

Programming in Year 4: An Analysis of the Design Research Process

Ivan Kalas¹^a, Andrej Blaho² and Milan Moravcik³

¹Department of Education, Comenius University, 842 48 Bratislava, Slovakia

²Department of Applied Informatics, Comenius University, 842 48 Bratislava, Slovakia

³Edix, Bratislava, Slovakia

Keywords: Primary Computing Education, Programming in Lower Primary, Educational Design Research, Computing with Emil.


Abstract: In this paper, we analyse the educational design research process in which we looked to develop an evidence-based intervention and professional development (PD) content for programming in primary Year 4*. We explain how the latest period of the Computing with Emil project extends our previous intervention for Year 3 by exploiting the same constructivist pedagogy and design principles. We recall these principles and the research strategy which we consider appropriate for the current state of knowledge in transforming computing education in lower primary years. As the concluding phase of the design research, we reflect on the design process in greater depth and formulate the characteristics of the intervention by focusing on its programming language and concepts. To frame this effort, we identify five interwoven contexts that shaped the development cycles of our inquiry, namely: (1) programming environment, (2) programming language, (3) content development, (4) workbook for pupils, and (5) teachers' PD design. Looking back, we analyse the iterative design process within these contexts through the lens of the five instruments of reflection we were continuously exploiting to inform our design, namely: (a) observing pupils work, (b) interviewing teachers, (c) analysing pupils' workbooks and onscreen content, (d) evaluating Emil Y4 pilot PD sessions, and (e) analysing Emil Y3 data. Among other findings, which have resulted from the analysis of the design process, we highlight in particular a thoughtfully designed pulsating range of the command set. Although we had gradually created the content ourselves, it was not earlier than the concluding retrospective analysis that exposed it so clearly.

1 INTRODUCTION

The interest of various educational systems in extending the modern concept of computing into lower primary years is considerable and has been augmented by (Royal Society, 2012) and other influential publications. We perceive a strong emphasis on *computing for all* (Sentance, 2019), an explicit rejection of reducing computing to developing digital literacy and computer skills, a broad and almost unanimous agreement on an important role of programming, and renewed interest in a better understanding of and exploiting the connection between the development of mathematical thinking and computational thinking.

Although we welcome these trends and strive to promote them, in our view, new computing education

still suffers from the absence of a systematic and comprehensive approach to support a holistic learning process over several years of primary and secondary stages. The Computing with Emil project (Emil for short) seeks to address this deficiency and explore the potential of programming at the primary level. Our goals are to (a) identify programming *concepts* and various *operations* performed with the concepts by the learners which are suitable for the age group, (b) recognise the productive gradation of 'appropriate steps' leading to a deep understanding of the concepts, and (c) design complex interventions to support the gradation and the learning process in a sustainable form. We are also interested in formulating and validating which *design principles* in the sense of (Van den Akker, 2013) and *pedagogies* should be employed so that a generalist teacher with no special

^a <https://orcid.org/0000-0003-4597-3028>

* In our educational system it means for pupils aged approx. 9 to 10. For short, we will often use Y4.

computing background are willing and confident to adopt the interventions in their classes.

As we want to support these goals and contribute to further knowledge, the educational design research theory (Van den Akker, 2013; Plomp, Nieveen, 2013) seems the right theory to help us do so. It stresses the importance of a thorough conclusive analysis of the process which would identify its design principles and thus help improve the understanding of the programming/algorithmic thinking development at the lower primary level.

2 BACKGROUND

Undoubtedly, the key concept of our area is computational thinking, coined by Papert in (1980), which is being studied more and more intensely nowadays, see (Grover, Pea, 2013; Brennan, Resnick, 2012) and others. Much attention is paid to developing computational thinking in extracurricular activities, after school clubs, summer camps etc. where participants are usually voluntary, gifted, or specially selected. Fewer studies target regular formal classroom settings (Lye, Koh, 2014). And yet, there are *compelling reasons for mandatory computing beyond economics and employment. Access to a high-quality education can be seen as an equity issue* (Sentance, 2019). ‘Computing for all’ is the highest priority in our project as well.

