This is the accepted manuscript (post-print version) of the article. The content in the accepted manuscript version is identical to the final published version, although typography and layout may differ.

How to cite this publication
Please cite the final published version:


Publication metadata

| Title: | The scope of autonomy when teaching computational thinking in primary school |
| Author(s): | Niklas Carlborg, Markus Tyrén, Carl Heath & Eva Eriksson |
| DOI/Link: | 10.1016/j.ijcci.2019.06.005 |
| Document version: | Accepted manuscript (post-print) |
| Document license: | CC BY-NC-ND |

General Rights
Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognize and abide by the legal requirements associated with these rights.

- Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
- You may not further distribute the material or use it for any profit-making activity or commercial gain
- You may freely distribute the URL identifying the publication in the public portal

If you believe that this document breaches copyright please contact us providing details, and we will remove access to the work immediately and investigate your claim.

If the document is published under a Creative Commons license, this applies instead of the general rights.
Accepted Manuscript

The scope of autonomy when teaching computational thinking in primary school

Niklas Carlborg, Markus Tyrén, Carl Heath, Eva Eriksson

PII: S2212-8689(18)30105-3
DOI: https://doi.org/10.1016/j.ijcci.2019.06.005
Reference: IJCCI 140


Received date: 31 October 2018
Revised date: 2 May 2019
Accepted date: 27 June 2019

Please cite this article as: N. Carlborg, M. Tyrén, C. Heath et al., The scope of autonomy when teaching computational thinking in primary school, International Journal of Child-Computer Interaction (2019), https://doi.org/10.1016/j.ijcci.2019.06.005

This is a PDF file of an unedited manuscript that has been accepted for publication. As a service to our customers we are providing this early version of the manuscript. The manuscript will undergo copyediting, typesetting, and review of the resulting proof before it is published in its final form. Please note that during the production process errors may be discovered which could affect the content, and all legal disclaimers that apply to the journal pertain.
The Scope of Autonomy when Teaching Computational Thinking in Primary School

Niklas Carlborg
RISE Interactive
Gothenburg, Sweden
niklas.carlborg@ri.se

Markus Tyrén
RISE Interactive
Gothenburg, Sweden
markus.tyren@ri.se

Carl Heath
RISE Interactive
Gothenburg, Sweden
carl.heath@ri.se

Eva Eriksson
Aarhus University & Chalmers University of Technology
Aarhus, Denmark & Gothenburg, Sweden
evae@cc.au.dk

Corresponding author: Eva Eriksson, email: evae@cc.au.dk
The Scope of Autonomy when Teaching Computational Thinking in Primary School

Abstract

During the 21st century, there has been an increased interest in the field of computational thinking as a consequence of the ever faster technical development. However, educating future generations in programming and computational thinking is not trivial. Many different platforms and teaching approaches can be used for this purpose. Inspired by the UK initiative with BBC microbit, this paper strives to identify what may be important to consider when designing teaching materials with the BBC microbit for training Swedish primary school learners’ computational thinking skills relating to mathematical and technical school subjects. This has been investigated in an iterative process, by conducting 21 workshops with the goal to support primary school teachers in developing BBC microbit teaching materials. The contribution of this paper is the Scope of autonomy model, which is based on the relation between learning potential, the risk of feeling overwhelmed, and the amount of choices provided in exercises. The model aim to support teachers in developing and appropriating material for teaching programming and computational thinking with individual progression in accordance with the new curriculum.

Author Keywords

Computational thinking; Autonomy; Education.

1. Introduction

Many countries face rapid changes in the primary and secondary school curriculum in order to incorporate computational thinking (CT) as part of the 21st-century skills and several countries are starting to formulate a curriculum for computational thinking [1]. The education systems differs widely, which makes it hard to compare and to learn from each other [2]. However, the general trend in the primary education is to introduce computing, often in the form of computational thinking (although this specific term is rarely used
explicit, but the ideas are included in some form), programming and digital competencies, while the trend in secondary education is to develop broader courses on computer science and its impact on society [2].

Sweden started to introduce programming in primary and secondary curriculum in 2009 [8]. The focus of the changes in the school curriculum is to enhance and emphasize the school’s role in strengthening the pupil's digital competences, varying from teaching stepwise instructions in the early year groups to encompass programming in later year groups. The changes are primarily concerned with introducing programming in multiple subjects throughout primary school, especially in technical and mathematical subjects. The students should able to work with digital texts, media and tools, be able to use and understand digital systems and services, and develop an understanding for the impact of digitalization on the individual and the society. There is further a focus on strengthen critical thinking skills, and be able to solve problems and realize ideas into action in a creative way using technology.

In the UK, the transition to more programming in school started in 2012 when The Royal Society recommended to reintroduce computer science in schools [5]. As part of the Make It Digital initiative in 2015, BBC together with Microsoft, Samsung and other partners, developed the BBC micro:bit for use in computer education [6]. Every year seven children (age 11-12) in the UK schools receive one of these small computers, or microcontrollers, that can be programmed and customized. Inspired by this initiative, the micro:bit was also introduced to pupils in grade 4 in Denmark in 2018 [7].

