Programming Concepts in Lower Primary Years and Their Cognitive Demands

Ivan Kalaš[(✉)] and Klara Horvathova

Comenius University, Bratislava, Slovakia
{ivan.kalas,klara.horvathova}@fmph.uniba.sk

Abstract. In our Computing with Emil project, currently in its 4th year, we are engaged in the design and research focused on productive constructionist learning of programming at the primary level, as a continuum from year to year. We are exclusively concerned with *computing for all learners* approach, being implemented by generalist primary teachers in their own classes. For similar purposes, national curricula usually list computational concepts, which pupils should learn and, in general agreement among educational experts, are considered developmentally appropriate for the primary school. However, in our work, we feel the vocabulary being normally used for this is too coarse-grained to clearly specify the learning goals, especially in the area of programming. Therefore, for each computational concept, we try to identify a set of related operations that primary pupils should learn in any particular year. In this research, we tried to verify whether our approach is comprehensible for teachers and prolific in outcomes for their pupils. We wanted to find out whether they realise that (a) different operations performed with each concept have different cognitive demands, and (b) these demands determine the arrangement of activities in an intervention. We addressed a large group of teachers who had already participated in our professional development sessions on the intervention for Year 3. We chose repetition as one of the concepts and designed six assessment tasks focused on various operations that pupils perform with it. We did not inform them of how we rank the tasks; we asked them to solve them and rank from the simplest to the most difficult, and explain their decision. We analysed the collected data by various methods, and in the paper, we discuss our findings. Teachers correctly distinguished different operations and helped us better understand challenges of projecting and assessing conceptual understanding.

Keywords: Programming in lower primary · Concepts and their cognitive demands · Generalist primary teachers' perspective

1 Introduction

In our research, we strive to better understand learning process of the primary level pupils (aged 5 or 6 to 10 or 11) in the area of computing and its programming. This arena is meagrely explored, especially if we are primarily interested in *computing for all* approach [1], i.e., in computing implemented in a regular class with all learners, possibly as a mandatory subject or integrated in mathematics. Additionally, in our case,

we are to build the educational content such that the generalist primary teachers feel confident to teach it themselves and consider it prolific for their pupils.

In this project, we focused on the assessment of cognitive demands [2] of the programming concepts which, according to the research literature and our own experience, we consider to be developmentally appropriate for primary school. Such concepts are often only briefly listed in policy documents, curricula or research reports, as e.g., in the current English programme of study at KS2 [3]: … *use sequence, selection, and repetition in programs; work with variables and various forms of input and output*. In our approach, however (see [4] or [5]), we prefer to use more detailed vocabulary when setting educational goals and projecting learning progression for individual years in KS1 and 2. Specifically, for each concept whose understanding we want to develop, we identify operations which we consider appropriate to perform with that concept. Subsequently, in the design research process [6], we transform the operations into gradation of activities and iteratively evaluate them in our partner primary classes.

Although such design requires a long period of time and is being evaluated in numerous iterations, here we present how we additionally tried to validate whether our *concepts and related operations* strategy is productive, i.e., whether primary pupils (and teachers) are aware of the differences between individual operations related with the concepts, perceiving them as different also in terms of their cognitive demands.

2   Programming Concepts at Primary Level and Their Difficulty

Although not always used with identical meaning and scope, several curricular documents exploit the strategy of identifying programming concepts which pupils are meant to learn (see e.g. [7, 8]). Similarly, many authors in their research and development reports take comparable approaches while presenting their area of interest and formulating their research problems. The concrete lists of concepts sometimes reflect selected programming environment. Bers [9] refers to *children learning basic concepts and powerful ideas of coding* in ScratchJr as they use programming blocks spanning *concepts from simple sequencing… to control structures*, and addresses *sequencing, repeat loop, sensors, and conditional statements* when they use KIBO.

