey & Stefaniak, 2001).

The strengths of this study include a large number of participants, and the use of diverse and generalised stimuli, making the tasks independent so that comparisons can be made across paradigms, languages, and pedagogic styles. The study combined different approaches and collected both qualitative and quantitative data, thus providing opportunities to compare the different factors. The study builds on existing work and uses some tests for which standardised data available. The main limitations of the study arise from the use of multiple experimenters, and include issues with respect to the consistency of the application of the study protocol, and consistency of coding, transcription and analysis.

It seems likely that a multi-factor model employing tasks such as those used in this study could be used as a reasonable predictor of success in introductory programming. However, this study suggests that further exploration of possible diagnostic tasks is required, as we must be careful to have a clear understanding of their inherent biases. It would also be useful to explore the extent to such tasks relate to general measures of IQ, or standard components of IQ such as verbal and spatial factors.

Acknowledgments

This study was supported by an ACM Special Interest Group in Computer Science Education (SIGCSE) Special Projects Grant, a grant from Computing Research and Education Association of Australasia (CORE), and a grant from the Computer Science Department of The University of Otago. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the funding organisations.

Thanks to Caroline Wills for assistance with workshop organisation and transcription; to Diane Hagan, CSSE, Monash University for her insight and assistance in conducting the study with Monash Students whilst a BRACE participant was on a Visiting Scholar Grant; to Warren Hill of Charles Darwin University in facilitating access to participants; to Charles Thevathayan of RMIT University and Dongmo Zhang of University of Western Sydney for facilitating access to participants, to Susan Snowdon for her contribution to the literature search; to Anthony Robins and Raymond Lister for exemplary workshop organization and support; to Box Catering Inc for phenomenal food.

References

Bailey, J. L., & Stefaniak, G. (2001). *Industry perceptions of the knowledge, skills and abilities needed by computer programmers.* Paper presented at the ACM SIGCPR conference on computer personnel research, San Diego, California.

Barker, R., & Unger, E. (1983). A predictor for success in an introductory programming class based upon abstract reasoning development. *ACM SIGCSE Bulletin, 15*(1), 154-158.

Benyon, D., & Wilmes, B. (2003). *The application of urban design principles to navigation of information spaces.* Paper presented at the People and Computers XVII - Designing for Society, Bath, UK.

Biggerstaff, T. J., Mitbander, B. G., & Webster, D. (1993). *The concept assignment problem in program understanding.* Paper presented at the 15th International conference on Software Engineering, Baltimore, Maryland.

Biggs, J. (1987). *Student approaches to learning and studying*. Melbourne: Australian Council for Educational Research.

Biggs, J., Kember, D., & Leung, D. Y. P. (2001). The revised two-factor study process questionnaire: R-SPQ-2F. *British Journal of Educational Psychology, 71*, 133-149.

Chowdhury, A., Van Nelson, C., Fuelling, C. P., & Mccormick, R. L. (1987). *Predicting success of a beginning computer course using logistic regression (Abstract only).* Paper presented at the 15th Annual Conference on Computer Science, St Louis, MO, USA.

Cope, C. (2002). *Using the analytical framework of a structure of awareness to establish validity and reliability in phenomenographic research.* Paper presented at the Current Issues in Phenomenography Symposium, Canberra, Australia.

Cox, A., & Fisher, M. (2004). *Navigating codespace: A new direction for spatial cognition research.* Paper presented at the International Society for Human Ethology.

Cross, E. (1970). *The behavioural styles of programmers.* Paper presented at the 85th Annual SIGCPR Conference.

Curzon, P. (2002). Games & Puzzles. *Interfaces, 42*, 14-15.

Ekstrom, R. B., French, J. W., Harman, H. H., & Dermen, D. (1976). *Kit of factor-referenced cognitive tests*. Princeton, New Jersey: Educational Testing Services.

Evans, G., & Simkin, M. (1989). What best predicts computer proficiency? *Communications of the ACM, 11*(November), 1322-1327.

Fincher, S., Petre, M., Tenenberg, J., Blaha, K., Bouvier, D., Chen, T., Chinn, D., Cooper, S., Eckerdal, A., Johnson, H., Mccartney, R., Monge, A., Moström, J., Powers, P., Ratcliffe, M., Robins, A., Sanders, D., Schwartzman, L., Simon, B., Stoker, C., Tew, A., & Vandegrift, T. (2004). *A multi-national, multi-institutional study of student-generated software designs.* Paper presented at the 4th Annual Finnish / Baltic Sea Conference on Computer Science Education.

Green, T. R. G. (1997). *Cognitive Approaches to Software Comprehension:*

Results, Gaps and Limitations. Paper presented at the Experimental Psychology in Software Comprehension Studies, Limerick, Ireland.