To implement computing in formal education, researchers have focused on studying computational concepts and their cognitive demand (in the sense of Blackwell, 2002) (refer to Brennan, Sentance, 2012; Kalas, Benton, 2017; Meerbaum-Salant et al., 2013). In their framework for studying computational thinking, Brennan and Resnick (2012) identify three key dimensions: *computational concepts* (sequences, iterations, events etc.), *computational practices* (testing, debugging, reusing, remixing etc.), and *computational perspectives* about the world around us.

Another issue frequently studied is the connection between developing *mathematical thinking* and *computational thinking* (see e.g., Benton et al., 2018a; Benton et al., 2018b) as per the reported findings of the ScratchMaths project. Three aspects of the ScratchMaths design process have also played an important role in our current Emil inquiry:

(i) *Design research strategy* (DR for short). The ScratchMaths intervention was subject to cycles of iterative design research (Plomp, Nieveen, 2013) which means that we had to (a) iteratively design an intervention to obtain an instrument to deal with our

research problem, and (b) iteratively research to continuously support and inform our design.

(ii) *Design principles and 5Es framework*. In the ScratchMaths design, we exploited several innovative decisions (e.g., pupils always working in pairs with one computer per pair) and created a new *framework of action* comprising of five pedagogical principles, namely *Explore, Explain, Exchange, Envisage, and bridgeE* (hence the *5Es*) which provide pupils with a continuous opportunity to construct their experience and understanding of the computational concepts in manifold ways (see Benton et al., 2018a; Benton et al., 2018b).

(iii) *‘From concepts to constructs’ approach*. As we elaborated in depth in (Kalas, Benton, 2017; Kalas, 2018a; Kalas, Horvathova, 2022), we consider the vocabulary which is normally used to define learning objectives in programming that are too coarsely constructed to capture subtle differences in grasping the concepts at pupils’ different stages of understanding. Therefore, we always strive to identify an appropriate *gradation of operations*; we want pupils to work with each concept to support a gradual construction of its meaning. For that, we use the term *constructs* to pair a concept with the operations to be performed with it by the learner.

3 COMPUTING WITH EMIL

Computing with Emil (Emil for short) is an ongoing design research project, currently in its 5th year. In April 2017, we initiated the design for Year 3 and from October that year, we started working regularly with pupils and their teachers in three *partner design lower primary schools* by using an early pilot implementation of our software and the content. The process for Year 3 (Y3 for short) was completed in December 2018. In February 2019, we continued the process for Y4, the final analysis of which comprises the focus of this paper. In (Kalas, 2018a), we presented in detail the outcomes of Emil Y3, its pedagogy, and the design principles, plus some of the findings learnt in that period that were adopted for the Y4 design as well. In short, the main outcomes of the design for Y3 were:

- a *research-based intervention* comprising (a) a programming environment, (b) a learners’ content implemented as a gradation of tasks partly presented in the software and partly in (c) a printed workbook for pupils, (d) teacher materials, and (e) a professional development (PD)

process. The gradation¹ of the tasks underpins a validated gradation of the computational concepts and operations developmentally appropriate for Y3 pupils,

- *a set of powerful ideas of computing* which primary pupils encounter and explore in Y3, thus, developing their deep understanding of the concepts and operations that are performed with them.

In what followed, we set out to explore how the design principles and pedagogy of Emil Y3 could be developed and extended in a coherent way to Y4.

4 EMIL FOR YEAR 4

Since February 2019, we continued our development for Y4. Our first lesson with pupils in our design school took place on April 2, 2019. Throughout the period, we proceeded with exploring and better understanding:

- what programming concepts and operations are developmentally appropriate for Y4 pupils, having had achieved the learning objectives of Emil Y3 intervention already,
- what curriculum gives pupils adequate opportunities to gradually construct an understanding of the selected concepts and operations to perform, and master them (e.g., from the perspective of the learning objectives of Anderson and Krathwohl's taxonomy of the cognitive process dimension, 2000),
- whether our design principles from Y3 can be extended to underpin a complex intervention in supporting a systematic learning process of programming for lower primary pupils (considering generalist primary teachers' perception as well).

