AN EDUCATIONAL PROGRAMMING ENVIRONMENT FOR THE SWEDISH SCHOOL

Lisa Lundgren
An educational programming environment for the Swedish school

Abstract

Programming is currently being introduced as an obligatory part of the K-9 education in Sweden with the aim to strengthen pupils’ digital competence and prepare them with skills adapt for the 21st century. The challenge is now to successfully integrate programming as a tool for teaching and learning within existing practices. Teachers are expected to support their pupils learning about programming despite their own limited chance to acquire knowledge on the subject. The Swedish National Agency for Education (Skolverket) has given some guidance but research on this area is rather limited and a lot is left to the teachers to decide and educate themselves in. Educational programming environments (EPEs), especially aimed at novices, can support teachers in this integration but needs to be suitable for the objectives from the Swedish school to not unintentionally direct the way teaching is done. This study aims to answer how an EPE should be designed to fit within the Swedish school context with a focus on the teachers’ perspective. To study this, seven interviews were conducted with educators in middle school with experience of working with programming in the classroom, followed by two observations during lectures with EPEs. The results provided implications that were used to form design guidelines for an EPE aimed at the Swedish school.

Keywords: Educational programming environment, Interaction design, User experience, Educational technology, Activity theory

1. Introduction

Digital competence has been realized as an important skill in today’s society and the Swedish government has decided that by the first of July 2018, a strengthened emphasis on digital competence and programming should be included in the new K-9 curriculum (The Swedish Government, 2017). There is no general agreement on what digital competence includes and it has been described to contain from two to fifty different aspects. In the Swedish K-9 curriculum, the definition is based on The European Commission’s (2016a) and Digitaliseringskommissionens (2014) description of digital competence. Deriving from these, the definition used in the Swedish curriculum constitutes of four parts; (1) understanding how the digitalization affects individuals and society, (2) understanding and knowing how to use digital tools and media, (3) critical and responsible use of digital tools and resources, and (4) ability to solve problems and implement ideas in practice (Skolverket, 2017a). Programming is considered part of all above-mentioned aspects of digital competence and is viewed more as an educational tool and problem-solving process than sole focusing on coding skills (Heintz, Mannila, Nordén, Parnes, & Regnell, 2017). In the new K-9 curriculum, this has been realized
with clarifications and reinforcements of digital competence and programming in several subjects, where programming is mainly added in areas of mathematics and technology. The new inscriptions for algebra include how algorithms can be formed and used in programming. In grades 4-6, the pupils are expected to work with visual programming environments and continue with other programming environments later on. In technology, programming is added as a part of learning about the work practice of developing technical solutions where pupils should encounter how to control objects with programming (Skolverket, 2017b).

Preceding these changes to the curriculum, the Swedish National Agency for Education (Skolverket) requested an overview of research and experiences of programming in school. Kjällander, Åkerfeldt, and Petersen (2016) took on this challenge and according to their research, it could not yet be stated that the introduction of programming in the Swedish school rests on a scientific basis since empirical research in this context was found to be limited. They did, however, see it as reasonable to include programming in schools due to other valuable aspects of democratic, gender equality, knowledge transfer, and economic gain. It was also highlighted that programming is much more than just coding and should be viewed to foster computational thinking. Computational thinking is often brought up internationally when arguing for programming in schools but not stated as clearly in the Swedish curriculum. The term originates from the work of Seymour Papert (1980) who was a pioneer in the field of educational technologies with his early thoughts on “the computer as a teaching machine” and that “computers may affect how people think and learn” (p. 3). In his book *Mindstorms: Children, Computers and Powerful Ideas* he presents his thoughts on how children would develop cognitive skills from learning to programme, valuable to other domains as well. Computational thinking was later popularized in an article by Wing (2006) where she argues for computational thinking as an attitude and skill set, that everybody, not just computer scientist, should learn. She later defined computational thinking as “the thought processes involved in formulating a problem and expressing its solution(s) in such a way that a computer—human or machine—can effectively carry out” (Wing, 2014). Computational thinking and digital competence are related terms that both emphasize that programming is about much more than just writing code. Digital competence also comprises of aspects on how the digitalization effects individuals and society with the broader aim to foster informed citizens with competencies required in a digitalized world.

Another rationale for introducing programming in schools is the labor shortage in the industry. Learners need to acquire this kind of knowledge to boost their employability in the ICT sector (Bocconi, Chiocciariello & Earp, 2018) and countries need to produce computer scientists and other related professionals to compete in a world driven by technology (Webb, Davis, Bell, Katz, Reynolds, Chambers & Syslo, 2017). It is hard to predict specific skills or knowledge that will be required of the computing/IT-workforce in the future, which is the rationale behind not specifying any particular programming languages or environments in the curriculum since the industry is prone to change rather quickly while the curriculum needs to stay relevant for a long time to come (Skolverket, 2017a).

The challenge now is to successfully integrate programming as a tool for teaching and learning within existing practices in the Swedish school. Teachers who are used to be the experts are now supposed to work with something they have limited experience of since
programming was not part of their formal education. The curriculum provides little guidance on how this should be realized in practice since it is not stated which programming environment to use, so much is up for the teachers to decide and educate themselves in. Groups of teachers have formed on platforms like Facebook and other websites to share their experiences of using programming in school. Teacher training material from Skolverket\(^1\) and other actors\(^2\) is available for teachers who are interested to learn, but programming is known for being a difficult endeavor and teachers do not have a lot of time to spare. Educational Programming Environments (abbreviated to EPE(s) from now on) developed to facilitate the process of learning how to program can give support for the teachers but to the best of my knowledge, little research has been done aimed at investigating how an EPE should be designed to fit within the Swedish school context. The aim of this study was to investigate this with a focus on the perspective of educators in middle school with the following research question: 

**RQ:** How should an EPE be designed to fit within the Swedish middle school?

- How are EPEs used today?
- What are perceived strengths and weaknesses with current EPEs?
- What are the goals of using EPEs in the education?

The expected outcome from this study is answers to these questions and from the results be able to derive a set of design guidelines for an EPE especially aimed at the Swedish middle school.

## 2. Related research

In this section, related research on the design of EPEs will be presented starting with associated difficulties with learning to program, what might be considered when children are the users and why the Swedish school should be considered a specific context. After that follows an overview of EPEs, previously derived design principles and guidelines for EPEs, and evaluations of EPEs.

### 2.1 The difficulties of teaching and learning programming

Programming is known for being a difficult endeavor. In a study by Lahtinen, Ala-Mutka and Järvinen (2005) where university students and teachers were surveyed on the subject, the most difficult issues were perceived as; understanding how to design a program to solve a certain task, dividing functionality into procedures, finding bugs in one’s program and understanding programming structures. These issues were stated to relate to the difficulty of understanding larger entities of a program and not just the details in a specific context. According to their finding, novice programmers are often limited to surface knowledge of programs, using a “line-by-line” approach rather than applying meaningful program structures and often fail in applying obtained knowledge (Lahtinen et al., 2005). Other

---

\(^1\) Webb course about programming from Skolverket: https://www.skolverket.se/kompetens-och-fortbildning/lararc/om-programmering-webbkurs-1.263574

\(^2\) internet courses from IIS: https://kurser.iis.se/kurser/programmering/, Introduction to programming from Kodboken.se https://www.kodboken.se/
difficulties are learning the verbosity and specificities of programming languages and dealing with abstract concepts since they seldom can be helped with any real-life analogies (Buitrago Flórez, Casallas, Hernández, Reyes, Restrepo & Danies, 2017). Ko, Myers, and Aung (2004) have also investigated the learning barriers encountered by beginning programmers and they identified the six most difficult areas as related to design, selection, coordination, use, understanding, and information.