Hestenes, D., Wells, M., & Swackahmer, G. (1992). Force Concept Inventory. *Physics Teacher, 30*, 141-158.

Leeper, R., & Sliver, J. (1982). Predicting success in a first programming course. *ACM SIGCSE Bulletin, 14*(1), 147-150.

Lister, R., Adams, E. S., Fitzgerald, S., Fone, W., Hamer, J., Lindholm, M., Mccartney, R., Moström, J., Sanders, K., Seppälä, O., Simon, B., & Thomas, L. (2004). A Multi-National Study of Reading and Tracing Skills in Novice Programmers. *ACM SIGCSE Bulletin, 36*(4), 119-150.

Lynch, K. (1960). *The image of the city*. Cambridge Mass.: MIT.

Mayer, D., & Stalnacker, A. (1968). *Selection and Evaluation of Computer Personnel - the Research History of SIG/CPR.* Paper presented at the 23rd ACM National Conference.

Mayer, R. E., Dyck, J. L., & Vilberg, W. (1989). Learning to program and learning to think: what's the connection? In E. Soloway & J. Sphorer, C (Eds.), *Studying the Novice Programmer*. Hillsdale, New Jersey: Lawrence Elbaum.

Mccracken, W. M., Almstrum, V., Diaz, D., Guzdial, M., Hagan, D., Kolikant, Y. B.-D., Laxer, C., Thomas, L., Utting, I., & Wilusz, T. (2001). A multi-national, multi-institutional study of assessment of programming skills of first-year CS students. *ACM SIGCSE Bulletin, 33*(4), 125-140.

Mcknight, C., Dillon, A., & Richardson, J. (1991). *Hypertext in Context*. Cambridge, UK: Cambridge University Press.

Mosemann, R., & Wiedenbeck, S. (2001). *Navigation and comprehension of programs by novice programmers.* Paper presented at the EEE 9th International Workshop on Program Comprehension (IWPC 2001).

Nasr, R., Hall, S. R., & Garick, P. (2003). *Student misconceptions in signals and systems and their origins.* Paper presented at the 33rd ASEE/IEEE Frontiers in Education Conference.

Onorato, L. A., & Schvaneveldt, R. W. (1986). *Programmer/non-programmer differences in specifying procedures to people and computers.* Paper presented at the First workshop on Empirical Studies of Programmers, Washington, D.C.

Passini, R. (1984). *Wayfinding in Architecture*. New York: Van Nostrand Reinh.

Petre, M., & Blackwell, A. F. (1999). Mental Imagery in Program Design and Visual Programming. *International Journal of Human-Computer Studies, 51*(1), 7-30.

Petre, M., Fincher, S., Tenenberg, J., Anderson, R., Anderson, R., Bouvier, D., Fitzgerald, S., Gutschow, A., Haller, S., Jadud, M., Lewandowski, G., Lister, R., Mccauley, R., Mctaggart, J., Morrison, B., Murphy, L., Prasad, C., Richards, B., Sanders, K., Scott, T., Shinners-Kennedy, D., Thomas, L., Westbrook, S., & Zander, C. (2003). *"My*

criterion is: Is it a Boolean?": A card-sort elicitation of students' knowledge of programming constructs (Computing Laboratory Technical Report No. 6-03). Canterbury: University of Kent.

Poucet, B. (1993). Spatial cognitive maps in animals: New hypotheses on tehir structure and neural mechanisms. *Psychological Review, 100*, 163-182.

Roddan, M. (2002). *The determinants of student failure and attrition in first year computer science*.

Rountree, N., Rountree, J., & Robins, A. (2002). Predictors of success and failure in a CS1 course. *ACM SIGCSE Bulletin, 34*(4), 121-124.

Werner, S., Krieg-Bruckner, B., Mallot, H. A., Schweizer, K., & Freksa, C. (1997). *Spatial cognition: The role of landmark, route,and survey knowledge in human and robot navigation.* Paper presented at the Informatik Aktuell (Informatik 97), Berlin.

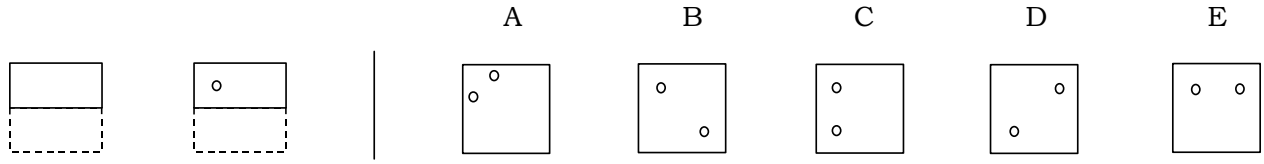
Wilson, B., & Shrock, S. (2001). *Contributing to success in an introductory computer science course: A Study of twelve factors.* Paper presented at the 32nd SIGCSE Technical Symposium on Computer Science Education.

