ISSEP 2022
15th International Conference on Informatics in Schools

Local Proceedings

Vienna, Austria, Sep. 25th – Sep. 28th, 2022

Andreas Bollin and Gerald Futschek (eds)

Preface

The International Conference on Informatics in Schools: Situation, Evolution, and Perspectives (ISSEP) is a forum for researchers and practitioners in Informatics education in both primary and secondary schools. The conference provides an opportunity for educators and researchers to reflect upon the goals and objectives of this subject matter, its curricula, various teaching and learning paradigms, and topics, as well as the connections to everyday life — including the various ways of developing Informatics Education in schools.

The ISSEP conference started in Klagenfurt, Austria, in 2005. It was followed by meetings in Vilnius (2006), Torun (2008), Zürich (2010), Bratislava (2011), Oldenburg (2013), Istanbul (2014), Ljubljana (2015), Münster (2016), Helsinki (2017), Saint Petersburg (2018), Larnaca (2019), Tallinn (2020), and Nijmegen (2021). This year the conference took place in Vienna, Austria, from Sep. 26th to Sep. 28th, 2022, and it was combined with a Doctoral Consortium on Sep. 25th, 2022.

In this volume, we are pleased to collect the papers that were accepted for presentation during the 15th International Conference on Informatics in Schools (ISSEP 2022) and that were not selected for the Springer LNCS proceedings. ISSEP 2022 had 24 full and four short paper submissions, and 12 full papers were selected for inclusion in Springer LNCS. Six other full papers and a short paper were accepted for presentation at the conference and are included here in the local proceedings. Additionally, the conference offered three invited presentations, eight different workshops in two parallel workshop sessions, and a poster session with seven poster presentations. Their abstracts are also included in these proceedings. The Doctoral Consortium attracted 10 Ph.D. students who presented and intensively discussed their work at the consortium with five international professors in the field of informatics didactics. Their work is included as extended abstracts in these proceedings, too. All submissions (including those of the Doctoral Consortium and the workshops) have been reviewed by at least three international reviewers in a double-blind peer review process using EasyChair.

We want to thank all the authors and presenters for their qualitative contributions and the Program Committee, who invested a lot of effort in reviewing and discussing the papers presented at the conference. We also would like to express our deepest thanks to the three keynote speakers, Torsten Brinda, Enrico Nardelli, and Sue Sentance, for their availability and talks, and all the local organizers and their team for making this event possible.

Andreas Bollin
Gerald Futschek
Klagenfurt/Vienna, 2022

Organization

Conference Chairs

Andreas Bollin	University of Klagenfurt, Austria
Gerald Futschek	Vienna University of Technology, Austria

Steering Committee

Andreas Bollin	(Chair) University of Klagenfurt, Austria
Valentina Dagienė	Vilnius University, Lithuania
Yasemin Gülbahar	Ankara University, Türkiye
Juraj Hromkovič	Swiss Federal Institute of Technology, Zurich, Switzerland
Ivan Kalas	Comenius University, Slovakia
Erik Barendsen	Radboud University and Open University, The Netherlands
Sergei Pozdniakov	Saint Petersburg Electrotechnical University, Russia

Program Committee

Andreas Bollin	(Chair) University of Klagenfurt, Austria
Peter Antonitsch	University of Klagenfurt, Austria
Andrej Brodnik	University of Ljubljana, Slovenia
Špela Cerar	University of Ljubljana, Slovenia
Christian Datzko	Wirtschaftsgymnasium und Wirtschaftsmittelschule Basel, Switzerland
Monica Divitini	Norwegian University of Science and Technology, Norway
Gerald Futschek	(Chair) Vienna University of Technology, Austria
Juraj Hromkovič	ETH Zurich, Switzerland
Mile Jovanov	Ss. Cyril and Methodius University Skopje, North Macedonia
Kaido Kikkas	Tallinn University, Estonia
Dong Yoon Kim	Ajou University, South Korea
Dennis Komm	ETH Zurich, Switzerland
Mart Laanpere	Tallinn University, Estonia
Martina Landman	Vienna University of Technology, Austria
Peter Larsson	University of Turku, Finland
Marina Lepp	University of Tartu, Estonia
Nina Lobnig	University of Klagenfurt, Austria
Birgy Lorenz	Tallinn University, Estonia
Piret Luik	University of Tartu, Estonia
Maia Lust	Tallinn University, Estonia
Kati Mäkitalo	University of Oulu, Finland
Tilman Michaeli	TU Munich, Germany

Mattia Monga	Università degli Studi di Milano, Italy
Tauno Palts	University of Tartu, Estonia
Stefan Pasterk	University of Klagenfurt, Austria
Hans Põldoja	Tallinn University, Estonia
Sergei Pozdniakov	Saint Petersburg Electrotechnical University, Russia
John-Paul Pretti	University of Waterloo, Canada
Ralf Romeike	Freie Universität Berlin, Germany
Barbara Sabitzer	Johannes Kepler Universität Linz, Austria
Carsten Schulte	University Paderborn, Germany
Giovanni Serafini	ETH Zurich, Switzerland
Vipul Shah	ACM India CSpathshala Education Initiative, India
Gabrielė Stupurienė	Vilnius University, Lithuania
Reelika Suviste	University of Tartu, Estonia
Maciej Syslo	UMK Torun, Poland
Michael Weigend	University of Münster, Germany
Albin Weiss	University of Klagenfurt, Austria
Markus Wieser	University of Klagenfurt, Austria

Doctoral Consortium Committee

Valentina Dagienė	(Chair) Vilnius University, Lithuania
Andreas Bollin	University of Klagenfurt, Austria
Gerald Futschek	Vienna University of Technology, Austria
Barbara Sabitzer	Johannes Kepler Universität Linz, Austria
Carsten Schulte	University Paderborn, Germany

Local Organizers

Gerald Futschek	(Chair) Vienna University of Technology, Austria
Franziska Tiefenthaller	(Organization) Vienna University of Technology, Austria
Martin Krajiczek	(IT support) Vienna University of Technology, Austria
Peter Kompatscher	(Event management) Vienna University of Technology, Austria
Stefan Pasterk	(Publicity chair) University of Klagenfurt, Austria
Melanie Ottowitz	(Publicity support) University of Klagenfurt, Austria

Table of Contents

Poster Session

Workshops

Doctoral Consortium

Keynotes

What all teachers should know about Informatics (Abstract)

Torsten Brinda

University of Duisburg-Essen: Essen, Nordrhein-Westfalen, Germany

1. Abstract

In recent years, the digital transformation of our everyday lives has increasingly led to a corresponding transformation in the education system to prepare students best possible for the digitalization-related requirements. During this process, numerous international, national, and federal competence frameworks were developed that describe which digitalization-related competences students and teachers will need in the future. Many of these frameworks set priorities in the area of teaching and learning with digital media, but the role of dealing with digitization as a learning object and thus the role of computer science was also increasingly emphasized. The required anchoring of teaching and learning with and about digital technologies in all school subjects leads to the question, which competences teachers will need for their work in the future across various recommendations in this area. At the University of Duisburg-Essen in Germany, a competence framework was developed as part of the work of an interdisciplinary working group, which wants to be understood as an attempt to integrate various existing frameworks and models in the broader field. A key sub-question is which computing competences teachers need for their digitalization and profession-related activities. For this purpose, an approach is presented that was developed by a working group of the German Informatics association with the aim of developing recommendations for computing education of non-computer science teachers. Closely related to this is the question of curricular anchoring in teacher education. For this purpose, the procedure and first results of a large joint project funded by the German Ministry of Education and Research are presented, in which a modular approach to computing education of non-computer science teachers was developed and tested with participants from all phases of teacher education.

2. CV

Torsten Brinda is a Professor for Computing Education Research at the University of Duisburg-Essen in Germany. His current research focusses on programming competency modelling,

Informatics in Schools. A step beyond digital education. 15th International Conference on Informatics in Schools: Situation, Evolution, and Perspectives, ISSEP 2022, Vienna, Austria, September 26-28, 2022

torsten.brinda@uni-due.de (T. Brinda)

https://www.ddi.wiwi.uni-due.de/team/torsten-brinda (T. Brinda)

0000-0001-6137-4258 (T. Brinda)

students' conceptions and interests, as well as on the contribution of informatics in the digital transformation of education. He has published about 150 papers in journals and conference proceedings. He is the current chair of IFIP working group 3.1 "Informatics and digital technologies in school education" and the past chair of the technical committee "Informatics and Education / Didactics of Informatics" within the German Informatics association (GI).

Informatics Education: the grand challenge for a XXI century school (Abstract)

Enrico Nardelli

Università di Roma "Tor Vergata", Rome, Italy

1. Abstract

Advances in Informatics, the scientific discipline behind digital technologies, are radically transforming society, affecting both professional and personal life. For the first time in the history of humanity, cognitive functions - once the exclusive privilege of people - are carried out by machines. Education is a key element to deal with this disruptive scenario but, unfortunately, the general level of Informatics education is rather shallow, and usually stops at the operational level of digital competencies. Instead, a science based approach, focused on understanding the principles of Informatics early in school, is needed to enable all citizens to actively contribute to the development of an economic and social prosperity for all.

2. CV

Enrico Nardelli is full professor of Informatics at University of Rome "Tor Vergata" and the President of Informatics Europe, the association representing Informatics university departments and research labs in Europe (https://informatics-europe.org). He is also member of the ACM Europe Council.

Since 2014 he coordinates "Programma il Futuro" (https://programmailfuturo.it), a project run by CINI (National Interuniversity Consortium in Informatics), in accordance with the Italian Ministry of Education, to introduce in Italian schools the basic concepts of Informatics as a scientific discipline. The project is present in 80% of the Italian schools and is followed by 3 million students.

He is the director of the national laboratory "Informatics and School" of CINI (where he has served as Management Board member from 2013 to 2021) and member of the Steering Committee of the "Informatics for All" coalition (https://informaticsforall.org), advocating the introduction of Informatics as component of fundamental education in all schools in Europe.

Informatics in Schools. A step beyond digital education. 15th International Conference on Informatics in Schools: Situation, Evolution, and Perspectives, ISSEP 2022, Vienna, Austria, September 26-28, 2022

nardelli@mat.uniroma2.it (E. Nardelli)
https://www.mat.uniroma2.it/~nardelli/ (E. Nardelli)
0000-0001-9451-2899 (E. Nardelli)

His current research activity is Informatics Education and interdisciplinary study of Informatics systems and their social impact, within the Link&Think Research Lab (https://link-and-think.blogspot.com) and the CINI National laboratory “Informatics and Society”.

Previously he did research in various fields of Informatics, from algorithms to databases, from geographical information systems to man-machine interaction and cooperative information systems.

He also carries out dissemination activity towards the general public regarding Informatics education and the role of Informatics in the digital society.

Addressing gender balance: the need for a multi-faceted approach (Abstract)

Sue Sentance

Raspberry Pi Computing Education Research Centre, University of Cambridge, United Kingdom

1. Abstract

Gender balance in computing education in most countries has long been a well-documented problem, along with the universal desire to broaden participation in computing more generally. Barriers identified in research that impact on gender include a lack of belonging, lack of relevance, lack of encouragement and teaching approach. Introducing mandatory computing into the school curriculum is one way of ensuring that all children have access to the same opportunities, especially at an early age when they may be less influenced by stereotypes existing in society and the media.

In this keynote, I will look at some of the research that has been carried out in this area to establish what we already know. I will then share details of a 3-year research project, consisting of a number of randomised-controlled trials and other interventions, that we have been running in England since 2019 with the aim of finding out what approaches might improve gender balance in computing in school. The results from this study are just emerging (but should all be available by the time I'm speaking at ISSEP!) but point to the need of a multi-faceted approach to addressing gender balance, including whole-school as well as subject-specific approaches.

2. CV

Sue Sentance is Chief Learning Officer at the Raspberry Pi Foundation and Director of the Raspberry Pi Computing Education Research Centre at the University of Cambridge, UK. She received her PhD in Artificial Intelligence from Edinburgh University in 1993, and since then has worked as a secondary teacher, teacher trainer, university lecturer and researcher, before joining the Raspberry Pi Foundation in 2018. At the Raspberry Pi Foundation, Sue's role is to advise on teaching and learning, and lead on research on computing education for young people. She has played a leading role in the DfE-funded National Centre for Computing Education, particularly around the development of the Teach Computing Curriculum, Isaac Computer

Informatics in Schools. A step beyond digital education. 15th International Conference on Informatics in Schools: Situation, Evolution, and Perspectives, ISSEP 2022, Vienna, Austria, September 26-28, 2022
sue@raspberrypi.org (S. Sentance)
https://www.mat.uniroma2.it/~nardelli/ (S. Sentance)
0000-0002-0259-7408 (S. Sentance)

Science, and online teacher professional development courses. She also leads the Gender Balance in Computing research programme. In 2020 Sue was awarded a Suffrage Science award for Maths and Computing and in 2017 the BERA Public Engagement and Impact Award for her services to computing education.

Full Paper Sessions

An Exploration of High School Students' Self-Confidence while Analysing Iterative Code

Claudio Mirolo*, Emanuele Scapin

University of Udine, 33100 Udine, Italy

Abstract

A number of studies on novice programming report that loops and conditionals can be potential sources of errors and misconceptions. We then felt the need to engage in a more systematic and in-depth investigation about the teaching and learning of iteration in some representative high schools of our regional area. As a medium-term outcome of this endeavour we expect to get fine-grained insights about the nature of students' difficulties, on the one hand, as well as to identify possible pedagogical approaches to be adopted by teachers, on the other. As a step of this project, we designed and administered a survey composed of a set of small tasks, addressing students' understanding of iteration in terms of *code reading* abilities. After summarising the motivations underlying the choice of the tasklets and the overall structure of the instrument, in this paper we will focus on a particular aspect which has not yet received extensive attention in the computer science education literature. Specifically, we will consider students' perception of self-confidence, in connection with their actual performance in each task, the specific program features, the cognitive demands (procedural vs. higher-level thinking skills), and the use of code vs. flow-charts. A noteworthy result of this analysis is that students' perception of self-confidence is poorly correlated to actual performance in the task at hand. The main implications of our study are twofold, pertaining our understanding of less conspicuous facets of the learning of iteration as well as possible pedagogical strategies to strengthen metacognitive skills.

Keywords

Informatics education, Program comprehension, High school, Iteration constructs, Novice programmers, Metacognition

1. Introduction

Learning to program is a complex, "slow and gradual process" [1]. Students' difficulties are well known and have been extensively investigated for tertiary education (e.g. [2, 3, 4]). Also such basic flow-control constructs as conditionals and loops turn out to be potential sources of several mistakes and misconceptions for novice learners [5, 6]. However, the overall picture as to the teaching and learning of iteration at the upper secondary level is still a bit fragmentary and calls for more systematic study.

In the attempt to make some progress in this direction, focusing on the high school context in our regional area, we are collecting teachers' and learners' insights from a range of perspectives,

Informatics in Schools. A step beyond digital education. 15th International Conference on Informatics in Schools: Situation, Evolution, and Perspectives, ISSEP 2022, Vienna, Austria, September 26–28, 2022

*Corresponding author.

claudio.mirolo@uniud.it (C. Mirolo); emanuele.scapin@uniud.it (E. Scapin)

0000-0002-1462-8304 (C. Mirolo); 0000-0001-8384-8231 (E. Scapin)

including subjective perception, instructional practice, ability and challenges to achieve program-related tasks. In particular, we have used an instrument designed to investigate students' ability and perception of self-confidence when analysing small programs based on iteration constructs. In accordance with Izu's et al. program comprehension perspective [7], the proposed test is based on a set of small *code reading* tasks, or *tasklets*.

After summarising the rationale behind the chosen tasklets and the structure of the instrument, in this paper we will mainly focus on a particular aspect which has not yet been deeply investigated in computing education, and particularly at pre-tertiary level. Specifically, we will consider students' perception of self-confidence in connection with their actual performance, the program features, the implied cognitive demands and the use of textual code vs. flow-charts.

Being able to monitor one's "current level of mastery and understanding" while trying to accomplish a task is indeed an important *metacognitive* competence [8], which appears to be particularly crucial in computer science [9]. As pointed out by Brown and Harris [10], "from both psychometric and learning theory perspectives, the accuracy of self-assessment [...] is critical. If self-assessment processes lead students to conclude wrongly that they are good or weak in some domain and they base personal decisions on such false interpretations, harm could be done, even in classroom settings."

The design of the tasklets and of the related questions was guided by the outcome of some preliminary work, namely; (i) A number of interviews asking experienced high-school teachers about the role of iteration in their practice and the related learning issues [11]; (ii) A "pilot" survey addressed to students, including questions about their subjective perception of difficulties as well as three short exercises on basic iteration constructs [12].

The rest of the paper is organised as follows. After mentioning in Sect. 2 some relevant background, Sect. 3 outlines scope, main features and organisation of the developed instrument. Then, Sect. 4 summarises the major findings concerning high school students' perception of self-confidence. Finally, in Sect. 5 we conclude with a few remarks and future perspectives.