2.2 When children are the users

A lot of the research on educational programming environments have been conducted on university students but children can be considered somewhat different since their cognitive abilities are still evolving to that of an adult. Bruckman and Bandlow (2003) highlight some differences relevant to HCI in accordance with the developmental stages by Jean Piaget. These categorizations might not hold true for individual children since every child develops differently but can be useful as a general description. In the preoperational stage (ages 2-7) children are described as having a brief attention span and only capable of holding one thing in memory at a time. Abstract concepts are difficult and they are still developing an understanding of situations from another person’s point of view. Some children might be able to read but it cannot be assumed. The use of the keyboard is often avoided and simple point-and-click interactions with the mouse are easier than drag-and-drop. In the concrete operational stage (ages 7-11), children are close to the cognitive abilities of an adult but they still have difficulties formulating hypothesis and understanding abstractions. They are often able to use grouping, categorization and use rather advanced software. Children’s ability to type and use quite fine control of the mouse is increased throughout this stage. In the formal operational stage (ages 12 and up) it can be assumed that children’s cognitive abilities are about the same as that of an adult, although interest and taste should be considered different. A study on college freshmen in physics showed that although the shift from the concrete operational stage to the formal operational stage should occur around the age of 12, it can still be very hard to grasp abstract concepts even for adults (Williams & Cavallo, 1995).

2.3 The Swedish school context

The field of Human-computer interaction (HCI) is concerned with the interaction between people and technology to better support the activities that people want to do in the contexts they are used (Benyon, 2014). The technology that we use should in that sense be formed by the way we want to use them and not the other way around. The context of use is therefore important to understand before trying to find technology that can support an activity in that context. Technology that achieves in this task exhibits a high level of usability, which is a central aim of any human-centered design. According to the official definition of usability (International Organization for Standardization [ISO], 2018) the context of use stands for the “combination of users, goals and tasks, resources, and environment” where “the ‘environment’ in a context of use includes the technical, physical, social, cultural and organizational environments.” (ISO, 2018). When introducing programming as a new tool for teaching and learning in the Swedish school, one part of the challenge is to find suitable technology. We need to consider the context to find tools that support teachers and learners to achieve their
goals of working with programming in an effective way. Existing technology cannot be assumed to fit since the circumstances within the Swedish school sets certain conditions for the use of an EPE. How it will be used is strongly related to the curriculum with the goal of strengthening pupils’ digital competence and the integration of programming mainly in mathematics and technology, subjects with their own learning goals. Different approaches to introduce programming in school have been taken by other European countries where programming has been added as a separate subject or as a cross-curriculum strategy (The European Commission, 2016b). The available resources, duration of use and culture of teaching might also influence the use and deviate from other contexts. Apart from affecting the way technology is used, the context has also been realized to influences learning in a substantial way.

Work in social psychology, cognitive psychology, and anthropology is making clear that all learning takes place in settings that have particular sets of cultural and social norms and expectations and that these settings influence learning and transfer in powerful ways. (National Research Council, 2000, p.4)

To make sure that the way programming is thought is not formed by available technology, it is then worth investigating how the design of an EPE should be to fit within the context of the Swedish school.

2.4 Educational programming environments

Educational programming environments can be described as software applications that facilitate the development of new software, especially aimed at novices. They can be grouped into two different categories based on how the programs are being created; text-based programming where the code is written in a specific programming language or visually-based programming where the code elements are specified by visual representations. Visually-based programming is sometimes referred to as block-based programming since the visual representations of code extracts are often in the form of blocks. Kiper, Howard, and Ames (1997) define visual programming as any system where users can specify a program in an (at least) two-dimensional fashion. Visual programming can be perceived as less authentic and powerful than text-based programming (Weintrop & Wilensky, 2015) but are considered suitable for beginners. In a review article by Bau, Gray, Kelleher, Sheldon, and Turbak, (2017) they ascribe the learnability of block-based programming to the way it helps to overcome three learning barriers for beginners. (1) The user doesn’t need to remember a word from the full vocabulary of a programming language but can pick a block from a palette and thus rely on recognition rather than recall. (2) Blocks reduce the cognitive load by chunking code into fewer entities of meaningful elements. (3) The user is constrained from making basic errors by the possible ways the code can be assembled.

Today the most well-known educational programming environment aimed at children is probably Scratch³ by the Lifelong Kindergarten group at the MIT Media Lab. Scratch is a visually-based programming environment with puzzle like blocks that snap together through

³ https://scratch.mit.edu/
drag-and-drop interaction. Another successful initiative is the Hour of Code webpage\textsuperscript{4}, organized by Code.org, a non-profit founded in 2013 supported, among others, by the Chan Zuckerberg Initiative and Bill Gates. The initiative constitutes of several one-hour-exercises aimed at introducing coding skills for children from 4 years and older. Kojo\textsuperscript{5} is another educational programming environment that supports both visual-programming and text-based programming with Scala, a programming language that has been translated into Swedish in a partnership with the Faculty of Engineering at Lund’s University. Other examples of EPEs are BlueJ, Alice, MIT App Inventor, Scratch Jr, Light bot, Greenfoot, Pex4Fun, objectKarel, Processing, Swift Playgrounds, Tynker, Codemonster, Codecombat, Daisy the dinosaur, etc. Some EPEs also incorporates a robot or other programmable hardware such as a Micro:bit, Blue Bot, Dash, Ozobot, Lego Mindstorm, Arduino and Raspberry pi. Positive effects of working with robotics and programming have been noticed in younger children’s abilities to tell stories and remember sequences of information (Kazakoff, Sullivan & Bers, 2013). To work with programming using robots have also been reported to induce the interest in programming amongst pupils (Yamanishi, Sugihara, Ohkuma & Uosaki, 2015).

2.5 Design principles and guidelines for EPEs
Design guidelines are recommendations towards good design practice that aims to give clear instructions on how to adopt certain principles. Design principles are more general than the design guidelines that help to understand how to implement a given principle. The following is a summary of previous guidelines and principles for educational programming environments.

Moons and De Backer (2013) designed an EPE influenced by constructivism and cognitive learning paradigms and their design was based on the following design principles;

• Make the design interactive to improve learning
• Use the guidelines discovered by cognitive science
• Use the characteristics of human perception in the design of the visualization

The first principle relates to the constructivism learning paradigm and promotes interactivity where learners are activated through a create-test-improve cycle. Their suggestions on how this could be incorporated in the design are among others to provide on-demand explanations of concepts, include pop-up quizzes during program executions and provide transport controls like next and play that can be used to step through the code. They also suggest functions to step back and advance to the next breakpoint in the code to enhance inspection possibilities.

The second principle covers guidelines discovered by cognitive science and the cognitive load theory. On this point, the researchers highlight those they found relevant to a program visualization environment and states the following. Related concepts (of similar or different media) should be placed closely together in both space and time. Information should be clear and concise, hiding information that is irrelevant for the moment to avoid information overload. Relevant information should also be highlighted. Their third design principle is based on the characteristics of human perception. Here they bring forward how gestalt-
principles, correct use of color, different shapes for different concepts and motion to draw attention can be applied to present more comprehensible visualizations to the learner.

Repenning, Webb and Ioannidou’s (2010) work on game design as an approach for educational programming environments led them to the creation of a checklist of requirements that a “computational thinking tool” should include in order to facilitate computational thinking in the public school. The checklist also considers circumstances related to the curriculum design in addition to the other mentioned guidelines. They suggest that a successful tool together with the curriculum should include;

- A low threshold: enabling a quick start and production of a working game.
- A high ceiling: making it possible to produce a game that exhibits sophisticated behavior such as complex AI.
- Scaffold flow: a curriculum that provides ways to make progress from acquired skills towards new challenges accompanied by the tool.
- Enabled transfer: both tool and curriculum should support transfer from game design to subsequent science applications for computational thinking.
- Support equity: learning activities should be accessible and motivational regardless of gender and ethnicity.
- Systemic and sustainable: it should be possible for all teachers to use the tool in combination with the curriculum to teach all learners. This means that it needs to support teacher training, learning standards, and other educational alignments.