Wolfe, J. (1971). *Perspectives on Testing for Programming Aptitude.* Paper presented at the 26th ACM National Conference.

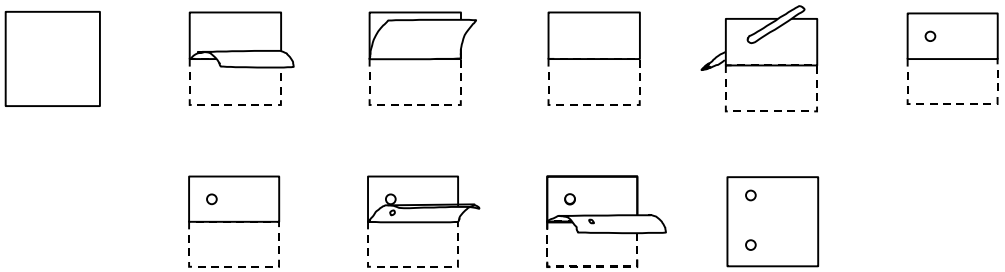
Appendix A: Instructions for the paper folding task

In this test you are to imagine the folding and unfolding of pieces of paper. In each problem in the test there are some figures drawn at the left of a vertical line and there are others drawn at the right of the line. The figures at the left represent a square piece of paper being folded, and the last of these figures has one or two small circles drawn on it to show where the paper has been punched. Each hole is punched through all the thicknesses of paper at that point. One of the five figures on the right of the vertical line shows where the holes will be when the paper is completely unfolded. You are to decide which one of these figures is correct and draw an X through that figure.

Now try the sample problem below. (In this problem only one hole was punched in the folded paper).



The correct answer to the sample problem above is C and so it should have been marked with an X. The figures below show how the paper was folded and why C is the correct answer.



In these problems all of the folds that are made are shown in the figures at the left of the line, and the paper is not turned or moved in any way except to make the folds shown in the figures. Remember, the answer is the figure that shows the positions of the holes when the paper is completely unfolded.

Some of the problems on this sheet are more difficult than others. If you are unable to do one of the problems, simply skip over it and go on to the next one.

You will have three minutes for each of the two parts of this test. Each part has one page. When you have finished Part One, STOP. Please do not go on to Part Two until you are asked to do so.

Appendix B: Protocol for the map task

- Offer the blank paper and pens/pencils of one colour.
- Direct the subject: "I would like you to draw me a map of the campus so that I can get from the <start location> to the <end location>. Please include important path markers, the clues that a stranger would need to make the right decisions to get to the <end location>. It's not important if you can't remember the names of streets and places. We don't expect an accurate drawing, just a sketch."
- Invite them to talk aloud as they work.
- Note the order in which they construct the map. (e.g., identifying both start and end point and then filling in the middle; or starting at start and drawing towards the end-point, adding features as they go).
- When the tell you they've finished, give the subject a different coloured pen and invite them to annotate their map with the "decision points" that someone would encounter if they used the map to walk the route.
- If they ask for a clarification, explain that "A decision point is an important path marker, the clue that a stranger would need to make the right decisions to get to the destination."
- For each decision point:
  Ask them to number the decision point.
  Ask: "How do I know when I've reached this decision point?"
  Ask: "What do I do at this decision point?"
- Make sure they number the points as they speak (so that the map can be correlated with the tape). They are free to append phrases or labels to their map if they wish to. If they wish to alter or correct the map, they may.
- Record their subject ID and the date on the back of their map. If there is more than one sheet, tape them together from the back *before you leave the room*.

Appendix C: Protocol for the phonebook task

- Pass the subject the local phone book.

Search One:

- Direct the subject: "I would like you to look for <insert name> in the phone book" (Assure them that this is not a trick question, and that the name really does appear in it.) Provide the name – as it appears in the phone book – on a piece of paper.
- When they have found the name, ask them: "Could you please describe to me what you just did to find that entry?" If they find it difficult to articulate, you may use the following probes:
  1. "How did you open the book?"
  2. "How did you find the page?"
  3. "How did you find the name on the page?

  You may follow each of the above probes with a single additional request to add further detail, if you consider it to be necessary. You may not probe further than the single additional request. At that point, use the subject's response verbatim.
- Note the subject's ability to articulate (good/average/poor).
- Note whether their articulation correctly reflects their actions or is wrong. If wrong, note how it is wrong.
- If the subject has not already done so, ask them to close the book.

Search Two:

- Direct the subject: "Could you now look for <Ian McDonald>, and as you do this task, please describe exactly what you are doing and how you get to the entry". Again, provide the name – as it appears in the phone book – on a piece of paper.
- Note the subject's ability to articulate (good/average/poor).
- During this (second) search, you should use no probes, and make no requests for further detail. Use the subject's responses verbatim.
- Note whether their articulation correctly reflects their actions or is wrong. If wrong, note how it is wrong.