Since many of the new curriculum changes happen quite rapidly, there is a shortage in teachers with experience and knowledge within the computing area. The common challenges for many countries are both to educate teachers fast enough [3], to keep up the teachers with relevant knowledge and skills, and to develop a suitable progression paired with teaching material [2]. As there are limited resources to educate teachers in CT, new methods have to be developed to support teachers over time. The basic principles and ideas in computational thinking remain the same, but new tools and materials are introduced at a regular basis, and teachers need to be able to evaluate and select the right tool for their particular teaching situation [4].

Inspired by the UK, a project to explore how micro:bit can be adapted to aid in the Swedish curriculum transition was initiated, as a collaboration between university researchers and a national research institute. This paper reports from that study, and the contribution is the Scope of autonomy model for developing teaching materials with a technological platform for training computational thinking in primary schools. The
results is based on non-exhaustive empirical design research and is limited to the microbit platform and the Swedish school context.

2. Background

Around the world, the number of initiatives for raising interest in technology and computational thinking in formal and informal learning activities are rising. Many of these activities focus on the tools or a specific technology, e.g. 3D printing, programming robots, and Scratch [9]. Acknowledging that the technology knowledge in CT includes e.g. how to use a variety of technologies to solve a task using technical processes, methods, and tools, and learn and adapt to new technologies. In addition, there is need for an increased emphasis on content knowledge in CT, e.g. conceptualization, skills of abstraction, and understanding computational processes [10], and not on their manifestations in particular programming languages or technologies [11]. However, we know less about the learners difficulties in developing abstractions beyond any programming tool or language [12]. Teachers have a solid general pedagogical knowledge that can be applicable to all content domains and that can scale up the learners progression [30]. However, this need to be complemented with the subject-specific pedagogical practices and lesson content in computational thinking, such as e.g. model how to problem solve or think about a problem in iterative and incremental ways, or explain a solution to a problem in a series of steps [12]. Also, in order to practice teaching in computational skills, the pedagogic principle, the technical and content knowledge needs to be integrated with the unique teaching context.

Having a curriculum is important, but preparing teachers to teach the curriculum is critical [1], as well as to support teachers in actually aligning the curriculum into their unique contexts and circumstances. While pedagogical analog technologies, such as e.g. a pencil, traditionally offers some stability and transparency of function [13] digital technologies are often more complex in their nature as they can be used in many different ways [14] and its function is often not visible [15]. Introducing digital technologies in teaching therefore present teachers with new challenges, not only to acquire new skill sets and appropriate the technologies, but also to fit with their pedagogical belief and social context. Teachers adapt technologies to their own needs for use in a particular school environment, a phenomenon called ‘appropriation’ [16]. Appropriation relates to the situatedness and dynamics of the technology, where initial needs and requirements may change over time, while it may also change the environment. Appropriation can also
contribute to a sense of ownership as people use a technology in their own way, sometimes in ways not intended. Considering specific classroom contexts, while acknowledging that teaching with digital technology is a complex, ill-structured task show that teachers need new ways of accommodating this complexity [13]. This includes designing teaching resources that support the transformation of content into forms that are understandable to individual learners.

One key challenge in teaching computational thinking is to support the progression of knowledge [9]. A learning progression is a sequence of subskills that needs to be mastered to reach a curricular aim [17], to start with something simple and move on to something more advanced, applying the knowledge in computational thinking in new ways. There are a rising number of tools for how to assess computational thinking, although with various focus. The Computational Thinking Test (CTT), strive to assess to academic performance (Informatics, Mathematics, and Language) and learning analytics in a Code.org course, in order to distinguish between ‘computational regular thinkers’ and ‘computational top thinkers’ (i.e., those who spontaneously accelerated from the ‘block-based’ programming environment to the ‘text-based’ one) [18]. Another approach is the MIT-model, where the level of CT development with the learners is assessed based on that knowing the definition of a computational concept is not useful if one cannot put it to use in practice [19]. The model propose three strategies that can assist: Artifact-based interviews let learners engage in conversations about their projects and practices, using examples to guide the conversation forward. Another way is to provide a set of design scenarios for the learners that they engage in, giving them four different angles to relate to, critiquing, extending; debugging; and remixing. Finally, documentation is about learners developing a sense of reflection on their own creations and ideas.

However, we do not yet know much about the “typical” development from basic to complex forms of many of the 21st century skills such as computational thinking [20]. Therefore, we do not yet have a model for best practice for how to guide teachers in what to expect from learners at different levels of skill, and how they can progress. Available teaching examples are often considered either too trivial or too difficult. Thus, finding examples that opens for experiential learning with good interaction between the digital and the analogue world, and that supports progression is key [9].