The argument that programming languages should have a low threshold and high ceiling is a longed lived principle that has also been argued for in the work of Papert (1980).

Good and Howland (2015) derived design guidelines from their own previous empirical investigations on how natural language fits into EPEs and arrived at the conclusion that EPEs for children should:

- Constrain the programming language: achieved using a set of graphical code blocks, eliminating the need to type code from scratch. Also achieved by making the natural language component of the language noneditable, thus preventing the user from inadvertently introducing syntax errors.
- Delineate natural language from code: in their EPE, the code is presented in a graphical code composition interface, with its natural language equivalent appearing in a plain English box below it. Natural language is used in the script (e.g. messages to be displayed to the player) and appears in a separate, color-coded block in the graphical language, and in quotes in the plain English box.
- Highlight natural language used as a computational construct: Their EPE uses correct computational terms to represent the underlying concepts on all interface menus and graphical programming blocks, allowing young people to begin to learn and use the "language of computation”. These computational keywords are automatically translated into a fuller, everyday language in the plain English box, further distinguishing computational language from natural language (reinforcing point 2).
- Highlight distinctions between computational categories: blocks are visually organized and color-coded by computational category (actions, conditions) to avoid
confusion between terms that are similar from a natural language perspective ("opens" vs. "is open") but computationally distinct (action vs. state).

- Provide support for errors of omission/commission at the program composition phase: blocks and slot fillers are differentiated by color and shape, with color used to link blocks with their corresponding slot fillers. This feature aims to support users' understanding of different types of commands and to help them avoid errors of commission. A further feature prevents blocks of the wrong type from snapping into place, thus preventing type errors. Errors of omission that prevent the script from compiling trigger a short error message.

These design guidelines were embodied in Flip, a bi-modal (text and visual) programming language for young people. An evaluation of the guidelines used in Flip showed that the main benefits in terms of support centered around issues of constraint, a clear separation between code and natural language, support for errors, and use of natural language for comprehension and debugging. The researchers conclude that full-sentence natural language is not well suited for code generation, but can provide helpful support for code comprehension, debugging and collaboration tasks.

The design of Scratch, that was described earlier in section 2.4, is based on three core design principles (Resnick, Maloney, Monroy-Hernández, Rusk, Eastmond, Brennan, Millner, Rosenbaum, Silver, Silverman & Kafai, 2009);

- make it more tinkerable,
- more meaningful,
- and more social than other programming environments.

Tinkering can be described as an act of trying out, experimenting and using something without focusing on any particular result. When describing the design of Scratch, Resnick et al. (2009) draw the analogy of kids playing with Lego bricks, "Given a box full of them, they immediately start tinkering, snapping together a few bricks, and the emerging structure then gives them new ideas" (s. 63). They wanted the process of programming with Scratch to have the same feel to it so they based their design on blocks that can be snapped together in different combinations where the shape of the blocks affords (suggests) how they can be put together. The block-based language simplifies the process of programming since users don't need to consider correct syntax, use punctuations or indentations that are prone to cause errors in text-based programming languages. The second design principle of Scratch, make it more meaningful than other programming environments, was realized by prioritizing two criteria. (1) diversity in the type of projects that can be made and (2) personalization of those project by enabling users to import their own media (e.g. sounds, photos, videos, graphics) into the project. This criterion is based on the notion that working on personally meaningful projects are the best way to learn while also enjoying what you do. Social connection and community are the third of Scratch's core design principles. Through the connected web community, Scratch users can seek support and inspiration from one another, collaborate on projects, comment and build on each other's work. This is stated as an important success factor since social opportunities serve as a motivational factor and the opportunity to learn from each other.
2.6 Evaluations of EPEs

In a study by Xinogalos, Satratzemi and Malliarakis (2017) students of higher education evaluated five different introductory programming environments: Scratch, BlueJ, objectKarel, Alice and MIT App Inventor. They found that the most ideal programming environment for novices should have a simple and user-friendly Graphical User Interface (GUI) that supports visualisation of objects and classes, includes a puzzle-like editor for program development, reports error messages in a simple and understandable language and lastly, provides the ability to follow the execution of the program in a step by step manner. None of their investigated environments contained all desired features but Scratch was found to be the most successful one. Some mentioned disadvantages with Scratch was related to insufficient error messaging and a distracting amount of functions. Regarding future work, they suggest the development of a programming environment that encompasses all of the above-mentioned features. In another study with Scratch it was also noted that although a user might show fluency in the use of coding blocks, this does not mean that the user understands the underlying computational concepts (Brennan & Resnick, 2012). This was e.g. noticed in a case where a Scratch user had produced a rather advanced project but copied a code extract from someone else and used it in the script (which is encouraged in the community). When asked about how it worked the user was unable to explain it which makes it hard to assess the level of knowledge by only looking at what has been produced.

3. Theoretical framing – Activity theory

To help structure the analysis of data, Activity theory was chosen as a conceptual framework for this study (its application will be further described in section 4.1.2.). The theory was considered suitable for highlighting the role of an EPE in the Swedish school context. Other theoretical frameworks were also considered but did not seem as applicable to this study. The educational theory of constructionism could have been used as a framework for outlining how pupils acquire knowledge using an EPE but since the study is more focused on the teachers perspective and design, activity theory was considered more suitable for this area of research.

Activity theory has become a rather long-lived tool for thinking about the mediating role of computational tools in HCI research (Kaptelinin, 2014). The theory is described by Nardi (1996) as more of a “clarifying descriptive tool than a strongly predictive theory” (p.7) where the objective is to understand the unity of consciousness and activity. Nardi (1996) continues to explain the benefits of Activity theory for HCI research as a way of understanding the context where computer-supported activities take place since it offers perspectives on human activity and concepts to help describe that activity. In its broadest sense, an activity can be described as an actor interacting with the world. In Activity theory, activities have two conditions that distinguish them from just any interaction: (1) subjects have needs that should be met through the interaction, and (2) the subject and activity mutually form one another (Kaptelinin, 2014). An activity involves both the subject (the one performing the activity) and the object (on which the activity is performed).
Activity theory has its roots in the social-cultural tradition of Russian Psychology with prominent researchers such as Lev Vygotsky, whose work has also had great influence on the field of education. Vygotsky’s mentee Aleksei Leontiev developed the original framework of Activity theory with inspiration from his mentor’s work. According to Leontiev (1981), human activity can be organized in a three-layered hierarchical structure (see figure 1). At the top-level is the activity, driven by an underlying motive. This motive can be somewhat hidden but is always present. Subordinate to the activity is conscious actions, related to goals. These actions are characterized by conscious planning. The actions are in their turn realized through operations, which are automated behaviors requiring little conscious attention if performed under profitable conditions. If we take the activity of writing this thesis as an example: This can be understood as an activity directed towards the motive of completing a full thesis. In this example, the writer is the subject and the thesis is the object. Doing a literature review and conducting interviews are subordinate actions, related to certain goals within this activity. Then we have the lowest level of operations of e.g. typing on the keyboard, reading a text or talking to someone that is executed rather autonomously under the conditions that everything goes well.