2. Background

Starting from the pioneering work on the cognitive implications of programming tasks in the early 1980s, e.g. [13, 2], empirical research has persistently shown that flow-control constructs such as conditionals and loops are frequent sources of errors and misconceptions for novice learners [5, 6, 14], especially when combined into nested constructs [15]. The reasons may be manifold, ranging from lack of problem solving skills to the need for accuracy and strenuous practice. According to Perkins et al. [16] programming is indeed "problem-solving intensive," in fact it requires multiple skills [17], and students may fail to develop a viable model of the underlying *notional machine* [18] or may not be able to see code execution at higher levels of abstraction to infer a program's purpose [19]. Moreover, part of the students' difficulties may also be related to the habits and expectations of both teachers and learners [17].

Here the focus is on students' perceived self-confidence while achieving a task, in fact one of the issues raised by our previous pilot study [12]. Being able to fairly assess the degree of self-confidence in the devised solution is connected to metacognitive skills, that can be generally meant as "one's knowledge about one's cognitive processes" [20]. While extensively

investigated in other fields, e.g. [21], metacognition is recently receiving wider attention also for programming education [22, 23]. Most often metacognitive skills have been explored in connection with students' success in tasks or exams [24, 25, 26] and some educators see pedagogical value in providing learners with opportunities to reflect on their metacognitive awareness under the guidance of the teacher [24, 20]

3. Characterisation of the instrument

The results of our pilot studies focused on iteration involving teachers [11] and students [12], also in light of the issues discussed in the literature on novice programming, raised a number of questions that seemed to us worth further investigation, in particular:

Q1. To what extent are students at ease with a range of features implied by iteration? (E.g., loop conditions, nesting of flow-control constructs, …)

Q2. How does the effort to trace code execution impact on the analysis of more abstract program properties?

Q3. Are students more at ease when using flow-chart or textual code representations of programs?

Q4. To what extent does students' perception of self-confidence correspond to their actual achievement in a task?

It is precisely around the above questions that our investigation instrument has been designed. To this aim, we have first identified the general areas and topics outlined in Table 1, where each area is labeled with the questions Q1–4 it pertains to. Then, we have defined two sets of 6 tasklets, whose scope covers the areas and topics listed in Table 1 in as balanced a way as possible. The size and complexity of a *taskset* are chosen in such a way that students could reasonably complete their work on it in about one hour — i.e. before losing concentration.

Each individual tasklet presents a program based on iteration constructs and asks one or two multiple-choice questions about its related properties. When the question brings into play high-level thinking skills (area A in Table 1), the student is also required to assess in a 4-grade Likert scale her/his perceived level of self-confidence on the provided answer. In addition, the programs of three tasklets may be shown either as textual code or as flow-charts (line FC in Table 1). Finally, this instrument has been made accessible as an online survey, the taskset and the code/flow-chart format of selected tasklets being assigned randomly.

To accommodate for the common practice in the considered high schools, the textual code is Java-like. In the following we will elaborate a little more on each of the areas reported in Table 1. For more details about the individual tasklets, English translations of the four versions (two tasksets × code/flow-chart modes) of the test-survey are available via the links provided in the online appendix http://nid.dimi.uniud.it/additional_material/iteration_tasksets.pdf .

A. Tasklets addressing higher-order thinking skills. This is a central area for our investigation and each tasklet includes at least a question of this type. It covers two broad

Table 1
Areas and topics addressed by the tasklets.

A. Tasklets addressing higher-order thinking skills (Q2, Q4)
   1. Abstraction on the computational model
      a. Equivalence (nested constructs, `for`/`while`, `do-while`/`while` …)
      b. Reversibility
   2. Relationships with the application domain
      a. Completion (condition, expression, statement …)
      b. Functional purpose

B. Tasklets addressing code features (Q1, Q3)
   1. Structural features
      a. Plain loop
      b. Nested conditional
      c. Nested loop
   2. Processing plan
      a. Exit condition
      b. Loop control variable
      c. Downward `for` loop

B. (continued)
   3. Conditions
      a. Simple condition
      b. Composite condition
      c. Boolean expression/variable

C. Tasklets addressing code execution, conceivably via tracing (Q1, Q2)
   1. Output/final state
   2. Number of iterations

D. Tasklets addressing data types (Q1)
   1. Numerical data (only)
   2. Non-numerical data
   3. Array data

SC. Perception of self-confidence (Q4)

FC. Flow-chart versus code (Q3)

categories, concerning abstraction over the computation structure and functional abstraction in connection with some problem domain. To test students' abilities in the former category we ask *equivalence* [27] and *reversibility* [28] questions. The tasklets in the latter include more common types of questions which ask either to choose an appropriate item (e.g., a condition, an arithmetic expression, a statement) to complete a program with a given purpose or to identify the intended purpose of a given program — in fact an instance of the recurrent "Explain in Plain English" theme [19]. As an example, Figure 1 shows an equivalence question with four answer options. Both equivalence and reversibility tasks require students to reason about program behaviour comprehensively, generalising what could be ascertained by tracing code execution for specific input data. The role of reversibility in learning, in particular, dates back to Piaget's work on cognitive development, where it is considered as an indicator of achievement of the concrete operational stage, and according to a neo-Piagetian perspective the learning stages apply regardless of age when approaching new knowledge domains [29]. Thus, reversibility seems to be an appropriate instrument to assess their comprehension in the early stages of learning to program.

B. Tasklets addressing specific code features. The code features listed in Table 1 are easily recognisable in a tasklet by program inspection and are connected with the findings of our pilot work [11, 12], indicating loop conditions and nested constructs as major sources of difficulties — nested loops being seen as the hardest challenge in the learners' subjective perception. Widespread issues and misconceptions regarding nested constructs, in particular, have also been identified in a number of studies [30, 31, 32, 15]. Additional difficulties worth consideration that can be ascribed to loop structures arise in the treatment of loop-control variables, see e.g. [2],

```
int x = m;
int y = n;

while ( x != y ) {

  while ( x < y ) {
    x = x + m;
  }
  while ( x > y ) {
    y = y + n;
  }
}
output( "x = " + x );
```

reference program

```
int x = m;
int y = n;

while ( x != y ) {

  while ( x < y ) {
    x = x + x;
  }
  while ( x > y ) {
    y = y + y;
  }
}
output( "x = " + x );
```

option 1

```
int x = m;
int y = n;

while ( x != y ) {

  do {
    x = x + m;
  } while ( x < y );
  do {
    y = y + n;
  } while ( x > y );
}
output( "x = " + x );
```

option 2

```
int x = m;
int y = n;

while ( x != y ) {

  if ( (x < y) || (x > y) ) {
    x = x + m;
    y = y + n;
  }
}
output( "x = " + x );
```

option 3

```
int x = m;
int y = n;

while ( x != y ) {

  if ( x < y ) {
    x = x + m;
  } else {
    y = y + n;
  }
}
output( "x = " + x );
```

option 4

T1.4 (ii) – *Which option is equivalent to the reference program?* The input requirements are that both the values of `m` and `n` are positive (> 0) integers and two programs are meant to be equivalent if the final states of their executions are always the same, whenever the initial states are the same — and provided the initial states comply with the given input requirements.

Figure 1: Example of equivalence question.

and while dealing with down-counting loops as opposed to more stereotypical up-counting loops, as pointed out in [33].

C. Tasklets addressing code execution. Small problems that can be solved at a low *operational* level by tracing the code execution are quite common, since tracing is deemed to be a basic ability "to build [...] higher-level comprehension skills upon" it [34]. In any case such ability should not be taken for granted; many students struggle, for instance, with tracing loops, especially `while`-loops [35]. Here, however, our main purpose is to address the investigation question Q2: whether code tracing can to some extent support higher-order thinking *in the task at hand*. In [12] we found indeed some cues suggesting that students' performance on more abstract tasks may improve when they are actually led to engage in some careful tracing, but that they tend to elude this effort to check their conjectures about program behaviour. Thus, the idea is to ask questions pertaining to the "abstract" area A, either including or not a previous question that can be answered via tracing, in particular to determine the program output or the overall number of iterations for given input data.

D. Tasklets addressing data types. The covered data types are essentially *numbers*, *booleans*, *characters*, *strings*, and *arrays*. The indexed access to arrays, in particular, can be problematic for novices, especially in connection with iteration — see e.g. the recent work [36, 37].

SC. Perception of self-confidence. Each tasklet requires to reason about a given program by asking at least one question in the area A, and after answering this question students must also

indicate their perceived level of self-confidence in a Likert scale ranging from 1 (not confident at all) to 4 (fully confident). Besides the reasons mentioned in the introduction and the potential pedagogical implications [24, 20], we included this feature since our study [12] suggested that students' perception of difficulty and actual performance in a task tend not to be consistent.

FC. Flow-chart versus code. According to [38], "flowcharts support novice programmers [...] and give guidance to what they need to do next, similar to a road-map." They may also help to identify difficulties and misconceptions [14]. On the other hand, Ramsey's et al. findings [39] seem to indicate that the use of flow-charts may not be natural for students and that working with code often gives rise to better performances. Thus, we tried to investigate more broadly on this topic: the impact of using flow charts should result from comparing the outcomes for two randomly assigned versions of the same tasklet, where the program is presented as flow chart vs. textual code.

4. Data collection and analysis of the results

The test-survey has been administered to 225 students from 16 high schools disseminated in the considered geographical area. 76% of the students followed a technical program, 15% a scientific one and 9% other types of curricula. Their age range was 15–18; more specifically 18% were attending the 2nd year, 54% the 3rd and 28% the 4th year (over 5 years of upper secondary instruction). All the students have engaged in the test in controlled situations, either during classwork (166) or at the beginning of a summer workshop (59). The data provided through the survey have been collected and processed in anonymous form.[1]

Of the investigation questions introduced in Section 3, Q4 has an explicit focus on students' perception of self-confidence, whereas Q1–3 may be addressed both in terms of performance and self-confidence. As mentioned earlier, here we will take the latter perspective and consider students' performance only in order to answer Q4. In the following analysis, we will refer to the tasklets (available online — see section 3) by labels in the format T$S.K$, where $S = 1|2$ denotes the taskset and $K = 1 \div 6$ the K-th tasklet in that taskset, possibly followed by the specific question, either (i) or (ii), when two multiple choice questions are asked.

Q1 – Self-confidence for different code features. The bar diagram reported in Figure 2 shows the distribution of students' perceived levels of self-confidence on their answers concerning abstract properties of the programs occurring in the tasklets. In summary, we can make the following observations:

- Both tasklets requiring to deal with boolean variables (T1.3 and T2.1) have been perceived as especially challenging.
- Most students feel not self-confident with string-processing code (T1.6, T2.3, T2.5); the only possible exception is tasklet T2.6, but only about 20% of the subjects is fully confident on their achievement and this is in fact the tasklet that registered the worst performance.

[1] Since no personal information was shared with any third party, and neither the students nor their institutions could be identified through the presented data, the research policies of our country do not require the approval by an ethics commission.

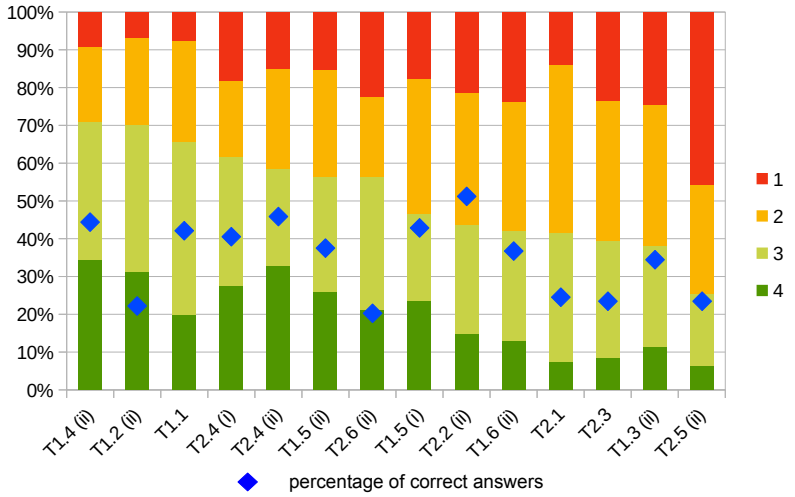


Figure 2: Students' perception of self-confidence for each question addressing higher-order thinking skills — sorted by decreasing "positive" level of self-confidence (Likert levels 3 and 4).

- Programs that carry out purely numerical (hence mathematical) computations, on the other hand, are perceived as easier to understand (T1.1, T1.2, T1.4); an exception in this respect is tasklet T2.2, which however requires to master the equivalence between different iteration constructs (`for`, `while` and `do-while`).
- Other code features, including nested constructs, seem to be perceived as more or less troublesome, depending on the context in which they occur.

For the sake of exemplification, the question shown in Figure 1 (T1.4) received the highest level of overall self-confidence. At the other extreme, of lowest self-confidence, the tasklet T2.5 includes a reversibility question for a string-processing program based on a plain loop.

It may be worth noting that most figures of our analysis do not distinguish between heterogeneous subgroups of students since we intend to convey a general idea about their perceived self-confidence, independently of specific contexts. However, the average overall level of self-confidence turned out to be essentially the same for technical (2.56) and scientific schools (2.55), and just a little lower for other types of schools (2.42). The differences appear more pronounced, on the other hand, over subsequent years of instruction, indicating an increase from the 2nd (2.20) to the 3rd year (2.70) and, unexpectedly, a decrease from the 3rd to the 4th year (2.46).

Q2 – Self-confidence in connection with tracing. We found no evidence that previous effort to trace code execution can increase the perceived level of self-confidence while analysing a program at a more abstract level. In particular, we may contrast the self-confidence bars in Figure 2 for T1.3(ii) and T2.4(ii). Both questions ask to identify a program's functional purpose: in the former case after tracing the program to count the iterations for a given input; in the latter after answering a more abstract reversibility question. The impact of tracing on program comprehension should however be investigated more accurately by means of a specifically designed instrument.

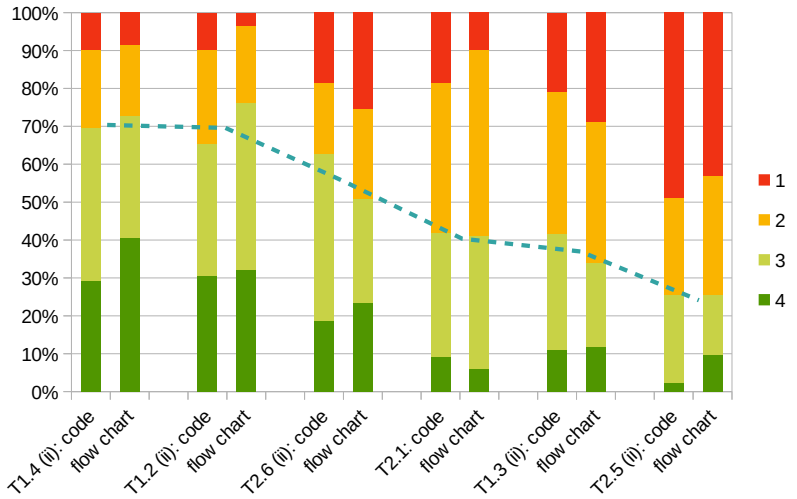


Figure 3: Students' self-confidence while analysing flow charts vs. textual code for all the tasklets whose programs could be presented in both forms; dashed green line: average percentages of "positive" levels of self-confidence (3 + 4).

Q3 – Impact of flow charts on self-confidence. As clearly illustrated by the bar diagram in Figure 3, students' self-confidence does not appear to be affected by reasoning on flow charts rather than on textual code — and in fact their performance does not either. Also in terms of average levels of self-confidence when using flow charts vs. textual code, the outcomes are hardly distinguishable: 2.76 vs. 2.69 for the first taskset and 2.26 vs. 2.25 for the second one.

Q4 – Self-confidence vs. performance. The most significant insight of the investigation, from a pedagogical perspective, is that students' perception of self-confidence on the provided solution is very poorly correlated with their actual performance in the task. To devise a quantitative measure, the options of the multiple-choice questions have been subdivided into three groups: *correct* answers, *incorrect* answers and *severely incorrect* answers.[2]

To begin with, after collecting the data the instrument turned out to be very well calibrated for the sake of the intended analysis, since it resulted into a balanced tripartition of the answers: 35% correct, 30% incorrect and 35% severely incorrect (more specifically, 37%, 29% and 34% for taskset 1; 33%, 31% and 36% for taskset 2).

Then, in order to establish a reasonable correspondence with the Likert range 1–4, these three groups have been assigned weights 4, 2 and 1, respectively. On this basis, Pearson's correlation coefficient between self-confidence and performance is only 0.235 for taskset 1 and 0.208 for taskset 2 (less than, for example, the correlation measured in similar tests in university courses [25, 26]). The correlation is low for most tasklets, the only exception being the 0.598 level for T2.4(ii), and it is not statistically significant for five items of the second taskset ($p > 0.05$); moreover, for three items of the first taskset $0.02 < p < 0.05$.

Alternatively, things can also be seen from a complementary standpoint by looking at individual subjects: in this respect the correlation between self-confidence and performance

[2]Refer to the online appendix to see how the options provided for each task have been classified in these terms.

