Chapter #11

THE INTRODUCTION OF PROGRAMMING IN K-12 TECHNOLOGY AND MATHEMATICS

Teacher choice of programming tools and their perceptions of challenges and opportunities

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
Many countries have started the process of involving programming in K-12 education. Most experts agree that this will be a positive change, but there are no concrete guidelines on which tools to use, and how to address challenges for the involved teachers. The aim of this study was to describe and analyse teachers' perceptions of integrating programming in technology and mathematics, and their view on programming tools. A case study strategy was used, with two versions of an introductory programming course as the case study units. For both course versions, technology and mathematics teachers taking the course could choose between textual programming in Python and block programming in Scratch. Data have been collected in a mix of submitted essays, programming solutions and researchers' observations. Findings show that a challenge in learning and integrating programming is the perceived time trouble, while an opportunity is that programming is perceived to be fun. Regarding the choice of tools, the majority of the teachers used Python themselves and mentioned that they could see a greater potential for it as a tool in education. However, many of them stated that they still will start off with Scratch, due to the lower threshold for novice programmers.

Keywords: programming tools, block programming, textual programming, K-12 education, teachers professional development.

1. INTRODUCTION

The integration of programming in K-12 settings is a worldwide and ongoing phenomenon (Balanskat & Engelhardt, 2015; Floyd, 2019; Dong et al., 2019). The expectations for this integration are that it should facilitate students' development of computational thinking and create skills that are useful for other school subjects, where some examples are self-efficacy, problem solving and reasoning skills in mathematics (Duncan & Bell, 2015; Psycharis & Kallia, 2017). However, this integration process also brings new challenges such as: access-limitations to computers and the internet, and lack of motivation and computer literacy.

Teachers also need to cope with time-issues in the learning of programming, and there is also the question of how motivating high-quality professional development courses for the teachers should be designed (Tundjungsari, 2016; Jawawi, Mamat, Ridzuan, Khatibsyarbini & Zaki, 2015; Mannila et al., 2014)? On the contrary, a successful integration of programming in K-12 schools might bring new positive opportunities, not only to computer science but also to other subjects. For example, programming might serve as an expressive tool for knowledge construction, and support students' growth from being passive consumers

to a new role as active producers (Feurzeig, 2010; Papert, 1993:142; Tundjungsari, 2016; Psycharis & Kallia, 2017).

In mars 2017 the Swedish government approved and presented a new curriculum for K-9 education that should be implemented no later than in the fall of 2018. A curriculum where digital competence and programming are introduced as interdisciplinary fields with explicit descriptions of programming and algorithm construction as new tools for problem solving in mathematics and technology (Heintz, Mannila, Nordén, Parnes & Regnell, 2017). Despite this rapid implementation plan, there was a lack of concreate guidelines on how programming should be involved in the mathematics and technology curricula (Government Offices of Sweden, 2017). An identified problem in some earlier studies is that many teachers find this stressful and do not know which tools to use (Humble & Mozelius, 2019; Mozelius & Hoff, 2019).

The aim of this study was to describe and analyse technology and mathematics teachers' perceptions of integrating programming, and their view on programming tools. The two important research questions to answer in the study 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. BACKGROUND

The first documented programming experiments were carried out as early as in the 1840s, in the famous collaboration between Ada Lovelace and Charles Babbage (Kim & Toole, 1999). About a hundred years later, Alan Turing constructed the foundation of modern computer programming by designing a model for computational instructions that were possible to store in electronic memories (Morris & Jones, 1984). Modern computational devices can only execute binary instructions that works for the specific processor. To write such programs directly as binary instructions is difficult, and time consuming for a human. To address this the very first assembly language was developed at the Cambridge University in the 1940s. This was done with the idea of replacing binary instructions with mnemonics that are easier to remember for a human brain.

Computer programs are for special purposes still written in assembly languages, but more common is to create the programs in various high-level languages. In a high-level language, a single instruction can correspond to a large number of machine instructions. (Gaddis, 2011) Modern programming languages such as Python and Java also have a design that make them platform independent and possible to run on different types of processors.