Alternatives:

- When they've finished, ask them once: "Can you describe any other ways in which you could have searched for that name, in this book? They need not be ways that you would use yourself."

Do not probe for further detail, but you may ask "Are there any other ways?" until the subject is certain that they are finished.

Appendix D: Study process questionnaire

This questionnaire has a number of questions about your attitudes towards your studies and your usual way of studying programming.

There is no *right* way of studying. It depends on what suits your own style and the course you are studying. It is accordingly important that you answer each question as honestly as you can. If you think your answer to a question would depend on the subject being studied, please give the answer that would apply to your programming course.

Please fill in <the appropriate circle alongside the question number on the *General Purpose Survey/Answer Sheet*. The letters alongside each number stand for the following response>

A – this item is *never* or *only rarely* true of me
B – this item is sometimes true of me
C – this item is true of me about half the time
D – this item is frequently true of me
E – this item is always or almost always true of me

- Please choose the one most appropriate response to each question. <Fill the oval on the Answer Sheet> that best fits your immediate reaction.
- Do not spend a long time on each item: your first reaction is probably the best one.
- Please answer each item.
- Do not worry about projecting a good image. Your answers are confidential.
- Thank you for your co-operation.

Questions:

1. I find that at times studying gives me a feeling of deep personal satisfaction.
2. I find that I have to do enough work on a topic so that I can form my own conclusions before I am satisfied.
3. My aim is to pass the course while doing as little work as possible.
4. I only study seriously what's given out in class or in the course outlines.
5. I feel that virtually any topic can be highly interesting once I get into it.
6. I find most new topics interesting and often spend extra time trying to obtain more information about them.
7. I do not find my course very interesting so I keep my work to the minimum.
8. I learn some thing by rote, going over and over them until I know them by heart even if I do not understand them.
9. I find that studying academic topics can at times be as exciting as a good novel or movie.
10. I test myself on important topics until I understand them completely.
11. I find I can get by in most assessments by memorising key sections rather than trying to understand them.
12. I generally restrict my study to what is specifically set as I think it is unnecessary to do anything extra.
13. I work hard at my studies because I find the material interesting.
14. I spend a lot of my free time finding out more about interesting topics which have been discussed in different classes.

15. I find it is not helpful to study topics in depth. If confuses and wastes time, when all you need is passing acquaintance with topics.
16. I believe that lecturers shouldn't expect students to spend significant amounts of time studying material everyone knows won't be examined.
17. I come to most classes with question in mind that I want answering.
18. I make a point of looking at most of the suggested readings that go with the lectures.
19. I see no point in learning material which is not likely to be in the examination.
20. I find the best way to pass examinations is to try to remember answers to likely questions.

Appendix E: Programming qualities: Frequency and count for each word/phrase

Participant	logical	problem solving	attention to detail	consideration of alternatives	mathematics	knowledge of programming	ability to learn	knowledge of computers	modularising	planning	memorising and recall	persistence	visualise	enthusiasm	enjoyment	smart	decision making	obsessive	patience	practice	quick	ability to follow instructions	competent	creativity	documentation	efficiency	time	competitive	dedicated	don't know	knowledge of languages	nothing in particular	organised	team player	Count
A01		1																																	1
A02		1							1		1						1				1														5
A03			2										1												1										3
A04		1		1									1		1																				4
A05		1	2								1																								3
A06			1										1		1			1											1						5
A07	1	1														1																			3
A08		1																																	1
A09		1		1													1									1									4
A10		2								1																									2
A11						1					1																								2
A12	1																			1							1								3
A13	1																																		1
A14		1								1																									2
A15	1				1																														2
B01			1	1		1			1		1																								5
B02	1		1																			1													3
B03				2																		1													2
B04	1			1								2												1											4
B05			1																																1
B07					1																														1
B08	1					1							1																						3
B09																1																			1
B10	2																																		1
E01								1						1																					2
E03		1	1		1														1																4
E06				1												1																			2
E10							2	1													1		1												4
E11								1										1									1								3
E13																														1					1
E15	1					1																									1				3
E16			1				1								1								1												4
E17	1	2	2						1																										4
E18		2			2			3																											3
E19								1																											1
F01					1											1																			2
F03							2																								1				2
F04	1	1				1						1		1					1	1								1							8
F05		1										1							1																3
F06				2				1	1		1																								4
F07		1				1					1																								3
F09	2																							1											2

