Aspects of Cognitive Style and Programming

Rebecca Mancy, Norman Reid
Centre for Science Education
University of Glasgow
mancyr@dcs.gla.ac.uk, N.Reid@mis.gla.ac.uk

Keywords: POP-VI.E. Computer Science Education Research; POP-I.B. Barriers to programming; POP-II.A. Learning Styles; POP-II.A. Individual Differences; POP-V.A. Short-term memory.

Abstract

There is widespread concern about low pass rates on introductory programming courses. While considerable research has been carried out to elucidate the reasons for this situation, many of the parameters leading to success or failure in the subject remain unknown. This article describes the results of an experiment to test two cognitive characteristics that have been shown to be important in other conceptual areas: working memory space and field dependency. These are related to examination results of around 150 students on an introductory programming course at the University of Glasgow. The results show that whilst working memory space appears to have only a marginal influence on levels of achievement on the course, field dependency is an important factor in determining success. The implications of this on the teaching of the subject are discussed briefly.

Introduction

Programming has always been, and remains, an important component of computer science degrees. However, introductory programming courses are known for their notoriously poor pass rates. At the University of Glasgow, the CS1P introductory programming course in the Department of Computing Science is no exception. In 2002-2003, only 50% of students obtained at least a C grade required for automatic progression to Level 2 (second year). According to other circumstances, some students obtaining a D grade are admitted to Level 2; 64% of students gained a D grade or better. These percentages are well below faculty averages, and although grade boundaries may be considered arbitrary, lecturers believe that these levels do represent the absolute minimum required to cope with later programming courses. Undergraduate students at the University of Glasgow take three subjects in their first year, concentrating on one or two in later years, so some of these students do not intend to study computing science as their main degree subject. Nonetheless, such low pass rates indicate that students experience difficulties with programming and anecdotal evidence confirms this.

Many studies have been carried out to try to elucidate the reasons for what are seen as poor levels of achievement in programming. Empirical research has been carried out into difficulties related to specific language structures such as variables (Sajaniemi, 2002), typical student misconceptions in programming (e.g. Putnam et al., 1989) and bugs in novice programs (e.g. Soloway, 1986). Several researchers have compared programming languages, paradigms and environments (Deek & McHugh, 1998; McIver & Conway, 1996; Green et al., 1991), evaluating their ease of use and appropriateness for novices.

A large body of research has concentrated on the differences in knowledge and skills between novice and experienced programmers. Many of these are concerned with the mental representations of knowledge possessed by novices and experienced programmers (Holt et al., 1987; Wiedenbeck et al., 1993). Other studies compare novice and experienced programmer processes (e.g. Kahney, 1983). However, "only indirect attempts have been made to infer what properties of attained expertise might mean for the acquisition of competence" (Glaser 1996), and little is known about the intermediate stages through which students must pass, or about the process of learning itself.

Practical studies have concentrated on applying educational theories to the teaching of programming in order to improve student learning, for example, Mayer (1989a), who used Ausubel's (1960) idea of

advance organisers. Work has been carried out on the use of metaphors and analogies in teaching programming (e.g. DuBoulay 1989). A current theme is that of using learning objects (Boyle, 2003) whilst further studies have used multifactor general strategies in an attempt to improve student performance (Chalk et al., 2003).

Nevertheless, there remains a great deal of variation in the level of skill achieved by different students during introductory programming courses. Kahney (1983) recognises this, talking of ordinary and "talented" novices: there is a need to understand the fundamental causes that make these learners different from others, and the same applies for those students who fail to understand even the basics of the subject. Clearly students differ in various ways, many of which may impact on their programming performance, and a better understanding of these factors will allow us to address them through appropriate measures. These differences range from previous experience and knowledge, through personal motivation, to social issues such as a feeling of integration in the university setting (Tinto, 1975; Perry 1999; etc.). The work described here concentrates on another type of factor: that of cognitive style and specific abilities. The focus is on working memory space (WMS) capacity and field dependency (FD).