4.1 Method

We want to present our research strategy of the process of designing Emil Y4 from two perspectives. First, we briefly elaborate its iterations. Then, we concentrate on the concluding analytical phase of the process when we reflected on the process in greater depth by looking for its structure and striving to formulate the *lessons learnt*.

In the process, we were working with two parallel classes (we refer to them as A and B) of one of our design schools. At that time, all pupils were already well familiar with our intervention for Y3. In class A, we ran 11 lessons with teacher T1 always present. Three more lessons were run after that by T1 alone, using and commenting our content. Another three times, pupils were given some '*without computer*' pilot worksheets² to solve as their homework. Intentionally, we started working with class B with a lapse of about eight weeks so that the software interface, programming language, and the worksheets would have already been several iterations ahead of the version initially used in class A. In class B, we ran eight lessons altogether.

All the 11 lessons in A, always observed by T1, were taught by one of the researchers and sometimes also by teachers T2 and T3. Teacher T1 was in all the eight lessons conducted in class B as well. Out of them, three lessons were run by herself. Two or three researchers were always present as well: one facilitating the lesson and other(s) acting as observer(s) with at least one teacher and one researcher observing.

The pupils were always working in pairs. Each pair had one tablet (as one of our main pedagogy principles), each pupil having their own workbook. As they were already familiar with the method, solving the problems in a collaborative way was all natural for them. Similarly, having regular whole class on-the-carpet discussions several times during each lesson was already a common practice.

Immediately after each lesson, T1 (sometimes also T2 and T3) and our team spent around an hour in a discussion where we analysed and evaluated the observations and all the collected materials. After finishing the visits to school in January 2020, we spent another two months in finalizing all the components of the intervention in a close collaboration with teachers T1, T2 and T3 again. The final curriculum and the workbook were then consulted for with a maths education expert and a language expert, and then reviewed by two computing education experts from different countries.

As with Emil Y3, we continued using the DR as our principal inquiry strategy (Van den Akker, 2013; Plomp et al, 2013; Design-Based, 2003). When the design process was completed, we identified more than 30 iterations of the software environment and the content integrated in it, and an even higher number of

¹ The reason why we avoid saying *progression* instead of *gradation* is to highlight the subtle difference between 'progression' and 'systematic progression' in the sense of (Garrison, D.R., Anderson, T., Archer, W., 2001).

² We do not consider these paper worksheets to be *CS Unplugged* tasks, as they are integral part of all other screen + paper activities. All worksheets were finally collected in a workbook for pupils.

iterations of the workbook representing the minor or major steps of our *designing for research* and *re-searching for design* endeavour (ibid). Only then, however, we were able to look back on the whole process and start thinking about which contexts of the design were iteratively addressed in these DR cycles of *studying and theorising, developing, deploying and evaluating, analysing*, and finally *re-theorising* again (see Design-Based, 2003). In the following section, we characterise some of the contexts, mostly those that played a central role in shaping the programming language of Emil Y4 itself. Then we explain what instruments we exploited to inform the design and guide our decisions.

4.2 Analysis

The process was led by (i) our expertise from the previous projects³, years of teaching computing at all stages from early education to CS students, and pre-service teachers of informatics, and (ii) our assumption that the design principles implemented in Emil Y3 can be applied productively in Y4 as well. The progression of the development was driven by regular internal design meetings of the authors where we tried to identify appropriate computational concepts and operations and strived to transform them into a productive gradation of expected *learning trajectories* (in the sense of Baker, 2018), an appropriate content, a programming environment and a programming language.

In the DR strategy, iterations of the design should mean improvements leading to the state where researchers conclude that the outcome of the design serves their need to answer their questions and they feel confident that the inquiry is validly evidence-based⁴.