One teaching strategy to tackle progression of learning is scaffolding instruction, which has been adopted as an educational approach to teaching software design and programming, e.g. [29, 32]. It originates from Lev Vygotsky’s sociocultural theory and the concept of the zone of proximal development (ZPD) [28], and is based on two foundations [30]. Firstly, the learner does not learn in isolation, rather, learning is influenced by social interactions that takes place in meaningful contexts. Secondly, is the ZDP, the distance between
what learners can do by themselves and the next learning that they can be helped to achieve with competent assistance [28]. There are six levels of scaffolding that the teacher operates with, namely to recruit motivation and attention to the task, reduce the degree of freedom in the task to manageable limits, maintain direction in the problem solving, mark critical features, control frustration, and demonstrate solutions when the learner can recognize them [30]. There are different types of scaffolding: one-to-one scaffolding where one teacher facilitating one learner and who offer the support directly to the learner’s needs, peer scaffolding refers to support being provided by fellow learners or between a more experienced learner supporting a weaker learner, in teaching software design pair programming is where two programmers share one keyboard and one monitor whilst cognitively collaborating to finish a programming task [31], and computer-based scaffolding. Scaffolding, together with modeling, fading, and coaching, are methods to promote the development of expertise in the theory cognitive apprenticeship, which refers to the learning-through-guided-experience on cognitive and meta-cognitive, rather than physical, skills and processes [33]. In cognitive apprenticeship, teachers need to identify the processes of the task and make them visible to the learners, situate abstract tasks in authentic contexts, so that learners understand the relevance, and finally vary the diversity of situations and articulate the common aspects so that learners can transfer what they learn. Scaffolding has many advantages, such as engaging the learner to not passively listen but is prompted to build on prior knowledge to build new knowledge, it can motivate students so that they want to learn, and minimize the level of frustration of the learner. However, there are also some disadvantages, such as using individualized scaffolds in a classroom setting is time-consuming. Scaffolding also requires that the teacher give up some of the control and allow the students to make errors, something that might be difficult for teachers to do [34]. Additionally, most often teaching materials and curriculum do not include examples of scaffolds or outlines of scaffolding methods that would be appropriate for the specific lesson content.

Self-Determination Theory (SDT), is a macro theory of human motivation that first and foremost states that motivation has more dimensions than simply the amount, and is highly relevant in education [21]. In SDT motivation is described as energizing behaviors and activities, and are divided into autonomous motivation and controlled motivation. Autonomous motivation consists of both intrinsic motivation and extrinsic motivations where people have identified with the value of a certain activity. Intrinsic motivation is where one is moved to act for the inherent satisfaction of doing the activity, not driven by any outcome separate from the activity itself. Whereas extrinsic motivation is described as an activity instrumental to reach an outcome separate from the activity at hand [22]. Controlled motivation on the other hand is described as extrinsic motivation such as rewards or punishments, or more introjected regulations such as approval.
motive, or avoidance of shame. In SDT, intrinsic motivation is sustained by satisfying the basic psychological needs for autonomy and competence. Autonomy is when learners act out of interest, for example, when they willingly devote time to their studies. Competence refer to the experience of behavior as effectively enacted, for instance when learners feel able to meet challenges of their studies. Both autonomy and competence needs to be satisfied in order to maintain intrinsic motivation [21]. Factors such as tangible rewards, threats, deadlines, directives, competition pressure and negative feedback undermines intrinsic motivation, while choice and opportunities for self-direction, as well as positive feedback has been shown to enhance intrinsic motivation [22]. Behaviors that are not intrinsically interesting to a person will require extrinsic motivation to be adopted. To make an extrinsic motivation more self-determined is the process of internalization and integration, when a learner truly understands the values of an activity, identifies with it and incorporates it with their sense of self. This can be achieved by addressing the basic psychological need of relatedness, by having the behavior valued by significant others to whom they trust and would like to feel connected. It is further argued that the need for competence has to be supported through challenges where the learners feel that they have the competence to succeed. However, to support internalization and integration to the extent that the regulation becomes autonomous, the basic psychological need of autonomy has to be supported by the environment in such a way that makes the learner feel free to explore new ideas and exercise new skills [22]. In an educational practice, intrinsic motivation and autonomous types of extrinsic motivation can relate positively to important academic outcomes. Classroom practices that support learners’ satisfaction of autonomy, competence, and relatedness are associated with both greater intrinsic motivation and autonomous types of extrinsic motivation [21].

Strategies for enhancing autonomy include providing choice and meaningful rationales for learning activities, acknowledging learners’ feelings about those topics, and minimizing pressure and control [21]. However, teaching practices do not occur in a vacuum. According to SDT, one explanation to why teachers choose not to use autonomy-supportive strategies in the classroom is that external pressures (for instance having to comply with the new curriculum) are placed on them [23]. The more that teachers’ satisfaction of autonomy is undermined, the less enthusiasm and creative energy they can bring to their teaching endeavors. Thus, if policy makers fail to consider the motivation of both teachers and learners, and instead rely on control and accountability, people involved in the learning process will suffer in motivation and learning outcomes [24]. So, this pose a challenge on how teachers can support progression in computational skills by enhancing learners autonomy while integrating new technology such as e.g. the micro:bit into their teaching to meet with the new curriculum. To help the teachers meet the programming requirements of the new curriculum changes, we investigate how teaching materials in computational
thinking that supports teachers pedagogical knowledge and unique contexts can be designed in order to enhance autonomy and scaffold the learners progression on an individual level.