Working Memory and Field Dependency

Working Memory

Memory is clearly important in learning to program, as with any skill. Whilst it is unlikely that models of information processing accurately represent reality, they allow us to visualise the interaction of different types of memory. Many models of information processing exist in the literature, largely based on the work of Atkinson and Siffrin (1971). The version below (figure 1), proposed by Johnstone (1984), suggests a simplified model of the learning process and enables us to understand the limitations of learning.

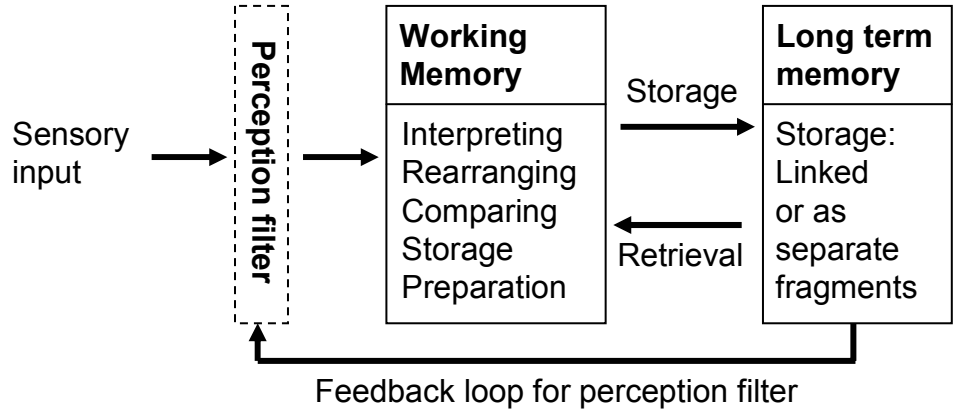

Figure 1. The information processing model

When we attend to a stimulus, it passes into working memory so our working memory contains those things to which our immediate consciousness pays attention. Working memory has at least two functions: to hold information temporarily and to process it into a form that can be used or stored. Use of the term "working memory" as opposed to the traditional "short-term memory" captures this idea of processing. According to Johnstone (1984), working memory is "that part of the brain where we hold information, work on it, organise it, and shape it, before storing it in the long-term memory for further use".

Working memory has two important characteristics: it has a limited capacity for storage, and memories stored in it decay rapidly. Without continued rehearsal, items can be held in working memory for around 20 seconds (Brunning, 1995). The capacity of working memory was investigated

by Miller (1956) who carried out a series of experiments and determined that an adult (defined as age 16 or over) is able to store about seven plus or minus two (7±2) units or separate "chunks" simultaneously. What constitutes a chunk depends on the individual and his or her knowledge, experience and acquired skills (Johnstone & El-Banna, 1986). It may be a word, a letter, or a digit, or even a whole concept. As we gain experience, concepts are grouped together to form fewer, higher level concepts and so there are differences in complexity and number of information units that are perceived by novices (child, beginner, etc.) and experts (adult, teacher, etc.). Chunking therefore allows us to retain more information simultaneously; for example, we can retain more letters if these are organised to make words than if they are in a seemingly random order.

As discussed earlier, working memory not only holds information, but it is within working memory that information processing takes place. Since working memory space is limited, the space has to be shared between storage and processing. If too much information is to be held, there is no capacity left for processing this information and this becomes impossible (Johnstone, 1997). Working memory overload occurs when there is too much information or too many manipulations are required simultaneously.

Field Dependency

One important and much-studied cognitive characteristic is field dependency. Witkin (1977) was the first to elaborate the construct, while researching the relationships between personality and the way subjects perceive and interact with their environment, notably looking at human perception of upright. Witkin's research indicated that other cognitive characteristics are correlated with the perception of upright and field dependency was seen to be linked to disembedding and cognitive restructuring abilities. Witkin and Goodenough (1981) defined field dependent and independent characteristics in the following way: an individual who can easily separate an item from an organised perceptual field is called field-independent, whilst those for whom this is difficult and who readily accept the dominating field or concept are described as field dependent. In other words, a person who is field-independent will more easily extract the "message" or "signal" from the "noise" or irrelevant information. A continuum exists between the field-dependent / field-independent categories, with those of intermediate ability being called field intermediate or field neutral.