So, what were those improvements in the case of Emil Y4? First, by analysing the process as a part of the final retheorising phase, we managed to identify five contexts that had gone through considerable development in our inquiry, namely: (1) *programming environment*, (2) *programming language*, (3) *content development*, (4) *workbook for pupils*, and (5) *PD strategy and teacher materials* (see the first column of Table 1). Each context was interwoven with all others, influencing them, and being influenced. They were all iterated in parallel throughout the process with the last context being intentionally postponed by several months. In what follows, we characterise only the first three of them (due to the focus of this paper).

Programming Environment. This context comprised the way how the screen was organised, how users navigated their way in it, how the icons and other items were presented etc.

We adopted all the main design principles of the environment of Emil Y3, such as navigation between small groups of related tasks (again denoted as units of tasks **A**, **after A**, **B**...), navigation within each unit, moving from one task to another, running a task anew etc.

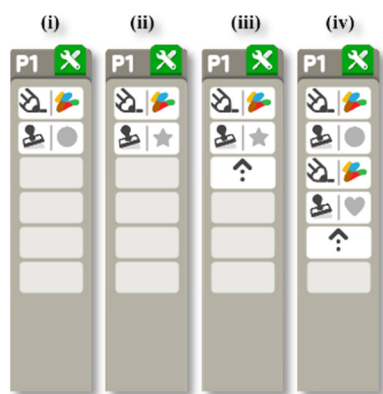


Figure 1. Final design of the Emil Y4 environment, showing task **J1** when solved. Pupils are given a simple definition of **P1**, see (i) on the left. They explore it and use it to build their own **P2**. Then they create various ‘bracelets’ (see the stage) by modifying **P1**, see (ii), (iii) and (iv).

³ Including Comenius Logo (or Super Logo in some countries), Thomas the Clown and Imagine Logo. One of the authors was a member of the ScratchMaths team.

⁴ Considering its *credibility, transferability, dependability, conformability*, and *authenticity*, as one possible set of criteria (Lincoln, Guba, 1985). Commenting them, however, would go beyond this paper.

What we intentionally changed and modified in numerous iterations was the visualisation of the panel with the history of commands given to Emil (see Figure 1, the green vertical panel on the left of the screen), an interface for new compound commands **P1**, **P2** and **P3**, and several other issues. To illustrate one of them, we explain why we decided to implement the onscreen *dragging of Emil* (with a finger or a mouse).

When pupils are provided with one or more compound commands, we encourage them to start by exploring them and only then modify them and use in their own programs and definitions (see section 5). To do so, as illustrated in Fig. 1, it is convenient to keep their experiments separate (see four line-patterns or ‘bracelets’ in the stage of Emil). Therefore, we introduced a possibility to drag Emil in *direct manipulation* mode (Kalas et al., 2018b), recorded in the panel of the commands history as a *meta-command*⁵.

Programming Language. In this context of the DR process, we focused primarily on the following three major categories of the computational concepts⁶:

- *Relative* (‘turtle-style’) *frame of reference*, i.e., controlling the direction of a sprite by turning it left or right is richly studied in Logo literature (see e.g., Abelson, diSessa, 1980; Chiocciariello et al., 1993) and is always considered an important but problematic issue for young learners. Controlling a physical programmable object using relative frame of reference by children of this age is all natural (due to the so-called *body syntonicity* (Watt, 1998) principle). However, most of the on-screen introductory programming environments for lower primary years choose the *absolute frame of reference*, as is the case of Emil Y3, when moving forward one step is implemented through four different commands – moving *up*, *right*, *down*, and *left*. In Emil Y4, we decided to facilitate the transition from absolute to relative frame by paying special attention to both (i) the visual of the **left** and **right** commands (which had been modified considerably throughout the process until getting the final form, see Table 2), (ii) clear and helpful animation of Emil when turning, and also (iii) the supportive gradation of activities of building this concept.