3. Method

This paper is based on a study taking a Human-Centered design approach in order to investigate what factors are important concerning using micro:bit for teaching computational thinking in Swedish schools. Initially, information was gathered about how teachers and learners perceive the introduction of programming in school through related literature. Additionally, we familiarized us with the platform, micro:bit, and iteratively developed exercises to be practiced during a series of workshops in schools. Insights from past workshops were incorporated the iterative development of new material. Over a period of one year, 21 workshops in school environments and educational conferences in different parts of Sweden were performed. The workshops were led by two of the authors. The participating schools and teachers were recruited from the extended network of a national educational Maker-project.

While field-testing the teaching formats and materials, feedback from the testers and the process was collected for later evaluation. During the workshops, most of the data gathering took place almost exclusively through exit tickets, and additional formative feedback. Evaluation and analysis of data were carried out at the end of each prototyping cycle to assess to what extent the implementation was done as design implied. Insights were extracted using an inductive approach and used to prepare for the next cycle. At a later stage, personas of learners and teachers were created from the gathered data, and used in a journey map in an attempt to extract even more insights. In the final evaluation phase, all data was evaluated with affinity clustering. The result from the study is the Scope of autonomy model presented later in this paper.

3.1 The workshops

In order to identify what factors are important for teaching computational thinking with micro:bit in Swedish schools, a series of workshops were initiated, see Table 1 for overview. All the workshops were performed by the same two facilitators, and lasted for about one hour.
Table 1: Overview of the 21 workshops

<table>
<thead>
<tr>
<th>Location</th>
<th>Age</th>
<th>Participants</th>
<th>Activities</th>
<th>Insights</th>
</tr>
</thead>
<tbody>
<tr>
<td>Teacher conference</td>
<td>Adult</td>
<td>100 Swedish teachers</td>
<td>First encounter with micro:bit workshops, for teachers.</td>
<td>Found useful to exercise examples</td>
</tr>
<tr>
<td>School</td>
<td>8-10 y</td>
<td>30 pupils, 1 teacher, a few parents</td>
<td>Got to know the context and students through a modified version of the previous workshop</td>
<td>Found exercise examples trends were observed in class Too much freedom lead to friction Hard to convey concept of variables</td>
</tr>
<tr>
<td>Lab</td>
<td>8-10 y</td>
<td>30 pupils, 1 teacher, a few parents</td>
<td>More guided presentation than last time. Tried imitation exercises.</td>
<td>Imagination is easy Coming up with own ideas is hard Hard for teacher to help all individually</td>
</tr>
<tr>
<td>Lab</td>
<td>9th grade</td>
<td>2 pupils</td>
<td>Tried to guide students more with a co-coding session at the beginning.</td>
<td>Explain that computers are stupid Free exploration fun but gets frustrating Helpful to see examples</td>
</tr>
<tr>
<td>School</td>
<td>4th-5th grade</td>
<td>20 pupils, 1 observing teacher</td>
<td>First student workshop with only iPads.</td>
<td>App-store passwords required Internet connection frustrating Pairing mode bug Text based instructions ineffective</td>
</tr>
<tr>
<td>3x same School</td>
<td>4th-5th grade, Adult</td>
<td>30 pupils, 1 observing teacher Teacher workshop 15 participants</td>
<td>Two mad workshops with a lot of troubleshooting. One computer workshop.</td>
<td>Bluetooth bug resolved through flashing Need for adapting exercises to students skill</td>
</tr>
<tr>
<td>6x same School</td>
<td>4th-6th grade</td>
<td>17 pupil</td>
<td>Shorter student workshops with computers.</td>
<td>Positive with freedom to customize Hard to find the right blocks Language can be a barrier, as there were complex words</td>
</tr>
<tr>
<td>School</td>
<td>mixed</td>
<td>30 pupils, 1 observing teacher</td>
<td>First student workshop without previous knowledge in Scratch.</td>
<td>Mismatch in difficulty and knowledge led to frustration</td>
</tr>
</tbody>
</table>
The exercises typically varied from analogue programming games to more advanced levels covering the basic concepts of programming: algorithms, loops, randomness, logic, variables, and finally debugging. Some of these exercises were developed based on material from the related Swedish project Makerskola [9]. The platform was micro:bit programmable microcontroller, and was provided by the workshop organizers. In addition, video material from the related Swedish project Makermovies.se was used [35].

The micro:bit was used together with the Microsoft MakeCode micro:bit editor, which is a free to use online and is a JavaScript/Blocks editor for programming the micro:bit. This means that it runs in the web browser and hence is cross platform compatible, both on different web browsers but also across different operating systems, such as OSX, Windows, iOS and Android. This also implies that an internet connection is required for using the editor. A workshop script was created for each workshop, outlining aims, methods and insights, see example in Table 2.
Table 2. Example of a workshop script