2.1. Textual programming in Python

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

Python can be described as a multi-paradigm programming language that was designed and developed with Guido Van Rossum as the main architect in the late 1980s. The language can be classified as multi-paradigm since it fully supports imperative and object-oriented programming, and also implements features that supports functional and aspect-oriented programming (Lutz, 2001; Van Rossum, 2007). Python has a high writability and unlike other dynamic and interpreted languages also a high readability.

High writability in this context means that complex techniques such as file handling or working with data collections just need a few lines of code. Python also has high readability in the sense that Python code is easy to read, understand and analyse. This combination places Python on a slightly higher level than other high-level languages such as Java and C#, which makes Python an interesting candidate for textual programming in K-12 education.

2.2. Block programming in Scratch

Block programming as a type of visual programming, can be understood by looking at the hierarchy of visual aids for programming (Singh & Chignell, 1992). A hierarchy that presents visual programming as a sub-group with different graphical interaction systems and visual language systems. Systems that in its turn consists of flow diagrams, icons, forms and tables (Singh & Chignell, 1992; Lavonen, Meisalo, Lattu & Sutinen, 2003). In a brief and broader definition, visual programming environments use graphical or visual representations of the code in a program (Lavonen et al., 2003).

The process that led to block programming environments can be said to start with the LISP-LOGO programming language. A language that was developed with the aim to offer a more understandable syntax than its predecessor LISP, with the use of graphical commands such as Forward and Right to make it easier to learn and use (Jehng & Chan, 1998). These visual elements have later been further developed in other programming tools where fundamental concepts as variables, functions, flow control and user interactions also have graphical representations (Lavonen et al., 2003).

The most widespread and well-known visual programming tool for a younger audience today is Scratch, a programming environment developed by the Lifelong Kindergarten research group at the MIT Media Lab. A tool where users can create programs by putting blocks of code together in the same way as building physical things with LEGO bricks (Resnick et al., 2009; Brennan & Resnick, 2012). A strength with Scratch, if seen as an educational programming tool, is its low starting threshold in combination with its potential for constructing larger and more complex projects (Shute, Sun & Asbell-Clarke, 2017; Resnick et al., 2009). In the continuously growing Scratch-community users can interact, and learn from each other by watching instructional videos and sharing projects (Brennan, Valverde, Prempeh, Roque & Chung, 2011; Brennan & Resnick, 2012). Furthermore, Scratch programming can be combined with art, music, storytelling and multi-media presentations (Maloney, Resnick, Rusk, Silverman & Eastmond, 2010). Options that are inclusive and have a potential to broaden the participation in programming and engineering among both girls and boys (Rusk, Resnick, Berg & Pezalla-Granlund, 2008).

3. METHODOLOGY

A case study approach was used as the overall research strategy with two instances of a programming course as the case units. The course, which is described in detail in the next section below, has a focus on fundamental programming for K-12 technology and mathematics. The first course instance was given during the autumn of 2018, and the second was given during the 2019 spring semester. As recommended for case studies, data was

collected from multiple data sources to gain a deeper understanding of the studied phenomenon (Yin, 2009:4; Creswell, 2009; Remenyi, 2012; Denscombe, 2007). The collection of data was in both case units conducted through workshop observations, essay assignments, and code submissions. A total of 60 K-12 teachers participated in the first course instance, and a total of 32 teachers participated in the second course instance.

To analyse and identifying topics of interests in the submitted essays a content analysis was carried out as described by Drisko and Maschi (2015:25-26), and Bryman (2016:283). Inductive coding was used in the analysis of the submitted essays (Drisko & Maschi, 2015:43) where the results later were compared to the analysis of the code submissions, and to the workshop observations. The analysed material consisted of: 49 essays, where 31 was gathered from the first instance of the course and 18 was gathered from the second instance of the course; 16 workshop observations (8 from each course instance in the form of campus meetings); and lastly, 209 code submissions, of which 146 code submissions were gathered from the first instance of the course, and 63 code submissions from the second instance of the course.

4. COURSE DESIGN

The programming courses referred to in this study are aimed at K-12 teachers in technology and mathematics. The teachers that took these courses had little or no previous experience in programming, and the use of it as a tool in their own teaching and learning activities. The programming courses are of a total of 7.5 ECTS each, given at a 25% study pace stretched over a twenty-week period and held in the Mid Sweden region. The courses consisted of both face-to-face meetings with lectures and workshops, and organised online learning in the virtual learning platform Moodle. The participating teachers were encouraged to share ideas and collaborate between meetings by creating local study groups.