Figure 1: Leontiev’s three-layered hierarchical structure of human activity

A version based on Leontiev’s framework was proposed in the 1980s by the Finnish educational researcher Yrjö Engeström (1987). He further developed the theory from focusing on subjects, activities, and objects to consider them as part of a larger context. Additional concepts of community, tools, the division of labor and rules were connected to the initial model and extended the perspective to form an activity system (see figure 2). The community then constitutes of other actors involved in forming the object. Tools are considered artifacts, instruments, signs or other means that mediates the subject-object interaction. The division of labor, in turn, mediates the relationship between the community and the object by a coordination of responsibilities and rules mediate the subject-community interaction by a set of implicit or explicit agreements on what is expected from being a member of the community.
4. Methodology

In order to answer the research question of **how an EPE should be designed to fit within the Swedish middle school**, qualitative research was conducted through interviews and observations. The research method was chosen since the study aimed to understand the setting around the technology and what makes it suitable for the Swedish schools. This cannot be easily measured or quantified which calls for more qualitative research methods. The interviews were semi-structured, all covering the same topics (see Appendix 1) but allowing for elaborations upon given answers (Rogers, Sharp, & Preece, 2011). Five interviews were held at the school where the participants worked. One interview was conducted at the participant’s home and one interview was held over the phone. Semi-structured interviews are a common approach used in design work and are one of the most effective ways of finding out what people want and what problems they have (Benyon, 2014). The advantages of using interviews over surveys are that interviews allow for deeper and more detailed answers and perspectives that surveys might miss due to their restricted format. One drawback, however, is the high effort it takes to conduct the interviews and analyze the data (Lazar, Feng & Hochheiser, 2010). The interview structure was planned based on the process and six stages described by Legard, Keegan, and Ward (2003) and a pilot interview was conducted as preparation for the interviewer and test of the interview questions. The pilot interview was not included as part of the final data collection. As with any post-experience measure, interview answers might suffer from relying on respondent’s memory. The method can, therefore, be supplemented with data collected in-situ, such as direct observations to provide insight into practices that are hard for respondents to remember or articulate. The observations in this study took place after initial
interviews as recommended by Benyon (2014) and was structured based on the framework suggested by Fox (1998) with an added focus on usability issues. This led to the following focus points during the observation:

- **Space**: the physical place or places
- **Actor**: the people involved
- **Activity**: a set of related acts people do
- **Object**: the physical things which are present
- **Act**: single actions that people do
- **Event**: a set of related activities that people carry out
- **Time**: the sequencing that takes place over time
- **Goal**: the things people are trying to accomplish
- **Feelings**: the emotions felt and expressed
- **Reflection**: your personal response to any of the above
- **Usability issues**: any problems that occurred while using the system

The role of the observer was nonobtrusive, trying not to interrupt the natural flow of the lecture and notes were taken on a laptop computer.

The collected data in the form of verbatim from the interviews and field notes from the observations were transcribed and analyzed (see section 4.3) to form the results of the study that can be found in section 5. These provided insights that together with previous research served as implications for design guidelines for an EPE especially aimed for the Swedish school that will be presented in the discussion.

### 4.1 Participants

The seven participants in this study were all educators in middle schools from the northern part of Sweden. They were purposely chosen with the inclusion criteria to have prior experience of using some EPE in their teaching with pupils from the middle school. They were a nonprobability sample meaning that they were not randomly chosen from the population of interest. Background information about the participants was gathered at the end of the interviews by filling in a form (see Appendix 2). On average, they had worked as educators in 15 years and responded to be strongly motivated about introducing programming in their teaching (Mean=6.714 on a scale from 1-7, SD=0.070). They rated themselves as more experienced with programming compared to other middle school teachers (Mean=5.143 on a scale from 1-7, SD=0.054) but still agreed somewhat to the statement “I would need more support for how I should work with programming in school” (Mean=4.143 on a scale from 1-7, SD=0.117). A description of the participants can be found in Table 1.

The participants were chosen because of their professional experience of working as an educator and working with EPEs in school. Since it is expected that these programming environments are a part of the education from the 4th grade, the focus was on middle school teachers. Although the pupils are an apparent user group for this technology, they were not part of the scope of this study. The teachers are central to a successful integration of these environments in school since they have the responsibility to teach their pupils on the subject. They were also considered a more informative source concerning the context surrounding the use of the technology.
Motivation to work with programming
Experience of programming compared to other middle school teachers
Need for more support on how to work with programming in school
The participant gave the score 7 on a seven-point Likert scale
The IT-educator’s responsibilities are often to support the school organization’s use of IT by educating staff members and working with IT-strategy, IT-policy issues etc. Being an IT-educator can also involve teaching other subjects than IT but with a thorough and balanced use of IT.

### Table 1: Description of the participants’ background

<table>
<thead>
<tr>
<th>ID</th>
<th>Years of teaching</th>
<th>Teaches</th>
<th>Motivation(^1)</th>
<th>Experience(^2)</th>
<th>Need(^3)</th>
<th>EPEs used during lectures</th>
</tr>
</thead>
<tbody>
<tr>
<td>Anna</td>
<td>11</td>
<td>Ma, NO, Tk</td>
<td>7(7)*</td>
<td>5(7)</td>
<td>3(7)</td>
<td>Scratch, Micro:bit</td>
</tr>
<tr>
<td>Erik</td>
<td>8,5</td>
<td>Ma, NO, Tk</td>
<td>7(7)</td>
<td>5(7)</td>
<td>4(7)</td>
<td>HTML, Python’s IDLE,</td>
</tr>
<tr>
<td>Ingela</td>
<td>12</td>
<td>Ma, Sv, NO, SO, Tk</td>
<td>7(7)</td>
<td>5(7)</td>
<td>4(7)</td>
<td>Blue-Bot, Scratch, Scratch Jr, Lego Mindstorms</td>
</tr>
<tr>
<td>Johan</td>
<td>20</td>
<td>IT-educator(^5)</td>
<td>6(7)</td>
<td>5(7)</td>
<td>5(7)</td>
<td>Code.org, Blue-Bot, Dash, Dot, Swift playgrounds, Micro:bit</td>
</tr>
<tr>
<td>Mona</td>
<td>14</td>
<td>Sv, SO, Ma</td>
<td>7(7)</td>
<td>6(7)</td>
<td>5(7)</td>
<td>Codemonkey, Code.org, Scratch, Sphero, Dash, Lego Mindstorms</td>
</tr>
<tr>
<td>Sofia</td>
<td>20</td>
<td>Ma, NO, Tk, Id</td>
<td>6(7)</td>
<td>5(7)</td>
<td>5(7)</td>
<td>Code.org, Scratch, Codable, Lego Mindstorms</td>
</tr>
<tr>
<td>Åsa</td>
<td>20</td>
<td>IT-educator(^4)</td>
<td>7(7)</td>
<td>5(7)</td>
<td>3(7)</td>
<td>Blue-Bot, Bee-Bot app, Scratch Jr, Lightbot, Code.org, Scratch</td>
</tr>
</tbody>
</table>

Ma=mathematics, NO=general science, Tk=technology, Sv=Swedish, So=social science, Id=sports

* Motivation to work with programming
* Experience of programming compared to other middle school teachers
* Need for more support on how to work with programming in school
* The participant gave the score 7 on a seven-point Likert scale
* The IT-educator’s responsibilities are often to support the school organization’s use of IT by educating staff members and working with IT-strategy, IT-policy issues etc. Being an IT-educator can also involve teaching other subjects than IT but with a thorough and balanced use of IT.

**4.2 Ethical considerations**

During the observations, a teacher who had previously taken part in the interview held a lecture on programming with pupils in middle school. During the first observation, Scratch was used in a mathematics lecture in two sessions with different classes. The first session included 13 pupils and the following 16. The second observation took place during a lecture of the pupil’s choice. Programming was voted to be the subject of the lecture and there were 14 pupils from the middle school participating. During this lecture, Python’s Integrated Development and Learning Environment (IDLE) was used.