<table>
<thead>
<tr>
<th>Workshop</th>
<th>Nr 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>Analog first workshop in a series of four. Familiarizing with sequence and loop.</td>
</tr>
<tr>
<td>Time</td>
<td>Feb 23, at 9.40-11.00</td>
</tr>
<tr>
<td>Location</td>
<td>Classroom</td>
</tr>
<tr>
<td>Participants</td>
<td>17 students, 2 facilitators, 1 observing teacher</td>
</tr>
<tr>
<td>Age</td>
<td>4th graders</td>
</tr>
<tr>
<td>Aims</td>
<td>Test if analog workshop works as introduction to programming for complete beginners. Test a physical game as method for introduce sequencing. Test if a physical game can work as a common experience to relate back to in future more detailed workshops. See how fluent the transitions between different sections and workshops are, is there a common theme?</td>
</tr>
<tr>
<td>Methods</td>
<td>Exit tickets from students, and video observations of the facilitators</td>
</tr>
<tr>
<td>Insights</td>
<td>Teacher better execute the code. Students like to present unique solutions</td>
</tr>
</tbody>
</table>

The digital workshops included one micro:bit per pair of pupils, and the exercise was introduced by a co-coding approach. Co-coding is when the teacher stands in front of the class, sharing their computer screen on a projector and solving programming exercises in dialogue with the class. This form is a useful hybrid between pure presentations and having pupils solve exercises on their own. Presentations were considered beneficial for introducing new knowledge, but it was undesirable to put the pupils in a rather passive seat. On the contrary, working individually with exercises, was considered to be more active but not ideal for introducing novel information. Co-coding hence evolved as a middle path between these two approaches. The workshops and exercises are further described in [25,26].

3.2 Process

Based on affinity clustering of all insights and observations from the workshop interventions, an emerging pattern was discovered relating to issues with balancing the level of freedom given to learners. Some observations revealed that learners did enjoy being able to explore and do things in their own way, but at the same time we saw some issues when learners were given too much freedom as this could lead to confusion and a feeling of being overwhelmed.

We were inspired by the idea of design spaces, and imagined a space where moving to a new point was equivalent to making a design decision. We mainly saw ourselves as facilitators, scaffolding and guiding the learners through some of these decisions towards the goal. However, at other times the learners were
allowed to be free enough to make decisions for themselves, as we saw making active decisions as an important part of the learning process. Viewing learning in this way also made it interesting to think about the many situations where there are more than one correct way to complete an exercise. In these cases we as facilitators have given the learners enough freedom to choose any out of the sometimes infinite possible designs that could satisfy the exercise.

We realized that we as facilitators never really had reflected on the amount of freedom we had given learners through various exercises. Sometimes we gave them no freedom, like when telling the learners exactly what to do and to simply ask them to imitate. Other times we had given them full freedom, by asking them to come up with their own ideas for a program they would like to build. We realized that we wanted to make an active decision in the creation of an exercise, about how much freedom was given to the learner within that exercise. Consequently the design decisions that would not be available to the learner would have to be made by the teacher.

The freedom that is given to a learner in their design decisions, we came to call the amount of autonomy that they were given. As we saw these levels of autonomy as increasingly larger areas to explore within the design space, we chose to call them scopes of autonomy. We chose to visualize these scopes as circular zones that would get increasingly larger with more autonomy, representing that the learner’s autonomy then covers more decisions (see Figure 1). This representation was also chosen to indicate that each new level of autonomy also includes the previous levels. The area outside of the learners scope of autonomy was chosen to represent the choices to be made by the teacher.

In parallel with this we tried to identify the different levels of autonomy present in working with the micro:bit, starting with the least amount, which is no autonomy at all. This means that the learners were not given any choices. This was discussed to be true for traditional lectures as well as pure imitation exercises. From our experience this was not desirable, and we found it more effective to teach programming through exercises that at least allowed for some level of autonomy, even for the very beginners.

4. The Scope of Autonomy Model

The scope of autonomy model aims to explain observed behaviors and phenomena regarding primary school pupils encounter with programming the micro:bit. The model consists of five levels of autonomy, and is
intended to be used as a tool when creating exercises, to help teachers bring awareness to the amount of choice expected of learners within exercises.

Figure 1: Scope of autonomy model

The scope of autonomy model illustrate and bring awareness to the distribution of autonomy between learners and teachers in relation to single given micro:bit exercise. The model is based on the premise that completing an exercise involves making a set of choices. The dark area in the center of the model in Figure 1 represents the choices made available to the learner, this is called the learners scope of autonomy. The area surrounding it represents the choices made by the teacher. A larger scope of autonomy hence implies fewer choices to be made by the teacher.

The model suggests that the larger the scope of autonomy becomes, the higher the learner runs a risk of feeling overwhelmed. The model also suggests that the larger the scope of autonomy becomes, the higher the potential is for the learner to improve their independent problem solving skills. Hence, there is a balancing act in the creation of an exercise, in that it provides the learners with enough choices to develop their independent problem solving skills, yet without exposing them to too many choices so that they feel overwhelmed. The model does not make any claims on how to determine what the appropriate level of autonomy is for any learner.

Five levels of autonomy were identified for working with the micro:bit. These are presented in a radial fashion to be compatible with the scope of autonomy model, see Figure 2. Any micro:bit exercise can be mapped onto a scope of autonomy disc onto this model. The more of these levels that are encompassed by an exercise the bigger scope of autonomy it has.
**Customization**

The first level of autonomy that was identified in relation to micro:bit exercises, was allowing learners to make smaller customization to a predefined design. In the case of a simple “hello world” program, this could mean allowing the learner to customize the text string to something else than “hello world”. Hence a customization is not something that alters the behavior of a design, but rather allows the learner to locally modify specific point of interest that have been selected by the person designing the exercise. From Figure 3 it is possible to see that a majority of the choices that have to be made regarding the exercise still has to be made by the teachers when an exercise has this scope of autonomy.

