Computer programming skills: A cognitive perspective

Graafsma, Irene

DOI:
10.33612/diss.168003240

Chapter 6

General Discussion

6.1 OVERVIEW

The research reported in this thesis aimed to contribute to a better understanding of computer programming as a skill in modern society and education. Three relevant areas of research within the field of cognition were identified: cognitive skills, personality traits, and processing in the brain. To study programming in the first two areas, it was necessary to lay methodological foundations to be able to measure programming skill. Specifically, the aim of Chapter 2 was to answer the methodological research question 1) Can we create two reliable parallel short versions of the Second Computer Science 1 (SCS1) programming test? Subsequent chapters then aimed to answer three further research questions: 2) Which cognitive skills predict success at the end of a programming course? 3) Do autistic traits predict success in a programming course? and 4) Is a programming language processed similarly to a natural language?

In this chapter, I first discuss the answers to these four research questions. Then, I focus on the findings across the different studies regarding the relationship between programming and language. Next, I reflect on implications of the findings for education. Finally, I give suggestions for future research, and discuss the broader impact of the outcomes of the thesis.

6.2 ANSWERS TO RESEARCH QUESTIONS

(1) Can we create two reliable parallel short versions of the Second Computer Science 1 (SCS1) programming test?

Chapter 2 reports the development of two shortened versions of the Second Computer Science 1 programming test (SCS1-S). The results showed that these versions could not be considered fully parallel, and that Version 1 was of questionable quality. However, Version 2 was of comparable reliability and validity to the full SCS1 that was validated by Parker et al. (2016) and Xie, Davidson et al. (2019). Hence, this research has resulted in a validated short version, allowing this test to be used in a wider variety of studies that require shorter testing time.

In addition, it was clear that the test was difficult for the current student population: they showed low accuracy and did not complete many items. This is in line with the findings

of both Parker et al. (2016) and Xie, Davidson et al. (2019). It suggests that our Australian undergraduate population performed similarly to the American undergraduate students, showing that this test can be used to measure programming skill across different courses, institutions and countries. However, this finding also means that the test is limited in its ability to measure differences in programming skill for populations with very little programming experience, and that the test cannot be used as a pre-test for computer science courses or research studies when students have no previous programming experience.

Based on the findings in Chapter 2 I give two suggestions for future use of the SCS1 and the SCS1-S. First, I suggest only using Version 2 of the shortened SCS1 test, as this version was found to be of similar quality as the full SCS1 test. Unfortunately, because testing for the studies in Chapters 3 and 4 happened simultaneously with the validation study in Chapter 2, Version 1 was still used in the current studies. This may have caused more noise in the independent programming measure, and therefore may have led to lower correlations than if we had only used Version 2. Second, I suggest that both the full SCS1 and the short versions of this test are most suitable to be used with students who are slightly more advanced than the students in the current studies. This means that the need for an easier test of basic programming skills still remains. Future studies could simplify the questions of the SCS1 or design an entirely new programming test that is more sensitive to differences in early gains in programming skill following the start of programming training.

(2) Which cognitive skills predict success at the end of a programming course?

In the study in Chapter 3, we examined the extent to which five cognitive skills, measured at the start of an Australian undergraduate programming course, correlated with programming performance on the SCS1-S, as an independent programming measure, and with the course grades at the end of the semester. A task measuring logical reasoning was the most consistent predictor, significantly predicting programming performance on both the course exam and the SCS1-S. Algebra and vocabulary learning tasks were also found to predict programming performance, but only on the independent programming test. This supports the idea that the predictive value of a cognitive skill depends on the way programming is assessed, as the testing format may emphasise certain aspects of

programming skills over others. When measuring programming skills with a stricter test and with more time pressure, rather than a more flexible format, participants may rely more on their problem-solving speed and their memorisation of the programming language, therefore increasing the predictive values of algebra and vocabulary skills. Additionally, the language component (consisting of grammar learning and vocabulary learning) did not predict programming skill on either outcome measure, while the algorithmic/mathematics component (consisting of pattern recognition, logical reasoning and algebra) predicted performance on both measures. For a more extensive discussion of the relationship between language and programming, see the section on "The role of natural language skills in programming" later in this chapter.