Participant	logical	problem solving	attention to detail	consideration of alternatives	mathematics	knowledge of programming	ability to learn	knowledge of computers	modularising	planning	memorising and recall	persistence	visualise	enthusiasm	enjoyment	smart	decision making	obsessive	patience	practice	quick	ability to follow instructions	competent	creativity	documentation	efficiency	time	competitive	dedicated	don't know	knowledge of languages	nothing in particular	organised	team player	Count
F11						2																													1
F12	1					1																													2
K01					2																														1
K02	1					1	1			1		1	1																				1		7
K03	1	1							1																										3
K04		1		1																															2
K05		1															1																		2
K06										1																									1
K07									1																										1
K08							2																												1
K09		1	1																																2
K10	1																																		1
M01							1	1																											2
M04								1																											1
M06			1							1																									2
M07																		1																	1
M10														1																					1
M14						1																													1
N06	1																																		1
N10	1									1			1								1														4
N13										1																									1
N21				1	1																														2
N22							1					1		1																						3
N23	2									1		1														1	1								5
P01				1									1																						2
P02					1																												1		2
P03	1								2																										2
P06	1	1			1																														3
P07														1	1																				2
P09	1	1																																	2
P12					1															1															2
P13											1																								1

Appendix F: Programming qualities: Set of Quotes for each word/phrase

Word/Phrase: ability to follow instructions

B02 The ability to follow instructions.
B03 and actually be able to do something in the correct order as well.

Word/Phrase: ability to learn

E10 they are quite readily to, to learn anything really
they are quite quick to learn a lot of other things as well, as programming. [Mm hm] Um, and, and not just in relation to computers; also in other aspects of their life.
E16 find the resource that helps them to do it, somebody who's done it before that they can learn from
F03 take in um … like concepts that ah seem to you, to be like alien to you, for example like learning how to programme is like learning another language, so um, 'cause that's essentially what you're doing, you're learning how to programme a computer which is another language,
ability to take in um like to accept technology as it's state of the art sort of thing
K02 Like to learn something from others
K08 Learn things step by step
Learn things in different ways
M01 you would need good learning skills
N22 Besides having a good teacher to teach you? An understanding I think, wanting to learn to do it I think, if you really want to learn to program then you will find it easy to pick it up. Also having a good teacher is also the other one. If you have good teachers you can pick things up. I think its if you want to learn, you will learn no matter what

Word/Phrase: attention to detail

A03 being able to describe, I mean a programme describes to the computer what to do, so the map drawing part is important, especially the sketch drawing and when you are writing the steps that you try not to leave anything out, it is vitally important you don't leave anything out and you really have to state the obvious
while you are writing a programme you need to document everything because if you assume somebody knows what a piece of code is doing you don't ????. You have to put comments in, if you don't put the comments in and assume someone knows what this code is doing, especially if ????? then if somebody grabs the code, they won't be able to do anything with it as they won't know what it is doing
A05 trying to think of every single possible point, every single possible detail
think everything through thoroughly
A06 Attention to detail - you must have attention to detail for programming
B01 Attention to detail
B02 Observation.
B05 Precision, you've got be, you can't just forget the second set of lights or something, just forget those, things don't work that way, people get lost.
E03 an eye for detail
E16 meticulous to detail
E17 possibly precise in the things that they do

the way you solve problems, really. I think if you can solve problems like that in a um, very meticulous manner, then you generally can understand programming.

K09 need to pay attention to details

M06 Must think of all the little steps getting from A to B

Word/Phrase: competent

E10 Adept.

E16 They don't just, sort of, sit down and hope that it works, they'll, if they can, do. A lot of the time, from what I can see, they're the sort of people who, may not be that welcome to getting help from other people, but they'll make an effort by themselves to go out and find the resource that helps them to do it, somebody who's done it before that they can learn from.

Word/Phrase: competitive

F04 who are competitive

Word/Phrase: consideration of alternatives

A04 how they need to be able to see a different way of what is going on

A09 realising that the most obvious solution that you start with might not be the best one to choose

B01 A way of being able to look at things differently. Figuring out, not accepting anything on face value.

B03 An open mind…
Because anything can happen…anything can happen and you can be asked to program anything, so you have to keep an open mind….It's unpredictable, actually

B04 have to be able to try about 10, 15 different possibilities and say…I don't know exactly, because I just keep trying to improve it

E06 that sometimes that they have a certain order in doing things and, don't like to, go either way about it. It's, um . . . I reckon that's the main one…Like there is more than one way, they always have a preferred way.