The participants were informed of the study purpose, how the information they provided would be used and that their participation was voluntary before taking part in the study. They were also asked for permission to be audio recorded. Their agreement was confirmed by signing an informed consent document. All the transcribed data was anonymized to not reveal the identity of the participants.
4.3 Data analysis

A Thematic analysis was used to synthesize the data and describe the content from the interviews and observations. Engeström (1987) activity system model from Activity theory was applied to structure information related to the context of use. The following describes how the data was analyzed to derive the results from the study.

4.3.1 Thematic analysis

The process was guided by the description given by Braun & Clarke (2006) where thematic analysis is divided into six phases. (1) Familiarization: In the first phase, audio recordings from the interviews were transcribed, writing down all verbal utterances and informative non-verbal communication. The transcriptions together with the field notes from the observations were then read through repeatedly to search for meanings and patterns, where initial ideas were noted. (2) Generating initial codes: In the second phase, initial codes were produced and organized, describing features of the data. Some extracts of text could be described by several codes and hence be attached to more than one code. (3) Searching for themes: When the entire dataset was coded the analysis continued to the identification of reoccurring patterns into potential themes. (4) Reviewing themes: The themes were then revised to make sure that they described the codes. In this phase, all the collected extracts from each theme were re-read to assure that they supported the broader theme. Themes were judged and reorganized based on that “Data within themes should cohere together meaningfully, while there should be clear and identifiable distinctions between themes.” (Braun & Clarke, 2006, s. 91). (5) Defining and naming themes: The themes were then continued to be refined and named to describe their essence and define what they were about. (6) Producing the report: In the final phase, the analysis was written down to tell the story provided by the data.

4.3.2 Applied theoretical framework

Engström’s (1987) activity system model from Activity theory was the major framework used in this study. By applying this model, the EPE was placed in the system revolving around the activity of teaching programming in school. In this system, the role of the EPE can be viewed as a mediator between the teacher and the pupil in the process of forming the pupil towards the desired outcome of strengthening the pupil’s digital competence. The teacher is the subject of this activity system and the pupil is the object that the activity is directed towards. Other actors such as colleagues, IT-educators, and the pupil’s classmates are also involved in the activity system but with less prominent responsibilities of forming the object mediated by the division of labor. Teacher colleges might share their experiences of working with EPEs with the teacher community and impact how it is used in the classroom. Colleagues at the same school might have a shared strategy for how the tool should be used and some schools have an IT-educator who is often involved in the selection of the EPE. The IT-educator’s responsibility is often to educate other teachers how to use IT in their teaching, but can also be to hold lectures with the pupils. The pupil’s classmates are also involved in that they share the attention of the teacher and can be a source for help if one of their peers get stuck on a problem. They might also motivate each other to be able to share accomplishments or create a competitive factor. During which classes and in relation to what goals the EPE is used is mediated by the curriculum. School policies and other rules that the actors are implied to
follow also affects how the EPE is used during class. In summary, this creates a system (see Figure 3) that can be used to describe the context surrounding an EPE in the Swedish school.

![Figure 3: The activity system model applied to this study](image)

### 5. Results

The themes from the analysis of the interviews and observations are divided into three major groups; use context, strengths and weaknesses, and desired outcomes. They each contain a subset of related themes (see Table 2). The interview and observation extracts are preceded by [Fictive name] (e.g. [Peter]) for participants and [Interviewer] for myself. The original Swedish conversations have been translated into English.

<table>
<thead>
<tr>
<th>Use context</th>
<th>Strengths and weaknesses</th>
<th>Desired outcomes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Work constellation</td>
<td>Engaging pupils</td>
<td>Subject connection</td>
</tr>
<tr>
<td>Progression of work</td>
<td>Gamification</td>
<td>Transfer of knowledge</td>
</tr>
<tr>
<td>Teachers’ experience level</td>
<td>Comparison between EPEs</td>
<td>Learning goals</td>
</tr>
<tr>
<td>Teachers’ motivation</td>
<td>Selection process</td>
<td>Problem-solving</td>
</tr>
<tr>
<td>Time and scheduling</td>
<td></td>
<td>Assessment and support</td>
</tr>
<tr>
<td>Working with programming</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Skolverket and the curriculum</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Computer and tablet resources</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Computer literacy</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Lecture plans</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Pupils’ experience level</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 2: Themes and sub-themes from the thematical analysis
5.1 Use context

Before holding any lecture, the teacher needs to gather suitable content for the session that resonates with the curriculum. Sometimes the content is created by the teachers themselves but it is also common to reuse that from previously planned lectures or use ready and available study material produced by others since it saves a lot of time in the planning phase. Similar to other lectures, the lecture with an EPE often starts with the teacher giving an introduction about what the class will be working on.

[Johan]: Well, I would like to say that it is actually not that big of a difference if you are working with programming compared to if you are working with whatever else, you start with some kind of walkthrough of an element or at least give an instruction of what it is they are going to do.

The introduction can also include solving problems together and going through relevant concepts e.g. what certain code elements represent in the programming environment and how they can be used.

[Mona]: Then it is like this, I’m not just throwing them out there. We do a lot together, to begin with [...] they take part and help to solve problems with me at first, to learn different blocks and what those blocks mean.

The lectures then continue with the pupils working hands-on with the EPE either alone, in pairs or in smaller groups. The pupils work in pairs [Ingela]: “in order to support each other”, [Sofia]: “out of a problem-solving-point-of-view, to be able to share questions and ideas with someone”, and because [Mona]: “one of them might have understood what the other one didn’t” and are then able to solve the problem together. The teachers often encouraged the pupils to help each other since they cannot always be the best in the classroom and keep up with the pupils. To work individually was mentioned as more suitable for certain tasks and sometimes preferred to avoid pupils becoming passive observers when working on a task.

[Interviewer]: How did they work during the lectures? Did they work together or more individually?

[Erik]: Individually as much as possible but they were welcomed to spy on each other or go and help others [...] and this is because I know that if it is something where they differ in knowledge from the beginning, then the tendency is that soon the one that has a skill for it does all the work while the other one is just watching.

To assign the pupils with roles was mentioned as a way to overcome the problem of pupils becoming passive during collaborations. Although work constellations are often deliberately chosen, it might also be forced due to limited computer resources.

[Interviewer]: Do you have computers so they can work one by one?

[Sofia]: Eh, <chuckles> we have the number of computers so that they can work one by one, but then those computers are totally useless so you should be happy if they work at all.
Most of the schools had a class set of computers that were used during lectures and laptops are the ones most common to use at this level of education (middle school), while tablets are mostly used with younger children. EPEs that are only available on tablets might hence be unfit for use in middle school. Computer literacy was not mentioned as a barrier but was noted to vary between pupils, some were a bit more unfamiliar with typing and using the mouse than others.

Working with programming in the classroom is centered around the pupils completing given assignments or tasks where the type of tasks largely depends on the EPE and the pupils’ previous experience with programming. Some task that the participants had used were; programming a robot to follow a pre-defined path, programming an object on the screen to complete a path, programming a Micro:bit to show a smiling or sad face, producing games, making an animation with their name, creating science exams, and making changes in a given code to see how the output changes. One of the challenges of working with programming in school is finding suitable assignments for the pupils since they often have very different levels of experience, to begin with.

[Åsa]: Many have an interest of their own and have been using Scratch at home or at some of these camps, and some don’t even have an idea of what it is. I think that challenge, in particular, has been the most difficult.

The teachers need to find suitable tasks for beginners and also provide challenges for the more experienced pupils so that everybody can get something out of the lecture. One participant mentioned, however, that [Anna]: “the ones who have worked with Scratch at home might not know all the basics” so lectures on the fundamentals of programming might still be beneficial for all. Some EPE’s have built-in instructions where the user is guided through the completion of a task in a step-wise manner (e.g. hour of code on code.org) others are more open in the type of tasks or projects that can be created. Many teachers use pre-made tutorials and teaching manuals to find suitable tasks in the beginning.