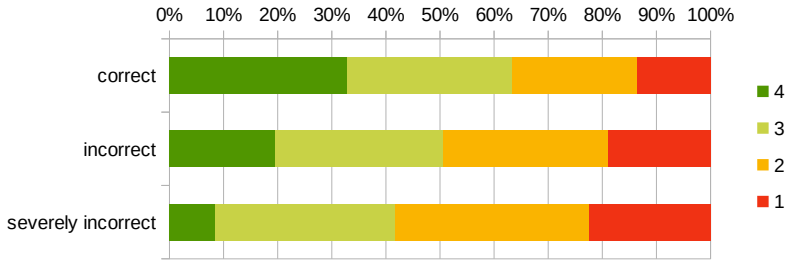


Figure 4: Visualisation of the poor correlation between self-confidence on the proposed solution and actual achievement in the tasklets.

is *negative* for 35% of the students. The general situation is probably better illustrated by the diagram in Figure 4, where we can see that, on the whole, about half of the incorrect answers as well as more than 40% of the severely incorrect answers are nevertheless associated with a positive perception of self-confidence on the provided solution. Conversely, more than 1/3 of the correct answers are paired with a negative perception of self-confidence.

In particular, remarkably high rates of positive self-confidence (Likert levels 3 and 4) for incorrect and severely incorrect answers have been registered for tasklet T1.2 (72%), for the equivalence question T1.4(ii) reported in Figure 1 (63%), for tasklet T1.1 (62%) and for T2.6(ii) (57%). Just to give a rough idea of the scope of these tasks, T1.2 asks a reversibility question about a `while` loop with a composite condition and a nested `if-else`. In T1.1 the student is required to identify the appropriate condition of a plain loop carrying out a simple numerical computation. Finally, T2.6 asks again to assess equivalence between simple `do-while` and `while` constructs used in string-processing programs.

5. Conclusions

In this paper we have attempted to analyse high school students' perception of self-confidence while engaging in small program comprehension tasks. In Section 3 we have outlined the main motivations underlying the choice of the tasks and the overall structure of the used instrument. In Section 4 we have then summarised a range of findings concerning high school students' perception of self-confidence. This will hopefully help to build a more detailed picture about the understanding of iteration in the high school context.

In a nutshell, the major results of our analysis can be stated as follows. First, students seem to be more at ease with mathematical computations than with string-processing tasks (what may be partly a consequence of the examples customarily presented in class), but boolean variables are probably troublesome to them. Second, being required to trace code execution does not appear to have a significant impact on the understanding of more abstract program properties. Third, our results do not support previous findings [39] that flow-charts may not be natural for students: the levels of self-confidence when using flow charts or textual code are essentially the same. The most salient outcome is, however, that students' perception of self-confidence is very poorly correlated to their actual performance in a task, what is likely to indicate weakness of metacognitive skills.

In light of the last observation as well as of the role of metacognitive skills in effective learning [40], the main implications of our study pertain to devising pedagogical strategies to strengthen learners' awareness in this respect. According to Schraw [21], indeed, metacognitive awareness *can* be taught. Thus, a more ambitious project could aim to elicit the *reasons why* students feel more or less confident about the provided solutions, from their own perspective, and to explore the impact of teaching metacognitive skills explicitly — in particular, if the type of tasks used in our present study could also be exploited for these purposes.

Currently, the test-survey has been administered to more than 200 students, but we are trying to broaden the scope of this empirical investigation, if possible by involving other interested educators working in different contexts. It would be especially interesting to compare the perception of self-confidence of girls versus boys. Indeed, besides noting that — unsurprisingly — their average level of self-confidence is slightly lower, we are unable to draw statistically significant insight from our present sample since the girls are only about 8% (19). It may also be worth elaborating on and extending the tasksets outlined here for instructional purposes. Indeed, we think that teachers could use them both as examples to illustrate different aspects connected to iteration and as instruments to assess students' understanding of this topic.

References

[1] E. W. Dijkstra, On the cruelty of really teaching computing science, Communications Of The ACM 32 (1989) 1398–1404.

[2] B. Du Boulay, Some difficulties of learning to program, Journal of Educational Computing Research 2 (1986) 57–73.

[3] A. Robins, J. Rountree, N. Rountree, Learning and teaching programming: A review and discussion, Computer Science Education 13 (2003) 137–172.

[4] A. Luxton-Reilly, Simon, et al., Introductory programming: A systematic literature review, in: Proc. Companion of the 23rd Annual ACM Conf. on Innovation and Technology in Computer Science Education, ITiCSE 2018, ACM, New York, USA, 2018, pp. 55–106.

[5] L. C. Kaczmarczyk, E. R. Petrick, J. P. East, G. L. Herman, Identifying student misconceptions of programming, in: Proc. of the 41st ACM Technical Symp. on Computer Science Education, SIGCSE '10, ACM, New York, 2010, pp. 107–111.

[6] Y. Cherenkova, D. Zingaro, A. Petersen, Identifying challenging cs1 concepts in a large problem dataset, in: Proc. of the 45th ACM Tech. Symp. on Computer Science Education, SIGCSE '14, ACM, New York, NY, USA, 2014, pp. 695–700.

[7] C. Izu, C. Schulte, et al., Fostering program comprehension in novice programmers: Learning activities and learning trajectories, in: Proc. of the ITiCSE Working Group Reports, ITiCSE-WGR '19, ACM, New York, NY, USA, 2019, pp. 27–52.

[8] J. D. Bransford, A. L. Brown, R. R. Cocking (Eds.), How People Learn: Brain, Mind, Experience, and School – Expanded Ed., National Academy Press, Washington, D.C., 2000.

[9] K. Falkner, R. Vivian, N. J. Falkner, Identifying computer science self-regulated learning strategies, in: Proc. of the 2014 Conference on Innovation & Technology in Computer Science Education, ITiCSE '14, ACM, New York, NY, USA, 2014, pp. 291–296.

[10] G. T. L. Brown, L. R. Harris, Student self-assessment, in: SAGE Handbook of Research on Classroom Assessment, SAGE Publications, Inc., Thousand Oaks; Thousand Oaks, California, 2013, pp. 367–393.
[11] E. Scapin, C. Mirolo, An exploration of teachers' perspective about the learning of iteration-control constructs, in: S. N. Pozdniakov, V. Dagienė (Eds.), Proc. of ISSEP 2019 – Informatics in Schools: New Ideas in School Informatics, Springer, Cham, 2019, pp. 15–27.
[12] E. Scapin, C. Mirolo, An exploratory study of students' mastery of iteration in the high school, in: Local Proc. of ISSEP 2020, CEUR Workshop Proceedings, 2020, pp. 43–54.
[13] E. Soloway, J. Bonar, K. Ehrlich, Cognitive strategies and looping constructs: An empirical study, Commun. ACM 26 (1983) 853–860.
[14] E. Rahimi, E. Barendsen, I. Henze, Identifying students' misconceptions on basic algorithmic concepts through flowchart analysis, in: V. Dagienė, A. Hellas (Eds.), Proc. of ISSEP 2017 – Informatics in Schools: Focus on Learning Programming, Springer International Publishing, Cham, 2017, pp. 155–168.
[15] I. Cetin, et al., Teaching loops concept through visualization construction, Informatics in Education - An International Journal 19 (2020) 589–609.
[16] D. Perkins, S. Schwartz, R. Simmons, Instructional strategies for the problems of novice programmers, in: R. E. Mayer (Ed.), Teaching and Learning Computer Programming, Routledge, New York, USA, 1988, pp. 153–178.
[17] T. Jenkins, On the difficulty of learning to program, in: Proceedings of the 3rd annual LTSN ICS Conference, 2002, pp. 53–58.
[18] J. Sorva, Notional machines and introductory programming education, Trans. Comput. Educ. 13 (2013) 8:1–8:31.
[19] R. Lister, B. Simon, E. Thompson, J. L. Whalley, C. Prasad, Not seeing the forest for the trees: Novice programmers and the solo taxonomy, in: Proc. of the 11th Annual SIGCSE Conf. on Innovation and Technology in Computer Science Education, ITICSE '06, ACM, New York, USA, 2006, pp. 118–122.
[20] M. Mani, Q. Mazumder, Incorporating metacognition into learning, in: Proceeding of the 44th ACM Technical Symposium on Computer Science Education, SIGCSE '13, ACM, New York, USA, 2013, pp. 53–58.
[21] G. Schraw, Promoting general metacognitive awareness, Instructional science 26 (1998) 113–125.
[22] P. Steinhorst, A. Petersen, J. Vahrenhold, Revisiting self-efficacy in introductory programming, in: Proceedings of the 2020 ACM Conference on International Computing Education Research, 2020, pp. 158–169.
[23] D. Loksa, L. Margulieux, B. A. Becker, M. Craig, P. Denny, R. Pettit, J. Prather, Metacognition and self-regulation in programming education: Theories and exemplars of use, ACM Trans. Comput. Educ. To appear (2022).
[24] L. Murphy, J. Tenenberg, Do computer science students know what they know? a calibration study of data structure knowledge, in: Proc. of the 10th Annual SIGCSE Conf. on Innovation and Technology in Computer Science Education, ITiCSE '05, ACM, New York, USA, 2005, pp. 148–152.
[25] P. Denny, A. Luxton-Reilly, J. Hamer, D. B. Dahlstrom, H. C. Purchase, Self-predicted and actual performance in an introductory programming course, in: Proc. of the 15th Annual

Conf. on Innovation and Technology in Computer Science Education, ITiCSE '10, ACM, New York, USA, 2010, pp. 118–122.
[26] P. Lee, S. N. Liao, Targeting metacognition by incorporating student-reported confidence estimates on self-assessment quizzes, in: Proc. of the 52nd ACM Technical Symp. on Computer Science Education, SIGCSE '21, ACM, New York, USA, 2021, pp. 431–437.
[27] C. Izu, C. Mirolo, Comparing small programs for equivalence: A code comprehension task for novice programmers, in: Proc. of the 2020 ACM Conf. on Innovation and Technology in Computer Science Education, ITiCSE '20, ACM, New York, USA, 2020, pp. 466–472.
[28] C. Mirolo, C. Izu, E. Scapin, High-school students' mastery of basic flow-control constructs through the lens of reversibility, in: Proc. of the 15th Workshop on Primary and Secondary Computing Education, WiPSCE '20, ACM, New York, USA, 2020, pp. 1–10.
[29] P. Sutherland, The application of piagetian and neo-piagetian ideas to further and higher education, International J. of Lifelong Education 18 (1999) 286–294.
[30] D. Ginat, On novice loop boundaries and range conceptions, Computer Science Education 14 (2004) 165–181.
[31] I. Cetin, Student's understanding of loops and nested loops in computer programming: An APOS theory perspective, Canadian Journal of Science, Mathematics and Technology Education 15 (2015) 155–170.
[32] M. Mladenovic, I. Boljat, Ž. Žanko, Comparing loops misconceptions in block-based and text-based programming languages at the K-12 level, Education and Information Technologies 23 (2018) 1483–1500.
[33] A. Kumar, G. Dancik, A tutor for counter-controlled loop concepts and its evaluation, in: 33rd Annual Frontiers in Education - FIE '03, volume 1, 2003, pp. T3C–7.
[34] R. Lister, E. Adams, S. Fitzgerald, et al., A multi-national study of reading and tracing skills in novice programmers, in: Working Group Reports from ITiCSE on Innovation and Technology in Computer Science Education, ITiCSE-WGR '04, ACM, New York, USA, 2004, pp. 119–150.
[35] M. Lopez, J. Whalley, P. Robbins, R. Lister, Relationships between reading, tracing and writing skills in introductory programming, in: Proc. 4th Int. Workshop on Comput. Educ. Research, ICER '08, ACM, New York, USA, 2008, pp. 101–112.
[36] L. Rigby, P. Denny, A. Luxton-Reilly, A miss is as good as a mile: Off-by-one errors and arrays in an introductory programming course, in: Proc. of the 22nd Australasian Computing Education Conf., ACM, New York, USA, 2020, pp. 31–38.
[37] C. S. Miller, A. Settle, Mixing and matching loop strategies: By value or by index?, in: Proc. of the 52nd ACM Technical Symp. on Computer Science Education, SIGCSE '21, ACM, New York, 2021, pp. 1048–1054.
[38] R. Smetsers-Weeda, S. Smetsers, Problem solving and algorithmic development with flowcharts, in: Proc. of the 12th Workshop on Primary and Secondary Computing Education, WiPSCE '17, ACM, New York, USA, 2017, pp. 25–34.
[39] H. R. Ramsey, M. E. Atwood, J. R. Van Doren, Flowcharts versus program design languages: An experimental comparison, Commun. ACM 26 (1983) 445–449.
[40] S. Bergin, R. Reilly, D. Traynor, Examining the role of self-regulated learning on introductory programming performance, in: Proc. of the 1st Int. Workshop on Comput. Educat. Research, ICER '05, ACM, New York, USA, 2005, pp. 81–86.

Achieving Computational Thinking competencies using the BBC micro:bit

Markus Wieser[1], Nina Lobnig[1], Stefan Pasterk[1] and Andreas Bollin[1]

[1]*University of Klagenfurt, Universitätsstraße 65-67, 9020 Klagenfurt am Wörthersee, Austria*

Abstract

Computational thinking becomes more and more important in society and teaching, as it is often part of digital literacy. In Austrian lower secondary education, "basic digital education" is implemented to cover these topics. Its curriculum includes a section for computational thinking, even though the included competencies are vaguely formulated and loosely connected to those promoted by literature. Another issue is that the specific competencies that characterize computational thinking are not clearly defined in literature either. This contribution presents a competency model for computational thinking to determine which topics are part of computational thinking. This model is then used to check to what extent the micro:bit units designed by the Informatics Lab at our university teach computational thinking and which areas are covered. For a meaningful comparison, the competencies of the learning materials are defined and then compared to the developed model of computational thinking. The results answer the question of which areas of computational thinking these micro:bit learning materials are covering and they make visible what is possible with the BBC micro:bit in this context.

Keywords

computational thinking, competencies, competency model, computer science education, micro:bit.

1. Introduction

Although it is often associated with programming, computational thinking is a really broad field and goes back to early work of Seymour Papert [1]. It became more important for society and teaching informatics in the last years, as it is often part of digital literacy. This situation was leading us to the Question: "How could the amount of computational thinking in learning materials be measured and improved?" In order to solve this question we had to start at the beginning, asking the question: "What is computational thinking?" However, there are quite some definitions, best known is the one formulated by Jeanette Wing [3]. Based on it this work aims to systematically collect all the necessary competencies and maps them to a competency model. This can be used on the one hand to get specific competencies, for example for curriculum developers, and on the other hand to find out how much computational thinking can be found in individual materials and how they can be improved in this area. For testing its applicability we created another competency model of the micro:bit materials designed by the Informatics Lab at our University for Students between the ages of 10 and 14. We then compared the

Informatics in Schools. A step beyond digital education. 15th International Conference on Informatics in Schools: Situation, Evolution, and Perspectives, ISSEP 2022, Vienna, Austria, September 26–28, 2022

markus.wieser@aau.at (M. Wieser); nina.lobnig@aau.at (N. Lobnig); stefan.pasterk@aau.at (S. Pasterk); andreas.bollin@aau.at (A. Bollin)

models asking the Question: "How much computational thinking do the materials contain?" and finally formulated some recommendations to improve the amount of computational thinking of the materials. In the following chapter computational thinking is defined and related work and "basic digital education" are discussed, then, in chapter three, a competency model for computational thinking is created. In chapter four we create another competency model based on the materials for the micro:bit, which we compare to the first model in chapter five, where we also formulate some recommendations to improve the materials. In the final chapter we talk about conclusions and future work.

2. Computational thinking

2.1. Definition and related work

When talking about computational thinking, there is no way around the paper of Jeanette Wing with the title "Computational Thinking" [2]. In her article she writes: "It represents a universally applicable attitude and skill set everyone, not just computer scientists, would be eager to learn and use." and "Thinking like a computer scientist means more than being able to program a computer. It requires thinking at multiple levels of abstraction." [2]. For Wing, computational thinking is conceptualizing, not programming, there are used fundamental, not rote skills, and it is a way that humans, not computers think. This means, computational thinking is about representing a real-world problem in an abstract way, not about the ability to program, even though this is of course very important for implementing the idea.

Another important point that is often misunderstood is, that computational thinking means thinking like a human. Computational thinking represents the way people solve problems. It does not aim at making people think or act like computers. The goal is to use computers to solve problems that probably could not be solved without them.

In her paper "Computational thinking. What and why?" published in 2010, Jeanette Wing answers the question what is computational thinking and she defines it as follows [3]:

> "Computational thinking is the thought processes involved in formulating problems and their solutions so that the solutions are represented in a form that can be effectively carried out by an information-processing agent.“

Informally, a problem should be formulated in a way it can be processed by humans, computers, or more generally, combinations of humans and computers.

Most definitions of computational thinking found in literature are related to the one of Wing. For example David Barr, a K12-Teacher, says [4] that computational thinking is a problem solving process including:

- Formulating problems in a way that enables us to use a computer and other tools to help solve them.
- Logically organizing and analyzing data,

- Representing data through abstractions, such as models and simulations,
- Automating solutions through algorithmic thinking (a series of ordered steps),
- Identifying, analyzing, and implementing possible solutions with the goal of achieving the most efficient and effective combination of steps and resources,
- Generalizing and transferring this problem-solving process to a wide variety of problems.

