Predictors of Success in a First Programming Course

Simon
University of Newcastle, Australia
simon@newcastle.edu.au

Sally Fincher
University of Kent, UK
S.A.Fincher@kent.ac.uk

Anthony Robins
University of Otago, NZ
anthony@cs.otago.ac.nz

Bob Baker
University of New South Wales, Australia
bob.baker@unsw.edu.au

Ilona Box
University of Technology Sydney, Australia
ilona@it.uts.edu.au

Quintin Cutts
University of Glasgow, UK
quintin@dcs.gla.ac.uk

Michael de Raadt
University of Southern Queensland, Australia
deraadt@usq.edu.au

Patricia Haden
Otago Polytechnic, NZ
phaden@tekotago.ac.nz

John Hamer
University of Auckland, NZ
J.Hamer@cs.auckland.ac.nz

Margaret Hamilton
RMIT University, Australia
margaret.hamilton@rmit.edu.au

Raymond Lister
University of Technology Sydney, Australia
raymond@it.uts.edu.au

Marian Petre
Open University, UK
m.petre@open.ac.uk

Ken Sutton
Southern Institute of Technology, NZ
ken.sutton@sit.ac.nz

Denise Tolhurst
University of New South Wales, Australia
d.tolhurst@unsw.edu.au

Jodi Tutty
Charles Darwin University, Darwin, Australia
jodi.tutty@cdu.edu.au

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 commonplace 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 correlation 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 success in programming.

Keywords: programming aptitude, education.

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; Mayer & Stalnaker 1968; Wolfe 1971). An alternative focus on academic success emerged during the 1980's. Studies exploring factors that might predict success in an introductory programming 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 & Silver 1982).

We believe that learning to program is problematic, and that the results achieved by students in programming courses do not correlate well with their other academic results. Our understanding of this phenomenon is patchy and poorly integrated, but it does seem clear that there are many influences at play. Factors suggested anecdotally and 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 (Hagan & Markham 2000), spatial reasoning and mathematical ability (Wilson & Shrock 2001), musical ability, logical reasoning ability, and previous academic background (Boyle, Carter, & Clark 2002). Measures of general intelligence correlate well (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, these indicators do not distinguish programming from other disciplines, and they have such a large effect that they 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). In order of diminishing significance, comfort level was found to be the best predictor of success, with mathematical background the second and attribution of success to luck (as a negative correlation) the third.

This paper reports on 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 can 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. The cost and challenges of carrying out empirical research at this scale mean that there are few precedents for programming-related studies of this size and scope. However, there has been a recent spate of such studies (Lister et al 2004; McCracken et al 2001; Petre et al 2003; Fincher et al 2004).

The full initial report on this study is published as a technical report (Fincher et al 2005), but the scope of the study is so great that it cannot be fully reported on in one conference paper. Therefore this paper presents a broad overview of the study, its aims, its method, and its conclusions, while other papers (to date, deRaadt et al 2005; Simon et al 2006; Tolhurst et al 2006) will go into more detail on each component of the study, giving a full analysis of the data and justification of the conclusions.

We must make clear at the outset that one particular simplification underlies our work. While we would really like to find correlations between performance on simple tasks and programming aptitude, we cannot do that because there is no accepted measure of programming aptitude. Therefore we have substituted it with the readily quantified measure of mark in a first programming course. We do not pretend that there is a linear relationship between programming aptitude and mark in a first programming course, or that different first programming courses are assessed comparably; but we have succumbed to the need for an easily measured quantity.

2 The study – an 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 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.

Several factors combine to set this study apart from others with the same basic goals.

- *Paradigm independence*: the use of diverse and generalised stimuli makes the tasks independent, so that comparisons can be made across programming paradigms, languages, and pedagogic styles.
- *Triangulation*: the study combines different approaches and collects both qualitative and quantitative data, providing 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 Concept Inventory' in Physics (Hestenes, Wells, & Swackahmer 1992; Nasr, Hall, & Garik 2003). Hence, like many researchers in this field, we relate our findings from the diagnostic tasks to the marks achieved by students in an introductory programming course, leaving somewhat open the question of the extent to which these marks reflect the students' actual programming ability.