The skills possessed by field-independent individuals are described as providing structure for interpreting a complex stimulus, for breaking this up into separate elements and for providing a different organisation than that suggested only by salient cues in the original information (Riding & Cheema, 1991). In other words, these learners can restructure material in their own way, applying internally generated rules arising from prior experience or developed from cues in the material.

In order to measure field dependency, the Embedded Figures Test was developed (Witkin et al., 1977). This test uses an individual's ability to disembed a simple shape from a complex visual field (pattern). A score is calculated as the number of shapes correctly identified and the student is situated along the field dependent – field independent continuum.

Working Memory Space and Field Dependency

Since working memory is of limited capacity, it is important to try to maximise its usage. Johnstone and Wham (1982) suggested that working memory overload appears to occur when the learner cannot differentiate the "message" or important information from the "noise"; the non essential and often irrelevant information that the teacher is transmitting to the learners. The field independent person is capable of using his or her working memory space more efficiently simply because it is not becoming cluttered with information irrelevant to the problem being faced.

Experiment Motivation

Most programming tasks encountered by novice programmers at university involve given data and specified outcomes or goals. The method may be largely uncertain. To be able to generate a method in the computational model, the learner must be familiar with the model itself and this requires a great

deal of conceptual knowledge (Mayer, 1988). Learning the concepts of the computational model depends at least partially on learning the language itself and through "finding patterns in linguistic output" (Chater & Vitanyi, 2003). Patterns are also important at a higher level as "experienced programmers can rapidly recognise cliched patterns" (Soloway 1984) in problem or program structure and apply or extract algorithms and plans, another skill that the learner must develop.

Research shows that working memory space and field dependency are useful predictors of success in conceptual areas such as mathematics (Christou, 2000) and statistics (Ghani, 2003). In problem solving, working memory space is important as problem solvers try to keep track of goals and subplans (Carpenter et al., 1990), both recognised skills for programming (Corbett & Anderson, 1995). Similarly, field-independent students are better problem solvers than their field-dependent colleagues (Ronning 1984). In learning the programming language itself and the patterns that appear within solutions, field dependency is almost certainly important as field-independent learners are generally more able to generate structural rules (Witkin 1977).

Measuring Working Memory Space and Field Dependency

In spring 2003, we administered tests for working memory space and field dependency to the cohort of first year students studying the CS1P introductory programming course in Ada at the Computing Science Department, University of Glasgow. The tasks involved were performed by all those present at a normal lecture.

Working memory space was tested using the digit span test (based on Jacobs, 1887). This test requires participants to listen to a list of digits, read out at a rate of one per second. When all the digits have been read out, the participant then has to write the digits down on a specially prepared sheet. Again, a second is allowed per digit to allow sufficient time to enter the digits on the test sheet. The test becomes more difficult as the number of digits in the list is increased, two chances being allowed for each level of testing (number of digits). We used two versions of the digit span test: "digit span forwards" and "digit span backwards". In the first, participants are asked to listen to the list of numbers and then to write them down in the original order. In the second, they are required to reverse the digits in their head and to write them down in reverse order, (without writing from right to left).

We expected, as in previous experiments (e.g. Johnstone & El-Banna, 1986) that the digit span backwards test would give on average scores corresponding to approximately one fewer space than those of the forwards test. We suppose that this is because one space is used up by the processing of the digits in order to reverse them. From a computational point of view, this is consistent with sorting algorithms which typically require at least one extra "space" (e.g. a variable) in order to carry out the sorting procedure.