As one can see, a lot of skills are necessary to achieve the goal of formulating a problem in a way, humans and machines are able to process it. Thus we are using the definition of Jeanette Wing for our paper and further explanations.

The definition of computational thinking is only the first step. Based on it, we will define a competency model. In literature there are many papers defining competencies of computational thinking like David Barr. These are for example Grover and Pea [5], Weintrop et al. [6], Computational thinking for professionals [7] and Jeanette Wing herself [3]. The competencies are very similar to each other. The main difference is that Jeanette Wing defined two sets of computational thinking skills. One basic skill set for everyone and one set of advanced skills for scientists, engineers, and other professionals.

David Barr [4] defined another set of meta skills, which support the learning and using of computational thinking, but are not necessary for achieving it. These are: *confidence in dealing with complexity*, *persistence in working with difficult problems*, *tolerance for ambiguity*, *the ability to deal with open-ended problems*, and *the ability to communicate and work with others to achieve a common goal or solution.* As these skills can be seen as requirements for computational thinking they are not considered in the process of the development of our competency model.

Apart from papers about computational thinking competencies there are also many tests and surveys for testing computational thinking skills, like CTT [8], BCTT [9], the Callysto CTi [10] or the Bebras Test [11]. We used these tests as base for creating the competency model first, but then learned, these tests mostly just check competencies related to algorithmisation and computational thinking is a lot more than just that.

2.2. Computational thinking in "basic digital education"

Taking a look into education, computational thinking appears in different curricula all around the world. One example for it is the current curriculum for "basic digital education" in Austria's first four years of secondary education [14]. The curriculum is competency-oriented and contains learning outcomes for different areas. It is structured into eight topics including one called "computational thinking". This topic consists of all together ten competencies, which are categorised into the subtopics "Working with algorithms" and "Creative use of programming languages". Sample competencies for the first subtopic are "Students formulate clear instructions for action (algorithms) verbally and in writing." or "Students discover similarities and rules (patterns) in action instructions." The second subtopic focuses with competencies like "Students

know different programming languages and production processes." or "Students master basic programming structures (branching, loops, procedures)." on programming. This shows that in this sample curriculum computational thinking is limited to algorithms and simple programming. The new competency model is intended to change this one-sided view of computational thinking.

3. Competency Model for Computational Thinking

The approach of creating a competency model of computational thinking based on the available tests was not as successful as we thought at the beginning, because it only covered a small part of computational thinking. Thus we decided to create our model based on the definition of computational thinking by Jeanette Wing [3] using a top-down approach. Starting from the definition she defined two sets of computational thinking skills. One for everyone and one for experts. Computational thinking for everyone means being able to:

- (WB1) Understand which aspects of a problem are amenable to computation
- (WB2) Evaluate the match between computational tools and techniques and a problem
- (WB3) Understand the limitations and power of computational tools and techniques
- (WB4) Apply or adapt a computational tool or technique to a new use
- (WB5) Recognize an opportunity to use computation in a new way
- (WB6) Apply computational strategies such divide and conquer in any domain

Computational thinking for scientists, engineers, and other professionals further means being able to:

- Apply new computational methods to their problems
- Reformulate problems to be amenable to computational strategies
- Discover new science through analysis of large data
- Ask new questions that were not thought of or dared to ask because of scale, but which are easily addressed computationally
- Explain problems and solutions in computational terms

Starting with these two sets of high level competencies, following our top-down approach, we had to break them down into lower level ones. In this step we found two types of competencies. Some could be derived semantically, others only syntactically. For example: If we want to split the competency "(WB3) Understand the limitations and power of computational tools and techniques" we could create the competencies "(WB3-1) Understand the limitations of computational tools and techniques" and "(WB3-2) Understand the power of computational tools and techniques" by splitting them analyzing the syntax of (WB3). Both are obviously child competencies of the first one. Deriving semantically on the other hand is not that simple. In order to create child competencies from "Understand which aspects of a problem are amenable to computation" one has to understand the meaning of the competency. First one has to understand a problem. This could be done by using models, diagrams or graphs. Then one has to decompose the now understood problem into smaller problems or aspects in order to find out which parts are amenable to computation. The next step is to understand computation. This is not as easy

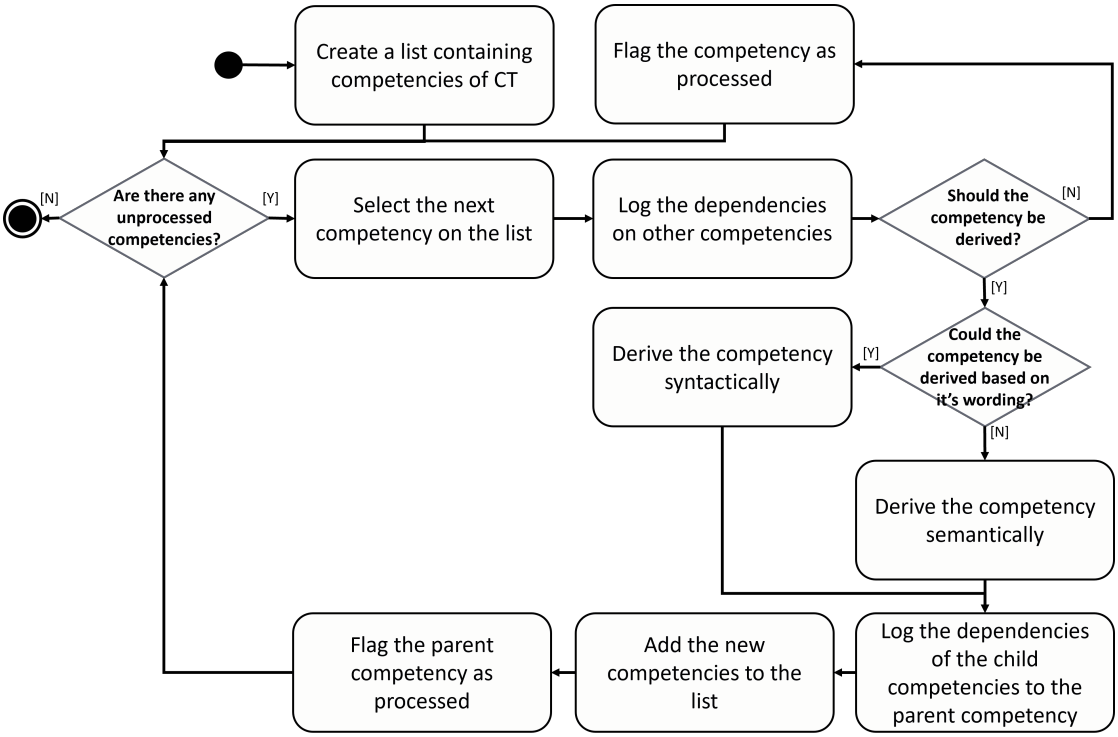


Figure 1: Workflow for deriving competencies

as it seems, but anyway it will be another competency. Based on this, one has to evaluate, if a problem is amenable to computation and how much power or time is required for the algorithm.

Now we created four new child competencies based on the competency "(WB1) Understand which aspects of a problem are amenable to computation". These are: "(WB1-1) Understand a problem using modelling, diagrams and graphs", "(WB1-2) Decompose a problem into smaller problems", "(WB1-3) Understand amenable to computation" and "(WB1-4) Evaluate, if a problem is amenable to computation and how much power or time is required for the algorithm". In Fig. 1 one can find the workflow for the derivation of the competencies.

If we look at the skill set, we learn, that the second competency (WB2), additionally depends on the first competency (WB1). "Understand which aspects of a problem are amenable to computation" is needed to achieve "Evaluate the match between computational tools and techniques and a problem". Based on the first competency (WB1) we get the structure seen in Fig. 2. The parent competencies are located in "Layer 0", the child competencies in "Layer 1". Each of these competencies could be derived and new child competencies could be created and the child competencies also could be derived again. To keep it as small as possible, we decided to limit the model to three layers, so each main competency is derived two times. The first layer competencies (Layer 0) are those defined by Jeanette Wing [3]. The main competencies also depend on each other and the basic ones are necessary to achieve the expert competencies.

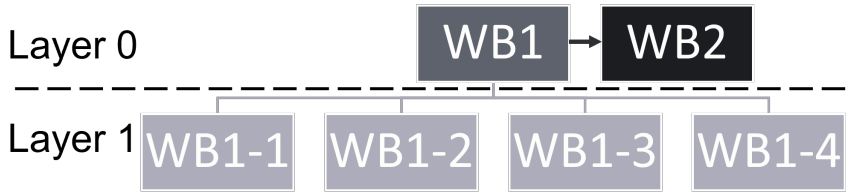


Figure 2: The competencies directly depending on WB1

In total, the competency model created consists of 130 competencies derived from the sets above. In this paper we will only show selected parts, the whole competency model could be found here [12].

The workflow described above has been applied to all the competencies of the basics set leading to the following competencies on "Layer 1":
Child Competencies of WB1:

- (WB1-1) Understand a problem using modelling, diagrams and graphs
- (WB1-2) Decompose a problem into smaller problems
- (WB1-3) Understand amenable to computation
- (WB1-4) Evaluate, if a problem is amenable to computation and how much power or time is required for the algorithm

Child competencies of WB2:

- (WB2-1) Understand and select computational tools and techniques
- (WB2-2) Validate whether a computational tool or technique is suitable for solving a problem.
- (WB2-3) Apply given computational tools and techniques to solve a problem.
- (WB2-4) Evaluate which computational tools and techniques are most appropriate for a problem.
- (WB2-5) Apply computer science tools and techniques to a problem

Child competencies of WB3:

- (WB3-1) Understand the limitations of computational tools and techniques
- (WB3-2) Understand the power of computational tools and techniques
- (WB3-3) Understanding computer architecture
- (WB3-4) Understand the concept and operation of register machines
- (WB3-5) Understand and apply basic algorithms and data structures

Child competencies of WB4:

- (WB4-1) Applying a computational tool for a new purpose

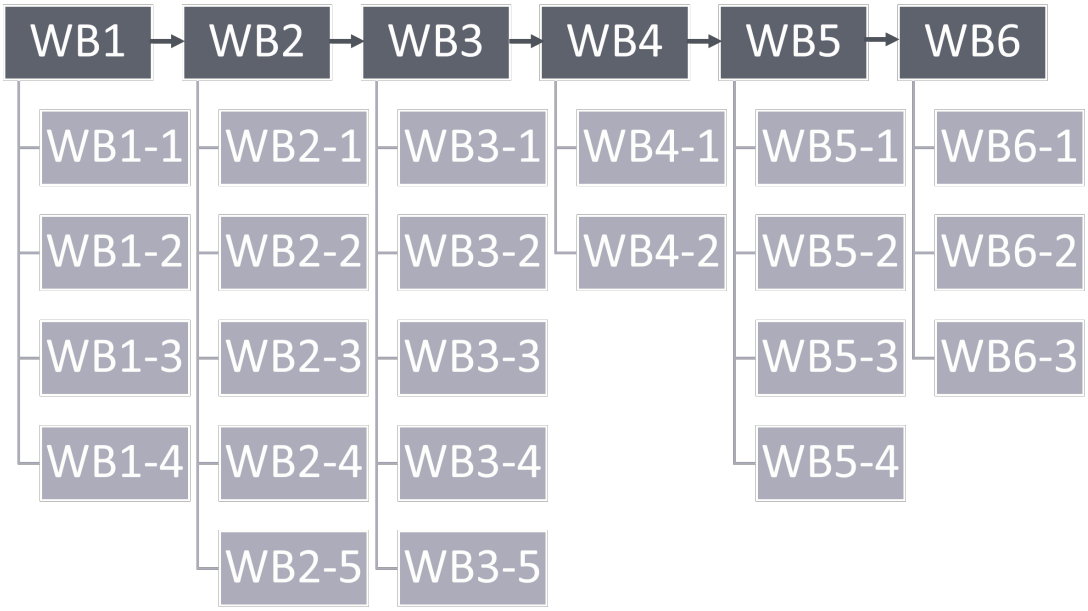


Figure 3: Overview of the structure of the basic skill set competency model in two layers

- (WB4-2) Adapting a computational tool to other requirements

Child competencies of WB5:

- (WB5-1) Identify commonalities between problems
- (WB5-2) Transferring a problem solving process to a different problem.
- (WB5-3) Generalization of a problem solving process to a class of problems.
- (WB5-4) Find similarities and differences between computational tools and techniques

Child competencies of WB6:

- (WB6-1) Divide problems into categories and assign a problem to a category
- (WB6-2) Apply design paradigms, such as Divide & Conquer or the Greedy Method, in any domain
- (WB6-3) Apply programming concepts, such as alternative or recursion in any domain

This leads to the competency model shown in Fig. 3.

4. Competencies achievable using the BBC micro:bit

After the creation of the competency model of computational thinking we created another competency model from the materials of the Informatics Lab to test what amount of computational thinking is covered by them. The materials were designed to teach programming to children between the ages of 10 and 14, using the BBC micro:bit. We decided to use the materials,

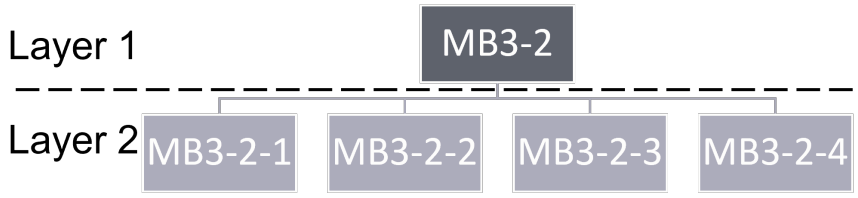


Figure 4: The competencies related to MB3-2

because they are designed for teaching programming, which requires algorithmization. This is part of computational thinking and therefore seems to be a good base for comparison. The materials are split into six parts, covering six different topics. They are: "Basics", "Inputs and random numbers", "Branches", "Variables", "Advanced Branching" and "Loops". Each topic contains tasks for practicing and builds on the previous ones. The materials can be found on the website of the Informatics Lab [13].

However, this time we have different requirements for creating the competency model. In the model of computational thinking we started from the definition and some high level competencies, now our source are the practicing tasks and the information given in the materials. So we had to use a different approach. Instead of top down, which was used to create the competency model of computational thinking, we now use a bottom up approach, starting with the practicing tasks and given information. We formulated competencies for every task and information found in the materials and listed them. For example, for the task "If the micro:bit is shaken, it should display a symbol of your choice, otherwise a question mark. If both keys (A + B) are pressed at the same time, it should randomly decide whether a large or a small heart is displayed." We have two possibilities in the second part, either a small heart or a large heart. This only can be resolved using branching and random numbers,l so we created the competency "Applying branching and random numbers". After the formulating process, we grouped the list into three categories: micro:bit specific competencies like "Use the programming environment", computational thinking competencies and math competencies like "Understand random numbers". This is due to the fact we had one main category about using the micro:bit environment and another one based on computational thinking. In the third category we put the competencies which could not be assigned to any of the previous ones.

After that, we categorized the competencies of computational thinking and formulated parent competencies for them. For example, the competencies "(MB-3-2-1) Be able to write programs sequentially", "(MB3-2-2) Understanding and applying branching", "(MB3-2-3) Understanding and applying variables" and "(MB3-2-4) Understanding and applying loops" are being merged to the parent competency "(MB3-2) Understand and master programming concepts such as branching, variables, and loops". This leads to the structure seen in Fig. 4.

Following the path to the top we get the following main competencies:

- Understanding and using the micro:bit user environment

- Apply (new) computational methods to known problems
- Reformulate problems to make them amenable to computational strategies
- Decompose a problem into individual sub problems, solve them, and combine the partial solutions
- Understanding and applying basic mathematics

The competencies MB2, MB3 and MB4 are connected to computational thinking, the other two are not. Sure one could argue without basic mathematics it is not possible to achieve computational thinking, but again basic mathematics are seen as requirement for computational thinking and for this reason not taken into account.

The whole competency model of micro:bit contains 81 competencies on 4 Layers. 39 of these are child competencies of "Understand and master programming concepts such as branching, variables, and loops". This shows that the main purpose of the materials, teaching basic programming concepts to children, is achieved in those areas. Another property of the micro:bit competency model is the fact that opposite to the computational thinking model, the main competencies are mostly independent.

5. Comparison and Recommendations

After creating both models, we compared them to each other by comparing the competencies. Competencies which are occurring in both competency models are:

- Reformulating problems to make them amenable to computational strategies
- Apply new calculation methods to known problems
- Adapting a computational tool to other requirements
- Recognize a way to use computational methods in a new way
- Finding a specific solution to a problem using appropriate computational tools and techniques
- Understand and master programming concepts such as branching, variables, and loops

As mentioned before, the materials for the micro:bit are designed for students between 10 and 14, so the competencies have to be adapted to the age level. Thus we decided to distinguish between fully achieved competencies and partly achieved ones. In a next step we matched the micro:bit competencies to their suitable counterpart in the computational thinking model. The result on the first two layers can be found in Fig. 5. The framed are competencies of computational thinking which are partly achieved by the micro:bit materials.