The authors' experiences from previous similar courses are that only a small part of the participants have experience in the practise of programming. Since the aimed participants of the course not only need to learn how to program but also how to use it as a tool in their teaching and learning activities, the course was dived into five specific sections that meet these needs (Mozelius, 2018).

The first section 'Programming in school, why, what and how?' allowed for a more general discussion with the participants about the overall digitalisation of education and computational thinking (why). While also introducing the basic concepts of programming (what) and give support in the installation of the programming environments for Scratch and Python (how).

The second section of the courses, 'The fundamental building blocks of programming', focused on developing the participants' knowledge on the fundamental building blocks in programming, such as variables, constants, selection and iteration. To be able to build their own programs the participants needs to know and practise these fundamentals of programming.

The third section 'Didactics for Technology and Mathematics' is perhaps what distinguish these courses the most from other programming courses. The participants in the courses are active K-12 teachers in technology and mathematics and in this section the aim is to develop knowledge and skills in how programming can act as a tool for knowledge building in their subjects.

Since the previous experience of programming will differ also among the participating teachers' future students, the participants will have to, to some extent, themselves teach others the basic concepts of programming. The fourth section 'Didactics for programming

education' therefore focuses on developing the knowledge and the skills of the participants to teach others programming, within the frame of technology and mathematics education.

The fifth and last section of the courses 'Project work' draws on the participants' knowledge and skills developed in previous sections and allows them to put these together in creating their own programming material. This is an important part of the courses since the participants in not only there to learn about programming, but to do something explicit that they can bring to their daily work and keep on developing with, or for, their students.

5. FINDINGS AND DISCUSSION

This section presents and discusses the findings from the analysed data in three sub-sections. The first sub-section, 'Challenges and opportunities', presents and discusses findings that relate to the first research question, 'What are teachers' perceptions of main challenges and opportunities in learning and integrating programming in K-12 technology and mathematics?'. The second sub-section, 'Choice of programming tools', presents and discusses findings that relate to the second research question, 'Which preferences do K-12 teachers have in the choice of programming tools and how might this be related to subjects and student age?'. In the third and last sub-section a more general discussion concerning the findings in the study is presented and related to previous research.

5.1. Challenges and opportunities

Regarding perception of challenges and opportunities in the integration and learning of programming in K-12 mathematics and technology, there was a greater consensus among the teachers about challenges. A majority of the teachers mentioned time, commitment, continuity and discussion in their essays as challenges for learning and integrating programming. Especially if the goal of the integration is that programming becomes a useful tool for other subjects, since that require a higher level of proficiency in programming. Another challenge mentioned in about half of the essays is that it is hard to learn programming since it requires the learner to learn new concepts, structures, logic and so forth in a secondary langue (most material is in English).

Although not as coherent as the perception on challenges, a mentioned opportunity that stood out in the teachers' essays where that programming is perceived to be a fun activity, which is mentioned in about a quarter of the essays. The easy access to a lot of learning material on the internet and in books is mentioned as another opportunity, also in about a quarter of the essays. The availability of this material is perceived as an opportunity both for the learning and integration of programming in K-12 mathematics and technology but also as an opportunity for further knowledge development.

5.2. Choice of programming tools

Concerning teachers' choice of programming tool to solve their own programming assignments in the courses there was a general consensus among them. The programming tool used in the majority of the code submissions was Python, regardless to which student age or subject the teacher in question taught. Since a majority of the essays mentioned that they perceived Scratch to be the easier and more fun of the two alternatives this result is quite interesting. This could be understood in the light of that more than half of the essays also mentioned that as a tool for education and other subjects they could see a greater potential for the use of Python than for Scratch.

However, some alignments in the choice of programming tools can be found in the workshop observations of the two course instances. An alignment for programming with younger students and in technology can be found for Scratch; and an alignment for programming with older students and in mathematics can be found for Python. During the workshops many teachers also stated that Python is perceived to be more of a programming language than Scratch and allowed for more freedom to do, for example, complex calculations, which suited older students and the subject of mathematics. These alignments can also be spotted in some of the teacher essays.