To assess development of pupils' computational thinking in Scratch, Brennan and Resnick [10] uses three key dimensions: computational concepts (namely *sequences*, *loops*, *parallelism*, *events*, *conditionals*, *operators*, and *data*), computational practices (*being incremental and iterative*, *testing and debugging*, *reusing and remixing*, and *abstracting and modularising*), and computational perspectives (*expressing*, *connecting*, and *questioning*). When Meerbaum-Salan et al. [11] investigate whether Scratch can be used to teach concepts of computer science, they studied pupils' understanding of *initialisation* (of position, direction, costume etc. up to variables), *bounded*, *conditional*, *and unbounded repeated execution*, *variables*, *conditional execution*, *message passing*, *events*, and *concurrency*.

In K-12 Computer Science Standards [12] the authors name the concepts for ages 8–11 in chapter Algorithms and Programming by stating: *Create programs that use variables to store and modify data* and *Create programs that include sequences,*

events, loops, and conditionals (p. 10) and also provide more specific justifications plus several developmentally appropriate examples of those concepts in use.

When designing their informatics curriculum, Dagiene et al. [13] specify each concept together with one or two related operations, situating them into recommended age group. For instance, *loops without parameters* (age group 7+) are presented as *First pupils recognise repetition of patterns in programs and can shorten the program description by applying repeat-loop. Secondly pupils recognise possible repetitions in task descriptions… and design programs with loops* (p. 350). Some authors specify the need to study cognitive demands of programming, such as Blackwell [2] in his discussion about why programming is deemed difficult. In 1976, Perlman [14, 15] also suggested considering *cognitive difficulties* while developing interventions for children.

Although the researchers and content designers usually have a clear idea of a certain appropriate order of the studied programming concepts, deeper understanding and systematic research in this direction is still in its early stage. Therefore, we aim to try to verify if the strategy of identifying different operations related to each programming concept and sorting them by their different cognitive demands is productive to design structure and learning progressions in curricula.

2.1   Computing with Emil: Project Background

The reason why we consider the combination of parallel research and development to be exceptionally fruitful is that we research to ensure that we can develop competently. Moreover, we develop so that we can study corresponding learning processes. These intertwined activities signify the essence of a research strategy named design research (DR for short) [6], which we employ. At the initiation of our Computing with Emil project in 2017, we wished to not only systematically support learning in informatics through all primary years, but also equip ourselves with powerful research instruments. Exploiting them already [16], we explore one of our key design principles – distinguishing between *direct manipulation*, *direct control*, and *programming*.

At the level of content development, we strive to support programming as a continuum in all primary years. We want pupils to experience it as an opportunity to encounter powerful ideas of computing, as an instrument for their computational curiosity, a way of thinking about and solving problems from different areas. So far, we have deployed Emil for Years 3 and 4 and Robotics with Ema for Years 1 to 4. All of these interventions complement each other and support a holistic learning progression, together with the ScratchMaths interventions for Years 5 and 6 [17].

Concepts, Operations, and Tasks.   In Emil Y3, we focused on the following powerful ideas of computing: *experiencing order and structure* (in data and commands), *learning to control and plan*, *coping with constraints*, *considering state and tools*, *learning to aggregate and abstract*, and *thinking about programs*. From the perspective of the programming concepts, we decided to concentrate on *sequence*, *repetition* (continued in Y4), and pre-concept of *procedure* (as a preparation for Y4). As with other interventions, in the DR process of Emil Y3, we proceeded by identifying

different operations related to each concept, and implemented them in small activities that constructed the whole gradation (systematic progression) of tasks. In Emil Y3, the overall gradation consists of 150^+ tasks. Some of them have multiple solutions, a few do not have any solution (and teachers invite pupils to explain why), while other tasks are partially 'unclear'. In this case, the teachers suggest that the group discuss the task, complete its wording and solve it. Thus, pupils become co-designers of the activities. Discussions and collaboration in the group are the most important part of our pedagogy.