3 The study – method

Most of the data was gathered in individual sessions between researchers and students, although at two of the participating institutions the paper-folding test was administered collectively. Participants completed the paper folding, map, and phone book tasks, and a short open-ended interview. Towards the end of their course participants completed the study process questionnaire task. The specifics of the four tasks and the interview are discussed in the following subsections.

One hundred and seventy-seven 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.

Eleven of the participating institutions contributed data. One institution contributed 32% of the 177 participants, but did not collect data for all tasks. The next highest contribution was 8%.

3.1 Task 1: paper folding

The purpose of this task was to explore the possibility of a correlation between mark in a first programming course and success in a cognitive task focusing on spatial visualisation and reasoning.

3.1.1 Background

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. 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. Researchers recorded the time taken to complete each set, along with the numbers of questions answered correctly, incorrectly, and not at all.

This study is not the first to use a paper folding test as a means of assessing aptitude for computer programming. Evans and Simkin (1989) used a paper folding test, but it accounted for only a small portion of their entire study.

3.1.2 Discussion

We found correlations between mark and paper folding score that are significant while not being particularly strong. We also found a slight negative correlation between the paper folding and a deep learning approach (see Section 3.4), which leads to some interesting speculation about the surface nature of the paper-folding task as compared with the deep nature of programming.

3.2 Task 2: map sketching

The purpose of this task was to explore the possibility of a correlation between mark in a first programming course and either the style of a map produced for a navigational task or the descriptions of the navigational decision points on that map.

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 (eg Curzon 2000). The goal is to assess participants' ability to design and sketch a simple map and to 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 somebody unfamiliar with the area. 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. Researchers collected the sketch maps with any annotations, an audio recording of the session, and their own notes regarding the order in which the map was drawn.

3.2.1 Background and motivation

In a study of expert programmers (Petre & Blackwell 1999), all of the experts reported using spatial representations for programming; for example, describing a problem space as a landscape.

Elsewhere, 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. In 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.

Werner et al (1997) suggest a hierarchy in acquisition of spatial knowledge. At the *landmark* level, knowledge is about unique objects at fixed locations. At the *route* level, knowledge is about fixed sequences of locations as experienced in traversing a route. At the *survey* level, a single model is synthesised from the known landmarks and routes. Poucet (1993) describes a similar model.

Our goal was to categorise the participants' maps according to Werner's hierarchy, then to see whether we could find a correlation between map category and mark in a first programming course.

3.2.2 Discussion

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

The nature of the relationship is a matter for conjecture at this stage, but the literature suggests that different navigational strategies may affect the way in which programmers are able to navigate programming code and in so doing to form a conceptualisation of the major features of this code.

Our results did show an interaction effect between mapping style and country (Australia, New Zealand, or Scotland). Because of the disparity in numbers between these countries, the interpretation of this result must be treated with caution. It might be due to general educational effects, the nature of the courses in each country, or the nature of the assessment items in each course.

Although a significant relationship was found between descriptions of decision points given by participants and their map style, no correlation was found between these decision point descriptors and the participants' final marks. This suggests that students' ability to represent navigational knowledge visually might 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' creation and comprehension of program structures.

We describe this component of the project in more detail in Tolhurst et al (2006).

3.3 Task 3: phone book searching

The purpose of this task was to explore the possibility of a correlation between mark in a first programming course and participants' ability to articulate their strategies for a commonplace search task.

The phone book 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, then to 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. Researchers made and transcribed an audio recording of the session, and took notes regarding the nature of the search and the quality of the articulation.