Despite that many teachers see a greater potential for Python in the integration and learning of programming in K-12 mathematics and technology and made the choice to solve their own programming assignments in the language of Python; many of them mentions that they probably will start with introducing Scratch to their students. The reason for this is declared to be a lower threshold for learning Scratch and that Python is perceived as complicated, for example in the use of Tkinter as event handling. The use of functions and function calls, the handling of local variables and the effect on semantics by indentation could also be supporting factors to a higher threshold for the Python programming language.

5.3. General discussion

As mentioned in previous research (Tundjungsari, 2016; Jawawi et al., 2015; Mannila et al., 2014) the teachers in our study also brought up the issue of having enough time to both learn programming themselves; and to teach it to their students and integrate it in their teaching activities in mathematics and technology in a meaningful way. The other important issue that the teachers mentioned, that programming is hard to learn, can be viewed as part of the time issue. The harder something is to learn, and to integrate in teaching activities, the more time is needed.

On the other hand, many teachers also mentioned that programming is fun and that there is a lot of material available for learning programming and developing one's knowledge in the field. This could mean that the teachers see the same potential in using programming in their teaching activities as mentioned in previous research. That is, that the integration of programming could bring positive opportunities as a tool for knowledge construction and activating the students (Feurzeig, 2010; Papert, 1993:142; Tundjungsari, 2016; Psycharis & Kallia, 2017).

As for the teachers' choice of programming tool, their preferences could be understood by looking at the history of the programming tools. A strength in Scratch is that it is developed to have a lower threshold than traditional text programming languages (Shute, Sun & Asbell-Clarke, 2017; Resnick et al., 2009). So that the teachers perceived it as easier should not be that surprising. This could also explain why there was a slight alignment towards Scratch by technology teachers and teachers of younger students. An easier language might be more suitable if the outcome of the constructed program is most important, for example, controlling a robot. In the same way, the slight alignment for Python towards teaching older students and mathematics could be understood by the textual programming languages' freedom from predesigned blocks. This makes the activity of writing code more flexible and could also explain why some teachers perceived a greater potential for Python in educational context.

6. CONCLUSION

Although this study is quite limited in that it is located only in the Mid Sweden region, the obvious finding of the study is supported by previous research. That is, the perceived lack of time that teachers mention for learning programming and integrating it properly in teaching activities. While many teachers still mentioned that they perceived programming as a fun activity. Findings also indicates that there is a perceived suitability that influence teachers' choice of programming tool; that is, that they reflect on the context in which it is to be used and make decisions based on that. This can be spotted in that many teachers own choice of programming language in the course was Python, even though many of them stated that they still, probably, will start off with Scratch with their students since it has a lower threshold more suitable to their level.

The conclusion of this study is that teachers need to be given time and opportunity to learn and implement programming in their teaching activities. Preferable, teachers should learn both a block programming tool and a textual programming tool to draw on the opportunities in both languages. This could also serve as a better preparation for the challenges and opportunities, as well as versatilities, in bringing programming to the classroom. Lastly, the authors would also like to stress the importance on discussing challenges and opportunities about implementing programming with one's peers, something that K-12 teachers in mathematics and technology also needs to get the time and opportunity to do.

7. FUTURE RESEARCH

This study has analysed the use of textual programming and block programming as teaching and learning tools in K-12 education. Another programming tool is unplugged programming where computational thinking is taught and learnt without the use of computers (AlAmer, Al-Doweesh, Al-Khalifa & Al-Razgan, 2015; Aranda & Ferguson, 2018). A next natural step would be to compare the concept of unplugged programming to textual programming and block programming. To better address the identified challenges in the teachers' professional development, it would be interesting to develop a model for learning analytics that is tailor-made for the target group. A straight-forward and understandable model that can be useful in the continuous revision of courses for teachers' professional development.

REFERENCES

AlAmer, R. A., Al-Doweesh, W. A., Al-Khalifa, H. S., & Al-Razgan, M. S. (2015, October). Programming unplugged: bridging CS unplugged activities gap for learning key programming concepts. In *2015 Fifth International Conference on e-Learning (econf)* (pp. 97-103). IEEE.