As an example, let us present how the operations related to *repetition* were identified in Emil Y3, here listed according to their increasing demands (as validated in the DR process). Here we refer to Fig. 1, which shows growing expressive power of the language along with the interface support for the way in which commands are displayed as small blue cards on the record/programming panel above Emil's stage:

- Directly control Emil using the *basic commands*. An *external record* of the steps taken is built in parallel on the panel (see an example on Fig. 1(i)).
- Experience the limitation of the panel's extent, which sometimes only possess 4 or 5 positions. This may affect or even preclude the solution of a problem.
- Read a given record (a simple sequence of *basic commands*) and recognise whether any command in the record is repeated more than once in a row. In Fig. 1(i) there are three such groups: two *baskets* to pick a mushroom, three *watering cans* to water a cherry tree to (1) grow, (2) bloom, and (3) ripen, and two *arrows down*.
- In direct control, discover a mechanism that any *basic command* repeated more than once in a row is automatically and immediately gathered in a *stack of commands* (see Fig. 1(ii)). This gathering is highlighted by an animation on the panel. In the example we can see three such stacks.
- Program a solution of a problem, making use of the fact that identical cards in a row are gathered in stacks. This saves positions on the panel considerably.
- When programming a solution, discover a procedure on how two consecutive *basic commands* can be merged on the panel into one 'double card', representing a pair of commands. Identical double cards in a row are then automatically gathered in *stacks of double cards* (see Fig. 1(iii) and (iv)).
- Build, read, analyse, explore, explain, simplify and compare different programs using *basic commands*, *stacks of basic commands* and *stacks of double commands*.
- When programming a solution, generalise the merging and stacking mechanism to three *basic commands*, which can later be changed to four (see Fig. 1(v)). Build, read, analyse, explore, explain, simplify and compare such programs.

Fig. 1. Different levels of the ways in which commands are displayed on the panel.

Our intervention for Y3 began to be implemented in primary schools in Slovakia, Czech Republic, and in smaller pilot hubs of schools in Norway and England a year before the pandemic. At that time, we also finalised the evaluation of intervention for Y4 and started running professional development (PD) sessions for teachers. Admirably, many schools continued using the interventions with their pupils in one way or another through the pandemic. The actual situation of 2020/21 school year, however, did not allow us to work directly with the pupils. Therefore, at this stage of the research, we focused our effort on the teachers, as we set out to study whether they realise the difference between varying operations which pupils perform with the same computational concept and how they characterise these operations. Therefore, we formulated our first research question as: *Are the teachers aware of different operations which pupils perform with each computational concept and how would they characterise them?* (RQ1) We decided to focus on repetition as one of the central concepts of the intervention in Y3 and designed a set of six paper-based assessment tasks. They represent six different operations and were intended to examine whether teachers are aware of the different cognitive demands of the operations. Therefore, this aspect was reflected in our second research question: *How do teachers perceive cognitive demands of different operations pupils do with the same concept?* (RQ2)

3   Method

Firstly, the participants of our research are characterised. We focused on all teachers who had already participated in our PD sessions[1] aimed at Emil Y3. This comprised (as of 30 April 2021) of 1079 teachers, 823 Slovak and 265 Czech. Out of these teachers, 302 agreed to proceed with our research assignment. As the final *volunteer sampling*, 62 teachers (21%) responded and undertook the challenge, as presented in Sect. 3.1.

---

[1] That is, one day session of face to face or two afternoons in the online model.

We will be referring to them as our *teacher sample*. These 62 teachers represent 49 schools, 37 Slovak and 12 Czech. More details of the same are presented in Fig. 2: in (a) it shows that 36 of the sample are primary teachers, 23 are lower secondary teachers who teach both primary and lower secondary pupils, one currently teaches lower secondary pupils only but is interested to know our intervention, and two are 'other'. In (b), we see that 27 teachers have already used the intervention in their class(es) and 35 have worked with it only in the PD session(s). In (c), we see the breakdown of the sample with respect to when the teachers participated in the PD session, grouped into six half-years, starting by the second half-year of 2018.