- *Commands-settings* and *commands with inputs*. While in Emil Y3, each basic command represents single on-screen action with a visible and unambiguous reaction and no input. In Emil Y4, we decided to gradually introduce two settings: **set colour** and **set size**, both manifesting in the next commands to draw lines and stamp. We also introduced two commands with inputs: **stamp shape** (one of the six shapes) and **fill with colour**. In the DR process, all of them have undergone several modifications, based on the data collected and analysed (see section 5), with the **fill** command undergoing the most of all.
- *Compound commands P1, P2 and P3*. The concept has already been ‘pre-introduced’ in the final part of Emil Y3 – as a simple means of abstraction. In Emil Y4, we continue the gradation by having three pre-defined or user defined/modified compound commands which operate as ‘my own blocks’ of Scratch with no input. Any of them can use any basic command or another compound command(s) in their definition with no way to end up with a direct or indirect recursion. Similar to other concepts listed above, the way how this one is presented to the learners had been considerably modified and reshaped in the process.

Content development is strongly connected with the programming language design and refinements in the environment. As with Emil Y3, it consists of one global gradation of more than 90 tasks organized in 18 units. Some tasks are presented only on-screen – in the software environment. Other tasks are only in the pupils’ workbooks. However, the majority of the tasks are presented partly on-screen, with the substantial part appearing in the workbook, such as task **J1** illustrated in Figure 1.

All tasks had been authored, designed, rearranged, and refined multiple times during the DR process in a special admin mode of Emil Y4. Developing and refining the content – and systematically analysing it at the final stage of the research endeavour – is one of the key contexts of the DR process, subject to dozens of iterations, under-going minor and major changes, deletions, insertions, and rearrangements. For the authors, such development must continue, informed and regulated mostly by the observations of the pupils’ work, interviews with them and their teachers and analyses of the pupils’ solutions. This is

filled polygons etc. Last line of Table 2 illustrates the complete programming language of Emil Y4.

⁵ No basic command can do the same, it is being done “above” the stage.

⁶ In Emil Y4, more computational concepts are being developed, like programming oblique lines, programming

done so until the authors conclude the intervention as a whole works well in the sense that the class of pupils involved enjoy it and master the operations with the concepts implemented in the gradation of developmentally appropriate small steps.

5 EVALUATION AND FINDINGS

During the design process of Emil Y4, we were continuously collecting multiple evidence to assess the development cycles and evaluate the interim outcomes. Nevertheless, as typical in the DR approach, only in the concluding retheorising and assessment phase of the inquiry, we succeeded in recognising certain systems and structure of the overall design process. Although it was us who designed and iteratively revised the intervention until we concluded a sufficient balance between the intended outcome and the real intervention had been achieved, it required this final phase to explicitly formulate (i) what had been modified and (ii) what were the instruments to advise the process. In this way, we managed to identify five interwoven contexts (three of them already presented). ‘Orthogonal’ to them were also five instruments of reflection we were continuously exploiting to advise and influence our design. These are (a) *observing pupils’ work*, (b) *interviewing teachers*, (c) *analysing pupils’ workbooks and on-screen content*, (d) *evaluating Emil Y4 pilot PD sessions*, and (e) *analysing the data obtained from deploying Emil Y3*. In Table 1, we depict in which context of the process which instrument advised us as the primary resource (✓) or as the supplementary one (✓).

As stated earlier, we worked together with pupils and their teachers in two design classes during all 19

lessons and considered this having a unique and irreplaceable impact on the intervention design. Usually all three of the researchers/authors were present in running the lesson or observing the pupils, talking to them, and writing down notes. After each lesson, we spent an extra time with teacher T1 in her office, analysing and assessing the lesson, activities, pilot worksheets, and pedagogy. Following the analysis and redesign (usually taking one or two weeks) significantly informed the programming environment development, programming language and content development, and plus, helped us refine the presentation of the tasks and inspired us in designing the new ones.