3.3.1 Background and motivation

Anecdotal evidence suggests that programmers are better than non-programmers at describing search processes. If true, this would imply that the metacognitive ability to describe strategy might be relevant to programming skill. 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 to make them relevant to students (eg Curzon 2000).

A phone book search task has previously been used in a study of naïve, novice/beginning, and experienced programmers (Onorato & Schvaneveldt 1987). 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, our study could be a useful complement to this one.

In designing the task we sought to explore two aspects of the ability to articulate strategies: accuracy and richness. By accuracy we meant how well a given description matched the researcher'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 researchers 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 students' articulations made by individual researchers during the task; and a similar but more focused assessment of all transcribed articulations made collectively by three of the researchers.

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.

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 or algorithmic manner?

3.3.2 Discussion

The observed trends are all in the expected direction. All measures show that increasing richness is associated with increasing mark in the programming course. The measures are not all statistically significant, although most become so if students who failed to complete the course are included in the analyses (these students come exclusively or at least predominantly from the weaker groups). Although individual measures are weak, taken together they appear to form a reliable picture.

In summary, this study provides some initial evidence that the question raised by Onorato and Schvaneveldt (1987) can be answered affirmatively: students who carry on to be successful programmers tend to have pre-existing strengths in a strategic / algorithmic style of articulation.

We describe this component of the project in more detail in Simon et al (2006).

3.4 Task 4: the study process questionnaire

The purpose of this task was to explore the possibility of a correlation between mark in a first programming course and aspects of participants' approach to study.

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. Researchers recorded the participants' answers to each question.

3.4.1 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 p135). This approach to learning will depend upon a number of factors that range from personal (eg motivation, available time, personal perception of task demands) through environmental (eg classroom climate, learning activities, assessment methods) to institutional factors (eg course culture, curriculum design). These different factors affect how students perceive the demands of a specific learning task and then how they choose to deal with it.

The R-SPQ-2F aims to measure two different learning approaches, 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. This approach is typically motivated by intrinsic interest in the material. Students adopting a surface approach build their view from facts and from the details of activities with the aim of reproducing material rather than making theoretical connections. This approach is typically associated with a fear of failure.

3.4.2 Discussion

We found 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).

These correlations, while significant, were only indications of a trend rather than strongly suggesting that 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 depends upon the complex interaction between the learning environment and the 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 will vary from institution to institution. Biggs et al (2001 p137) write that at the end of the course the scores 'may describe how teaching contexts differ from each other'. We found some variation between the institutions in the strengths of the correlations; this variation might be indicative of such differences between teaching contexts.

The discussion so far has 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 they are included all of the correlations described above become stronger. Not surprisingly, those who discontinued at this late stage were dominated by surface learners.

We describe this component of the project in more detail in de Raadt et al (2005).

3.5 The interview questions

The purpose of this part of the study was to elicit the qualities or skills that entry-level undergraduate students express as important to the successful learning of programming.

This 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.

Having completed the first three tasks, participants were asked:

- What do you think we were trying to find out?
- How do you think the sketch-map task might relate to programming?
- How do you think the phone book task might relate to programming?
- What qualities or skills do you think are important to learn programming well, to 'get it'?

An audio recording of each session was made and transcribed.

3.5.1 Background and motivation

We were unable to find any prior studies designed to elicit the qualities, knowledge, skills, or abilities that students perceive as important to learning programming.

There are non-discipline specific studies about students' preparation for and perception of learning at university, such as the work by Biggs (1987). While these studies apply to students in general, they provide no insight into specific student perceptions of what it takes to learn programming well.

Bailey and Stefaniak (2001) conducted a survey of industry perceptions of the knowledge, skills, and abilities (KSAs) needed by entry-level computer programmers. The survey questions were developed from focus groups of a few individuals from five companies. The 85 KSAs that emerged were divided into 53 technology skills, 20 soft skills, and 12 business concepts. In the absence of any studies more pertinent to our own interests, we were interested in exploring the similarities between their work and our own.