In parallel, we also identified 13 Slovak and Czech expert informatics researchers[2] and educators in the academic population with the same request. We received eight responses, therefore deciding to use them as another validation of our method (see below). In this paper, we will be referring to this purposive sample as our *expert*.
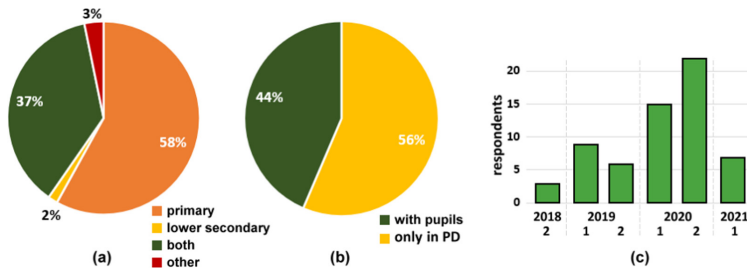

Fig. 2. Breakdown of the teacher sample.

3.1 Designing the Assessment Tasks

For this research, we designed six assessment tasks similar to activities performed by pupils (and the teachers) with repetition in Y3 intervention. Yet, the assessment tasks are dissimilar to the activities of the intervention as they are not open-ended and can be unambiguously assessed (although tasks 3 and 4 have multiple solutions). We ordered the tasks in the way corresponding to the increasing cognitive demands as implemented in the intervention. For the needs of the data analysis and the presentation of the findings, we numbered and colour-coded the data. Table 1 briefly characterises the essence of each task, while Fig. 3 illustrates one of the tasks.

---

[2] Familiar with our approach and our interventions.

Table 1. Cognitive characterisation of the assessment tasks focused on repetition in Y3.

Task 1		analyse a given record (program) of the single *basic commands*, recognise and mark each group of single commands repeated in the record next to each other
Task 2		read a given record of the *basic commands* and *the stacks of basic commands* and execute it; draw a resulting path of Emil through the stage
Task 3		plan a solution, i.e., program a path using *basic commands* and *stacks of basic commands* (single cards), with limited number of steps
Task 4		plan a solution, i.e., build a program that uses *basic commands* and a given 'double command', with limited number of steps
Task 5		read a given program with *stacks of cards* and *double cards* of the basic commands, and indicate the *basic command* Emil will run first, second, third etc.
Task 6		read a given program with *stacks of cards, double cards and triple cards*, and indicate the *basic command* Emil will run first, second, third etc.

3.2 Data Analysis

We randomly shuffled the set of our assessment tasks and distributed them (in the same random order) to all teachers of the volunteer sample of 302 and our purposive sample of 13 experts. We asked them to (a) solve the tasks, (b) rank them by the cognitive demands – as they perceive it – from the easiest to the most difficult one, (c) explain the reasons for ranking each task as they did, and (d) share the scans of their solutions. We obtained 70 complex responses (eight in the expert sample and 62 in the teacher sample) and analysed the data using various descriptive and analytical statistical methods, along with analyses of the explanations and the solutions. Although we applied various methods to collected rankings, we consider this research endeavour as qualitative and harness all results as a means to:

- Validate the approach of *refining granularity* of the programming concepts specifications in school informatics by identifying gradations of operations connected with each concept. Thus, we wish to learn more about how to design structure of the content and support the learning trajectory of the pupils in informatics.
- Inform our effort to better understand cognitive demands of the operations, thus improving the understanding of the programming/algorithmic thinking development at the primary stage.

To achieve this, we formulated research questions RQ1 and RQ2 (see above). As discussed in the final section of the paper, in the second period, we plan to extend this research by addressing the pupils of the teachers involved.
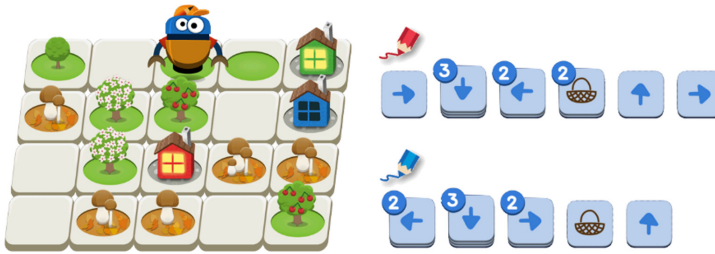
Fig. 3. Task 2, as presented to the teachers, with the following assignment: *This is Emil's stage and two records of the paths that he could take.* **Draw both paths through the stage.**