However, it must be noted that although processing is necessary in order to reverse the digits, we cannot be sure that no processing is required on the digit forwards test. For an answer to be marked as correct, the participants must write the numbers down in the order that they are given; it is not sufficient to be able to remember the numbers, they must also be remembered in the correct order. Perhaps, more than our ability to hold an (unordered) set of numbers, we are testing the number of pieces of information that can be held *meaningfully* in the working memory.

Scores were calculated for the tests based on the number of correct answers, obtaining means of 6.92 and 5.92 for the forwards and backwards scores respectively. Although the digit forward test was essentially used for practice, information was removed from the sample for a small number of students where forwards and backwards scores were highly inconsistent, as was the case when the score sheet was illegible, etc. Only the scores for the digit backwards test were used subsequently as chunking strategies are harder to develop on this test and so these scores are considered more representative.

Field dependency was measured using an embedded figures test. Witkin's original test was modified slightly and comprised 20 complex figures, plus two additional introductory figures used as illustrative examples. Students were given a booklet containing the set of complex figures and the simple geometrical shapes to be found. They indicated their responses by tracing the outline of the

shape in the booklet. A score was calculated as the number of correct shapes found within the complex figures. A total time of 17 minutes was given to complete the test.

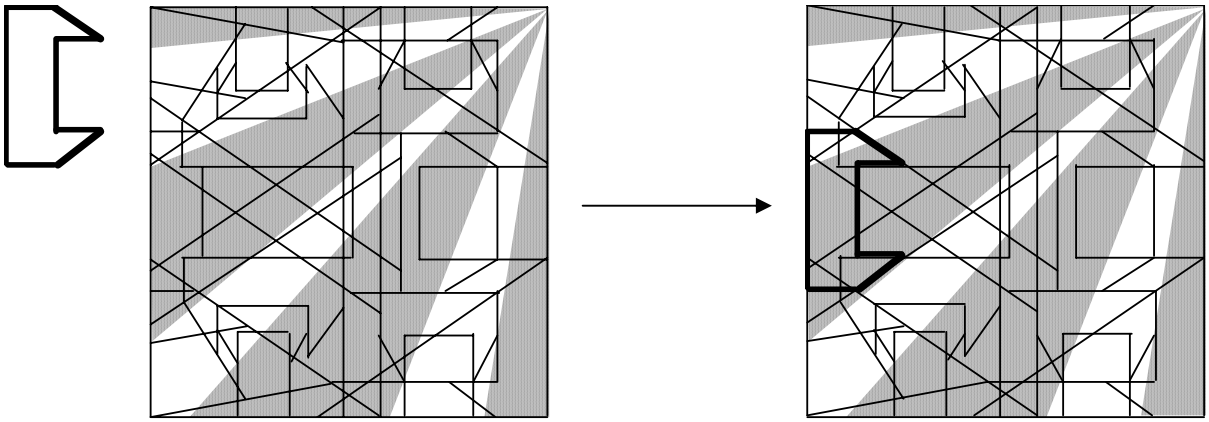

Figure 2. Example item from the embedded figures test

Although means are quoted above for results on the working memory space test in order to compare with research elsewhere, absolute measurement was not required. The aim was to see how students compared with one another and not with any predefined levels.

Results

During the course of the year, students on the CS1P course sit 4 examinations: a practical examination (Practical 1) and a traditional written examination (Class Test) in January, and a second practical (Practical 2) and a further written examination (Final Exam) in June. The final examination is worth 70% with the other 30% of the marks spread evenly over the other examinations.

Whilst the written examinations test many of the skills of programming (problem solving, syntax, etc.), there are certain skills that are impossible to test in this way. The practical examinations are designed to complement these by testing detailed coding ability, debugging skills and effective use of the programming environment. A programming problem is given to students in advance and students prepare as they feel necessary. During the practical examination, students have no access to their usual file store and standard examination conditions are respected.