Aranda, G., & Ferguson, J. P. (2018). Unplugged Programming: The future of teaching computational thinking. *Pedagogika, 68*(3), 279-292.

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). 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). 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.

Dong, Y., Cateté, V., Lytle, N., Isvik, A., Barnes, T., Jocius, R., ... & Andrews, A. (2019). Infusing Computing: Analyzing Teacher Programming Products in K-12 Computational Thinking Professional Development. In *Proceedings of the 2019 ACM Conference on Innovation and Technology in Computer Science Education* (pp. 278-284). New York, NY: ACM.

Drisko, J. W., & Maschi, T. (2015). *Content analysis*. *Pocket Guides to Social Work Research Methods*.Oxford, NY: Oxford University Press.

Duncan, C., & Bell, T. (2015). 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). New York, NY: 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.

Floyd, S. P. (2019). Historical High School Computer Science Curriculum and Current K-12 Initiatives. In *Proceedings of the 50th ACM Technical Symposium on Computer Science Education* (pp. 1287-1287). New York, NY: ACM.

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

Government Offices of Sweden / Regeringskansliet (2017). "Stärkt digital kompetens i läroplaner och kursplaner" ["Strengthened digital competence in curricula and syllabi"], Retrievedfrom http://www.regeringen.se/pressmeddelanden/2017/03/starkt-digital-kompetens-i-laroplaner-och-kursplaner/

Heintz, F., Mannila, L., Nordén, L. Å., Parnes, P., & Regnell, B. (2017). 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. (2019). Teacher perception of obstacles and opportunities in the integration of programming in K-12 settings. In *Proceedings of International Conference on Education and New Learning Technologies* (pp. 350-356). IATED.

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). 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). New York, NY: 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.

Mozelius, P. (2018). Teaching The Teachers to Program: On Course Design and Didactic Concepts. In *Proceedings of the 11th annual International Conference of Education, Research and Innovation* (Vol. 11, p. 8031-8037). IATED.

Mozelius, P., & Hoff, C. (2019). The Introduction of Programming in Compulsory School: A Multi-Stakeholder Perspective. In *Proceedings of the 12th annual International Conference of Education, Research and Innovation* (vol. 12, 2032-2040). IATED.

Papert, S. (1993). *The children's machine: Rethinking school in the age of the computer*. New York, NY: BasicBooks.

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.

Shute, V. J., Sun, C., & Asbell-Clarke, J. (2017). Demystifying computational thinking. *Educational Research Review*, *22*, 142-158.

Tundjungsari, V. (2016). E-learning model for teaching programming language for secondary school students in Indonesia. In *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.

ADDITIONAL READING

Lye, S. Y., & Koh, J. H. L. (2014). Review on teaching and learning of computational thinking through programming: What is next for K-12?. *Computers in Human Behavior*, *41*, 51-61.

Papert, S. (1980). *Mindstorms: Children, computers, and powerful ideas*. New York, NY: Basic Books, Inc..

ACKNOWLEDGEMENTS

Firstly, we would like to thank the course participants for their open-hearted way of sharing ideas and reflections in the essays. Your intelligent and constructive contributions were far beyond expectations. Secondly, we like to thank the arrangers of the END conference for the opportunity to present and share our ideas in a well-organized international forum.

AUTHORS' INFORMATION

Full name: Niklas Humble
Institutional affiliation: Department of Computer and System Science
Institutional address: Akademigatan 1 – Hus Q, 831 40 Östersund, Sweden
Email address: niklas.humble@miun.se
Short biographical sketch: PhD Student in Computer and System Science at Mid Sweden University

Full name: Peter Mozelius
Institutional affiliation: Department of Computer and System Science
Institutional address: Akademigatan 1 – Hus Q, 831 40 Östersund, Sweden
Email address: peter.mozelius@miun.se
Short biographical sketch: Senior Lecturer in Computer and System Science at Mid Sweden University

Full name: Lisa Sällvin
Institutional affiliation: Department of Information Systems and Technology
Institutional address: Holmgatan 10, 851 70 Sundsvall, Sweden
Email adress: lisa.sallvin@miun.se
Short biographical sketch: Lecturer in Information Systems and Technology at Mid Sweden University