For brevity, we present here only one method out of several which we used to analyse the collected data. Namely, we wanted to consider the degree of similarity between our understanding of the cognitive demands of the different operations' pupils in Y3 perform in the context of repetition represented as a tuple (1, 2, 3, 4, 5, 6), and collected rankings of the tasks' demands as perceived by the teachers represented as permutations of the initial tuple. To measure the ordinal association between two tuples, we used the Kendall rank correlation coefficient [18]. The value of the coefficient lies between the interval of −1 (a perfect disagreement, in this case for (6, 5, 4, 3, 2, 1)) and 1 (a perfect agreement). The correlation is high when an assessment of the perceived cognitive demands of the tasks is similar to the intended one.

Kendall's correlation coefficient, referred to as *Kendall's tau*, is calculated as the difference between the number of *concordant* and *discordant* pairs within the tuple, divided by the binomial coefficient of the tuple's length. From the CS perspective, the Kendall's tau can be easily derived[3] from the minimal number of necessary swaps between the items of a tuple when sorted by bubble sort. For example, if the perceived ranking is (2, 3, 4, 1, 6, 5), four swaps are needed to sort the tuple and the similarity between the tuple and (1, 2, 3, 4, 5, 6) is 0.467 when represented as the Kendall's tau.

4   Results

To investigate whether teachers are aware of the differences between various operations performed by the pupils with repetition in Y3, we asked them to solve our six assessment tasks and rank the task they considered the easiest to the most demanding. The resulting rankings can be analysed from two (closely related) alternative perspectives: we can review, which of our tasks was ranked by the participants as the easiest, which was ranked as the 2nd etc., while the other possible view would consider which rankings were assigned to our task 1, which rankings were assigned to our task 2 etc. Both perspectives proved to be interesting and instructive.

---

[3] More precisely, tau = (n – 2 * m) / n, where n is the minimal number of swaps needed in the worst case and m is the minimal number of necessary swaps for this particular tuple.
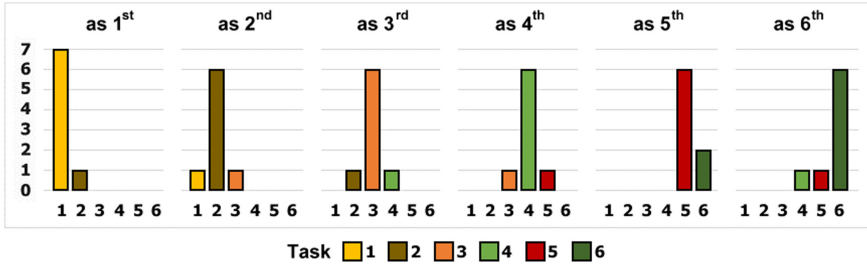
Fig. 4. Expert sample (n = 8) rankings of the tasks. (Color figure online)

In Fig. 4, the rankings of the experts are analysed (n = 8). For example, our task 1 (yellow) was assessed seven times as the simplest one and once as the 2nd. Task 2 was once assessed as the easiest, six times as the 2nd and once as the 3rd. In general, we see that experts significantly validated our understanding of the increasing cognitive demands of the different operations, although the variance in their rankings slightly increases towards more demanding tasks. As we noticed similar trend in the teacher sample, we will provide our interpretation of this in the discussion.
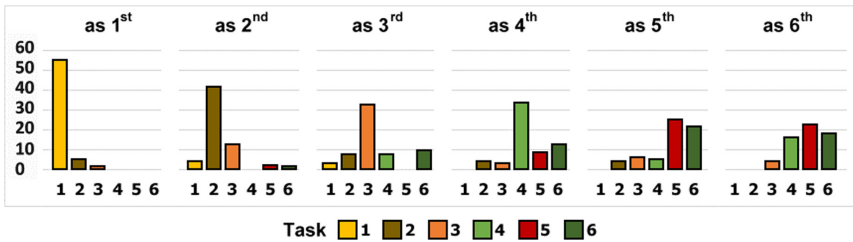