(3) Do autistic traits predict success in a programming course?

In the study in Chapter 4, we measured autistic traits at the start of an Australian undergraduate programming course and correlated these with programming performance on the independent SCS1-S measure and the course-related grades at the end of the semester. Autistic traits were found to be higher in the current student population compared to the general population. However, they did not predict programming skill. There were also no significant correlations between autistic traits and the cognitive skills, nor were there any significant correlations between the individual subscales of the Autism Spectrum Quotient (AQ) and programming skill. These results suggest that despite popular stereotypes of programmers as people with autistic characteristics, autistic traits do not play a role in predicting programming aptitude.

(4) Is a programming language processed similarly to a natural language?

In the study in Chapter 5, we studied Event Related Potentials (ERPs) measured with Electroencephalography (EEG) in response to violations in participants' native language (Dutch), second language (English) and a programming language (Java). Violations in all three languages elicited positive ERP deflections. However, there were differences between the languages with regard to onset, offset and scalp distribution of the effects. Specifically, the early onset and offset of the effect in Java, as well as its frontal and bilateral scalp distribution suggested that this type of bracket violation was processed differently to the subject-verb violations in the two natural languages. Based on both timing and scalp

distribution, the effect for Java could be interpreted as either an early P600 or a late P300 effect, and therefore we cannot draw a conclusion as to whether this effect was language related. However, the effect in response to the bracket violations was similar to that observed in response to orthographic violations in natural language (Vissers et al., 2006), suggesting that programmers may perceive it as incorrect spelling. In sum, based on the results of the current study, we cannot conclusively answer the question if Java is processed similarly to a natural language.

6.3 THE ROLE OF NATURAL LANGUAGE SKILLS IN PROGRAMMING

Based on the levels of the PGK-hierarchy model of the programming process (named after Perrenet, Groote and Kaasenbrood who first defined it; Perrenet et al., 2005) as described by Armoni (2013), I speculated that the first two levels of the model, the problem level and the object level, rely most strongly on algorithmic/mathematical skills. For the third level of the model, the program level, I speculated that this level relies more strongly on knowledge of the programming language and, thus, on language skills. It was thus expected that language skills would be predictive of programming ability. However, the results from Chapter 3 suggest that programming skill, following the course undertaken by our participants, relied more on algorithmic/mathematical skills and less on language skills, with only vocabulary learning predicting performance only on the SCS1-S. In Chapter 5, the Java bracket errors were found to be processed differently from Dutch and English subject-verb violations. This means that the results of the research in this thesis show only a limited relationship between language and programming. Based on the results of the current thesis and previous studies, I will now present two possible explanations for the limited role for natural language as found in the studies reported in Chapters 3 and 5.

First, it is possible that programming languages do resemble natural languages and are processed similarly, but that the specific type of programming language violation tested in Chapter 5 was just not grammatical in nature. If programming languages do indeed resemble natural languages, then the lack of predictive value for language skills in the study in Chapter 3 may have been due to teaching focus. Just like many modern-day programming courses, the undergraduate course taken by our participants did not teach the programming language explicitly (Hermans, 2020; Portnoff, 2018), and, therefore, focussed

on the first two levels of the PGK-hierarchy but not on level 3. Therefore, programming performance in the current course may have relied more on algorithmic/mathematical skills and less on language skills. It is possible that language skills would be more predictive of outcomes from a course that teaches programming languages more explicitly. Additionally, it is also possible that language skills are more predictive of the speed of the learning process, rather than end performance. This is supported by the findings of Prat et al. (2020), who measured both how well people performed at the end of the course, and how quickly they progressed throughout the course. They showed that language skills were most predictive of how quickly people learned. The study in Chapter 3 did not include a measure of learning speed. It is possible that the current language skills measured in Chapter 3 (grammar learning & vocabulary learning) would have been more predictive of such a measure.