After the comparison we analyzed the merged model and the micro:bit materials again, focusing on optimization. With the help of the competency models we found some areas of computational thinking which could easily be covered by the micro:bit materials just with a few modifications. These areas could be:

- A higher focus on understanding and explaining in the tasks

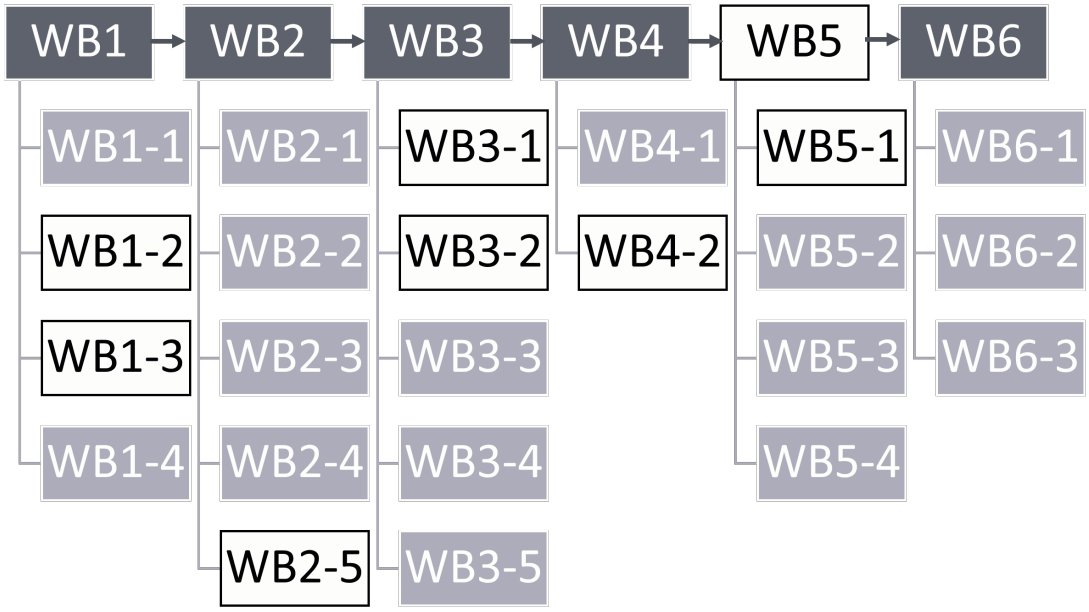


Figure 5: Partly achieved competencies of computational thinking by the micro:bit materials

- Selection of methods and techniques
- Consider debugging
- Discuss classifying problems
- Optimize solutions
- Add basic algorithms and data structures

All these recommendations could easily be applied to the materials with just a few modifications except for the basic algorithms and data structures, which should be added as an extension. These recommendations have a great impact on the coverage of the computational thinking competency model as can be seen in Fig. 6. The black competencies are the ones, which would be additionally covered by applying the recommendations.

6. Conclusion and Future Work

Out of the 49 Competencies of the first two layers of the computational thinking model, there are 15 at least partly achieved by the micro:bit materials of the Informatics Lab. If we apply the recommendations, we talk about 12 added and 27 in total. So we have an initial coverage of 30.6% and after applying the recommendations, which are covering 24.5%, we have a total coverage of 55.1% of at least partly achieved competencies of computational thinking in the micro:bit materials.

Although it is only weakly represented in the some approaches like for instance the

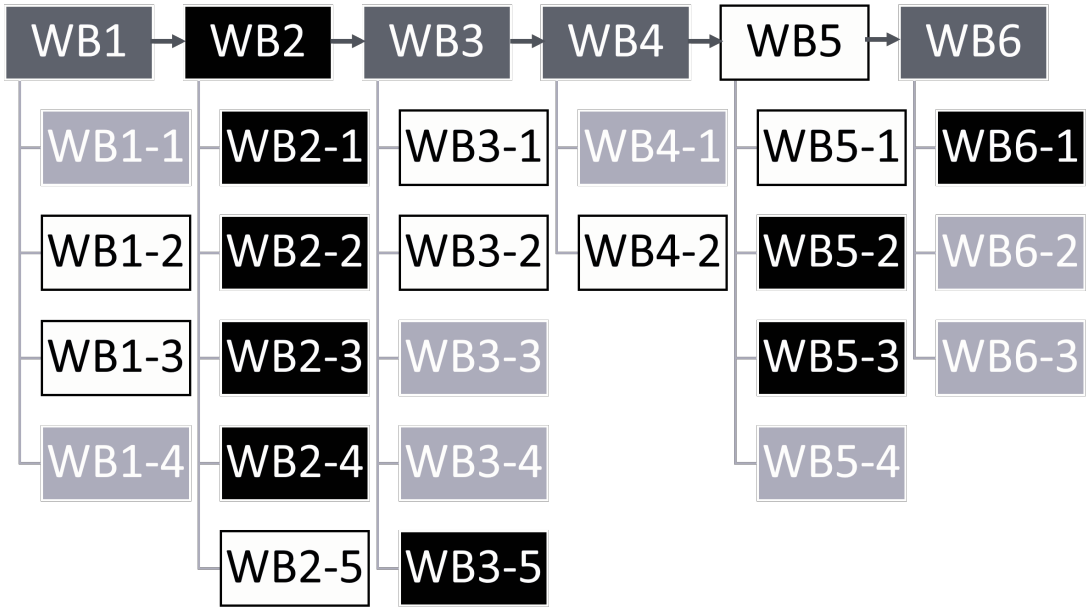


Figure 6: Additionally covered competencies by applying the recommendations

curriculum of "basic digital education" in Austria, computational thinking is a really broad field. Our competency model breaks it down into smaller parts that are easier for curriculum writers, for example, to work with. The model can also be used to check how much computational thinking is found in certain documents. However, this assumes that a competency model of the materials exists. Based on the comparison, recommendations can be made as to what extent the amount of computational thinking could be increased. Our computational thinking model is therefore a good way to analyze, which competencies are covered by some learning materials and to what extent this coverage still could be optimized.

To make it even more usable, an extension by using Bloom's taxonomy would probably be desirable. This would make the future model even more extensive, but the improved structure would make it easier to use, which in turn could encourage more curriculum authors to use the model. This could also lead to an even stronger anchoring of computational thinking in the curricula, which would be a very welcome development.

The competency model of computational thinking could also be relevant for the Informatics Lab at our university. This makes it possible to check all materials of the lab for their content of computational thinking and to determine which competencies are covered and which are missing. For the missing competences, new materials could be developed or existing ones could be extended in order to cover as much of computational thinking as possible.

References

[1] Papert, S.: Mindstorms, Children Computers and Powerful Ideas. Basic Books (1980)
[2] Wing, J.: Computational thinking. Communications of the ACM, **49**, 33–35 (2006)
[3] Wing, J.: Computational thinking. What and why? (2010)
[4] Barr D., Harrison J. and Conery L.: Computational thinking: A digital age skill for everyone. Learning and leading with technology, **38**, 20–23 (2011)
[5] Grover S., Pea R.: Computational thinking in k–12 a review of the state of the field. Educational Researcher, **42**, 38–43 (2013)
[6] Weintrop D. et al.: Defining computational thinking for mathematics and science classrooms. Journal of Science Education and Technology, **25**, 127–147 (2016)
[7] Denning P., Tedre M.: Computational thinking for professionals. Communications of the ACM, **64**(12), 30–33 (2021)
[8] Bati, K.: Computational thinking test (ctt) for middle school students. (2018)
[9] Zapata M. et al.: Computational thinking test for beginners: Design and content validation. (2020)
[10] PIMS. Callysto, https://www.callysto.ca/computational-thinking-tests/. Last accessed 18 May 2022
[11] Bebras, https://www.bebras.org/. Last accessed 18 May 2022
[12] Wieser M., Entwicklung eines Kompetenzmodells des Informatischen Denkens im Kontext blockbasierter Programmierung am Beispiel des micro:bit, https://resolver.obvsg.at/urn:nbn:at:at-ubk:1-43345. Last accessed 18 May 2022 (German)
[13] Regionales Fachdidaktikzentrum Informatik: micro:bit Grundlagen. https://www.rfdz-informatik.at/grundlagenmicrobit/. Last accessed 26 May 2022
[14] Bundesministerium für Bildung, Wissenschaft und Forschung Österreich: Lehrplan Digitale Grundbildung. https://www.ris.bka.gv.at/Dokumente/BgblAuth/BGBLA_2018_II_71/BGBLA_2018_II_71.html. Last accessed 20 May 2022
[15] Bundesministerium für Bildung, Wissenschaft und Forschung Österreich: Digitale Grundbildung - Begutachtungsentwurf. https://www.ris.bka.gv.at/Dokumente/Begut/BEGUT_25796D77_3C78_4325_A420_58ADC71458CC/BEGUT_25796D77_3C78_4325_A420_58ADC71458CC.pdf. Last accessed 20 May 2022

Creating Learning Resources based on Programming concepts

Cristiana Araújo[1], Pedro Rangel Henriques[1] and João José Cerqueira[3]

[1]Centro ALGORITMI, University of Minho, Portugal
[3]Life and Health Sciences Research Institute (ICVS), University of Minho, Portugal

Abstract

Nowadays, computer scientists and pedagogues recommend that citizens shall acquire, since childhood, a set of skills required to properly solve problems using a computer. That set of skills is usually referred to as Computational Thinking. However, the acquisition of mental abilities can not be only studied on books or classes. Skills to efficiently execute complex tasks shall be trained doing appropriated exercises repeatedly. This paper is concerned with that topic, the design of adequate Learning Resources (LR) to help young people train their Computational Thinking. The contribution that we intend to introduce, and illustrate with some examples, is the design of such resources based on concepts used in computer sciences like *annotation* or *text markup*, and *finite automata* or *regular expressions*. We believe that the approach can lead to original resources, promote the creativity of their authors, and be twofold for practitioners — training their Computational Thinking capabilities, and smoothly introducing important concepts for knowledge structuring and modelling.

Keywords

Computational Thinking, Learning Resources, OntoCnE

1. Introduction

Learning is changing the brain. We are always learning because the brain is always changing. However, in the beginning of our life, between Childhood and Adolescence, this changing process has its biggest impact. As Computing teachers, the statement — *It is at young ages that we can make big differences* — motivated us to prepare young people to solve problems by computer.

This intention can be achieved by resorting to Computational Thinking (CT) principles that advocate the acquisition of a set of reasoning abilities.

In that context, we started years ago by characterizing the concept of Computational Thinking and formalizing its relationship with the Computer Programming (CP) discipline[1]. That research came out with an ontology for Computing at School, called OntoCnE. Although many others

Informatics in Schools. A step beyond digital education. 15th International Conference on Informatics in Schools: Situation, Evolution, and Perspectives, ISSEP 2022, Vienna, Austria, September 26–28, 2022
decristianaaraujo@hotmail.com (C. Araújo); prh@di.uminho.pt (P. R. Henriques); jcerqueira@med.uminho.pt (J. J. Cerqueira)
https://epl.di.uminho.pt/~cristiana.araujo/ (C. Araújo); https://www.di.uminho.pt/~prh/ (P. R. Henriques); http://www.icvs.uminho.pt/about-icvs/people/jcerqueira (J. J. Cerqueira)
0000-0002-9656-3304 (C. Araújo); 0000-0002-3208-0207 (P. R. Henriques); 0000-0003-3155-2775 (J. J. Cerqueira)

[1]CP branch of instruction or learning.

are working on CT and Computing at School subjects, to the best of our knowledge we do not know any effort to create an ontology to describe that domain. In the light of OntoCnE we can identify the thinking skills that students need to learn programming. At the same time, OntoCnE enables us to identify Learning Resources (LR) to train the referred skills. This new network of connections, linking resources to CT skills, is being worked out at moment and is leading to a richer ontology.

To disseminate our beliefs and pave the way to prepare primary and secondary school students according to the referred ideology, we are working with IT[2] and CP Teachers, engaged in a Master's course in the areas of Education and Computing. Those Master's Students are creating nice LR resorting to original approaches that in our opinion sound promising and we intend to introduce in the present paper. By the moment, the goal is not to discuss quantitative or qualitative insights concerning experiences in classrooms, but to present the approach. For that purpose we selected from that set just a few samples that are more illustrative.

This article is structured in four sections. Section 2 presents the concept of Computational Thinking and an ontology that describes the interception of two domains: Computational Thinking and Computer Programming. Section 3 presents the design of Learning Resources to train Computational Thinking. Finally, Section 4 discusses the conclusions and future work.

2. Computational Thinking

Computational Thinking (CT) is not a new topic, but over the years it has increasingly attracted the attention of many researchers.

Jeannette Wing was the first to utter the concept of Computational Thinking (CT). Wing defined Computational Thinking as "a method for solving problems, or designing systems and understanding human behavior, based on the fundamental concepts of computer science" [1, 2]. According to her, CT is a key skill for all people, and should be trained from the earliest years.

Since then, several definitions have emerged in the literature. Syslo and Kwiatkowska defined Computational Thinking as a set of thinking skills that may not necessarily result in computer programming. For him, CT should focus on computing principles rather than computer programming skills [3].

The key purpose of Computational Thinking is to be a way to reinforce concepts and complement education in coding or programming; and is not an alternative to learning to code. It is intended that, through CT, students develop stronger mental models to help them solve problems and lastly make them better software engineers. The challenge is to use the CT approach to be useful and effective in all disciplines [4].

The International Society for Technology in Education (ISTE) and the Computer Science Teachers Association (CSTA)[3] presented a Operational Definition of Computational Thinking for K–12 Education. Computational Thinking is a problem-solving process that includes at least these six characteristics [5]: formulate problems in a way that makes it possible to use a computer and other tools to help solve them; logically organizing and analyzing data; represent data through abstractions (models and simulations); automate solutions through algorithmic

[2]Information Technologies

[3]Accessible at: https://www.iste.org/explore/computational-thinking/computational-thinking-all

thinking (using ordered steps); identify, analyze and implement possible solutions with the aim of achieving the most efficient and effective combination of steps and resources; and generalize and transfer this this problem solving process to any problem.

These skills are supported and enhanced by various attitudes that are essential dimensions of Computational Thinking, namely: confidence in dealing with complexity; persistence in working with complicated problems to solve; tolerance for ambiguities; the ability to deal with open ended problems; and the ability to communicate and work with others to reach a common solution [5].

The organization Computing At School [6] have a definition very similar to ISTE and CSTA.

Over the years, several initiatives have been carried out to promote CT in the educational field. In 2016, a report was published by the European Commission that presents a comprehensive view: of Computational Thinking skills; the main trends in the integration of CT in compulsory education in Europe and in the World; and examples of initiatives to promote CT [7]. Very recently, in 2022, a review report was published that updates and expands on the findings presented in the report published in 2016 [8].

In the following subsection we will present an ontology developed by us that aims to describe the domain of Computational Thinking and the domain of Computer Programming.

2.1. OntoCnE – Ontology for Computing at School

OntoCnE (Ontology for Computing at School) is an ontology that aims at providing a detailed and rigorous description of two knowledge domains and their interception: Computational Thinking (CT), and Computer Programming (CP). We decided to structure this ontology in three layers in order to accurately state [9, 10]: what concepts are needed to train Computational Thinking (layer 1); what to train at each level of education and how deep to go (layer 2); what material to use to train Computational Thinking at the various levels of education (layer 3).

To build the ontology, a set of concepts that describe the domains of CT and CP were collected. In Listing 1, a fragment of the concept list is displayed.

```
Concepts{
   Algorithm, Abstraction, Automaton, Array, Assignment, Block, Collaboration,
    Computational_Thinking, Debug, Data_Structure, Decomposition, Generalization,
    Instruction, Logical_Reasoning, Machine_States, Operation, Pattern,
    Pattern_Recognition, Perseverance, Program, Problem, Programming, Read,
    Regular_Expression, Repetition, Resolution, Tree, Write, ...
}
```

Listing 1: Concepts that describe the domains of CT and CP (fragment)

To associate or link two concepts it is necessary to have a relationship between them. Listing 2 shows some of the relationships defined in the ontology.

```
Relations{
  defines, divides, detects, equivalent, guides, has, is_composed_of, is_used, involves,
    manipulates, materialized_by, models, may_be, may_have, requires, solved_by, validates
    , ...
}
```

Listing 2: Relationships to relate the concepts that describe the domains of CT and CP (fragment)

The connection of two concepts through a relationship is called a triple. A triple is composed of '(Subject, Predicate, Object)' where Subject and Object are concepts and Predicate is a relation.