3.5.2 Discussion

The qualities and skills identified most often by our participants were logical thinking and problem-solving. The next top eight were attention to detail, consideration of alternatives, mathematics, knowledge of programming, ability to learn, knowledge of computers, modularising, and planning. The top 10 KSAs in 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, teamwork skills, knowledge of structured programming fundamentals, ability to implement programs, knowledge of multiple programming languages, and ability to visualise/ conceptualise.

There is some alignment between the two studies, but there are also significant differences. For example, none of our participants mentioned listening skills, which ranked quite high in the industry study.

Because we did not use probing questions to clarify the participants' meanings, a number of their utterances remain open to interpretation. For example, do these two utterances mean the same thing? (1) 'Problem-solving'; (2) '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'.

Other aspects of the study lead us to conclude that the reliability of the data source is weak. Even so, the results suggest that the students lack an awareness of what it takes to learn programming well, at least in comparison with the expert opinion generated by the industry study.

4 Conclusions

This study has explored a number of issues that may influence success in learning to program. Researchers at eleven participating institutions used the same protocol to gather data from students in introductory programming courses 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, generally in accordance with our expectations and with predictions drawn from 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 at the higher end of mark scale.

Only a small positive correlation was found 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 (Mayer et al 1989).

In the map-sketching task a progression of map-drawing styles identified in the literature, from landmark to route to survey, has a significant correlation with marks. For this effect there is some interaction with the institution's country which 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.

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.

The strengths of this study include the large number of participants and the use of diverse and generalised stimuli, the latter making the tasks independent so that comparisons can be made across paradigms, programming 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 are available. The main limitations of the study arise from the use of multiple researchers, and include issues with respect to the consistency of the application of the study protocol and of the 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 which such tasks relate either to general measures of IQ or to standard components of IQ such as verbal and spatial factors.

5 Acknowledgments

This study was supported by an ACM Special Interest Group in Computer Science Education (SIGCSE) Special Projects Grant and by a grant from Computing Research and Education Association of Australasia (CORE).

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 while a participating researcher was on a Visiting Scholar Grant; to Warren Hill of Charles Darwin University, 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 organisation and support; and to Box Catering Inc for phenomenal food.

6 References

Bailey, J.L., & Stefaniak, G., (2001). Industry perceptions of the knowledge, skills and abilities needed by computer programmers. *Proc 2001 ACM SIGCPR conference on Computer personnel research*, San Diego, CA, USA, 93-99, ACM Press.

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.

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**(1):133-149.

Boyle, R., Carter, J., & Clark, M. (2002). What Makes Them Succeed? Entry, progression and graduation in Computer Science. *Journal of Further and Higher Education*, **26**(1):3-18.

Chowdhury, A., Van Nelson, C., Fuelling, C.P., & McCormick, R.L. (1987). Predicting success of a beginning computer course using logistic regression (Abstract only). *Proc 15th ACM Annual Computer Science Conference*, St Louis, MO, USA, 449.

Cox, A., & Fisher, M. (2004). *Navigating codespace: A new direction for spatial cognition research*. Poster presented at the Biannual Convention of the International Society for Human Ethology, Ghent, Belgium. http://users.cs.dal.ca/~amcox/pubs/pubs.html. Accessed 3 Oct 2005.

Cross, E. (1970). The behavioral styles of computer programmers. *Proc 8th Annual SIGCPR Conference*, Maryland, WA, USA, 69-91.

Curzon, P. (2000). Learning Computer Science through Games and Puzzles. *Interfaces* **42**:14-15. http://www.bcs-hci.org.uk/interfaces/interfaces42.pdf. Accessed 3 Oct 2005.

de Raadt, M., Hamilton, M., Lister, R., Tutty, J., Baker, B., Box, I., Cutts, Q., Fincher, S., Hamer, J., Haden, P., Petre, M., Robins, A., Simon, Sutton, K., Tolhurst, D. (2005). Approaches to learning in computer programming students and their effect on success. In Brew, A., & Asmar, C. (2005). *Higher Education in a changing world: Research and Development in Higher Education*, **28**:407-414.