F06 not being too tied down to um y'know thinking the way that most people think
I suppose um always be open to, to new ideas and new ways of doing things

K04 Think outside the circle

N21 sometimes its not logical but sometimes you have to try the logical way first and then do the opposite

P01 See different ways a problem can be solved, see which one looks best, and then translate them into code

Word/Phrase: creativity

B04 probably creativity at some point,

F09 maybe a bit of creativity as well just when you create a solution, if you're creating the solution

Word/Phrase: decision making

A02 making decision

A09 identify when you need to make a decision

K05 be good at making right decisions, well not really right decisions but being able to tell people the right things

Word/Phrase: dedicated

A06 Dedicated

Word/Phrase: documentation

A03 while you are writing a programme you need to document everything because if you assume somebody knows what a piece of code is doing you don't ????. You have to put comments in, if you don't put the comments in and assume someone knows what this code is doing, especially if ????? then if somebody grabs the code, they won't be able to do anything with it as they won't know what it is doing

N23 good documentation

Word/Phrase: efficiency

A09 trying to find out the most efficient, or not the most efficient way that someone would just do it without necessary trying to develop a really strong process or most efficient method - it might not be the most efficient method

N23 being efficient about it, writing things efficiently if you have got it, if you have already written it put it as essential code rather than rewrite it again, again and again

Word/Phrase: enjoyment

A04 the idea that processes more fun than the actual outcome

A06 I'm not enjoying it so I'm not a special kind of person.

E16 generally enjoys solving problems in a, structured, manner

P07 want to enjoy it

Word/Phrase: enthusiasm

E01 enjoy computer games they might wanna, um, learn how to program one of them. And, yeah, so someone with a drive . . . to . . . to do that, someone with an enthusiasm. Yeah.

F04 they are largely, they are interested

M10 If you are interested in it, you must do it [??]

N22 get you enthused as well sometimes, sometimes your enthusiasm wanes a bit and they can really get you going again

P07 a desire to want to do it

Word/Phrase: knowledge of computers

E01 a good knowledge of computers already. Maybe even someone who's enthusiastic about computers, and, computer games, as well. If they enjoy computer games they might wanna, um, learn how to program one of them.

E10 they're quite adept on computers

E11 really likes computers

E18 IT inclined or whatever, that have used computers, for a long time
I'd say like people that have had a bit more experience, or have just used computers in general, before, would take to it a lot quicker
Not necessarily just programming but just any form of computer use.

E19 be good with all, any type of computer, to start with being like, not just programming but, making a computer and all that kind of stuff

F06 be able to at least use a computer

M01 Skills in using computers, basic background of computing and everything, probably

M04 someone who plays games, person who has to run and jump (?). If you've used some of the office style programs, that'll help

Word/Phrase: knowledge of languages

F03 I'm not saying you have to be um able to easily understand other languages but it, it certainly helps

Word/Phrase: knowledge of programming

A11 Understanding it and ??????? and knowing how to use it

B01 You need to have a reasonable memory to remember the language, but that would be secondary

B08 he has to know all class content

E15 they need to understand what they're doing, I spose, um, and have a good good feel for it in that regard

F04 when you do programming you have to learn the language. And ah, yeah when you do programming you have to learn the language, first thing you learn the language then I think that means you know ah that means you know that like, like you know English but then you've got to talk you have to put, you know like make a sentence, right, so that's what programming is. You learn the language and as you learn you try to make ah concepts out of it that are in computer or …

F07 understanding um machine language, ah so we know how to write a programme so that you can process it

F11 a lot of practice in computer knowledge of course, or else there is no way of …Programming skills I guess

Train or taught young so you've got some idea when you started off.

F12 … 'spose just the proper usage of everything, like all the commands and whatnot so that you can effectively use it to programme something,

K02 Read more examples from other programmers and know other how to program.

M14 Most important thing in programming is understanding how to do it

Word/Phrase: logical

A07 logical

A12 Logic

A13 Logical thinking

A15 Logic. I think it seems to me that programming is the most logical thing. If you lack logic ????

I think really that logic is the ????

B02 Skills of logic and ordering things.

B04 logic

B08 what we have to do first and second and third.

E: So he can make it step by step

B08: yeah yeah.

B10 has to be able to very analytical

also be able to logical, logically analyse situation and be able to draw out the step before he actually program

E15 gotta have reasonable logic in terms of um, knowing what they're, knowing what they're doing instead of um, there's knowing how to do something, but knowing what, exactly what they're doing, um, at the time

E17 to get programming you need to sort of, um, be able to order your thoughts in that manner, to break down your thoughts

F04 a logical ah process, it's very logical

F09 organising things into a logical order,

probably someone with an objective mind has a, a good chance of becoming the person who's skilled because it makes them think critically about what they're actually doing and to make them see their mistakes easily.

F12 logical, like computers are logical machines so if you think logically you can figure out the problem and if you're not logical about it you're not gonna come up with any answer, so, yeah it's … that's all I've got, just logic um, maybe being able to organise things well so to kind of programming's not typed all over the place it's all in a logical sequence so, no

K02 Think logically

K03 A logical brain

K10 ability to think logically be able to do things like this step by step to reach the end target

N06 Logic, and put things into sequence and put things in a logical way

N10 think logically

N23 thinking logically,
Logical

P03 think about things logically

P06 I think the logic skills are helpful.