[Ingela]: There were these short YouTube clips with like step-by-step, that were very educational, so we watched one of those clips, and then they got to try, and we watched the next clip, very structured like that, but then once they had finished that they were quickly on track and got going.

The lectures were around 40 to 90 minutes long and occurred more or less frequently depending on different schools, from once, up to approximately 10 times a year. In some cases, the EPE was used during dedicated programming lecture as parts of the pupil’s choice while in other cases it was included as an element of already existing subjects such as mathematics and technology but as uttered by one of the participants:

[Johan]: Because this is so new in school, I believe that, from this fall, now as all schools are starting to work with it, it will, you will get to see examples of how schools are working with things and also, how schools are correcting their initial idea and their initial plan based on what others are doing. I think there will be a lot of things happening this following year as we are starting to practice this.
The curriculum is not specific on which languages to used or how the progression should look like and many participants found it to be quite vague.

[Interviewer]: Is it clear on what level it should be on?
[Anna]: No, that's the thing. If you read the text, you can interpret it quite broad in my opinion. What I think it says is that you don't need specific languages.

5.2 Strengths and weaknesses
Participants were asked why they chose the EPE’s that they had worked with. Their answers give away some important aspects for an EPE to succeed in the school context. Some mentioned factors were that the EPE was available for free or at a low cost, perceived as easy to start with and that there is ready available course material to take inspiration from or to use during lectures. Another important aspect is the promotion from influential figures, such as exposure in Skolverkets web courses and tips from colleges and keynote speakers.

[Anna]: Scratch is a freely available program, so that's why I took it [...] and I thought it was easy to work with blocks especially. That's what I wanted us to start with [...] then we were at this lecture at the university where this Finnish lady, ah, I can't remember her name, she was promoting Micro:bits and I talked a lot with her [...] and I talked a lot with our IT-educator, and him and I decided that we should buy it.”

Participants also talked about possibilities for progression within the EPE as a reason behind their choice [Erik]: “Python might be quite easy to learn, although you are still not that limited, considering that I want much more; to be able to go a long way with it.” To be able to use the same tool across different stages was also mentioned [Åsa]: “I try to consider, that it should be suitable for all ages. How can we buy things that last in all ages, that we can further develop?”

The participants were also asked if they saw any advantages or disadvantages between EPEs. Many of the participants then mentioned the benefit of block-based programming as easier to work with than text-based programming. Since the script is created by combining visual representations of code extracts instead of written texts, the pupils don’t need to focus on the syntax. One of the participants who had tried text-based programming with the pupils had experienced some of its disadvantages.

[Erik]: Well, in hindsight I can say that it might have been better to start with Scratch for some because even if they could write it, all of them sometimes didn’t know exactly what they were doing. [...] For some of the pupils who might already have dyslexia, it was very difficult to get these incomprehensible words to spell right.

The visual representations were also described as more visually interesting and appealing for the pupils. A reason to still work with text-based programming even in earlier stages is the perceived authenticity of it, [Erik]: “Why I chose Python instead of Scratch was because I wanted them to feel like this is for real, that now we are programmers”. During the observations where Python (text-based programming) and Scratch (visually-based
programming) was used respectively, it was noticed that the pupils asked more math related questions during the lecture with Python. This was probably because the tasks they performed were more math dependent than the task during the lecture with Scratch. Pupils were also more focused on visual aspects of what they created during the Scratch lecture where they worked on an interactive valentines-day card.

Code.org was the second most used environment after Scratch and described as simple, easy to use and a great way to start, with clear instructions and a nice progression. Another strength is that it seems to capitalize on the children’s interests.

[Mona]: I have children in a pretty perfect age who are playing with e.g. angry birds, all these games that are coming up. So, I knew that these kids they know of this. It felt good to begin there because this is something that they recognize so that’s why I chose code.org.

A mentioned weakness with code.org was uniformity, on the risk of becoming tedious [Mona]: “Children today are like click-click-click-click (demonstrates fast motions with the forefinger), it should go fast, it should happen new things, so you can... It can become monotonous”. Scratch was experienced as more creative and open in the type of tasks that could be created using the environment, compared to environments that follow a stricter flow of actions built around completing certain tasks, [Sofia]: “Scratch is more like, ‘here you are (hands out two open palms), do something!’, in my experience”. Scratch was also described as easy to start with but more advanced than environments like code.org and Blue bot, [Mona]: “I’m pretty glad that I waited quite long with that one (Scratch) because it is more advanced in my opinion”.

Programmable hardware like Micro:bits or robots was described as having the benefit of making programming more concrete in that it causes something to happen outside of the computer screen. Micro:bits was also described as one of the most widely applicable hardware options since:

[Johan]: “They contain quite a lot of sensors and other technology that makes it possible to apply its use in a way that you may not be able to with these, how should I put it, more play-like robots”

Robots were mentioned as a great way to capture the pupils’ attention and raise an interest from an early start. One of the participants, who had also worked with Scratch, Scratch Jr, and Lego Mindstorms stated that

[Ingela]: “As a way of introducing and getting started with the work, the Blue Bot was without a doubt the best in that it got the children all along with it, there is no question about it. They thought that was a lot of fun.”

Another participant did, however, choose not to work with Blue bots in middle school since [Anna]: “Well at least the fifth graders have come such a long way that our IT-educator thought it would be a step backward”. Another participant expressed that working on tasks with the Blue bot could be developed endlessly but that the Bee bot, on the other hand, was quite
limited. In the Bee bot app you are not presented with the code you input before running it which makes it hard to follow and know where possible errors might have occurred. Lego Mindstorms was mentioned twice as too advanced to fit in middle school. Codemonkey was described to progress quite quickly to the most advanced level, which was experienced as too difficult by one participant while another thought of it as more suited for younger children, reflecting different experiences.

The language in the interfaces is in some EPEs translated into Swedish which was expressed as beneficial since understanding the English words could sometimes be a barrier for pupils at this level. When instructions and tutorials were clear and mainly visual, the language barrier was not considered to be as pronounced. Short video-clips as instructional material was thought to be a very pedagogical way of explaining tasks within the environment and worthwhile, even if it was more time-consuming. Another mentioned benefit was the option to log in with an existing Google account since that didn’t require creating new accounts.

When asked about the pupils’ experience of working with programming, the participants gave predominantly positive reports although some pupils were described to be less excited than others. The pupils were perceived to become engaged when working with programming was varying, creative and matched their interests. Another factor that was mentioned to be important was that the pupils got to feel that they succeeded,

[Ingela]: “[Programming] is this thing where you get to be curious and creative and get confirmation that you actually can do it. That is; kids need success and it’s easy to succeed with this.”

When asked about how pupils coped with encountered problems a lot of participants reported that they raised their hands and asked for help although some worked insistently on solving problems by themselves. During the observations, it was also noticed that some of the pupils were not as keen to ask for help and sat quietly stuck for a long time. Gamification was also a topic that came up during the interviews and mentioned as something that could motivate the pupils and make them more persistent in finishing a task, [Sofia]: “To get your stars, your cucumbers or your points [...] I believe it to be quite important actually, a little bit silly but very important”. Some skepticism towards the gamified approach was also raised [Erik]: “the risk is that if you do it too much, then they’ll get used to that that’s the way it is supposed to be and then they’ll think that it’s not that fun to continue working with it”.

5.3 Desired outcomes

When asked about their desired outcomes from working with programming with the pupils, many of the participants talked about problem-solving skills and the ability to collaborate with others. To think logically and the ability to give clear instructions was also frequently mentioned.