Besides the first three instruments of reflection (observing pupils, working with teachers, and assessing pupils’ solutions of the pilot content), we also made significant use of the other two. In the later stage of the project, we initiated the pilot PD sessions with the generalist primary teachers (already using Emil Y3 in their classes) as we know that each opportunity to work with the practitioners in any design research project is a valuable source of feedback and it might be late to get it only when the development is completed. During the entire DR process for Y4, the intervention for Y3 had been used in schools in four different countries. That provided our research team with rich and unique opportunities to collect various data from the teachers and their pupils and exploit them in our design for Y4.



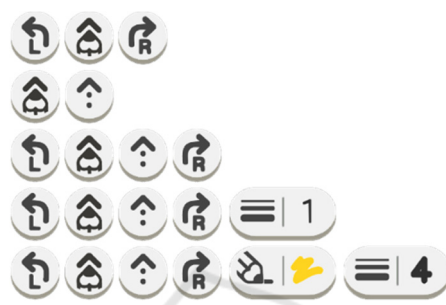




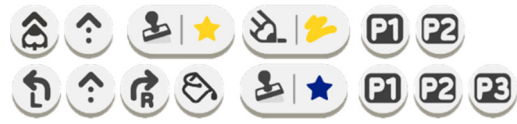

The most significant finding that resulted from the concluding analysis of the DR process was when we looked at the range of the command sets in the individual tasks of the overall gradation. In general, the range enlarges itself from two simple commands at the beginning to a complete⁷ programming language of Emil Y4.

Table 1: Contexts of Y4 DR process and different instruments of reflection used to inform it.

instruments of reflection	observing pupils’ work	interviewing teachers	analysing content	inputs from PD sessions	analysing Emil Y3 data
contexts					
programming environment	✓	✓			
programming language	✓	✓	✓		
Y4 content design	✓	✓	✓	✓	✓
Emil Y4 workbook	✓	✓	✓	✓	✓
PD strategy & teacher material		✓	✓	✓	✓

⁷ Naturally, not in the sense of Turing completeness.

Table 2: Gradation of the tasks, revealing the 'pulsating range of the command set'.

<p>1 move within the grid; stamp this shape</p>	
<p>2 turn left and right; stamp this shape; set (choose) colour for stamp</p>	
<p>3 draw line; turn; move and draw; set colour and size for the draw command</p>	
<p>4 draw a closed outline of a polygonal area; fill it with a chosen colour</p>	
<p>5 use pin to draw oblique line in the grid; draw and fill polygons with oblique sides</p>	
<p>6 use compound command, prebuilt by authors</p>	
<p>7 use one or two compound commands plus some other basic commands</p>	
<p>8 modify, debug and finally define from scratch our own compound commands</p>	
<p>9 use complete programming language of Emil Y4</p>	

However, when we lined up the ranges of all the tasks and reduced repetitive or similar lines, a pulsating structure was revealed (see Table 2). It clearly illustrates one of our key design principles: *Every new and, therefore, cognitively more demanding concept that pupils are going to explore should be presented in a way that they can fully focus on with as little distraction as possible*. In our context, it means working with as few commands as necessary. The ‘pulsating waves’ that have become readable in the analysis identify exactly these moments: The table shows distinct segments of the gradation ① to ⑨, separated by clear milestones of the intended learning process. While in Table 2, we explain which commands are used in each segment, here we characterise the milestones:

- ① Intuitive entry into the basic moves of Emil. We make use of stamping so that pupils immediately create something new on Emil's stage such as a visible output product.
- ② Towards the end of ① in A3, a kind of cognitive tension is intentionally built: pupils start commenting the need to make Emil turn. The very next task (see first line of segment ②) will allow the **right** turning Emil while the following tasks will help in turning Emil to the **left**.
- ③ A new type of moving Emil forward is added, namely, drawing a line segment. This is highlighted in B4 by having only two commands available: moving forward (to the next grid point) and drawing a line segment (to the next grid point).
- ④ Pupils discover how to reposition Emil – not by a command of the language but by dragging. They also discover how to colour an area they have just specified by drawing its closed polygonal outline.
- ⑤ Drawing in direct control mode leads to programming oblique lines within the grid. Pupils discover that they have to drop a pin to start such a line, make Emil draw and finally pull the pin at the end.