P09 Need to be able to think through steps.

Word/Phrase: mathematics

A15 possibly maths, basic algebra ??????

B07 You have to be good at maths

E03 some sort of mathematical knowledge

E18 usually a bit more mathematical minded, like, um, there are some people, you know, don't get maths or whatever, they probably, usually are the ones that don't get programming as easy, like, well, the programming that I've done anyway
So your people who are good problem solvers usually are mathematically minded also

F01 really like maths, because maths is a very fast subject, there's always dealing with figures and stuff and many programming is like that

K01 Maths
I think Maths was a big one

N21 probably good maths

P02 good to be doing better at Maths

P06 I'm quite good at Maths

P12 maths has something to do with it, because it's logical just maths, being able to manipulate questions, manipulate stuff, knowing what you can do

Word/Phrase: memorising and recall

A02 memory, memorising all the things and being able to recall real quick.

A05 memory

A11 and basically just memorizing it

B01 You need to have a reasonable memory to remember the language, but that would be secondary

F06 I suppose it's a bit of a memory thing as well, you know if you can't remember it then there's no way you can do it because looking up in a book every time, what to, what to do when you want to do a particular thing really slows the um eventually people tend to not bother with having to do it, um, yeah, yeah, any other skills.

F07 relating it to something so you can understand or remember it by like um sometimes you don't know what a the command is or say if you relate it to something else, maybe a song or something you can remember it and type it back in

P13 Need to have a good memory for programming. Both long and short term memory

Word/Phrase: modularising

A02 breaking up cause its one big problem, breaking it up into little ones - it's like getting to here, then getting to there, rather than telling someone to go straight to there - its just too complex.

B01 The ability to break a task down into its components

E17 I think when you can do that you tend to be able to break down problems, and, um, solve them with more precision because of the breaking down into smaller problems, um, instead of trying to solve it all as one big problem and doing more or less than you need to.

F06 you need to sort of see problems in their simplest form

K03 Someone that can just break up a problem into small steps

K07 you just step by step, take several, a few parts of …just do it

[R] Break it into parts?

[S] Yes yes

[S] And break into small programs and write

P03 need to be able to break it down into steps

Talks about stepwise refinement. It can be overwhelming with a new problem but breaking it down into constituent steps makes it easier

Word/Phrase: obsessive

A06 obsessive compulsive disorder

E11 typical guy that, really likes computers and stuff like that, you know, and they spend heaps of time and sometimes they don't, really, do another stuff, in, their personal life,

M07 Explain to other people, tell others instructions, important for programming

Word/Phrase: organised

P02 Need to be organised to program. It'd be harder to find the problem if you weren't

Word/Phrase: patience

E03 patience

F04 I would say they are patient, because if you're patient sometimes you have to practice something over and over again to make it ah to get the concept to get what, what the output is, so yeah I think, and then I would say, so the second one, the second thing is being patient.

F05 Patience

Word/Phrase: persistence

B04 persistence

I just keep trying to improve it over and over and over and make something better, just persistence

F04 They won't let it go

F05 determination to actually learn it because it, like you don't learn it in the one day and it takes work to learn all the different skills and stuff like that

K02 try things many times

N22 even if you get stumped with something you will overcome that by doing or asking questions to get yourself a way out or over your stump
N23 finishing things basically

Word/Phrase: planning

A10 learn to plan ahead because when you are there already it is harder to go back and do the programming and sort of like, when you are at the end of the programme you are there already and get some problems along the way it is harder to start and plan again coz you're there already and it's harder to think how to do it in a better way
A14 If you have a good plan for solving problems I think it is a good way for finding a place or solving a problem
K02 Be prepared… before program do some like pseudocode, do some deskcheck…And you trial whether this way can reach the result or not…Do a desk check and practice on paper…Write some pseudocode start the program not? directly write a program.
K06 Must know very clearly how to get to the destination and to know each step
M06 The planning is crucial get every information out of what you want to happen
N10 Good programmers work to a plan I'm sure
N13 To know what you are actually trying to do and whats going to happen
N23 having a diary recording what you have done

Word/Phrase: practice

A12 practice
F04 The third thing is practicing how, so if a person is good in practicing, but he does prac- like he or she does practicing in their, a lot, for example, in the programming I reckon they can do it, everyone can do it, just they do it if they work hard, if they practice, if they're patient enough to practice that's what, I think that's how they are related patience and practicing because when you are working with computer therefore ah you get tired, you know, ah sometimes that's computer might crash down, I don't know, and ah some- something happens then you might get ah disappointed or just don't want to do it anymore but some people do just keep doing it, keep doing it until they get it, they get it right.
P12 Need experience, practice to become a good programmer