We calculated correlations (Pearson r) between the scores obtained for working memory space and field dependency, and marks on each of these examinations. The results are shown below:

	Practical 1	Class Test	Practical 2	Final Exam
Working Memory Space	0.15	0.13	0.26**	0.16
Field dependency	0.27**	0.24**	0.36**	0.40**
Number of students	158	159	145	154

*Table 1 – Pearson correlations between scores on the working memory space and field dependency tests (to 2dp). ** indicates that this correlation is significant at the 99.9% level. All other correlations were not significant at the 95% level.*

As it can be seen from the table, working memory space appears to be of marginal importance on the introductory programming course at the University of Glasgow: only one correlation, for the second practical examination, is statistically significant. This result is difficult to interpret and only tentative suggestions are offered here. Working memory space becomes a limiting factor where several pieces of information must be held in memory simultaneously. We might hypothesise that only this examination contained material where the information to be held exceeded the working memory space for a significant number of students. If this is the case, it would seem to indicate that working memory space is not a limiting factor during the learning stage of this course, but that it may be during examinations. However, other explanations are possible and further research would be required to confirm this.

The results for field dependency show statistically significant correlations across all examinations. Students who scored well on the field dependency test – and are thus considered to be field-independent – achieved, *on average*, better marks in the examinations. The division of the sample into three groups of approximately equal size, as shown below, is given as a visual illustration of the way the field dependency related to marks in the final examination. (Statistical significance is not given.)

	Field dependent	Field intermediate	Field independent
Score range on test	0-11	12-16	17-20
Number of students	55	54	45
Average final exam mark	44.0%	53.5%	63.5%

Table 2 – Average mark (to 1dp) on final examination for each group: field-dependent, field-intermediate, field-independent

Field dependency is clearly a critical skill and the correlation of 0.4 on the final exam, which the CS1P lecturers consider as the most representative of programming ability, is extraordinarily high. Although our aim here was not to predict success or failure in programming, it is worth noting that general measures of nonverbal intellectual ability used to predict programming ability, such as in the IBM Programmer Aptitude Test (PAT) or Aptitude Assessment Battery Programming (AABP), usually correlate with programming test scores in the range of 0.3 (Mayer, 1989b). In Mayer's study, higher correlations were obtained using tests that directly measure "specific thinking skills or specific cognitive components" of programming such as problem translation skills. A more recent study, using a test developed by Huoman that consists of a collection of programming-like tasks that do not require previous programming experience, found a correlation of around 0.5 with final examination marks on an introductory programming course (Tukiainen, 2002).

Conclusion

It is interesting that although working memory space limitations have been shown to pose considerable difficulties in related areas of science education, this would appear not to be the case for introductory programming. This may be inherent in the subject itself, or it may be that the course lecturers already (and probably unintentionally) follow guidelines for avoiding working memory overload.

However, field dependency appears to be a critical skill in learning to program. It seems likely, given the general nature of field dependency that this will occur across programming paradigms, but research needs to be carried out to confirm this. Lecturers and course organisers should be aware of the importance of this cognitive skill and develop course material in line with this, perhaps by focusing overtly on key ideas or restructuring information so that important concepts are highlighted. For example, some researchers have suggested that use of colour may help field-dependent students to see structure (Moore & Dwyer, 1991). In a recent questionnaire at the end of the CS1P course, 74%

of the students indicated that they had experienced difficulties with debugging. Debugging skills are not particularly emphasised during the course with links between compiler error messages and the errors that can provoke them never explicitly taught; students must see the patterns and generate a rule themselves. Perhaps explicit teaching could assist students in this area.

A further question to ask is whether the skills associated with field-independent individuals are teachable and can be developed. Perhaps students can be trained to better see the key concepts for themselves and to restructure their ideas. Potentially, testing students' levels of field dependency would allow extra support to be provided for these individuals, specifically aimed to help them with the skills they lack.