However, it is also possible that we found little predictive value for language skills because programming languages are different from natural languages and are thus processed differently. Based on the results of Chapter 5 there are two possible options. First, it is possible that a programming language is processed in the same way as natural languages by the brain, but that its linguistic properties are different. For example, programming languages may contain more orthographic and fewer syntactic rules than natural languages. However, it is also possible that programming languages are not processed like a language at all, and that programming is an altogether different skill from language. This was suggested in a recent study by Ivanova et al. (2020), where they found that reading a programming language activated different brain areas to reading sentence problems. However, their results still showed that reading programs did require parts of the language system, and therefore could not exclude that a programming language is in part processed like a language. In order to determine which of these explanations is correct, future ERP studies will have to study a wider variety of linguistic elements of programming languages. More specific suggestions for future ERP studies can be found in the upcoming section “Suggestions for future studies”.

6.4 POTENTIAL IMPLICATIONS FOR EDUCATION

The studies reported in this thesis aimed to unravel the relationship between programming and various cognitive skills. The ultimate aim of this field of research is to guide programming education. Although more experimental research is still needed, this thesis does give some indicators which may be helpful for educators to consider.

First, in Chapter 3, results show a convincing role for logical reasoning skills when predicting programming skill at the end of a typical undergraduate course. Therefore, educators could use logical reasoning assessments to identify students who are not strong in this area and so may have difficulties during the course and, conversely, those with strong logical reasoning skills, who may do well. Additionally, it is possible that more explicit teaching of logical reasoning could improve programming skills. This would be in line with suggestions from educational researchers stating that teaching computational thinking prior to teaching programming would be beneficial for the learning process (e.g., Lu & Fletcher, 2009). Grover et al. (2019) showed that for middle schoolers, using non-programming activities to teach programming skills and concepts such as Boolean logic, loops and variables did indeed improve their programming performance in Scratch. Future training studies will have to confirm whether such teaching methods, including explicit teaching of logical reasoning, also improve programming performance in undergraduate students. In the meantime, it may already be worth considering unplugged teaching activities in educational programs.

Second, cognitive skills that predicted programming performance varied depending on the way in which programming skill was assessed. It is important for educators to note that not just the content, but also the format, of assignments and exams may favour students with certain skills over others. For example, assessments with formats like the SCS1, with strict limitations regarding the use of outside sources and with time pressure, may favour students with relatively good vocabulary learning and algebra skills. Therefore, it is important for educators to carefully consider the skills that they wish to measure, and to design their assessments accordingly.

Finally, at present, empirical results, both of the current thesis and of previous studies, are inconclusive when it comes to the relationship between programming and

natural language. Some studies suggest that language skills do predict programming performance, and that programming languages are processed similarly to natural languages in the brain (e.g., Floyd et al., 2017; Prat et al., 2020; Siegmund et al., 2014). Other studies show differences in processing between natural languages and programming languages and argue that these cognitive skills are distinct (e.g., Ivanova et al., 2020). Based on the current state of the findings in the literature and on the findings of the current thesis, I argue that language skills may play a role in programming, but that this role could be influenced by the way that programming is taught. In current educational systems programming is typically taught with a constructivist approach, in which students are expected to learn the programming language by using it, without much direct instruction (Hermans, 2020; Portnoff, 2018). This lack of explicit instruction may diminish the emphasis on language in programming courses. Some researchers have argued for more explicit programming language teaching (e.g., Hermans, 2020; Hermans & Aldewereld, 2017; Portnoff, 2018). The exact role of language in programming needs to be studied further to inform decisions regarding the best methods to teach programming languages.

6.5 SUGGESTIONS FOR FUTURE STUDIES

Based on the experimental chapters in the current thesis, in this section I will discuss four directions for future research: 1) Interactions between course content, test format, and cognitive skills, 2) Further disentangling the relationships between cognitive skills and programming, 3) Autistic traits and programming, and 4) How to take advantage of the newly developed ERP presentation method.