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

Evans, G., & Simkin, M. (1989). What best predicts computer proficiency? *Communications of the ACM* **32**(11):1322-1327.

Fincher, S., Petre, M., Tenenberg, J., Blaha, K., Bouvier, D., Chen, T-Y., 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. *4th Annual Finnish / Baltic Sea Conference on Computer Science Education*, Koli, Finland, 1-8.

Fincher, S., Baker, B., Box, I., Cutts, Q., de Raadt, M., Haden, P., Hamer, J., Hamilton, M., Lister, R., Petre,

M., Robins, A., Simon, Sutton, K., Tolhurst, D., Tutty, J. Programmed to succeed?: a multi-national, multi-institutional study of introductory programming courses (2005). *Computing Laboratory Technical Report 1-05*, University of Kent, Canterbury, UK.

Hagan, D., & Markham, S. (2000). Does it help to have some programming experience before beginning a computing degree program? *ACM SIGCSE Bulletin* **32**(3):25-28.

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

Leeper, R.R., & Silver, J.L. (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.

Mayer, D.B., & Stalnaker, A.W. (1968). Selection and Evaluation of Computer Personnel – the Research History of SIG/CPR. *Proc 1968 23rd ACM National Conference*, Las Vegas, NV, USA, 657-670.

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

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.

Nasr, R., Hall, S.R., & Garik, P. (2003). Student misconceptions in signals and systems and their origins. *Proc 33rd ASEE/IEEE Frontiers in Education Conference*, Boulder CO, USA, T2E23-T2E28.

Onorato, L.A., & Schvaneveldt, R.W. (1987). Programmer-nonprogrammer differences in specifying procedures to people and computers. *The Journal of Systems and Software* **7**(4):357-369.

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 6-03*. University of Kent, Canterbury, UK.

Poucet, B. (1993). Spatial cognitive maps in animals: New hypotheses on their structure and neural mechanisms. *Psychological Review* **100**(2):163-182.

Roddan, M. (2002). *The determinants of student failure and attrition in first year computing science*. Final-year undergraduate project. http://www.psy.gla.ac.uk/~steve/localed/mrodd.html. Accessed 3 Oct 2005.

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

Simon, Cutts, Q., Fincher, S., Haden, P., Robins, A., Sutton, K., Baker, B., Box, I., de Raadt, M., Hamer, J., Hamilton, M., Lister, R., Petre, M., Tolhurst, D., Tutty, J. (2006). The ability to articulate strategy as a predictor of programming skill. *Proc Eighth Australasian Computing Education Conference*, Hobart, Australia, Jan 2006.

Tolhurst, D., Baker, B., Hamer, J., Box, I., Cutts, Q., de Raadt, M., Fincher, S., Haden, P., Hamilton, M., Lister, R., Petre, M., Robins, A., Simon, Sutton, K., Tutty, J. (2006). Do map-drawing styles of novice programmers predict success in programming? A multi-national, multi-institutional study. *Proc Eighth Australasian Computing Education Conference*, Hobart, Australia, Jan 2006.

Werner, S., Krieg-Brückner, B., Mallot, H.A., Schweizer, K., & Freksa, C. (1997). Spatial cognition: the role of landmark, route, and survey knowledge in human and robot navigation. In Jarke, M., Pasedach, K., & Pohl, K. (Hrsg.), *Informatik 97: Informatik Aktuell* 41-50, Berlin.

Wilson, B.C., & Shrock, S. (2001). Contributing to success in an introductory computer science course: a study of twelve factors. *ACM SIGCSE Bulletin* **33**(1):184-188.

Wolfe, J.M. (1971). Perspectives on Testing for Programming Aptitude. *Proc 1971 26th ACM National Conference*, Chicago, IL, USA, 268-277.