However, in order to fully address these issues, a better understanding is required about how students extract messages and how the concepts are represented in their minds. Teaching almost certainly plays a role in this, but the experience of learning different concepts may differ partially as a function of the concepts themselves. Initially, work should concentrate on those concepts that students find particularly difficult.

Acknowledgements

The authors wish to thank the following who have contributed to this research: Steve Draper, Quintin Cutts, Rob Irving, Gail Reat and the CS1P students (2002-2003) who participated in this research.

References

Atkinson, R. and Siffrin, R. (1971) The control of short-term memory. *Scientific American*, 225, 82-90.

Ausubel, D. P. and Robinson, F. (1960) The use of advance organizers in the learning and retention of meaningful verbal material. *Journal of Educational Psychology*, 51, 267-272.

Boyle, T., (2003) Design principles for authoring dynamic, reusable learning objects. *Australian Journal of Educational Technology*, 19(1), 46-58.

Brunning, R. H., Schraw, G. J., and Ronning, R. R. (1995) *Cognitive Psychology and Instruction*, Prentice-Hall, New Jersey (Second edition).

Carpenter, P. A., Just, M. A. and Shell, P. (1990) What one intelligence test measures: a theoretical account of processing in the Raven's Progressive Matrices Test. *Psychological Review*, 97(3), 404-431.

Chalk, P., Boyle, T., Pickard, P., Bradley, C., Jones, R., Fisher, K. (2003) Improving pass rates in introductory programming. *Proceedings of LTSN-ICS 2003*, Galway 2003.

Chater, N. and Vitanyi, P. (2003) Simplicity: a unifying principle in cognitive science? *Trends in Cognitive Sciences*, 7(1), 19-22.

Christou, K. (2001) Difficulties in solving algebra story problems with secondary pupils, MSc. Thesis, University of Glasgow

Corbett, A. T. and Anderson, J. R. (1995) Knowledge decomposition and subgoal reification in the ACT programming tutor. *Artificial Intelligence and Education, Proceedings of AI-ED 95*. Charlottesville, VA: AACE, 243-254.

Deek, P. F., McHugh, J. A. (1998) A survey and critical analysis of tools for learning programming. *Computer Science Education*, 8(2), 130-178.

du Boulay, B. (1989) Some difficulties of learning to program. In E. Soloway and J. C. Spohrer (eds.) *Studying the Novice Programmer*. New Jersey: Lawrence Erlbaum, 283-300.

Ghani, S. (2003) Personal communication.

Glaser, R. (1996) Changing the agency for learning: acquiring expert performance. In K. A. Ericsson (ed.), *The Road to Excellence – the Acquisition of Expert Performance in the Arts and Sciences, Sports and Games*. Lawrence Erlbaum, New Jersey, 303-311.

Green, T. R. G., Petre, M. and Bellamy, R. K. E. (1991) Comprehensibility of visual and textual programs: a test of superlativism against the "match-mismatch" conjecture. In J. Koenemann-Belliveau, T. Moher, and S. Robertson (eds.), *Empirical Studies of Programmers: Fourth Workshop*. Ablex, Norwood, NJ, 121-146.

Holt, R. W., Boehm-Davis, D. A. and Chultz, A. C. (1987) Mental representations of programs for student and professional programmers. In G. M. Olson, S. Sheppard and E. Soloway (eds.), *Empirical Studies of Programmers: Second Workshop*, Ablex, NJ, 33-46.

Jacobs, J. (1887). Experiments on 'prehension'. *Mind*, 12, 75-79.

Johnstone, A. H. and Wham, A. J. B. (1982) Demands of practical work. *Education in Chemistry*, 19(3), 71-73.

Johnstone, A. H. (1984) New stars for the teacher to steer by? *Journal of Chemical Education*, 61(10), 847-849.

Johnstone, A.H. and El-Banna, H. (1986) Capacities, demands and processes: a predictive model for science education. *Education in Chemistry*, 23(3), 80-84.