Example illustrating how an exercise with a LED-tiara using the micro:bit and a block editor could be defined at customization level: Based on the Scope of Autonomy model:

**Customization** - Starting from a finished tiara and a complete program the learner is invited to try to change specific values such as animation color or animation speed.
Solution Procedure

The next level of autonomy that was identified is related to the solution procedure of an exercise. This level of autonomy relates to what subparts of a solution to tackle in what order. When this level of autonomy lies within the learner’s scope, the learner is free to choose the order in which to create the solution. When the solution procedure does not lie within the scope of the learner’s autonomy, the learners are asked to follow a stepwise procedure instructed by the teacher. In the case of creating an animation, this could be the difference in starting with drawing the desired animation and then figuring out the best timing between frames, or doing it the other way around. This way there is more freedom for the learner to make choices about the way they solve an exercise but the target design is still chosen by the teacher, as illustrated by Figure 4.

Example illustrating how an exercise with a LED-tiara using the micro:bit and a block editor could be defined at solution procedure level based on the Scope of Autonomy model:
**Solution procedure** - The learners have to puzzle the program on their own, given a picture showing how the finished program will look. Therefore, they know what to build, but they themselves have to find the correct blocks and put them together in any order. The aim of the exercise is still that the learner’s code should look like the “target image” the teacher gave them. (with the exception of animation color and animation speed then, when these were optional).

**Design**

The third level that was identified is concerned with the design that a learner makes to complete an exercise. This is a rather interesting level of autonomy, as setting the learners scope of autonomy to encompass this level means that the teacher no longer knows what the final design will look like, as it is up to the learner. In contrast to a scope of autonomy that only encompasses the level of customization, a scope that encompasses the level of design allows for completely new design solutions to an exercise, and not only the modification of predetermined placeholders. The learner is however still restricted by the teachers choice of blocks to be used with this scope of autonomy, as illustrated by Figure 5.

![Design Level Scope](image)

**Figure 5: Scope set at Design Level**

Example illustrating how an exercise with a LED-tiara using the micro:bit and a block editor could be defined at design level based on the Scope of Autonomy model:

**Design** - Here, the teacher can no longer provide a complete picture to show how the program should look, but it is about the teacher limiting learners’ choice by specifying which blocks are available to the learner. This can be based on a micro:bit project where all blocks that may be used are scattered on the workspace, such as free, some pieces of puzzles. So the teacher says “here’s what you have to work with, try to build a program of these blocks that animates your tiara”.
Block Selection

The fourth level of autonomy that was identified is related to block selection, see Figure 6. Blocks are the building pieces that are used to create a design. When this level is not encompassed in the learners scope of autonomy in an exercise, it means that the teacher has predetermined what blocks the learner should use to create his or her design.

Figure 6: Scope set at Block selection level

This level of autonomy has two parts. The first of which relates to the type of blocks and the second one relates to the quantity of blocks. For instance, the teacher can give an exercise where the learners are asked to create an animation on the micro:bit using any number of blocks of the “loops” and “show LED” variety. This way the types of blocks are chosen by the teacher but the learner is free to choose the number of blocks. Another exercise could be to make a step counter using only four blocks in total. Here the teacher decides the number of blocks but their type are free to be chosen by the learner. These two examples illustrate that an exercise can be created in ways where the learners scope of autonomy only encompasses one of these two block selection levels. Likewise none of them can be encompassed, which means that the teacher decides exactly what blocks ought be used. And lastly when both of them are encompassed by the learner’s scope of autonomy, it means that the learner is free to create a design out of any block type or quantity, as long as it satisfies the assignment.

Example illustrating how an exercise with a LED-tiara using the micro:bit and a block editor could be defined at block selection level based on the Scope of Autonomy model:

*Block selection*: here the teacher can assume the above instructions but add “Now you can use how many blocks you want from the already mentioned blocks”, or “now you can use which blocks you want of all blocks in the entire editor”.
Assignment

Lastly, an autonomy level was identified relating to the very assignment itself. This relates to decisions about the topic and aims of an exercise. In the case where this level is not encompassed by the learners scope of autonomy, the teacher defines what the learner ought to do in order to complete the exercise. When this level is encompassed by the learners scope of autonomy, learners make the decisions about what the exercise is going to be about. These kind of exercises might initially only be associated with higher educational projects, it is however just as true for exercises where the teacher tells learners to create whatever they want. Having to create your own assignment is basically the same as having to come up with an original project idea. To be able to handle this level of autonomy, learners are recommended to have reached a rather high level of experience and be comfortable with making various decisions, or they might run the risk of feeling overwhelmed. As illustrated by Figure 7, this level of autonomy does not require the teacher to make any decisions.