Word/Phrase: problem solving

A01 First of all I think you need to know how to approach the task if you are given a mathematical problem you will have to solve it on paper first and then apply the solution into the programme.
A02 Problem solving
A04 Problem solving
A05 problem solving
A07 be able to sort out problems.
A08 ability for solving problems
A09 Problem solving and being able to identify what the problem is and being able to solve that, sort of finding, making sure you understand how the problem works and what you want to achieve, what your goals are and then trying to develop a method that solves that particular problem.
A10 how people try to solve problems and find ways on how to solve that problem.
learn to plan ahead because when you are there already it is harder to go back and do the programming and sort of like, when you are at the end of the programme you are

there already and get some problems along the way it is harder to start and plan again coz you're there already and it's harder to think how to do it in a better way

A14 If you have a good plan for solving problems I think it is a good way for finding a place or solving a problem

E03 problem solving

E17 probably that good problem solving ability
the way you solve problems, really. I think if you can solve problems like that in a um, very meticulous manner, then you generally can understand programming.

E18 tend to be, um, suppose better at solving problems and things that are new to them
So your people who are good problem solvers usually are mathematically minded also

F04 who try to learn thing, ne-new things, new ways of solving problems and looking new ways of yeah solving problems,

F05 you have to have problem solving skills, you've gotta like be able to look at a problem and think of a solution to that problem and being able to actually code it

F07 learn how to solve problems, so maybe if you're a good problem solver that would be good

K03 Good problem solving skills

K04 have to be good at problem solving

K05 be a good problem solver.

K09 need to be a problem solver

P06 Just knowing how to go about solving something, what steps to take.

P09 All of them are problem solving

Word/Phrase: quick

A02 being able to recall real quick.

E10 Adept.

N10 can think on their feet a fair bit

Word/Phrase: smart

A07 have to be particularly smart

B09 perhaps being exceptionally bright wouldn't be very good

E06 normally that they're smart

F01 you should also be good in understanding, that's be good in like probably English, because many problems started as a one page problem and it's all in words and stuff then you have to figure it out

Word/Phrase: team player

K02 Easier like team work if you cannot solve by yourself ask for help from others not always by individual. Teamwork important.

Word/Phrase: time

A12 time

E11 Cause you have to spend heaps of time in the computer

Word/Phrase: visualise

A03 how to visualise a problem

A04 Just being able to see all the steps in a process, sort of to visualize them and to be able to see the starting point and ending point before you've done all the rest, the guts of it

A06 visualize things in their mind

B08 Maybe he will be a person who knows space very well around here

K02 First show that generally thinking about your programming…You try to write a program you should imagine what's the results of view be

N10 And some people aren't very good at explaining things and if you can't explain things very well then it might hard for you to visualise in your mind what is going on which may make it hard for you to write programs because you document things very well, spend more time writing everything out and it would take to long

P01 visualisation clearly important

Appendix G: Programming qualities: Word/phrase occurrence with other word/phrases

Matrix Nodes (a cell value is the number of documents with an intersection of one word or phrase with another)	ability to follow instructions	ability to learn	attention to detail	competent	competitive	consideration of alternatives	creativity	decision making	dedicated	documentation	don't know	efficiency	enjoyment	enthusiasm	knowledge of computers	knowledge of languages	knowledge of programming	logical	mathematics	memorising and recall	modularising	nothing in particular	obsessive	organised	patience	persistence	planning	practice	problem solving	quick	smart	team player	time	visualise
ability to follow instructions	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ability to learn	0	7	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
attention to detail	0	0	11	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
competent	0	1	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
competitive	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
consideration of alternatives	0	0	0	0	0	10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
creativity	0	0	0	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
decision making	0	0	0	0	0	0	0	3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
dedicated	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
documentation	0	0	1	0	0	0	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
don't know	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
efficiency	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
enjoyment	0	0	0	0	0	0	0	0	0	0	0	0	4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
enthusiasm	0	0	0	0	0	0	0	0	0	0	0	0	0	5	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
knowledge of computers	0	0	0	0	0	0	0	0	0	0	0	0	0	1	8	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
knowledge of languages	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
knowledge of programming	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	10	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
logical	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	22	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
mathematics	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	10	0	0	0	0	0	0	0	0	0	1	0	0	0	0
memorising and recall	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	7	0	0	0	0	0	0	0	0	0	1	0	0	0
modularising	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	7	0	0	0	0	0	0	0	0	0	0	0	0
nothing in particular	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
obsessive	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	3	0	0	0	0	0	0	0	0	0	0
organised	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
patience	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3	0	0	0	0	0	0	0	0
persistence	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	6	0	0	0	0	0	0	0
planning	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	8	0	2	0	0	0	0
practice	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3	0	0	0	0	0
problem solving	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	2	0	21	0	0	0	0
quick	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	3	0	0	0
smart	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4	0	0
team player	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
time	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2
visualise	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	7