Johnstone, A.H. (1997) Chemistry teaching, science or alchemy? [The Brasted Lecture], *Journal of Chemical Education*, 77(3), 262-268.

Kahney, H. (1983) Problem solving by novice programmers in Green, T. R. G. and Payne, S. J. (eds.) *The Psychology of Computer Use: a European Perspective*, Academic Press, London, 121-141.

Mayer, R. E. (1988) From novice to expert. In Helander, M. (ed.), *Handbook of Human-Computer Interaction*, Elsevier, Amsterdam, 569-580.

Mayer, R. E. (1989a) The psychology of how novices learn computer programming. In Soloway, E. and Spohrer, J. (eds.) *Studying the Novice Programmer*, Lawrence Erlbaum Associates, Hillsdale, NJ, 129-159.

Mayer, R. E. (1989b) Learning to program and learning to think: what's the connection? In Soloway, E. and Spohrer, J. (eds.) *Studying the Novice Programmer*, Lawrence Erlbaum Associates, Hillsdale, NJ, 113-124.

McIver, L & Conway, D.M. (1996) Seven deadly sins of introductory programming language design, *Proc. Software Engineering: Education and Practice* (SE:E&P'96), University of Otago, Dunedin, NZ, IEEE Computer Society, 309-316.

Miller, G. D. (1956) The magical number seven plus or minus two: some limits on our capacity for processing information. *Psychological Review*, 63, 81-97.

Moore, D. M. and Dwyer, F. M. (1991) Effect of color coded information on students' levels of field dependence. *Perceptual and Motor Skills*, 72, 611-616.

Perry, W. G. (1999) Forms of intellectual and ethical development in the college years: a scheme, Jossey-Bass, San Francisco.

Putnam, R. T., Sleeman, D., Baxter, J. A., Kuspa, L. K. (1989) A summary of misconceptions of high school basic programmers. in Soloway, E. and Spohrer, J. (eds.) *Studying the Novice Programmer*, Lawrence Erlbaum Associates, Hillsdale, NJ.

Riding, R. and Cheema, I. (1991) Cognitive styles – on overview and integration. *Educational Psychology*, 11(3/4), 193-215.

Ronning, Royce R., McCurdy, D. and Ballinger, R. (1984) Individual differences: a third component in problem-solving instruction. *Journal of Research in Science Teaching*, 21(1), 71-82.

Sajaniemi J. (2002) Visualizing roles of variables to novice programmers. *Proceedings of the Fourteenth Annual Workshop of the Psychology of Programming Interest Group (PPIG 2002*) (eds. J. Kuljis, L. Baldwin, R. Scoble), London, U.K., June 2002, 111-127.

Soloway, E., and Ehrlich, K. (1984) Empirical studies of programming knowledge. *IEEE Transactions on Software Engineering*, 10(5), 595-609.

Spohrer, J., and Soloway, E. (1986) Analyzing the high-frequency bugs in novice programs . In E. Soloway and S. Iyengar, (eds.) *Empirical Studies of Programmers*, Ablex, New York, 230-251.

Tinto, V. (1975) Dropout from higher education: a theoretical synthesis of recent research. *Review of Educational Research*, 45, 89-125.

Tukiainen, M. and Mönkkönen, E. (2002) Programming aptitude testing as a prediction of learning to program. *Proceedings of PPIG 14*, Brunel, London, 18-21 June 2002, 45-57.

Wiedenbeck, S., Fix, V. and Scholtz, J. (1993) Characteristics of the mental representations of novice and expert programmers: an empirical study. *International Journal of Man-Machine Studies*, 39(5), Academic Press, 793-812.

Witkin, H. A., Moore, C.A., Goodenough, D. R., and Cox, P. W. (1977) Field dependent and independent cognitive styles and their educational implications. *Review of Educational Research*, 47, 1-64.

Witkin, H. A., and Goodenough, D. R. (1981) *Cognitive Styles: Essence and Origins, Field Dependence and Field Independence*. International University Press, New York