[Åsa]: Well, to be clear, meticulous [...] I mean creativity, clarity, logical thinking, that I think is very important. But also, this thing with patterns and sequencing and troubleshooting, ehm. How can we solve this in a different manner? To be able to see that a problem can be solved in many ways. Can it be more efficient or not?
Some participants also talked about wanting the pupils to get a grasp on what programming is and to not feel intimidated by it. To get an understanding of how programs are built and how programming is used in systems all around us was also mentioned as a desired outcome. It was asked if they could see any transfer effects from working with programming onto other areas of work as well. On this question, participants mentioned that learning the work method of problem-solving in programming could be seen to influence mathematics and other subjects in a beneficial way but it was not that clear.

[Sofia]: I believe that what you with a little bit of good will can see effects on is problem-solving in mathematics. This thing where you take things step-by-step. They might become better at reading instructions. Reading, reflecting, what is it I’m supposed to do?

It could also work the other way around, that programming becomes an area where previously acquired knowledge from other subjects can be applied. The pupils did however not always make the connection themselves and needed the teacher’s help to make it explicit that their knowledge about e.g. angles could be applied in a programming task. That programming is being included in already existing subjects such as mathematics raised a concern that it would take time from other important parts of the education.

[Erik]: “I mean just because it is quite limited in time if you are to fulfill all the goals in math, then I feel that you cannot, I believe it might do more harm than good […] if you dedicate too much time to programming, then it is not sure that they will have time with everything else they need to do”

The participants were also asked what they needed to know about the pupils’ performance to assess their knowledge level and give them the right kind of support. This was considered a difficult question to answer but in general, the completion of a task served as a good enough receipt that the pupils had attained the acquired knowledge.

6. Discussion

The aim of this study was to investigate how an EPE should be designed to fit within the Swedish school. The focus was on the teachers’ perspective and answers to how they use EPEs in their teaching, what strengths and weaknesses they could see with these environments and what the goals of using them were. The results give some implications for design that were used to derive the following design guidelines:

- Low threshold. The environment should be easy to start with for beginners and enable all to succeed in solving a programming task.
- High ceiling. The environment should be usable to an extent and possible to use for a variety of programming tasks.
- Subject connection. Programming tasks should clearly relate to the curriculum and the learning goals for mathematics, technology, and other subjects.
- Connection between visual- and text-based programming. The EPE should present code both as block elements and as their text-based counterparts.
• Learning analytics. The EPE should visualize the pupil’s work to help the teacher in supporting their pupil.

• Support collaboration. The pupils should be able to work collaboratively on solving problems through shared projects and teachers should be able to share and be inspired by programming tasks used by other teachers.

• Swedish/English interface. The text in the interface should be available in both Swedish and English translation.

• Support control of physical objects. It should be possible to connect hardware that can be programmed through the EPE.

In the following sections, these guidelines will be discussed in relation to the results from this study and previous research.

The first and second guideline; that the EPE should have a low threshold and a high ceiling have been previously recommended by Repenning, Webb and Ioannidou’s (2010) and can be traced back to the work by Papert (1980). They are long-lived recommendations and the results of this study bring further support for their relevance. Participants stated that “children need to succeed” and for an EPE used in school, this becomes extra important. It should be possible for everyone, not only pupils with talent, to get the feeling of accomplishment while using the EPE. The pupils need to feel that programming is something they are capable of but to get that feeling from the start must be deliberately designed for since programming is associated with a lot of difficulties (Lahtinen, Ala-Mutka, & Järvinen, 2005; Buitrago Flórez et al, 2017; Ko, Myers & Aung, 2004). To work with programming at an early age could otherwise risk discouraging pupils from pursuing careers within the area of computer science. Visually-based programming is recommended for beginners and built-in instructions for pre-defined tasks provides even further support. There are a lot of already available EPEs that scaffold beginners in this way, making programming easier but also more limited. An EPE with this kind of structure will eventually become too simple and participants reported that it could become tedious for pupils to work with. An EPE built solely on this approach might also be discarded by decisionmakers who are looking for a tool that can be used to go far.

This brings us to the second guideline, that the EPE should have a high ceiling. For the EPE to be a sustainable tool it needs to allow for more open-ended programming as well and the possibility for creative use. That programming is a creative activity was something that seemed to enthuse both teachers and their pupils and should be utilized in the EPE. Teachers should be able to form their own programming tasks and pupils should be able to use the EPE with imagination once they have learned the basics. Repenning, Webb and Ioannidou’s (2010) motivates the same guideline for EPEs with a game design approach but their argument is based on a need for the possibility to create games with complex behaviors so that the pupils do not lose their initial excitement. To be able to create complex behaviors does not seem to be such a present need in the context of this study where the motivation for the guideline rather comes from the teachers and pupils need for creative use. The possibility for more open-ended programming within the same environment that was earlier restricted might, however, become confusing. The users initial mental model for how the system can be used might have to change as the EPE evolves towards more open possibilities. For beginners, the proposed approach is largely built on following pre-defined tasks but when enabling more creative use,
the way that the EPE can be used changes. How to avoid confusion around this transition is something that might need to be further explored.

In the requirement for a high ceiling, Repenning, Webb and Ioannidou’s (2010) also state that it is essential that the tool helps the pupils progress towards educational STEM applications. This subject connection is also a very present need in the Swedish school context. The EPE should be connected to the subjects in school since it is intended that programming becomes integrated as a tool in existing subjects once the pupils have reached basic knowledge and experience in programming (Skolverket 2017a). Some participants had already started with this integration but it also raised a concern that it could have a negative effect on the learning outcome in mathematics if too much time becomes dedicated to programming. To avoid this, programming needs to be effectively integrated into these subjects but this will require practical knowledge regarding programming which is a scarce resource in many schools today (hence the focus on teacher training). The suggested guideline is, therefore, that the EPE should provide subject connected tasks related to the curriculum to make it easier for teachers to apply programming tasks to their lectures. The tasks should be focused on, but not limited to, mathematics and technology since other application areas could be useful as well. The creation of these tasks should preferably be formed by experts in the field of education and programming to ensure high quality of the material. Teachers should also be able to edit tasks using the EPE to make programming more adaptable to specific learning activities. Since the focus is not on programming skills but rather on digital competence, the applicability of programming in the school subjects should be emphasized in the EPE.

The guideline of presenting a connection between visual- and text-based programming is recommended to support the pupil progressing from working with visually-based programming. Visually-based programming is suitable to start with but when moving on to later stages of the education, it is likely that text-based programming will be encountered. This should not come as a surprise and can be avoided by making a connection between visual- and text-based programming from an early stage. The curriculum does not state that text-based programming is the given next step but that pupils in later stages should work with various programming environments where text-based programming is mentioned as an example. The occurrence of the text-based code could become questioned since it is not expected in middle school. Pupils might only need to know that this connection exists without having to make it explicit in the EPE. Moons and De Backer (2013) recommends that information that is irrelevant for the moment should be hidden to avoid information overload. The relevance of this feature might in regard to this recommendation have to be revised.

The EPE is also recommended to help the teacher keep track of the pupils’ progression and present an overview of their work at the end of the lecture. This could streamline the process of assessing the pupils’ performance and alleviate some of the pressure on teachers programming knowledge. The teacher can then focus on how to best support their pupils. In the classroom observations, it was noticed that the teachers were very busy supporting pupils who were eager to get help. Some pupils, on the other hand, was not as quick to ask for help and insisted on solving the problem by themselves which might end up with pupils falling behind without being noticed by the teacher. Learning analytics have the potential to flag for struggling pupils and provide teachers with actionable insights to improve the didactical
design of their teaching. What kind of data that should be gathered and how it should be presented to be useful is something that needs to be further investigated given the implications it could have for the assessment of the pupil. Providing the right kind of analytical data is not an easy task as the study by Brennan & Resnick (2012) could show and considerations need to be taken on the impact it could have to provide this kind of data. Relevant information for the teachers based on the results of this study seem to be if the pupils have managed to solve the problem and how long they worked on it.

