DOI: 10.36315/2019v1end099
Education and New Developments 2019

TEACHER CHALLENGES AND CHOICE OF PROGRAMMING TOOLS FOR TEACHING K-12 TECHNOLOGY AND MATHEMATICS

Niklas Humble[1], Peter Mozelius[1], & Lisa Sällvin[2]
[1]Department of Computer and System Science, Mid Sweden University (Sweden)
[2]Department of Information Systems and Technology, Mid Sweden University (Sweden)

Abstract

A current ongoing process in many countries today is to implement programming in K-12 education. With this comes challenges for the involved teachers on how to best teach and integrate programming in their subjects. On the other hand, the introduction of programming could also open opportunities for programming as a new and improved way of learning and understanding technology and mathematics. For Swedish K-12 teachers this should be rapidly implemented, but without any concrete guidelines for how or for which tools to use. The aim of this study was to explore teachers' perceptions of learning and integrating programming in technology and mathematics, and their preferences of programming tools.
The overall research strategy was a case study approach, with two instances of a programming course as the case study units. In both course instances the main choice has been between block programming with Scratch, and textual programming with Python. Data was collected in a combination of submitted essays, programming assignments and workshop observations. Findings from a content analysis of the submitted essays have been compared to workshop observations, and to the analysis of programming assignments.
Results suggests that the main challenge in learning and integrating programming is the perceived time trouble. In parallel, many teachers highlight the potential benefits of renewing their teaching and learning sessions with programming-based problem solving. Considering the choice between block programming and textual programming several teachers brings up the idea of combining the two rather than excluding one of them. Furthermore, there seems to be minor differences in the preferences of programming tools between teachers with different subjects and different age groups of students. Finally, the most positive finding is the improved self confidence that many teachers show, when their own ability to manage programming in their classrooms increase after learning the fundamentals of programming.

Keywords: *Programming tools, block programming, textual programming, teachers professional development.*

1. Introduction

The integration of programming in K-12 education has been and is a worldwide trend (Balanskat & Engelhardt, 2015). The hopes in this integration are that it will help students develop computational thinking and skills useful for other subjects, such as self-efficacy, problem solving and reasoning skills in mathematics (Duncan & Bell, 2015; Psycharis & Kallia, 2017). However, with this integration challenges follows in a number of field, such as: access-limitations to computers and the internet, lack of students' motivation and technical skills, coping with time-issues of learning programming, and providing motivating high-quality professional development courses for the teachers that will lead this integration (Tundjungsari, 2016; Jawawi, Mamat, Ridzuan, Khatibsyarbini & Zaki, 2015; Mannila et al., 2014). On the other hand, if done successfully the integration of programming could bring with it opportunities, not only to computer science but to other subjects; for example, as an expressive tool for knowledge construction and supporting the users growth from passive consumer to active creator (Feurzeig, 2010; Papert, 1993:142; Tundjungsari, 2016; Psycharis & Kallia, 2017).

In mars 2017 the Swedish government approved a new curriculum for K-9 education to be implemented by the latest in the fall 2018; where digital competence and programming was introduced as interdisciplinary traits and with explicit mentions of programming, algorithms and problem solving in the subject of mathematics and controlling physical artefacts in the subject of technology (Heintz, Mannila, Nordén, Parnes & Regnell, 2017). Despite this rapid implementation, teachers perceive a lack of concreate guidelines and expectations of the integration (Humble & Mozelius, in press).

The aim of this study was to explore teachers' perceptions of learning and integrating programming in technology and mathematics, and their preferences of programming tools. The two research questions to answer were: 1) What are teachers' perceptions of main challenges and opportunities in learning and integrating programming in K-12 technology and mathematics? 2) Which preferences do K-12 teachers have in the choice of programming tools and how might this be related to subjects and student age?