From segment ⑥ until the end of the intervention, compound commands become the central computational concept of the gradation. Through all the following tasks, a thread⁸ winds, focused on various operations with compound commands. We identified and had a trial of whether these operations were developmentally appropriate for the pupils. This helped

⁸ We use this word to denote various sub sequences of the tasks of the global gradation, not necessarily closely following each other, that step by step develop deep understanding of a certain concept. The thread that deals with compound commands has already started in the last part of Emil Y3, plays central role in Emil Y4, then continues

in the gradual discovery of this key means of abstraction in programming. The final analysis of the DR process helped us articulate those steps in even more detail than as in milestones ⑥ to ⑨. They are:

- In direct control mode, pupils **use** pre-defined compound command **P1** with five basic commands (without settings or inputs).
- In direct control mode, they use **P1** plus other basic commands.
- They use **P1** and **P2** that, in their pre-definitions, use basic commands with settings.
- Pupils continue developing their emerging understanding of general repetition by repeatedly applying **P1**
- Still in direct control, they use the pre-defined **P1**, **P2** and **P3** together with other basic commands.
- They **modify** the last command in the pre-defined **P1**, then use it to solve a problem.
- They modify a command inside the definition of **P2** by analogy with **P1** and then use both commands together (plus other basic commands) to solve a problem.
- They **program P2** by analogy with **P1** as **P2** should draw similar shape as **P1**.
- They program **P1** so that the given sequence of direct control commands (including **P1**) will draw a given outcome.
- They program **P2** which uses **P1** and then use **P2** repeatedly.
- Finally, as the ultimate step of the gradation in the context of *my own compound commands*, pupils program **P1**, then **P2** which uses⁹ **P1**, and then **P3** which uses **P2** – to solve a complex task.

It is rewarding for the designers/researchers to observe pupils – and their generalist primary teachers – solving the tasks that cannot be solved without building compound commands and exploiting them. The whole design research endeavour would however remain unfinished with its potential unharnessed if we did not analyse the process thoroughly, strive to understand and formulate its structure, criteria for iterative revisions, and the lessons learnt. Due to the limited space, we could not comment on all of those. Instead, we decided to focus on analysing the programming language of Emil Y4 and studying some properties of the duration of the pupils’ learning process.

by using and building *my own blocks* in Scratch, see (Kalas, 2017), and later by using and defining functions in Python.

⁹ In CS style we say **P2** calls **P1** plus other basic commands, **P3** calls **P2** plus other commands but itself.

ACKNOWLEDGEMENTS

Our thanks go to several hundreds of teachers whom we have met lately in countless PD sessions across different countries, online or in person. Observing their work and having the interactions together provided us with unique experiences to help improve the interventions. Our special thanks belong to the teachers who looked for all possible ways on how to run their computing lessons despite the pandemic situation. Our thanks also go to Indicia, the non-for-profit organisation funding the development.

No research ethical principle (in the sense of Petousi, Sifaki, 2021) has been breached in the project. This work has been funded in part by VEGA Slovak Agency under project *Productive gradation of computational concepts in programming in primary school 1/0602/20*, and Slovak Research and Development Agency under the Contract no. APVV-20-0353.