The work with programming in schools is often done in pairs and collaboration skills were mentioned as one of the desired outcomes of working with programming. It should then be easy to collaborate with peers. When the pupils work on tasks together with one computer their collective progression should be logged on all pupils' accounts. It should also be possible to share projects and work collaboratively on two or more computers at the same time since collaborative work is not always due to resource limitations. As stated by the participants in this study and by Resnick (2009), social connection and community are important for the opportunity to learn from each other. In Scratch, this is realized with the connection to an online community but for an EPE used in the classroom, the community support can be found amongst classmates which makes an additional online community a redundant feature. Teacher on the other hand, who are also in need of social connection and community might not have the same access to this support in their physical environment and should be enabled to find this kind of support within the EPE by making it possible to share and be inspired by tasks used by the community of teachers.

Some pupils at the level of education where these environments are being introduced might need the interface to be in Swedish to fully understand it although the English terms will be important to learn as well as preparation for the future. The work by Good and Howland (2015) would suggest that the most beneficial use of natural language would be in the instructions and error messages while there needs to be a clear separation between code and natural language. The words used in the code elements might then benefit from being kept the same regardless of the user's native language since their semantic meaning needs to be understood separately from plain English.

In the curriculum, it is stated that pupils should be able to control objects with programming. This should be supported in the environment to make it more applicable for the school. Positive effects from working with programmable hardware have been found in previous research (Kazakoff, Sullivan & Bers, 2013) and have been reported to heighten the interest in programming amongst pupils (Yamanishi, Sugihara, Ohkuma & Uosaki, 2015). Participant in this study also mentioned that robots served as a great way of introducing the children to programming and that the use of robots and programmable hardware made the programming more concrete. To work with more tangible objects might be extra beneficial for children since their ability to understand abstractions is more limited compared to adults (Jacko and Sears, 2003).

When looking at this study from the perspective of activity theory it highlights the central activity it concerns, namely the activity of teaching programming in school and the role of the EPE in that activity. Another related activity that was not the focus of this study is the activity of learning programming in school where the pupil is the subject of the activity. How EPEs are
used and experienced by the pupils has not been thoroughly investigated in the scope of this study but is an area of interest for future research. How to support a transfer of knowledge from using these tools onto other domains is not yet thoroughly understood and needs further investigation as well to find effective methods for teaching. There is also a need for research on what computational concepts are most appropriate to be taught at what level of school and clearer directives on what level of programming the pupils should have reached when leaving middle school. Research on integrated assessment within these environments is yet another area for further exploration in order to provide teachers with better actionable insight.

All research has limitations and the ones that could have affected the reliability and validity of this study is considered to be the following. The design guidelines that were derived from this study have not been validated. The intention was to evaluate the guidelines with a new round of participants but this was not realized in the scope of this study. A prototype was built based on the guidelines and evaluated with one participant which gave valuable insights to the discussion although it was too limited to be considered as results in this study. The thematic analysis of the data could have been conducted by more than one researcher to confirm the themes and avoid bias caused by one researcher’s perception. The data and analysis could have been made stronger if validated by the participants to avoid wrongful assumptions and interpretation of their response. The sample of participants was purposefully chosen and not a random sample which means that some underlying variables could have influenced the results. Since working with EPEs is not yet mandatory for all middle school teachers and the teachers had to have prior experience of using EPEs to be able to contribute to the study, a random sample would not have been possible. The participants’ early adoption might incline a more positive attitude towards technology in general and motivation to include programming in their teaching then what is representative for all middle school teachers. If this affected the results is hard to say but in future research, it would be interesting to compare these results to a more heterogeneous group of teachers.

Ethical considerations were taken when conducting this study to ensure that the participants were informed about the purpose of the study, what they agreed to when participating, how the information that they provided would be used and that their participation was voluntary. The data was handled with care and anonymized to not reveal the identity of the participants. The only harm from participating is considered to be the time it took from their usual undertakings. No one declined to take part in the study after reading the information regarding the study and there were no defections. The permission to observe during the lecture was granted by the teacher but it was not considered to gather consent from the pupils as well. This is justified by the observation being seemingly harmless and without any associated risk for anyone involved. Only field notes and no personal information about the pupils were gathered and the observer’s role was nonobtrusive, trying not to disturb the natural flow of the lecture.

7. Conclusion

The question that this study set out to answer was how an EPE should be designed to fit within the Swedish middle school. Although there are a lot of good options available, there is room
for improvements. The results from this study provided descriptions surrounding the use of EPEs, their perceived strengths and weaknesses and desired outcomes from using them, that gave implications for how an EPE should be designed to fit within the Swedish middle school. The recommendations derived from this study is that such an EPE should have a low threshold and a high ceiling, provide subject connected tasks, present a connection between visual- and text-based programming, including learning analytics, support collaboration, be available in both Swedish and English and support control of physical objects. The main contribution of this study is considered to be how EPEs are used and experienced by teachers in this context since research on this area is rather limited. However, the question of how an EPE for the Swedish middle school should be designed also needs to be studied from the pupils’ perspective which is an area for future research. Although the design guidelines that were derived from this study has not been validated they can still be useful for developing an EPE for the Swedish school which is relevant to the field of HCI but may also give some direction for teachers who are choosing amongst available options.

8. Acknowledgment

This study would not have been complete without acknowledging the efforts of people who have helped me in his work. I would like to thank my supervisor for his guidance, friends and family for their support and my supervisors at Dohi Agency for consulting me along the way.
References


Appendix 1: Interview script

Context of use
- How do you normally structure your lesson plans?
  *More in general terms than specific to programming.*
- Can you tell me about how you have work with programming during your lessons?
  - Do the pupils work in pair, group or alone?
  - Approximately how long do they work?

Choosing programming environment
- Which programming environment(s) have you used?
  - Are there any more that you know of?
- What was it that made you decide on that programming environment?
- How does the outside support (from Skolverket or similar) look for how you should work with programming in school?

Current issues and suggestions on improvements
- If you were to compare the EPEs that you have experience of/other ways of working with digital competence in school, which are pros and cons?
- How has it been going for the pupils to work with programming?
- What are common problems for them?
  - How do they cope?

Digital competence (goals and support)
- What kind of skills would like your pupils to develop through programming?
  - Can you notice any knowledge transfer from working with programming onto other areas?
- What would you need to know about your pupils’ performance with these tools to be able to give them the right support?
- What are your thoughts on the progression of using programming?

Background information
*Filled in by the participant on a separate paper (see Appendix 2)*

Closing question
- Is there anything that you would like to add?
1. I vilka ämnen undervisar du?
________________________________________________________________________

2. Hur många år har du jobbat som lärare?
________________________________________________________________________


   $1 = \textit{inte alls motiverad}, \quad 7 = \textit{starkt motiverad}$

   $1 - 2 - 3 - 4 - 5 - 6 - 7$

4. På en skala från 1 till 7, hur skulle du skatta din egen erfarenhet av programmering jämfört med andra mellanstadielärare? Ringa in ditt svar.

   $1 = \textit{ingen erfarenhet}, \quad 7 = \textit{mycket erfarenhet}$

   $1 - 2 - 3 - 4 - 5 - 6 - 7$

5. Nu vill jag att du tar ställning till följande påstående:
   Jag behöver mer stöttning kring hur jag ska jobba med programmering i skolan. Ringa in ditt svar.

   $1 = \textit{tar helt avstånd från}, \quad 7 = \textit{instämmer helt}$

   $1 - 2 - 3 - 4 - 5 - 6 - 7$