With all of these all suggestions for future research, it is important to note that programming languages vary widely in their syntax, rules and other properties. The current thesis studied two languages: Java and Processing. Consequently, based on the studies in this thesis, we are unable to determine the extent to which the results would generalise to other programming languages. Therefore, it will be important for future studies to extend the work reported here by investigating the same issues in a wide variety of programming languages.

6.5.1 Interactions between course content, test format, and cognitive skills

In the current thesis I make two important points about course content and test format. First, results suggest that course content influences performance on the SCS1-S. Second, results suggest that course content and test format may influence which cognitive skills are predictive of programming skill. In this section I give recommendations for future research on these two points.

The results presented in Chapter 2 show that the SCS1 items are able to successfully measure programming skills across different institutions and in different countries. However, the results also suggest that course content or format can influence scores on the SCS1 items to some extent: some items were easier while other items were more difficult for our population compared to the populations studied by Parker et al. (2016) and Xie, Davidson et al. (2019). To get a better idea of the relationship between course content and performance on the SCS1 test items, it would be beneficial to validate this test in a wider variety of courses. In addition, researchers should carefully consider *a priori* how the test relates to the curriculum used in the study.

Additionally, the results of the experiments in Chapter 3 suggest that course content and test format may influence which skills are predictive of programming performance at the end of the course. Specifically, algebra and vocabulary learning skills were found to be more predictive of performance on a programming test with a strict format than on a test with a more flexible format. In addition, we argued that the way in which programming syntax is taught may influence the importance of language skills in the learning process. Even in ERP experiments, like the one conducted in Chapter 5, it is possible that the way a programming language is processed also depends on the way it was taught. For example, if educational programs focus more explicitly on the syntax rules of the programming languages, violations may be processed more as syntactic violations. This hypothesis is supported by findings of natural language studies that show that the method of second language instruction can influence the size, shape and scalp distribution of ERP effects (Morgan-Short, 2012). Therefore, it is relevant for future studies to investigate whether different methods of instruction (e.g., explicit vs implicit syntax teaching), influence 1) the relationship between programming performance and cognitive skills, and 2) the brain

responses to specific syntactic elements. This will help to further clarify the ways in which instruction may or may not influence the role of different cognitive skills in programming.

6.5.2 Further disentangling the relationships between cognitive skills and programming

The results presented in the current thesis show that there are relationships between cognitive skills and acquiring programming skill. In Chapter 3 we showed these relationships within a correlational study design. This design is suitable and convenient for exploratory studies aiming to get an impression of which subskills are important for learning to program. Also, this design allows us to test hypotheses in a natural learning context. However, future studies are still necessary to determine the directions of these relationships. This could be achieved by testing the hypotheses from the current thesis with experimental training studies. These studies could include a control condition or manipulation of instruction methods. For example, studies can test whether more explicit teaching of logical reasoning skills improves programming performance.

Secondly, to further refine the PGK-hierarchy (Armoni, 2013; Perrenet et al., 2005), it would be interesting to test whether specific cognitive skills relate to specific sub-skills of programming. This could be done by splitting programming assessments into validated assessments of programming subskills (e.g., problem solving, programming syntax knowledge, etc.), corresponding to the three levels described in the model, and testing which cognitive skills correlate with which specific level.

Additionally, to further disentangle the relationship between programming and language skills, as mentioned previously, future studies should first investigate whether teaching formats with more emphasis on programming syntax show a stronger relationship with language-related cognitive skills. Future studies could also use the ERP methodology established in the current thesis (see “How to take advantage of the newly developed ERP presentation method” later in this chapter) to test more different elements in different programming languages to determine the extent to which different elements of programming languages have language-like properties. Specifically, based on the results of a recently published study by Ivanova et al. (2020), programming languages with a more

text-based structure, such as Python, are expected to show stronger linguistic resemblance to natural languages than graphical languages such as Scratch Junior.