In Listing 3 a fragment of the set of triples is presented. Note that all fragments shown are described in a formal language, OntoDL [11, 12] a DSL specifically designed to facilitate the process of defining and instantiating an ontology.

```
Triples {
   Algorithm=[
        requires=> Computational_Thinking;
        is_composed_of=> Block;
        manipulates=> Data_Structure;
        materialized_by=> Program ];
    Array=[ is-a=> Data_Structure];
    Tree=[  is-a=> Data_Structure];
    Block=[ is_composed_of=> Instruction ];
    Computational_Thinking=[
        requires=> Decomposition, Abstraction,
        Pattern_Recognition, Logical_Reasoning, Perseverance;
        involves=> Collaboration ];
    Decomposition =[ divides=> Problem];
    Problem=[
        requires=> Resolution;
        solved_by=> Programming ];
    Automaton=[
        guides=> Program;
        models=> Machine_States;
        equivalent=> Regular_Expression];
    Regular_Expression=[
        is_used=> Pattern_Recognition;
        defines=>  Pattern]
}.
```

Listing 3: Triples of the ontology that describes the Computational Thinking and Computer Programming domains

When reading the triples, displayed in Listing 3, it is possible to see that the concepts are linked together, creating an understandable discourse. Let's look at a concrete example: `Algorithm =requires=> Computational_Thinking` (line 3), `Computational_Thinking =requires=> Decomposition` (line 11), `Computational_Thinking =requires=> Perseverance` (line 12), and `Algorithm =materialized_by=> Program` (line 6), `Automaton =guides=> Program` (line 19), `Automaton =equivalent=> Regular_Expression` (line 21), and `Regular_Expression =is_used=> Pattern_Recognition` (line 23).

Figure 1 displays an excerpt (formally defined in Listings 1–3) of layer 1 of OntoCnE.

As mentioned before, layer 2 of OntoCnE indicates the concepts that are trained at each level of teaching and to which level of depth. In this layer, relationships play a very important role, as they determine the level of depth with which concepts are approached each year, that is, relationships are the unit of measurement. In addition to these new relationships, concepts are also added that will represent each school year (Ano1/1st Year, Ano2/2nd Year, etc.) An example of the depth to which a concept is taught in different grades is shown below: `1st Year =introduce=> Algorithm` and `2nd Year =reinforces=> Algorithm`.

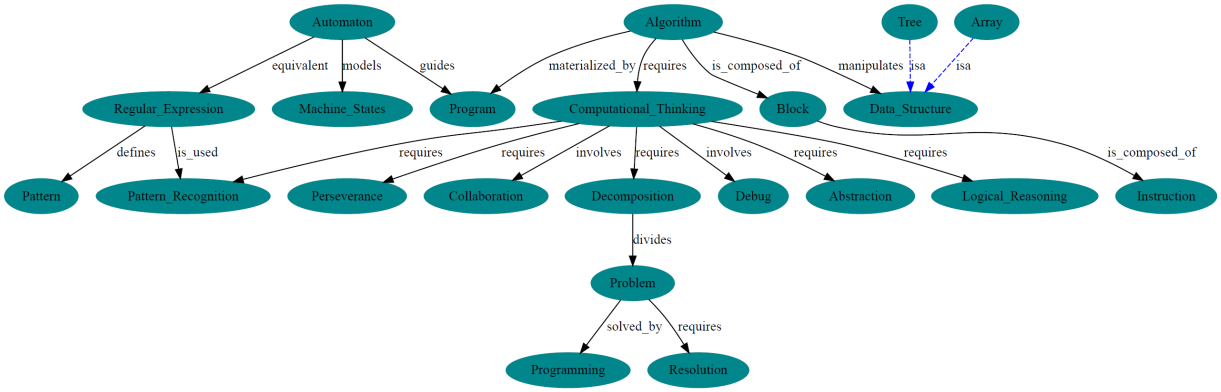


Figure 1: OntoCnE: Ontology for Computing at School – layer 1 (fragment)

But to train and teach the concepts of CT and CP present in the ontology, it is necessary to have adequate Learning Resources (LR). These LRs will populate layer 3 of OntoCnE. Each resource will be associated with the layer 1 concepts that it allows to train and in which school years it can be applied. In this way, it is possible to find adequate resources for training the CT and teaching the concepts of PC.

In the following subsection, the concept and type of Learning Resource and its role in CT training are presented.

2.2. The role of Learning Resources for Training Computational Thinking

Learning Resource is "a device used for educational purposes in any format, real or virtual, that illustrates or supports one or more elements of a course of study; and may enrich the learning experience of the pupil or teacher" [13]. The Learning Resource aims to: put in practice or train previous knowledge; develop new knowledge; encourage the process of understanding, organization and synthesis of educational content, logical reasoning, communication and interaction; and contribute to the development of different skills and acquisition of student values, as well as the retention of desirable knowledge, and attitudes [14]. To train Computational Thinking, two types of resources can be used: *plugged* (need electronic devices) and *unplugged* (do not need any kind of electronic devices).

The resources must be adequate to the training of CT so that the students can attain the different abilities and acquire the attitudes required. Interdisciplinarity is a key aspect that must be promoted through LR. This allows the training of Computational Thinking to be transversal to the various subjects of the students' curriculum.

It is important to take into account that the challenges presented in LR have more impact on learning if they are similar to the real day-to-day context. Our brain more easily remembers facts and skills that are integrated into real scenarios. This is due to the fact that memories are processed in the hippocampus, and this has the function of labeling the context in the memories. This tagging allows the brain to remember faster when that context comes up [15].

In addition, it is also important that each Learning Resource, besides its description and usage

rules, has a guide on how to apply it so that the training is effective. This guide is similar to a prescription for a medication [10].

In the next section we will present a template for designing LR and several examples of features.

3. Designing Learning Resources

We decided some time ago to create a template that gathers the main characteristics of a LR, so that teachers can appropriately select a resource adequate to teach a given subject to a given class. Such a template is composed of the following attributes: Title; Author Learning Objective; Category (eg: game, video, slides, worksheet, etc.); Type (unplugged or plugged); Education level (K0-K12) or Age; Computational Thinking Skills & Attitudes trained; Description; Usage rules; Material (eg: computer, robot, paper and pencil, etc.); Subjects involved (eg: Math, History, English, Natural Science, Programming, etc.); Language; Duration[4]; and Requirements[5]. Notice that the field *Computational Thinking Skills & Attitudes trained* shall be fulfilled with concepts belonging to OntoCnE (layer 1) introduced in the previous section. In this way, as a side effect, doing the characterization of new LR we populate layer 3 of OntoCnE linking CT and CP concepts to Learning Resources.

The set of attributes listed above is relevant to choose one LR, however it shall also be used by any author to propose a new LR, creating it from scratch or adapting an old one, as those that will be discussed in this section. It will also guide the LR presentation in the following subsections.

3.1. Using Annotation

The LRs presented in this subsection are inspired by document annotation. This annotation is performed in Computer Programming, using Markup languages such as XML. In Markup languages, *Tags* are defined that allow marking text in order to associate with it some structural role in the document or some interpretation (assigning meaning to the tagged term). Annotation preserves completely the original document but adds meta-data that allows computers to process it. Markup is also usual nowadays in Software Engineering as an agnostic data description language to assure interoperability and allow different applications share databases. The goal of annotation using markup languages is to enable the automatic extraction of structured data from text files, making that information available for further processing by different programs aiming at knowledge exploration [16].

Below, two LR will be presented: Recycle Annotation, a worksheet to markup a text about recycling; Riskatintas, a completely different worksheet to annotate the source code of programming exercises to emphasize the relevant concepts involved in a program.

Recycle Annotation

[4] Average time necessary to complete the activity.
[5] Topics or skills necessary to execute that activity.

Author: Joana Leitão;
Learning Objective: develop Computational Thinking, identify and apply concepts associated with Environmental Education (recycling bins and their colors, organic vs inorganic materials, correct separation of garbage by the different recycling bins), analyze and interpret texts, and identify and apply tags in text;
Subjects involved: Natural Sciences & Environmental Education, Natural Language interpretation (in the case, Portuguese), Programming (document annotation);
Category: Worksheet;
Type: unplugged;
Age: >8;
Computational Thinking Skills & Attitudes trained: Abstraction, Pattern Recognition, Decomposition, Perseverance;
Description: Students receive the worksheet that contains a text about daily life materials, garbage and their recycling and a set of tags (marks or labels) previously defined by the teacher in order to emphasize the most important concepts present in the given text. Tags are provided with a short explanation so that students can understand their meaning. The objective is to annotate all the given text using the given tags.
Listing 4 exhibits a fragment of the annotated text.

```
%%Predefined TAGs%%
<DEFINITION concept=""> => Definition of a concept
<CONCEPT> => Identification of a concept
<MATERIAL organic="SIM|NAO" type="material" recyclingBin="Amarelo|Verde|Azul"> =>
    Characterization of a material
%%TEXT ANNOTATED%%
<CONCEPT>Materiais inorganicos</CONCEPT><DEFINITION concept="Materiais inorganicos">e tudo
    o que nao possui origem biologica, ele e produzido por meios nao-naturais, ou seja,
    produzidos pelo homem, como o <MATERIAL organic="NAO" type="Plastico" recyclingBin="
    Amarelo">plastico</MATERIAL>, <MATERIAL organic="NAO" type="metal" recyclingBin="
    Amarelo">aluminio</MATERIAL>, <MATERIAL organic="NAO" type="Vidro" recyclingBin="Verde
    ">vidro</MATERIAL>, MATERIAL organic="NAO" type="Papel" recyclingBin="Azul">jornais</
    MATERIAL> e outros materiais.</DEFINITION>   ....
```

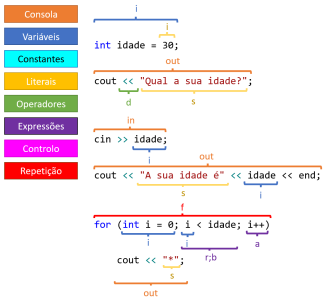
Listing 4: Recycle Annotation Example – TAGS and Markup Text

The next LR aims highlight variables, statements and expressions on a program using a similar annotate technique.

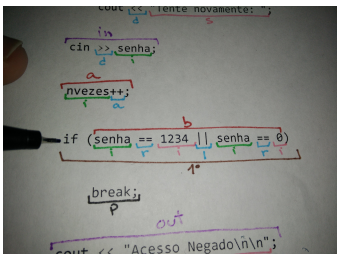
Riskatintas

Author: Pedro Ferreira;
Learning Objective: develop Computational Thinking, analyze and interpret source code in programming languages, and identify and apply tags in texts;
Subjects involved: Programming and ICT (analyze and interpret source code in high-level programming languages; document annotation), and Natural Language interpretation (Portuguese, in that example);
Category: Worksheet;
Type: unplugged;

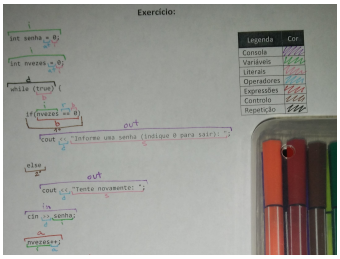
Age: >15;
Computational Thinking Skills & Attitudes trained: Decomposition, Abstraction, Pattern Recognition, (detailed) Analysis, Perseverance;
Description: Students receive the worksheet containing the source code to be annotated, and instructions on how to complete the challenge, i.e., the concrete set of tags and their attributes to used in the annotation. Students shall use colored pens/pencils or markers to highlight the text fragments to associate with each tag. The first task is to assign a unique color to each tag. In the example tags are: *consola* (input/output statements), *variables* (attr: data type), *constants*(attr: data type), *literals* (attr: data type), *operators* (attr: operator type), *expressions* (attr: expression type and data type), *control structures* (i.e., selections) and *repeating structures* (i.e, cycles). The annotation of the program elements, according to the concepts referred above, must respect the list of attributes. Data types considered in the example are: (i)nt, (c)har, (s)tring, (f)loat, (b)ool, and (a)rray. Expression types are: (b)oolean and (a)rithmetic. Operator types considered: (a)rithmetic, (r)elational, (l)ogical, (c)oncatenation, (d)shift, (at)assigment. The repetition can be: (f)finite, (d)dependent, (p)cycle stop, (c)cycle continuity. It is important to point out that the marks chosen for this exercise can be used to annotate any source code, that is, they are generic for imperative languages.
Figures 2a, 2b and 2c illustrate Riskatintas challenge.



(a) Tags and Colors Map and a markup example



(b) A program fragment being annotated



(c) Another example of a markup exercise

Figure 2: Riskatintas, some illustrative photos

This annotation feature is very important in Programming subjects because many of the students have difficulties reading and interpreting the source code in any programming language[17, 18, 19].

In the next section we will describe LR to train Computational Thinking through automata and regular expressions.

3.2. Using Automata and ER

Finite Deterministic Automata (FDA) are mathematical objects used as formal definitions (models) for the behavior of state-machines defining the transition function which associates to each state and each stimulus[6] a new state. A FDA defines all the valid paths (sequences of

[6] A token that triggers the system and forces it to react.

stimuli or tokens) from the start state to the final states. On the other hand, Regular Expressions (RE) are also mathematical models that use a simple set of algebraic operators[7] to write compact formulas that describe a valid set of sentences, i.e., sequences of tokens. For a given set of tokens (alphabet), FDA and RE are equivalent in the sense that they can describe precisely the same set of paths or sentences. The conversion of a RE into a FDA is always possible and systematic.

In this subsection we introduced two LR inspired by the referred concepts (Automata & RE): the first, Computational Tourist, aims to describe a short story (namely a visit program suggested to a tourist just arrived at a town) through an automaton and an ER; the second, Colored Automata, a kind of a labyrinth traversal, aims to interpret given Automata or ER using colored plastic bottle caps.

Computational Tourist

Author: Nelson Vaz and João Paulo Vasconcelos;
Learning Objective: develop Computational Thinking, introduce Programming and Math concepts like automata, and regular expressions, and also cultural heritage;
Subjects involved: Programming and Math;
Category: game plus worksheet;
Type: unplugged;
Age: >14;
Computational Thinking Skills & Attitudes trained: Decomposition, Abstraction, Pattern Recognition, Logic Reasoning, Algorithmic approach, Perseverance;
Description: This game consists of magnetic pieces that represent touristic sites in a town (monuments, cultural or gastronomic points) and a worksheet that introduces the theme and guides the game. In this specific case, the theme is the city of Porto but the idea can be adapted to any location; also the magnetic pieces proposed[8] can be replaced by images printed on sheets of paper. Students are given a simple story and they have to build an automaton or a regular expression that represents the story.
Let's look at an example: *The tourist wakes up and decides to leave the HOTEL and explore the city of Porto. On the first day morning, he can choose to visit CLÉRIGOS TOWER or visit the MUSIC HOUSE. After choosing Clérigos or Music House he can enter there ONE OR MORE TIMES. After the visit, the tourist will have a FRANCESINHA for lunch. After lunch, he goes straight to RIBEIRA neighborhood. At RIBEIRA he can choose to GO or NOT GO to the BARS.*
With the magnetic symbols given, the student's task is to construct the regular expression and the automaton that represents this day of the Tourist.

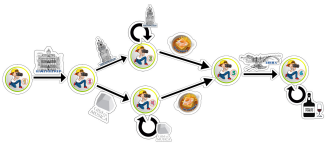
Figures 3a and 3b show the regular expression and the automaton that describe the previous example, respectively. Figure 3c presents another example solved by the students, placing the magnetic pieces on the whiteboard in the classroom.

The following LR is also based on a story but uses plastic caps to interpret the automata and RE given.

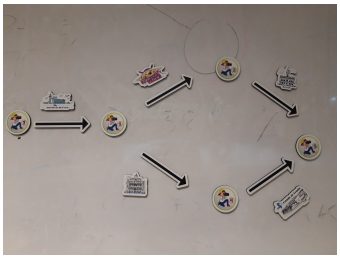
Colored Automata

[7] Concatenation, Union/alternative, Exponentiation, Transitive and Kleene closures.
[8] To be easily attached to the classroom whiteboard.

(a) Regular Expression that represents the story



(b) Automaton that represents the story



(c) Another example solved in the classroom

Figure 3: Computational Tourist, some illustrative pictures

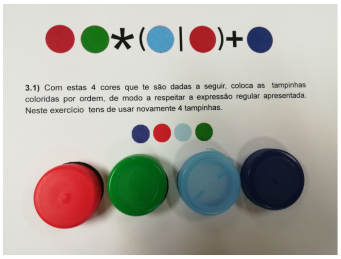
Author: Francisco Borges, Miguel Tinoco and Mónica Oliveira;
Learning Objective: develop Computational Thinking, introduce Programming and Math concepts like automata, and regular expressions;
Subjects involved: Programming and Math, and Natural Language (in the case, classic fairy tale in Portuguese);
Category: worksheet;
Type: unplugged;
Age: >13;
Computational Thinking Skills & Attitudes trained: Decomposition, Abstraction, Pattern Recognition, Logic Reasoning, Algorithmic approach, Perseverance;
Description: Students receive a worksheet that contains several exercises based on the fairy tale "Little Red Hood". As in the previous one, this LR can be adapted to any story. A story scene is told and a Automata with colored edges is displayed over the forest to represent possible paths of the tale's actors. The objective is to find paths on the given graph (network) using colored plastic bottle caps observing the colors code. Alternatively some possible paths are described by a *colored-caps ER* and the students are asked to find a valid way according to the pattern exhibited.
Let's look at an example: *The Bad Wolf heard the hunter approaching. Help Bad Wolf choose the shortest path from point 2 to point 7, so that he can hide on the other side of the forest. Each yellow paw indicates a step the Wolf has to walk to go from one place (state) to the next place (state) allowing to associate weights with each edge. Choose the correct colored caps and place them in the right order to build the shortest path.*

Figures 4a illustrates the use of an automaton to represent possible paths and the choice of one built up from the caps given; Figure 4b shows the resolution of the exercise described above which goal is to find the shortest path. In Figure 4c an alternative exercise using an ER is presented.