Fig. 5. Teacher sample (n = 62) rankings of the tasks. (Color figure online)

In Fig. 5, we see corresponding visualisation of the teacher sample (n = 62) rankings. In general, it shows that teachers perceive the differences between the operations related to repetition in Y3 and classify their cognitive demands in accordance with the experts and the authors. However, the variance in their rankings is significantly higher and considerably increases towards the harder tasks, with our tasks 5 and 6 being undistinguished in their cognitive demands. This is clearer when we use the alternative perspective on the rankings (see Fig. 6), showing the variability of the tasks which experts and teachers assessed as the 1st (the easiest), as the 2nd, and so on.
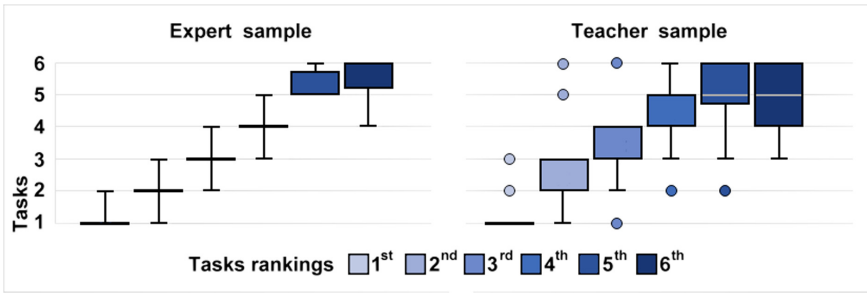
Fig. 6. Variation of the rankings in the expert and teacher samples.

Single points located *distant* from either the lower or upper quartiles (i.e., outliers) are depicted here as small circles. Note that in the expert rankings, only the last two tasks show non-zero interquartile range (Q_1, Q_2 and Q_3 being identical), with medians still being 5 in the 5th place and 6 in the 6th place. The variability in the teacher rankings is significantly higher, with the medians being 1, 2, 3, 4, 5, and 5. Clearly, while all operations are being recognised by the teachers as different, the interquartile range grows considerably from left to right and the cognitive demands of our tasks 4, 5 and 6 become blurred, especially in the last place of the rankings.
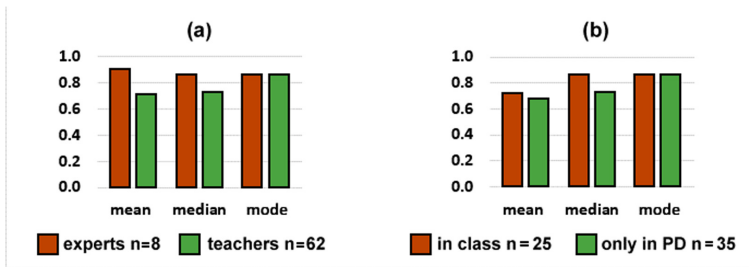

Fig. 7. Kendall's tau correlations (a) in expert and teacher samples, and (b) in the breakdown of the teacher sample into those who have used the intervention in the class and who have not yet.

To analyse the data from another perspective, we measured the ordinal association between collected rankings (cognitive demands of the tasks as perceived by the experts and by the teachers) and our implemented ranking, deriving the Kendall rank correlation coefficient (Kendall's tau) for each. Figure 7 demonstrates a few basic statistics descriptors of the results, after excluding one outlier with negative tau (based on the related textual comments). In (a) certain differences in the expert and teacher samples can be seen, most significantly in the mean and median. Diagram (b) describes the rankings of the teachers (n = 62) breakdown into those who have already used the intervention in their classes and those who have so far taken part only in the PD session. Any significant difference in those two data sets was not observed, except that the *in-class* data set has higher median and its upper 50% scores are between 1 and

0.8667, i.e., only one or less necessary bubble sort swap between the items of their rankings and our understanding of the increasing cognitive demands (1, 2, 3, 4, 5, 6).