![Figure 7: Scope set at Assignment level.](image)

Example illustrating how an exercise with a LED-tiara using the micro:bit and a block editor could be defined at assignment level based on the Scope of Autonomy model:

**Assignment** - Here it may be that the learner will take the hardware for the tiara and find a whole new use for it, maybe the learner wants to write a program where the tiara now instead of showing an animation can be used to show how mood states, or maybe the learner want to fix it on a bike and blink if the bike is moving?
5. Discussion

Learners tend to learn better and are more creative when intrinsically motivated, particularly on tasks requiring conceptual understanding. According to SDT, the way in which teachers introduce learning tasks impacts learners’ satisfaction of the basic psychological needs for autonomy and competence, thereby having an impact on the learners intrinsic motivation [21]. When creating an exercise for working with micro:bit there seems to be some importance in making a conscious decision regarding its scope of autonomy. The exercise should match the learner’s current level as good as possible, and provide them with an opportunity to improve their independent problem solving skills without being too overwhelmed. This is in line with staying within the zone of proximal development [28]. However, the scope of autonomy model differs from scaffolding instruction in that instead of having the teacher simplify the task by reducing the acts required to reach a solution [30], the teacher instead allows learners to perform some level of customization for the task, based on that choice and opportunity for self-direction has been shown to enhance intrinsic motivation [22]. Five levels of autonomy were identified for working with the micro:bit, namely Customization, Solution procedure, Design, Block selection, and Assignment. This set might very well need to be changed or be expanded with more levels. It can for instance be discussed if there are more layers outside of the one called assignment. An remotely autonomous assignment with the micro:bit still is an exercise limited to the hardware micro:bit, it is reasonable to say that there could be a level of hardware and maybe editor outside of the existing levels.

Our observations show that it is beneficial to provide some scope of autonomy in exercises, as learners tend to become pacified when they did not have any way to affect the outcome of the exercises. When working with micro:bit, always providing some level of autonomy would translate into always allowing learners to perform some level of customization in any exercise they are involved in. This relates to Papert’s findings that learners exposed to environments with creative freedom have a higher tendency to learn the necessary knowledge in order to realize their ideas [27].

Learners are on different knowledge levels, and require different scopes of autonomy in their exercises, which is a challenge when working with a whole class, which is also one of the main problems in practicing scaffolding where one teacher tailors the support directly to the learner’s needs. As it is hard to give every learner individually adapted exercises with scopes of autonomy that matches their individual needs, teachers instead need to find exercises that can be given to the entire class. This means that a class with a wide span in individual progress, a single exercise can be perceived as anything from boring to useful to overwhelming. This can be tackled in various ways. One way is to try to minimize the skill span in the class,
and unify the individual levels. A second approach is to expand a single exercise’s scope of autonomy to be more flexible. In this way, one single exercise can be given to an entire class, but different modifications or customizations can be used to increase or decrease the scope of autonomy for the exercises, to better adapt it to individual learners. To expand the scope of autonomy relates to increasing complexity and diversity in sequencing [33].

In Self-Determination Theory, autonomy is both a basic psychological need as well as the causality orientation described as “acting out of interest” [24]. This shows that the word autonomy can be a bit arbitrary, and hence might differ slightly from the way it is used throughout this paper. The way it is used in this paper is more relating to the amount of choices available to a learner in a certain exercise situation. This positions our use of autonomy more as a term relating to the way of learners’ activities, the exercises, can be designed to satisfactorily support both the basic psychological needs of autonomy as well as competence. Both relates to providing the learner with enough choices to feel autonomous, yet not provide them with too many choices, as this runs the risk of having them feel that they lack the competence to succeed, which is how the basic need of competence is defined according to SDT [24].

What it means that a student is on a certain level of progress is not really understood at this point and we do not know how to measure it. So far, the activity of adapting exercises scope of autonomy to fit the student’s levels has been relying on the teacher’s ability to customize exercises on the fly when helping students. Co-coding sessions were found helpful for trying to estimate how progressed students were in programming. This method is however limited by the fact that the most knowledgeable students are most probable to answer any question the teacher might raise. This leads to the risk of having a few advanced students answering a teacher’s co-coding questions, while others sit quietly without understanding. Yet it can be considered to be helpful that these students then at least get the opportunity to see how it is supposed to be done.

5.1 Workshops

By experimenting with different methods of teaching, co-coding was found useful as it both allows the teacher to probe learners current skill level with questions and rate the discussions in the class, and also to adapt the level of guidance. This stands in contrast to the method modeling as used in cognitive apprenticeship, where the teacher performs a task so students can observe [33]. Co-coding can be an effective method to teach new concepts within the context of an exercise, rather than as a separate presentation. Co-coding should be practiced often as it provides an including activity for the whole class,
prepares learners by carefully giving them new tools to work with and also lets the teacher get an overview of the general understanding towards the subject in the class. The workshops covered the basic concepts of programming: algorithms, loops, randomness, logic, variables, and debugging. The concepts were introduced and practiced in parallel to each other, rather than separately, in series. This was due to the difficulty to create interesting exercises based on single programming concepts, why combinations of multiple programming concepts were practiced.