In the following section we will present the conclusions and future work.

(a) Build a valid path from point 1 to 8 using the 4 caps

(b) Build the shortest path from point 2 to 7



(c) Write a valid sequence of caps according to the RE

Figure 4: Colored Automata, some illustrative images

4. Conclusion

In this paper, some Learning Resources to train Computational Thinking with K0-K12 students were presented. Those games, built specifically for that purpose by future teachers of primary or secondary schools[9], illustrate how simple and adequate resources can be built with imagination guided by an ontology that formalizes the concepts involved in Programming and Computational Thinking. Moreover, we emphasized the possibility of create LR that complementary to the train of abstraction, algorithms, strategic and logic thinking, can smoothly introduce students to important concepts used in Computer Programming like annotation (or markup languages), automata and regular expressions.

As future work, we intend to populate OntoCnE-layer3 with all the resources developed by the students, and of course any other found meanwhile. For this we are already building a web platform that will store and grant access to that repository of LR to train CT. We also intend to test these resources with students, in schools, in order to measure their impact on CT training. The definition of strategies and exercises to assess the impact of training CT skills in the easiness of learning CP is under development in the context of a Ph.D. project. The outcomes of the desired experiments will be the subject of a future publication.

Acknowledgments

This team work has been supported by FCT – Fundação para a Ciência e Tecnologia within the R&D Units Project Scope: UIDB/00319/2020.
The Ph.D. work of Cristiana Araújo is supported by FCT – Fundação para a Ciência e Tecnologia, Research Grant, with reference 2020.09845.BD.
We are in debt to all our students of the Master's in Education and Informatics (ed. 21-22) for having developed the Learning Resources that made possible this paper.

[9]Students of a Master's Program in Education for IT and Computer Programming.

References

[1] J. Wing, Computational thinking, Journal of Computing Sciences in Colleges 24 (2009) 6–7. URL: http://dl.acm.org/citation.cfm?id=1529995.1529997.

[2] J. M. Wing, Computational thinking, Commun. ACM 49 (2006) 33–35. doi:10.1145/1118178.1118215.

[3] M. M. Sysło, A. B. Kwiatkowska, Informatics for all high school students, in: I. Diethelm, R. T. Mittermeir (Eds.), Informatics in Schools. Sustainable Informatics Education for Pupils of all Ages. ISSEP, volume 7780, Springer Berlin Heidelberg, Berlin, Heidelberg, 2013, pp. 43–56. doi:10.1007/978-3-642-36617-8_4.

[4] F. J. García-Peñalvo, A. J. Mendes, Exploring the computational thinking effects in pre-university education, Computers in Human Behavior 80 (2018) 407–411. URL: https://www.sciencedirect.com/science/article/pii/S0747563217306854. doi:https://doi.org/10.1016/j.chb.2017.12.005.

[5] C. . ISTE, Operational Definition of Computational Thinking for K–12 Education, https://cdn.iste.org/www-root/Computational_Thinking_Operational_Definition_ISTE.pdf?_ga=2.140613553.243754824.1652693570-2014042818.1652693570, 2011. Accessed: 2021-11-07.

[6] CAS-Barefoot, Computational Thinking Concepts and Approaches, https://www.barefootcomputing.org/concept-approaches/computational-thinking-concepts-and-approaches, ND. Accessed: 2021-11-07.

[7] S. Bocconi, A. Chioccariello, G. Dettori, A. Ferrari, K. Engelhardt, Developing Computational Thinking in Compulsory Education - Implications for policy and practice, Scientific analysis or review LF-NA-28295-EN-N (online), Luxembourg (Luxembourg), 2016. URL: https://publications.jrc.ec.europa.eu/repository/handle/JRC104188. doi:10.2791/792158.

[8] S. Bocconi, A. Chioccariello, P. Kampylis, V. Dagienė, P. Wastiau, K. Engelhardt, J. Earp, M. Horvath, E. Jasutė, C. Malagoli, V. Masiulionytė-Dagienė, G. Stupurienė, Reviewing Computational Thinking in Compulsory Education, Scientific analysis or review KJ-06-22-069-EN-N (online), Luxembourg (Luxembourg), 2022. URL: https://publications.jrc.ec.europa.eu/repository/handle/JRC128347. doi:10.2760/12695.

[9] C. Araújo, L. Lima, P. R. Henriques, An Ontology based approach to teach Computational Thinking, in: C. G. Marques, I. Pereira, D. Pérez (Eds.), 21st International Symposium on Computers in Education (SIIE), IEEE Xplore, 2019, pp. 1–6. doi:https://doi.org/10.1109/SIIE48397.2019.8970131.

[10] C. Araújo, Training Computational Thinking: exploring approaches supported by Neuroscience, Ph.D. thesis, University of Minho, 2022. Ph.D. - prethesis.

[11] A. M. C. Dias, ONTODL+, An ontology description language and its compiler, Master's thesis, Minho University, Braga, Portugal, 2021. MSc dissertation.

[12] L. Martins, C. Araújo, P. R. Henriques, Digital Collection Creator, Visualizer and Explorer, in: R. Rodrigues, J. Janoušek, L. Ferreira, L. Coheur, F. Batista, H. G. Oliveira (Eds.), 8th Symposium on Languages, Applications and Technologies (SLATE 2019), volume 74 of *OpenAccess Series in Informatics (OASIcs)*, Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, Dagstuhl, Germany, 2019, p. 15:1–15:8. URL: https://www.dagstuhl.de/dagpub/978-3-95977-114-6.

[13] R. Saskatchewan, The Education Regulations, Technical Report, 2015.

[14] R. Bušljeta, Effective use of teaching and learning resources, Czech-Polish Historical and Pedagogical Journal 5 (2013) 55–70. doi:10.2478/cphpj-2013-0014.

[15] M. Gazzaniga, R. B. Ivry, G. R. Mangun, Cognitive Neuroscience: The Biology of the Mind: Fifth International Student Edition, International student edition, Norton, 2019.

[16] C. Araújo, Building the Museum of the Person Based on a combined CIDOC-CRM/ FOAF/ DBpedia Ontology, Master's thesis, Universidade do Minho, Braga, Portugal, 2016. URL: http://hdl.handle.net/1822/47730, MSc dissertation.

[17] A. Gomes, C. Areias, J. Henriques, A. J. Mendes, Aprendizagem de programação de computadores: dificuldades e ferramentas de suporte, Revista Portuguesa de Pedagogia (2008) 161–179. doi:10.14195/1647-8614_42-2_9.

[18] A. A. Lawan, A. S. Abdi, A. A. Abuhassan, M. S. Khalid, What is difficult in learning programming language based on problem-solving skills?, in: International Conference on Advanced Science and Engineering (ICOASE), 2019, pp. 18–22. doi:10.1109/ICOASE.2019.8723740.

[19] R. P. Medeiros, G. L. Ramalho, T. P. Falcão, A systematic literature review on teaching and learning introductory programming in higher education, IEEE Transactions on Education 62 (2019) 77–90. doi:10.1109/TE.2018.2864133.

The pioneers of introducing informatics in K-12 education – Do not fall into the same hole, we did

Mária Csernoch

University of Debrecen, Kassai út 26., Debrecen, 4028, Hungary

Abstract

Hungary was one of the first countries to introduce informatics education at all levels of school education, from as early as 1995. However, since their first appearance, the newly designed curricula clearly show both the changes in the requirements and the lack of clear concepts behind the newer versions. The present paper provides details of the new National Base Curriculum 2020, its connection to the frame curricula and course books, and some of the consequences of its introduction, which other countries can take into consideration in advance when introducing their own curriculum and its ac-companying materials.

Keywords: national base curriculum, informatics curricula, knowledge transfer, teacher education.

Keywords

national base curriculum, informatics curricula, knowledge transfer, teacher education

1. The introduction of Digital Culture 2020

1.1. Scheduling

A new National Base Curriculum (NBC 2020) [1] and its accompanying frame curricula (FC) [2, 3, 4] were introduced in Hungary in the academic year of 2020/2021, with a renamed informatics (computer science) subject, entitled *Digital Culture*. The subject is compulsory in Grades 3–11 with 1 or 2 classes per week (Table 1).

Table 1

The number of classes per week in NBC 2012 and 2020.

	1	2	3	4	5	6	7	8	9	10	11	12
2012						1	1	1	1	1		
2020			1	1	1	1	1	1	2	1	2	

According to NBC 2020 [1], the first grades who study *Digital Culture* are 5 and 9 in 2020, while the others do not (Grades 3–4 and 11), or continue their studies according to NBC 2012 [5] (Grades 6–8, 10). Consequently, the introduction will only affect all grades in the academic year 2023/2024. The number of classes and the years of introduction are presented in Table 2. It is not clear from either NBC 2020 [1] or FC [2, 3, 4] how students in Grades 5 and 9 in 2020, and in Grades 6 and 10 in 2021 etc., can learn the contents of the missing years. There is no continuity since they must study the new content without completing the requirements of the previous grades [6, 7, 8, 9]. Furthermore, they must take the maturation exam in *Digital Culture* instead of *Informatics* [10]. The most serious is the situation in Grade 9 in 2020 with students who did not study *Digital Culture* in Grades 3–8 but *Informatics* in Grades 6–8 [5]. They will leave K-12 education in 2024. The last year when students will leave K-12

Informatics in Schools. A step beyond digital education. 15th International Conference on Informatics in Schools: Situation, Evolution, and Perspectives, ISSEP 2022, Vienna, Austria, September 26–28, 2022

EMAIL: csernoch.maria@inf.unideb.hu

ORCID: 0000-0002-7088-7714

with this mixed schedule will be 2028. Instead of introducing *Digital Culture* as early as possible in Grade 3 to avoid this knowledge gap for many years, the system presents a false view of digitally educated students.

Table 2
The schedule of the introduction of Digital Culture and the publication of course books in the consecutive academic years in Hungary.

	1	2	3	4	5	6	7	8	9	10	11	12
2020					×				×			
2021						×				×		
2022			×				×				×	
2023				×				×				

1.2. The content of the frame curricula and the course books

Considering the content of NBC 2020 [1], it is in complete accordance with the *Informatics Reference Framework for Schools* [11][11]: "Compulsory informatics education should not only prepare pupils for the present and the future, but also provide a fascinating and useful insight into the connections of informatics to other subjects." [11]. The publishing dates of the two documents might suggest that the Hungarian NBC is way ahead of others and would serve as an example for those countries who want to introduce informatics in the following years.

Table 3
The topics and the number of classes assigned to the topics in Grades 5–6.

Topic	suggested classes	number of years	suggested per year	reality per year
algorithms, block programming	14	2	7	6
online communication	5	2	2.5	1.5
Robotics	11	2	5.5	4.5
word processing	12	1	12	11
creating presentations	8	1	8	7
creating multimedia elements	8	2	4	3
information society, e-world	6	2	3	2
using digital tools	4	2	2	1
Total	68	2	7	6

However, NBC is a base curriculum, based on which the frame curricula (FC), based on which the course books are written which reach students and teachers.

Table 4
The topics and the number of classes assigned to the topics in Grades 9–10.

Topic	suggested classes	number of years	suggested per year	reality per year
algorithms, use of formal programming language	25	2	12.5	11.5
information society, e-world	3	2	1.5	0.5
mobile technology skills	4	1	4	3
word processing	11	1	11	10
computer graphics	14	2	7	6
creating multimedia documents	4	1	4	3
online communication	4	2	2	1
publishing on the Internet	14	2	7	6
spreadsheet management	12	2	6	5
database management	5	1	5	4
using digital tools	6	2	3	2
total	102			

The number of subjects dealt in FC and in the published course books are presented in Table 3 and Table 4. Officially, there are 34 weeks in a school year; however, 30–31 is a reasonable total. All the topics must be evaluated with a mark, which in most cases means testing, further reducing the number of classes assigned to the topics. To demonstrate the huge amount of content, we can sample the 36 pages of programming in Grade 9, assigned to 11 classes [12]. Similarly, to this chapter all the other topics are overloaded.

2. Course books – The realization of NBC

The course books of *Digital Culture* matching the requirements of NBC 2020 are not ready [1]. Each year since 2020 only those *Digital Culture* books have been published, including two in 2020 [6, 7] and another two in 2021 [8, 9]. This scheduling of course book publishing means that for the grades in between the later published contents are not available. The chance to catch up, even for the most interested and motivated students is almost impossible. One further negative effect of this scheduling is that books are published without knowing what is taught in the previous classes. For example, this belated publishing of the Grade 6 book [8] caused the repetition of word-processing materials in the Grade 9 book, and programming is introduced in the Grade 9 book [7] without knowing what should have been taught in the previous grades.

2.1. Examples of tool-centered tasks

Word processing. Word processing is taught both in Grades 6 and 9 (and according to FC, also in Grade 7–8, although we do not know this yet). The Grade 9 course book does not mention that this topic is not new but starts everything from the very beginning. It provides examples of decontextualized tasks (Figure 1) or tasks paying no attention to the content, styles (usually Normal style is ignored and over-formatted), and typography rules [15, 17] (typographic errors in Figure 2: underline, mixture of font types and font styles; there is no connection to the content). The focus is on tools only.

Lorem·ipsum·dolor·sit·amet,·consectetuer·adipiscing·elit.·Maecenas·porttitor·congue·massa.·Fusce·posuere,·magna·sed·pulvinar·ultricies,·purus·lectus·malesuada·libero,·sit·amet·commodo·magna·eros·quis·urna.·Nunc·viverra·imperdiet·enim.·Fusce·est.·Vivamus·a·tellus.·Pellentesque·habitant·morbi·tristique·senectus·et·netus·et·malesuada·fames·ac·turpis·egestas.·Proin·pharetra·nonummy·pede.¶

Mauris·et·orci.·Aenean·nec·lorem.·In·porttitor.·Donec·laoreet·nonummy·augue.·Suspendisse·dui·purus,·scelerisque·at,·vulputate·vitae,·pretium·mattis,·nunc.·Mauris·eget·neque·at·sem·venenatis·eleifend.·Ut·nonummy.·Fusce·aliquet·pede·non·pede.¶

Figure 1: Random, decontextualized text and task in Grade 9 course book.

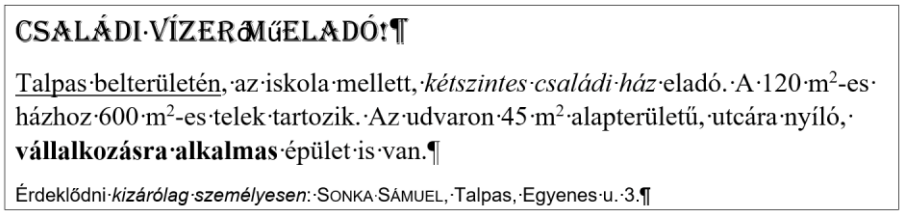
Family house for sale!
Two-story family house for sale in Talpas, near the school. The 120 m^2 house is on a 600 m^2 plot. In the yard there is a 45 m^2 building overlooking the street, suitable for a business.
Please contact, in person only: Hugo Ham, Talpas, 3, Straight Street.

CSALÁDI HÁZ ELADÓ!

Talpas belterületén, az iskola mellett, *kétszintes családi ház* eladó. A 120 m^2-es házhoz 600 m^2-es telek tartozik. Az udvaron 45 m^2 alapterületű, utcára nyíló, **vállalkozásra alkalmas** épület is van.

Érdeklődni *kizárólag személyesen*: SONKA SÁMUEL, Talpas, Egyenes u. 3.

Figure 2: Text for typing and formatting with typographic errors.


CSALÁDI·VÍZERőMűELADÓ!¶

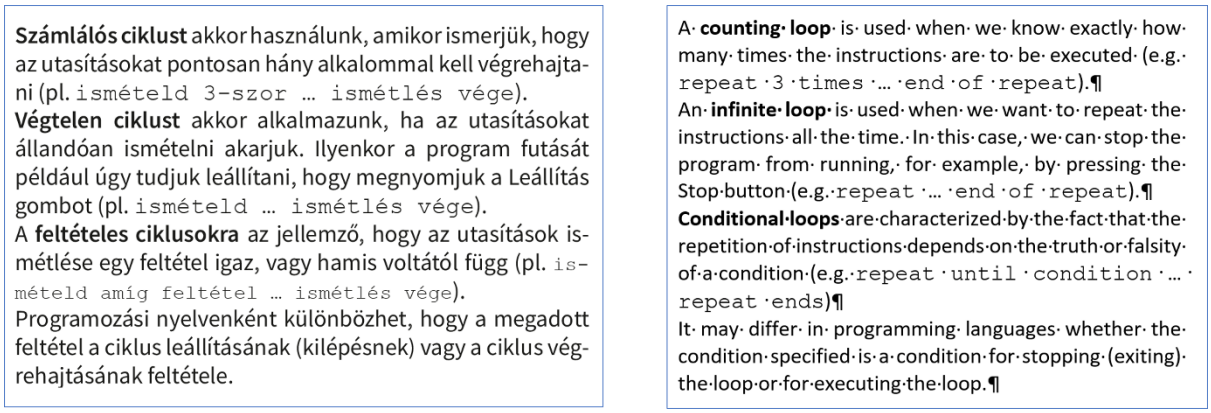
Talpas·belterületén,·az·iskola·mellett,·*kétszintes·családi·ház*·eladó.·A·120·m²-es·házhoz·600·m²-es·telek·tartozik.·Az·udvaron·45·m²·alapterületű,·utcára·nyíló,·**vállalkozásra·alkalmas**·épület·is·van.¶

Érdeklődni·*kizárólag·személyesen*:·SONKA·SÁMUEL,·Talpas,·Egyenes·u.·3.¶

Figure 3: The result of incorrect font type used in a word processing task in the Grade 9 course book. The word *ház* is changed to *vízerőmű* which contains ő and ű letters.