REFERENCES

- Abelson, H., diSessa, A.A. (1980). *Turtle Geometry: The computer as a medium for exploring mathematics*, MIT Press, Cambridge.
- Anderson, L.W., Krathwohl, D.R., Bloom, B.S. (eds.) (2000). *A taxonomy for learning, teaching, and assessing: A revision of Bloom's Taxonomy of Educational Objectives*. New York, Longman.
- Bakker, A. (2018). *Design Research in education: A practical guide for early career researchers*. Routledge.
- Benton, L., Kalas, I., Saunders, P., Hoyles, C., Noss, R. (2018a). Beyond jam sandwiches and cups of tea: An exploration of primary pupils' algorithm-evaluation strategies. *J of Comp Assisted Learning* 5; 590–601.
- Benton, L., Kalas, I., Hoyles, C., Noss, R. (2018b). Designing for learning mathematics through programming: A case study of pupils engaging with place value. *Int J of Child-Computer Interaction* 16, 68–76.
- Blackwell, A.F. (2002). What is Programming? Proc. of *14th Workshop of the Psychology of Programming Interest Group*, pp. 204–218.
- Brennan, K., Resnick, M. (2012). New frameworks for studying and assessing the development of computational thinking. Proc. of the *2012 Annual Meeting of the American Edu Research Association*, Vancouver.
- Chiocciariello, A., Leccioli, N.C., Oreste, C. (1993). Four steps to the right. In: diSessa, A.A., Hoyles, C., Noss, R., Edwards, L.D.: *Computers and Exploratory Learning* (eds.), doi.org/ 10.1007/978-3-642-57799-4.
- Design-Based Research Collective: Design-based research (2003). An emerging paradigm for educational inquiry. *Educational Researcher*, 32 (1), 5–8.
- Garrison, D.R., Anderson, T., Archer, W. (2001). Critical thinking, cognitive presence and computer conferencing in distance education. *American Journal of Distance Education*, 15 (1), 7–23.
- Grover, S., Pea, R. (2013). Computational thinking in K-12: A review of the state of the field. *Educational Researcher* 42 (1), 38–43.
- Kalas, I., Benton, L. (2017). Defining procedures in early computing education. In: Tatnall, A., Webb, M. (eds.): *Tomorrow's learning: Involving everyone. Learning with and about technologies and computing. WCCE 2017. IFIP Advances in Information and Communication Technology*, vol 515. pp. 567–578. Springer, Cham.
- Kalas, I. (2018a). Programming in lower primary years: Design principles and powerful ideas. Proc of *Constructionism, Computational Thinking and Educational Innovation*, pp.71–80, Vilnius.
- Kalas, I., Blaho, A., Moravcik, M. (2018b). Exploring control in early computing education. In: Sergei N. Pozdniakov and Valentina Dagienė (eds.) *Informatics in Schools. Fundamentals of Computer Science and Software Engineering. ISSEP 2018*. LNCS, vol 11169, pp. 3–16. Springer, Cham.
- Kalas, I., Horvathova, K. (2022). Programming concepts in lower primary years and their cognitive demand, accepted for IFIP OCCE 2021 DTEL post conference book.
- Lincoln, Y.S., Guba, E.G. (1985). *Naturalistic enquiry*. Beverly Hills, CA: Sage, 416 p.
- Lye, S.Y., Koh, J.H. (2014). Review on teaching and learning of computational thinking through programming: What is next for K-12? *Computers in Human Behaviour* 41, pp.51–61.
- Meerbaum-Salant, O., Armoni, M., Ben-Ari, M. (2013). Learning computer science concepts with Scratch. *Computer Science Education* 23 (3), 239–264.
- Papert, S. (1980). *Mindstorms. children, computers, and powerful Ideas*, 230 p. Basic Books, New York.
- Petousi, V., Sifaki, E. (2021). Contextualizing harm in the framework of research misconduct. Findings from a discourse analysis of scientific publications, *Int J of Sustainable Development*, 23 (3/4), 149–174.
- Plomp, T., Nieveen, N. (eds.) (2013). *Educational design research. Part A: An Introduction*, 204 p. SLO, Netherland.
- Royal Society (2012). Shut down or restart: The way forward for computing in UK schools, royalsociety.org/education/ policy/computing-in-schools-report/.
- Sentance, S. (2019). Moving to mainstream: developing computing for all. In: *Proceedings of WiPSCE 2019*, doi.org/10.1145/3361721.3362117.
- Van den Akker, J. (2013). Curricular development research as a specimen of educational design research. In: (Plomp, 2013), pp. 52–71.
- Watt, S. (1998). Syntonicity and the psychology of programming. In Proc of *10th Annual Workshop of Psychology of Programming Interest Group*, pp. 75–86.