In hindsight we realize we might had progressed too fast in some of the workshops, the learners were not ready and got stuck and confused. As an example, we observed a pair of learners that managed to complete the rock-paper-scissor game fast, and where the other learner looked to them and eagerly wanted to do the same. Soon everyone were trying to code their own so that they could play against each other. This led to that many learners tended to focus on the result rather than the process of learning; they basically wanted us to make the game for them so they could play with their friends. This kind of behavior has both positive and negative impacts, the good part being that they get motivated and suddenly have a purpose to learn how to code a micro:bit. On the other hand, they often get impatient and try to skip a few steps in the learning process to reach the end result fast, which is not always desired from a learning perspective.

During our early workshops, we encountered trends as a social phenomenon among the participating learners. It began when a pair of learners programmed a funny game and started playing; others took notice and wanted to join in. Soon the whole workshop had turned focus towards this game and everybody wanted to make it. During this period, the learners were highly involved, motivated and interacted with each other to a great degree, even helped one another. This behavior has been seen multiple times throughout the study, and though it is positive, it seems hard to predict. However, we believe there is potential to capitalize on if the teacher can seize these moments when trends erupt and adapt the teaching to these opportunities.

5.2 Generalization

Many different technologies and development platforms are available and new ones are constantly being developed. Choosing what technological platform to invest in might be a very relevant question to educators, however this is not within the scope of this paper. This study uses the micro:bit, as a given educational platform as the client stakeholders considered it to be affordable, already well spread in the
UK, and having a lot of potential with its many onboard sensors. Furthermore there are multiple different editors available for working with the micro:bit, thus in this study we only consider the Microsoft MakeCode BBC micro:bit editor, and specifically the block editor part of it.

Throughout the study, we have always related to the basic concepts of programming in our design process. These concepts permeate through all programming teaching activity and promote computational thinking and problem solving. Even though our model was made for micro:bit specifically, there are many similarities that makes it versatile. Most of the platforms used to teach programming for primary school uses a block type editor, just like micro:bit. Other platforms can be better suited to teach some of the concepts in a vacuum, which is not the case for micro:bit, but the workflow still applies as it promotes a progress that aims to keep learners at a balanced level of stimulation when learning. Since the underlying theories on the behavior of learners and how they learn is supported in the model, we believe the model can be claimed to have a wider applicability even though data to support the long term effects of using such model is non-existent in this paper.

A majority of the participants in the workshops had a positive bias towards curriculum changes and digitalization. Therefore, there is no claim that the teachers represent an accurate image of the average mindset and motivation a teacher might have regarding programming.

The Scope of Autonomy model suggested in this paper is merely an attempt to explain behaviors and phenomena that were observed throughout the study. A next step would be to see if it is possible to design a study that can validate or falsify the suggested correlation between the amount of choices presented in an exercise, and show the learners potential to develop problem solving skills and feelings of being overwhelmed. The effect of such model need to be measured in a live setting where conclusions for large sample sizes can be made in order to truly gain any significance.

### 6. Conclusion

Through an iterative process, a total of 21 workshop interventions were conducted to collect qualitative data and gain insights about Swedish primary school learners and their teachers interactions with micro:bit teaching materials. The results from the study suggests that it is important for teachers to consider the
amount of free choices that are given to learners in any given exercise, when designing teaching materials
with the micro:bit for training primary school pupils computational thinking skills. The contribution of this
paper is the scope of autonomy model, which is based on observed relationships between learning
potential, the risk of feeling overwhelmed and the amount of choices provided in exercises. More work is
needed to validate the model. The scope of autonomy model is a result from investigating what is
important to consider when designing teaching materials with the micro:bit for training Swedish primary
school pupils computational thinking skills, and can be a support for teachers to meet the programming
requirements of the new curriculum changes.

7. Acknowledgements

We thank all the pupils, teachers, and school leaders involved in this research. The research is funded by
Vinnova grant nr 2015-02319.

8. References

Inroads 1, 1 (March 2010), 61-66. DOI: http://dx.doi.org/10.1145/1721933.1721953

computational thinking, computer science and computing in K-12 education. Frontiers in education


Digital Competence in Swedish K–9 Education. 10th International Conference on Informatics in
Schools: Situation, Evolution, and Perspective (ISSEP).


[6] About the BBC. BBC Head of BBC Learning, Sinead Rocks. BBBC micro:bit, groundbreaking initiative
to inspire digital creativity and develop a new generation of tech pioneers.
http://www.bbc.co.uk/mediacentre/mediapacks/microbit/, 2016. [Online; accessed 2 March-
2018].
[8] Regeringskansliet. Stärkt digital kompetens i läroplaner och kursplaner - regeringen.se
   Vol. 16 s. 9-15. DOI: 10.1016/j.ijcci.2017.10.001
   Computational Thinking Curriculum Framework: Implications for Teacher Knowledge. Educational
   Technology & Society, 19, 47-57.
   Contemporary Issues in Technology and Teacher Education, 9(1).
   Schuster.
   English: A Theoretical Framework for Research on Learning to Teach. American Journal of
   https://www.jstor.org/stable/10.1086/218593
   Educational Leadership, Volume 64, Number 7, Pages 83-84.
   computational talent be detected? Predictive validity of the Computational Thinking Test.
   https://doi.org/10.1016/j.ijcci.2018.06.004
   learning. Retrieved October 23, 2018 from https://www.brookings.edu/blog/education-plus-