2. Extended background

The very first attempts to program a computer were carried out in the 1840s, in the collaboration between Ada Lovelace and Charles Babbage (Kim & Toole, 1999). A hundred years later, Alan Turing built the foundation of modern computer programming by designing computational instructions that could be stored in an electronic memory (Morris & Jones, 1984). Modern computers can only execute programs that are written in machine language with binary instructions. Writing such programs is both difficult and time consuming for a human. To address this issue the first assembly language was developed at Cambridge University in the 1940s, with the idea of replacing binary instructions with mnemonics. Computer programs are sometimes still written in assembly languages, but more common is to write programs in high-level languages where an instruction can correspond to a large number of machine instructions. (Gaddis, 2011)

2.1. Textual programming in Python

Since the first high-level programming language FORTRAN was constructed by IBM in the 1950s, textual programming has been the standard mode of programming. In textual programming statements, selection, iteration and all other standard constructions are built up by combinations of textual code that later are syntactically checked by a compiler or an interpreter (Erwig & Meyer, 1995). Reading and analysing code can be hard in traditional programming languages such as FORTRAN, C and Perl. In the 1990s new programming languages like Java and Python strived to have a higher readability (Lutz, 2001).

Python is a multi-paradigm programming language designed by Guido Van Rossum in the late 1980s. The language is multi-paradigm in the sense that it fully supports imperative as well as object-oriented programming and implements several features that supports functional and aspect-oriented programming (Lutz, 2001; Van Rossum, 2007). Python has, like other dynamic and interpreted languages a high writability and unlike other dynamic and interpreted languages also a high readability.

High writability means that quite complex features such as file handling just needs a few lines of code, and high readability in the sense that Python code is relatively easy to understand and analyse. A combination that places Python on a bit higher level than other high-level languages, and makes Python an interesting candidate for textual programming in primary and secondary education.

2.2. Block programming in Scratch

Block programming, which is a type of visual programming, can be understood by taking a closer look at the hierarchy of visual aids for programming (Singh & Chignell, 1992). In this hierarchy visual programming is presented as a sub-group consisting of graphical interaction systems and visual language systems, containing in its turn of flow diagrams, icons, tables and forms (Singh & Chignell, 1992; Lavonen, Meisalo, Lattu & Sutinen, 2003). In other words, visual programming is a graphical or visual representation of the code in a program (Lavonen et al., 2003).

The development that led to block programming tools can be said to start with the programming language LISP-LOGO. The language was developed to have an easier syntax than its predecessor LISP, with graphical commands such as Forward and Right to make it easier to learn and use (Jehng & Chan, 1998). These visual elements have a connection to later developed programming tools and even purely visual programming languages where variables, functions, flow control, user interactions etc. are represented in graphical expressions or icons (Lavonen et al., 2003).

One of the most widespread visual programming tools in K-12 education is Scratch, developed at MIT Media Lab by the Lifelong Kindergarten research group, where the user can build their programs by putting together block of code in the same way as one might use LEGO bricks (Resnick et al., 2009; Brennan & Resnick, 2012). A strength mentioned about Scratch as an educational programming tool is its low threshold with a potential for larger and more complex projects (Shute, Sun & Asbell-Clarke, 2017; Resnick et al., 2009). The steadily growing Scratch-community that interact with and learn from each other through shared project and instructional videos is another valuable resource for both teachers and students (Brennan, Valverde, Prempeh, Roque & Chung, 2011; Brennan & Resnick, 2012). Further, Scratch combines programming with art, music and storytelling (Maloney, Resnick, Rusk, Silverman & Eastmond, 2010), which are some of the key factors to broadening the participation in programming and engineering among both girls and boys (Rusk, Resnick, Berg & Pezalla-Granlund, 2008).

3. Methodology

The study was conducted with a case study approach where a combination of data sources was used to generate a deeper understanding of the analysed phenomenon (Yin, 2009:4; Creswell, 2009; Remenyi, 2012; Denscombe, 2007). This study was based on two instances of a programming course on fundamental programming for teachers in K-12 mathematics and technology. The first course instance had 60 participating teachers and the second instance had 32. Data was collected through an essay assignment, workshop observations and code submissions during each of the course instances.

Content analysis was used to analyse the submitted essays and identifying topics of interests (Drisko & Maschi, 2015:25-26; Bryman, 2016:283). The codes where developed during the analysis process through inductive coding (Drisko & Maschi, 2015:43) and later compared to the workshop observations and analysis of the code submissions to find additional common or differential topics. In total the analysed material consists of 49 submitted essays (31 from the first course instance and 18 from the second), 16 workshop observations (from 8 campus meetings in each course instance), and 209 code submissions (146 from the first course instance from 51 different participants and 63 from the second course instance from 31 different participants).