4.1 Experts and Teachers' Justifications of Their Rankings

Another source for our research effort, interesting in the context of both RQ1 and RQ2, were the textual explanations provided by the experts and teachers, in which they justified their rankings. During the analysis, we relied on the corresponding correlation coefficients of the rankings and our initial intended ranking of the tasks' demands. The first important finding achieved in this manner is that all participants are clearly aware of the differences between operations with the same concept and can characterise them properly. Furthermore, we obtained better insight into why the variance in the rankings increases towards more demanding tasks (Table 2).

Table 2. Selected justifications of the experts' and teachers' rankings.

Experts confirming intended ranking (with Kendall's tau between 0.8667 and 1)
- (Task 1) *simple representation, simple reading of a sequence of commands*
- (Task 2) *stepping through a program with simple stacks of single basic commands*
- (Task 3) *understanding basic commands, their repetition, the rules of moving Emil*
- (Task 4) *looking for a solution which uses given double card repeatedly*
- (Task 5) *understanding what double card represents, how to step through a program*
- (Task 6) *same as 5, but containing triple cards*

Experts justifying different rankings (with Kendall's tau between 0.7333 and 0.8667)
- (Task 3 as the 4th) *programming with limited number of steps, considering alternatives*
- (Task 4 as the 6th) *programming with a given double card*
- (Task 5 as the 6th) *understanding how repetition works*

Teachers confirming intended ranking (with Kendall's tau > 0.5)
- (Task 1) *simple representation, no algorithm, only reading a sequence of simple steps*
- (Task 2) *reading a program with known notation, stepping through simple program*
- (Task 3) *programming simple steps and stacks of simple steps*
- (Task 4) *planning a solution with stacks of simple steps and double steps*
- (Task 5) *understanding the repetition of double cards*
- (Task 6) *harder than 5 because it requires reading a program with triple cards*

Teachers justifying different rankings (with Kendall's tau $\leq$ 0.5)
- (Task 3 as the 1st) *simple programming with commands gathered in stacks*
- (Task 5 as the 2nd) *simple counting the commands*
- (Task 6 as the 3rd) *understanding the order of the steps, understanding the notation*
- (Task 6 as the 4th) *understanding the order of the steps when double cards and triple cards are used in the program*
- (Task 3 as the 5th) *programming with limited number of steps, understanding the number of repetitions*
- (Task 3 as the 6th) *programming (planning a solution) is required*

5   Discussion

In this research, our aim was to investigate whether our *granularity refinement strategy* (as we put it in [19]) is productive in setting learning goals. We tried to not only better understand which concepts are developmentally appropriate for the primary stage, but to assess this adequacy in more detail, in the context of the various operations that pupils have to perform with each concept in different KS2 years.

We contacted more than 300 teachers and computing education experts, who had already attended PD session(s) on our intervention for Y3 and asked them to solve a set of six assessment tasks, ranking them from the easiest to the most difficult and comment on their choice. We gathered 62 + 8 responses and applied various statistical methods to the data obtained in this way. For instance, we studied the degree of similarity between our understanding of the cognitive demands of the different operations implemented in the tasks and participants' own rankings.

Both expert and teacher samples clearly confirmed our assessment of different cognitive demands of the operations and their proper gradation. However, especially in the teacher sample, the variance in the rankings increased towards more demanding tasks, with tasks 5 and 6 becoming almost equal in their demands. Hence, when planning or analysing the suitability of activities gradation for certain age group and expectations, at least two factors must be considered: one stems from the inherent cognitive demands of different operations pupils must perform with a concept, the other represents the pedagogy applied. Naturally, there are several ways depicting how the intervention can implement specific progression (still respecting the inherent gradation of the cognitive demands). Our pedagogy is based on the constructionist epistemology, which always starts with concrete experience and *direct control* [16] to acquire an opportunity to discover new operation (in close collaboration in the pairs and regular whole class discussions), new mechanism or new reaction, such as the one that repeats *commands* automatically gather in a *stack of commands*, Fig. 1(ii to v). Only then the gradation proceeds to *programming control* [ibid] and practising a new piece of knowledge. This, however, has the consequence that if pupils or teachers have already made a discovery, new operation or mechanism becomes known and as such 'simple': in the assessment tasks they see well known notation and its meaning they already know.