Finally, future ERP studies could compare ERP effects across programmers with different levels of expertise to examine how proficiency in a programming language is reflected in the linguistic processing. Floyd et al.'s (2017) functional Magnetic Resonance Imaging (fMRI) study found that, for expert programmers, brain activity whilst reading code was virtually indistinguishable from brain activity during natural language processing, while for less experienced programmers brain activity during the different types of language processing was more distinct. These differences in expertise may also be visible in EEG signals of participants of varying levels of proficiency, where we may expect more proficient natural second language speakers to show ERPs that are more similar in timing and distribution to those in a native natural language (Kotz, 2009; Rossi et al., 2006; Weber-Fox & Neville, 1996). The study in Chapter 5 did not have a large enough sample size with enough variation in proficiency to test this hypothesis for programmers, but it is possible that the more proficient the programmer, the more the response to Java would resemble the response to Dutch or English.

6.5.3 Autistic traits and programming

The results of the study presented in Chapter 4 did not show any evidence for a relationship between autistic traits and programming skill. Therefore, if aiming to predict programming performance, it is probably more useful to focus on cognitive skills in future studies. However, it is possible that the lack of a relationship between autistic traits and programming in the current study was due to the limited reliability of the AQ subscales used in Chapter 4. Therefore, future similar studies could be conducted using a test that is more suitable to measure separate autistic traits. Specifically, in Chapter 4, the Subthreshold Autistic Traits Questionnaire (SATQ; Kanne et al., 2011) is suggested as a potentially appropriate measure. This test is more sensitive to different autistic traits than the AQ, allowing investigation of the relationship between individual autistic traits and programming skill more precisely. This may either show that there is a relationship between specific autistic traits and programming skills which was missed with the AQ, or it may confirm that there is indeed no relationship.

Secondly, it is possible that there is a relationship between autistic traits and interest (rather than aptitude) in programming. This could be studied by administering questionnaires asking students about their reasons for choosing a course at the start, and about course enjoyment at the end of programming courses. Here we may expect students who take the course as an elective to show higher autistic traits than students who take it as a compulsory course. We may also expect students with higher autistic traits to express more course enjoyment than students with lower autistic traits.

6.5.4 How to take advantage of the newly developed ERP presentation method

In Chapter 5, an ERP experiment was conducted to study responses to grammatical and ungrammatical snippets of a programming language, and to compare these responses to responses to grammatical violations in Dutch (first language) and English (second language). This required the adaptation of typical ERP stimulus presentation to the programming context. Specifically, the full length of the stimulus was presented in asterisks on the screen at once, after which the words or symbols were filled in by replacing the asterisks word by word (or per semantic entity for code). Stimuli for all three languages (Java, Dutch, and English) were presented in this way. The effects for Dutch and English were similar to the effects found in previous studies, thus suggesting that this new method of presentation for ERP stimuli elicits similar ERP effects as the traditional method, whilst presenting stimuli in a way that is more natural for programming languages. This method can also be considered for ERP studies with natural languages, as this way of presentation resembles the natural reading experience more closely than some currently used methodologies. To ensure that this method of presentation is reliable for eliciting the whole range of ERP effects, it would be advisable to replicate and extend the results found in Chapter 5 with various types of violations in different languages.

6.6 OVERALL CONCLUSION AND IMPACT

Based on the results of the studies in this thesis, I conclude that logical reasoning is the most reliable cognitive skill to predict programming performance following an introductory programming course. Additionally, algebra and vocabulary learning also seem to be of importance, but only when programming skill is measured with a test that is administered

under time pressure and that limits use of outside sources. Autistic traits did not predict programming skill, but I hypothesised that they may predict an interest in programming.

Based on the value of vocabulary learning in predicting programming skill, and the finding that programming language processing shows some similarities to of natural language processing, I suggest that natural language skill *does* play a role in programming acquisition. However, I suggest that this role is dependent on the way the programming language is taught.

The results of this thesis provide us with essential new knowledge on the cognitive nature of programming and facilitate future research into this area. Ultimately, the findings in this field will shape our understanding of programming as a skill.