4. Findings and discussion

In this section the findings from the analysed data is presented and discussed. The findings have been separated into two sub-sections where the first presents and discuss findings relating to the first research question and the second presents and discuss the findings related to the second research question.

4.1. Challenges and opportunities

Concerning the question of teachers' perceptions of main challenges and opportunities in learning and integrating programming in K-12 technology and mathematics there was a greater consensus in their answers about challenges than in opportunities. The vast majority of the essays mentioned that a challenge in learning and integrating programming is that programming takes a lot of time, commitment, continuity and discussion. Especially if you want to get to the level of proficiency were programming becomes useful for other subjects. About half of the essays also mentioned that a challenge in learning and integrating programming is that it is hard to learn, both that there are new tings to learn (concepts, structures, logic and so forth) and that much of the learning requires knowledge in the English language.

Regarding opportunities in learning and integrating programming the answers were not as coherent as about the obstacles. But what did stand out was that about a quarter of the essays mentioned that an opportunity with learning and integrating programming is that it is fun. About a quarter also mentioned that an opportunity in learning and integrating programming is that there is a lot of learning material available on the internet and in books that can be used to further develop one's knowledge.

4.2. Choice of programming tools

Concerning the question of which preferences K-12 teachers have in the choice of programming tools there is a general consensus across all teachers in their choice of programming tool to solve their own code assignments. The vast majority of code submissions was done with Python, regardless to what subject or student age the teacher taught. This is quite interesting since the majority of the essays mentioned that they perceived Scratch as easier and more fun. However, more than half of the essays also mentioned that they could see a greater potential for using Python in education.

Some smaller difference in the attitudes towards the two programming tools can be spotted in the essays, with an alignment towards technology and younger students for Scratch and an alignment towards mathematics and older students for Python. This is also supported by the workshop observations in the two courses where many teachers declare that they perceive Python as being more "programming" than Scratch and that it is more suitable for older students that needs freedom in their programming (for example in doing complex calculations in mathematics). However, many teachers also declare that they probably will start off with introducing Scratch to their students due to its lower threshold; and the fact that the event handling with Tkinter in Python is perceived as complicated. Other factors in the Python programming language that might contribute to the higher threshold could be functions and function calls, the handling of local variables and the fact that indentation affects semantics in the language.

5. Conclusions

Regarding the first research question the obvious finding is that teachers perceive a lack of time to learn and integrate programming properly. In the answering of the second research question findings indicate that teachers' choice of programming tool is dependent on the perceived suitability for the context in which it should be used. Authors' recommendation is that teachers learn a textual programming tool and a block programming tool in a context where they have time and opportunity to discuss learning and integration with their peers. This will probably better prepare them for the versatility of challenges and opportunities that programming will bring to the classroom.

6. Future research

This study compared textual programming to block programming in K-12 settings. An interesting next step would be to explore the potential of unplugged programming as a complementary tool in this context. Another idea would be to compare the results from this small case study with findings from other types of research in this area. What is similar and what is specific for the Swedish context?

References

Balanskat, A., & Engelhardt, K. (2015). *Computing our future: Computer programming and coding-Priorities, school curricula and initiatives across Europe*. European Schoolnet.

Brennan, K., & Resnick, M. (2012, April). New frameworks for studying and assessing the development of computational thinking. In *Proceedings of the 2012 annual meeting of the American Educational Research Association, Vancouver, Canada* (Vol. 1, p. 25).

Brennan, K., Valverde, A., Prempeh, J., Roque, R., & Chung, M. (2011, June). More than code: The significance of social interactions in young people's development as interactive media creators. In *EdMedia+ Innovate Learning* (pp. 2147-2156). Association for the Advancement of Computing in Education (AACE).

Bryman, A. (2016). *Social research methods*. Oxford university press.

Creswell, J. W. (2009). *Research Design, Qualitative, Quantitative and Mixed Methods Approaches*, Sage Publications Inc

Denscombe, M. (2007). *The good research guide for small-scale social projects*. *Maidenhead, England: McGraw Hill*.

Drisko, J. W., & Maschi, T. (2015). *Content analysis*. Pocket Guides to Social Work R.