In spite of this, all experts and teachers in general agreed on our perception of growing difficulty of the operations, with tasks 1, 2, 3 always preceding 4, 5 and 6, with 4 easier than 5 and 5 easier than 6 for most of the experts. Most importantly, all experts and teachers were aware of the differences between the operations related to repetition, correctly naming and distinguishing between them.

We also noticed other interesting details, which, however, need to be followed in more detail, such as whether the rankings are influenced by the time span since attending the PD session, whether they already have real experience from the class (Fig. 7b) etc.

References

1. Sentence, S.: Moving to mainstream: developing computing for all. In: Proceedings of the WiPSCE 2019 (2019). https://doi.org/10.1145/3361721.3362117
2. Blackwell, A.F.: What is programming? In: 14th Annual Workshop of Psychology of Programming Interest Group, pp. 204–218 (2002)
3. Department for Education: National curriculum in England: computing programmes of study (2013). www.gov.uk/government/publications
4. Gujberova, M., Kalas, I.: Designing productive gradations of tasks in primary programming education. In: Proceedings of the WiPSCE 2013, pp. 109–118. ACM, Aarhus (2013)
5. Kalas, I., Benton, L.: Defining procedures in early computing education. In: Tatnall, A., Webb, M. (eds.) WCCE 2017. IAICT, vol. 515, pp. 567–578. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-74310-3_57
6. Plomp, T., Nieveen, N. (eds.): Educational Design Research. Part A: An Introduction. SLO, Enschede (2013)
7. CSTA and ACM: Computer Science K-8: Building a Strong Foundation (2012)
8. MŠMT: Rámcový vzdělávací program pro základní vzdělávání, Praha (2021)
9. Bers, M.U.: Coding, playgrounds and literacy in early childhood education: the development of KIBO robotics and ScratchJr. In: Proceedings of the 2018 IEEE Global Engineering Education Conference (EDUCAN), pp. 2094–2102 (2018)
10. Brennan, K., Resnick, M.: New frameworks for studying and assessing the development of computational thinking. In: Proceedings of the 2012 Annual Meeting of the American Edu Research Association, Vancouver (2012)
11. Meerbaum-Salant, O., Armoni, M., Ben-Ari, M.: Learning computer science concepts with Scratch. Comput. Sci. Educ. **23**(3), 239–264 (2013)
12. CSTA: K-12 Computer Science Standards, Revised 2017 (2017). www.csteachers.org/page/standards
13. Dagiene, V., Hromkovic, J., Lacher, R.: Designing informatics curriculum for K-12 education: from concepts to implementations. Inf. Educ. **20**(3), 333–360 (2021)
14. Perlman, R.: Using Computer Technology to Provide a Creative Learning Environment for Preschool Children. AI Memo 360. MIT, Cambridge (1976)
15. Morgado, L., Cruz, M., Kahn, K.: Radia Perlman: a pioneer of young children computer programming. In: Current Developments in Technology-Assisted Education, pp. 1903–1908. Formatex, Badajos (2006)
16. Kalas, I., Blaho, A., Moravcik, M.: Exploring control in early computing education. In: Pozdniakov, S.N., Dagienė, V. (eds.) ISSEP 2018. LNCS, vol. 11169, pp. 3–16. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-02750-6_1
17. Benton, L., Hoyles, C., Kalas, I., Noss, R.: Bridging primary programming and mathematics: some findings of design research in England. Digit. Experiences in Math. Educ. **3**(2), 115–138 (2017). https://doi.org/10.1007/s40751-017-0028-x
18. Cohen, L., Manion, L., Morrison, K.: Research Methods in Education. Routledge, London (2017)
19. Webb, M., et al.: Moving on with informatics/computer science curricula – challenges and opportunities. In: Symposium Presented at IFIP TC3 OCCE 2021 DTEL (2021)