Duncan, C., & Bell, T. (2015, November). A pilot computer science and programming course for primary school students. In *Proceedings of the Workshop in Primary and Secondary Computing Education* (pp. 39-48). ACM.

Erwig, M., & Meyer, B. (1995). Heterogeneous visual languages-integrating visual and textual programming. In *Proceedings of Symposium on Visual Languages* (pp. 318-325). IEEE.

Feurzeig, W. (2010). Toward a culture of creativity: A personal perspective on Logo's early years and ongoing potential. *International Journal of Computers for Mathematical Learning*, *15*(3), 257-265.

Gaddis, T. (2011). *Starting out with Python*. Addison-Wesley Publishing Company.

Heintz, F., Mannila, L., Nordén, L. Å., Parnes, P., & Regnell, B. (2017, November). Introducing programming and digital competence in Swedish K-9 education. In *International Conference on Informatics in Schools: Situation, Evolution, and Perspectives* (pp. 117-128). Springer, Cham.

Humble, N. & Mozelius, P. (in press). *Teacher perception of obstacles and opportunities in the integration of programming in K-12 settings*. In proceedings of EDULEARN 2019.

Jawawi, D. N., Mamat, R., Ridzuan, F., Khatibsyarbini, M., & Zaki, M. Z. M. (2015, May). Introducing computer programming to secondary school students using mobile robots. In *2015 10th Asian Control Conference (ASCC)* (pp. 1-6). IEEE.

Jehng, J. C. J., & Chan, T. W. (1998). Designing computer support for collaborative visual learning in the domain of computer programming. *Computers in human behavior*, *14*(3), 429-448.

Kim, E. E., & Toole, B. A. (1999). Ada and the first computer. *Scientific American*, *280*(5), 76-81.

Lavonen, J. M., Meisalo, V. P., Lattu, M., & Sutinen, E. (2003). Concretising the programming task: a case study in a secondary school. *Computers & Education*, *40*(2), 115-135.

Lutz, M. (2001). *Programming python*. " O'Reilly Media, Inc.".

Maloney, J., Resnick, M., Rusk, N., Silverman, B., & Eastmond, E. (2010). The scratch programming language and environment. *ACM Transactions on Computing Education (TOCE)*, *10*(4), 16.

Mannila, L., Dagiene, V., Demo, B., Grgurina, N., Mirolo, C., Rolandsson, L., & Settle, A. (2014, June). Computational thinking in K-9 education. In *Proceedings of the working group reports of the 2014 on innovation & technology in computer science education conference* (pp. 1-29). ACM.

Morris, F. L., & Jones, C. B. (1984). An early program proof by Alan Turing. *Annals of the History of Computing*, *6*(2), 139-143.

Papert, S. (1993). *The children's machine: Rethinking school in the age of the computer*. BasicBooks, 10 East 53rd St., New York, NY 10022-5299.

Psycharis, S., & Kallia, M. (2017). The effects of computer programming on high school students' reasoning skills and mathematical self-efficacy and problem solving. *Instructional Science*, *45*(5), 583-602.

Remenyi, D. (2012) *Case Study Research*, Reading: Academic Publishing International Limited

Resnick, M., Maloney, J., Monroy-Hernández, A., Rusk, N., Eastmond, E., Brennan, K., Millner, A., Rosenbaum, E., Silver, J., Silverman, B. & Kafai, Y. B. (2009). Scratch: Programming for all. *Commun. Acm*, *52*(11), 60-67.

Rusk, N., Resnick, M., Berg, R., & Pezalla-Granlund, M. (2008). New pathways into robotics: Strategies for broadening participation. *Journal of Science Education and Technology*, *17*(1), 59-69.

Singh, G., & Chignell, M. H. (1992). Components of the visual computer: a review of relevant technologies. *The Visual Computer*, *9*(3), 115-142.

Tundjungsari, V. (2016, February). E-learning model for teaching programming language for secondary school students in Indonesia. In *2016 13th International Conference on Remote Engineering and Virtual Instrumentation (REV)* (pp. 262-266). IEEE.

Van Rossum, G. (2007). Python Programming Language. In *USENIX annual technical conference* (Vol. 41, p. 36).

Yin, R. K. (2009). *Case study research: design and methods 4th ed.* SAGE Publications, Inc.