There is one further problem with text formatting tasks in the related topics, namely the selection of font type. In the Hungarian language, there are letters which are not included in most of the font types. The course books simply ignore this fact and provide examples with incorrect font types. In this case, the characters are replaced with characters of the Normal style (Figure 3).

Programming. Programming is taught in all grades; however, we only have access to Grades 5–6 and 9–10. What is obvious in the programming chapters of the Grades 5–6 course books is that they rather serve as glossaries or dictionaries (Figure 4) than teaching-learning materials. The definitions and the descriptions are usually accompanied with lists of tasks; however, there are no solutions or guidance provided with the presented tasks. Full guidance to novice programmers should be extremely important [13] but both teachers and students have to solve these problems on their own without any help. Furthermore, there is no continuity or knowledge transfer between the different programming chapters, which further deepens the problem we are faced with (Figure 5). Along with this, it is never mentioned that spreadsheeting is functional programming, and the algorithms behind the functions are never discussed.


Számlálós ciklust akkor használunk, amikor ismerjük, hogy az utasításokat pontosan hány alkalommal kell végrehajtani (pl. `ismételd 3-szor … ismétlés vége`).
Végtelen ciklust akkor alkalmazunk, ha az utasításokat állandóan ismételni akarjuk. Ilyenkor a program futását például úgy tudjuk leállítani, hogy megnyomjuk a Leállítás gombot (pl. `ismételd … ismétlés vége`).
A **feltételes ciklusokra** az jellemző, hogy az utasítások ismétlése egy feltétel igaz, vagy hamis voltától függ (pl. `ismételd amíg feltétel … ismétlés vége`).
Programozási nyelvenként különbözhet, hogy a megadott feltétel a ciklus leállításának (kilépésnek) vagy a ciklus végrehajtásának feltétele.

A·**counting·loop**·is·used·when·we·know·exactly·how·many·times·the·instructions·are·to·be·executed·(e.g.·`repeat·3·times·…·end·of·repeat`).¶
An·**infinite·loop**·is·used·when·we·want·to·repeat·the·instructions·all·the·time.·In·this·case,·we·can·stop·the·program·from·running,·for·example,·by·pressing·the·Stop·button·(e.g.·`repeat·…·end·of·repeat`).¶
Conditional·loops·are·characterized·by·the·fact·that·the·repetition·of·instructions·depends·on·the·truth·or·falsity·of·a·condition·(e.g.·`repeat·until·condition·…·repeat·ends`)¶
It·may·differ·in·programming·languages·whether·the·condition·specified·is·a·condition·for·stopping·(exiting)·the·loop·or·for·executing·the·loop.¶

Figure 4: The introduction of loops in Grade 5. Three types of loops are listed without tasks to understand and practice them. Furthermore, the fourth paragraph talks about the exit of loops, which students without firm programming background would not understand.


counting loop (for)
infinite loop
conditional loop (while)

conditional loop (while)
stepping loop (for)

Figure 5: The three types of loops introduced in Grade 5 (left), and the two in Grade 9 (right). However, it is not mentioned that all the listed loops are conditional, nor are the connections between the listed types are given.

2.2. The quality of figures

The quality of figures in Grades 5 and 9 course books is extremely low [6, 7] (Figure 1, Figure 2). In general, figures must be presented in excellent quality in any course book. However, in books where the quality of images and editing pictures are handled as topics, this shortcoming of the books is extremely annoying (Figure 6).

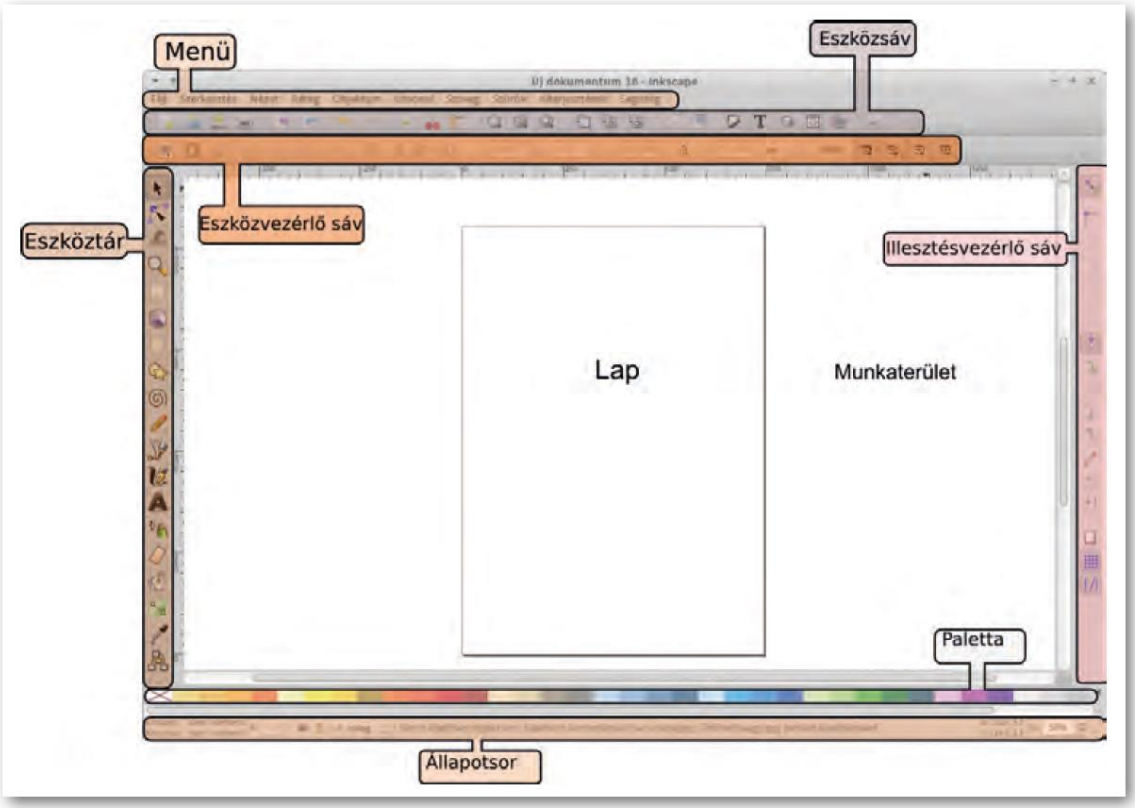
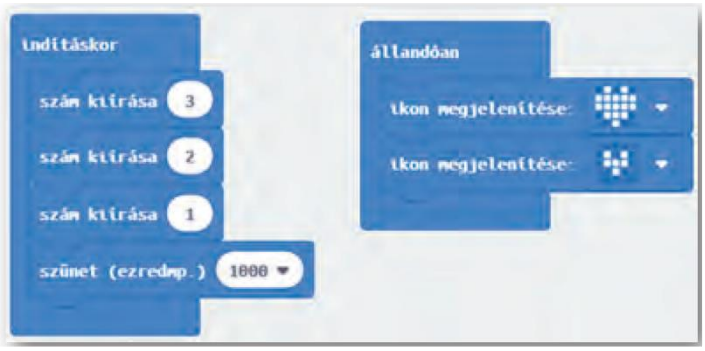
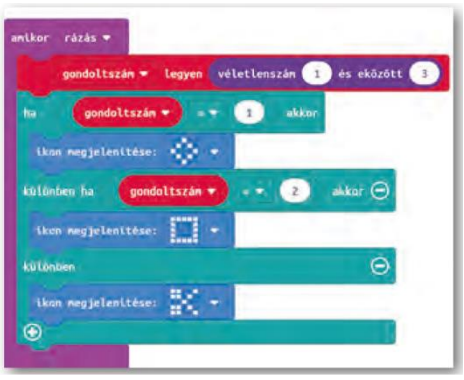


Figure 6: An illegible figure in Grade 9 course book explaining the window of Inkscape.

The frustration caused by the low quality of pictures does not decrease when tasks should be solved based on illegible figures (Figure 7).



Let us try what will happen in the simulator/device as a result of the blocks shown here.



Try the code shown here! Let's play some games in pairs!

Figure 7: Illegible figures in Grade 5 course book giving codes which students have to understand and try out (the size of pictures is similar to their size in the printed book).

2.3. Handling files

Saving files might seem to be a minor issue, but this is not so. Without a firm knowledge of handling files, students cannot facilitate fast thinking [18] when the sharing/uploading/downloading of their or others' work is required. This knowledge is also fundamental when classes other than informatics require file sharing and downloading [1].

Actually, saving is hardly mentioned in the *Digital literature* course books. Programming topics are exceptions, where it is emphasized that "usually one-word names are used without accents as program names" [7]. However, in other cases, saving is not practiced, and the examples carry accented filenames of several words or meaningless filenames (e.g., first.py). Students must learn and practice how to name a file not only in a programming context but also in various circumstances to build up firm schemata [18, 19] and help communication between the sender and the receiver.

The source files added to the course books only occasionally follow the suggested "one-word filename without accented and special characters". Good examples might help students to learn how to name files correctly, which might lead to good practice.

In connection with sharing files, compression of folders is first mentioned in Grade 9 [7] and the whole chapter is repeated verbatim in Grade 10 [9]. Compression must be taught a lot earlier, since students must upload their digital products to the school's learning platform.

2.4. General view

Consistency. There is no agreement between the authors of the course books on how to spell fundamental expressions (e.g., Clipboard vs. clipboard, Ctrl + S vs. Ctrl S, Ctrl + C vs. Ctrl C, domain vs. domén). These expressions must be written in the same way in all the course books, according to the grammar of the language and the terminology of the subject. There are various contents which are presented from different points of view in the different books and chapters without connections being built between them.

Continuity. As mentioned in Section 2.1, Grade 9 word processing ignores anything taught in previous grades. However, it is not only previously taught contents which are ignored but also materials from the same course books (Figure 8).

	Hétfő	Kedd	Szerda	Csütörtök	Péntek
1.					
2.					
3.					
4.					
5.					
6.					
7.					

Figure 8: Students are requested to type both the numbers and days (Hétfő, Kedd, Szerda, Csütörtök, Péntek, in English: Monday, Tuesday, Wednesday, Thursday, Friday) after studying one- and multi-level automated numbering in the previous chapter of the Grade 9 book.

In a similar way, after studying six years of programming in Grades 4–8, the Grade 9 course book starts programming as something completely new. The book ignores that studying programming in Python is connected to previously learned programming languages, including spreadsheeting, block programing, robotics. The title of the programming chapter in the Grade 9 course book is "Our first programs" and one of its subtitles is "Our very first program", which is not the case. It seems that the previous programming knowledge is not brought into programming classes in Grade 9, which is sad. Not one imperative code of the block programming codes is checked, compared, or analyzed.

Knowledge connected to data types is also ignored, taught as something completely new. Furthermore, algorithms are not mentioned in connection to programming, only to coding.

The publishing order of course books and the lack of continuity have undesired effects on students' attitudes towards the subject.

- No one, either students or teachers, cares what is taught or should have been taught in the previous years.
- The contents of the higher-level classes can be taught without any previous knowledge being brought into class, which questions the quality of the teaching-learning materials.
- There is no continuity between the classes and the contents. The contents are taught separately, without building up knowledge-transfer elements and connections between the pieces, which makes the teaching-learning process less effective.
- If Grade 9 can be taught without the knowledge brought into the class from Grades 3–8, then what is the purpose of these classes?

Confusing information specialists (computer scientists) with school students. It is obvious that both the frame curricula and the course books want to teach too much in the assigned number of classes. Another problem with the course books is that they do not focus on the simplest tool, thus allowing space for real problem-solving, but want to show the tools in their historical development or to provide full descriptions (e.g. the 4th paragraph in Figure 4). For a novice, not necessarily motivated student – of which we have many, we cannot deny –, a simple tool would serve better than three more old-fashioned, obsolete, or comprehensive descriptions of tools.

For example, in the introductory section in Grade 9, ways to run programs are shown in command lines and explained on long pages. This method of running programs should not be the focus of a course book designed for general studies in informatics. It might fit in the discussion section later when students have a general idea of programming. In a similar way, word processing is introduced in Grade 6 with boring typing and copying, and decontextualized searching tasks in Notepad. The Grade 9 book explains the tabulator of typewriters, which is completely out of context, and one of the major sources of the erroneous use of default tabulator positions.

3. Knowledge transfer and guidance

3.1. Knowledge transfer between school subjects

One of the most valuable aspects of the course books is that – apart from the meaningless, decontextualized tasks –, the contents derive from other school subjects. With this concept, the subject *Digital culture* tries to fulfill the requirements of Technological Pedagogical Content Knowledge (TCPK) [20]. This can be a first step to integrate computational thinking skills into the 3Rs [21]. We must call attention, however to two issues in connection to content transfer. Teachers of informatics are not the polyhistors of the 21st century; consequently, there might be topics where teachers do not have the appropriate content knowledge. The other consideration is that there are teachers of other subjects who have the content knowledge but do not have the technological knowledge, a shortcoming which does not allow them to complete the presented tasks in other classes. To develop students' computational thinking skills effectively, first the computational thinking skills of the non-informatics schoolteachers must also be developed (and not infrequently, the skills of teachers of informatics also). In the case of the 3Rs, this is obvious. The 3Rs skills are not only developed in the reading, writing, and maths classes but in all the classes. Those who cannot read, write, and do fundamental arithmetic cannot be teachers. However, having fundamental computational thinking skills – and fundamental ≠ ECDL – is not a requirement, which is something which must be changed. However, no knowledge transfer links are built up from informatics to other subjects. It is never mentioned that informatics would bring new points of view or new solutions to other sciences, which is one further shortcoming of the subject in the way it is presented.

3.2. Knowledge transfer within informatics

While the obvious presence of knowledge transfer between informatics and the other subjects increases the quality of the course books, one of their gravest shortcomings is that the informatics topics (computer sciences) are taught in isolation. We are faced with various forms of this problem.

- Topics are repeated without using and relying on knowledge officially built up in previous grades and chapters (Section 2.4). For example, text formatting is taught in word processing, presentation, and spreadsheeting, instead of being taught once, and the built up knowledge applied in the other contexts.
- Topics are introduced late without considering their use and necessity (Section 2.3).
- Topics are introduced early without considering students' age and background knowledge (e.g., presentation in Grade 5, and never again).
- Topics and book chapters are repeated verbatim in upper grades.
- There is no connection between statements, and sometimes they are contradictory. For example, spreadsheeting is taught as an office application without mentioning that it is functional programming which would serve novice and end-user programmers better than imperative languages.
- The presented tasks should be accompanied with source files to reduce typing load, allow work with real/adapted data, and to practice handling files.

However, the biggest burden of this material is that algorithms and handling data are only mentioned in the programming and database management chapters. It is completely ignored that other applications are also algorithm- and data-based, and activities in these applications should also be algorithm- and data-driven. Without seeing the algorithms and data behind computer activities problem-solving is reduced to handling tools and interfaces with low efficiency.

4. Teacher education

According to NBC 2012 and 2020, the number of classes of informatics increased from 5 to 11 (Table 1). The subject has a new name and content, and the requirements of the maturation exam have also changed. At present, teachers of informatics use the widely accepted tool-centered low-mathability approaches [22, 23] which are well presented in the course books mentioned. These teachers and the authors of the course books are experienced but mostly folk-teachers, according to the definition of Lister [24]. We need more; we need expert teachers and authors [25, 26] who are able to implement the results of computing education research [27] into their work to improve the effectiveness of the teaching-learning process [24].

Furthermore, we must calculate. At present, we do not have enough teachers of informatics to fill all the positions according to NBC 2012. The number of classes will more than double when all grades reach their time for the introduction of *Digital culture*. Both in primary and secondary education schools will not have qualified teachers who can teach informatics. Who is going to teach?

As mentioned, one of the shortcomings of the course books is that solutions and solution guidelines are not added to them. Without solutions and guidelines, neither students nor teachers (who do not necessarily have firm background knowledge) can decide whether their results are correct or not. This negligence of developing teaching-learning material is dangerous since it might lead to misconceptions and false assessment (e.g., from parents and unqualified teachers) and self-assessment.

5. Conclusions

It seems that it is easier to write a good National Base Curriculum than to prepare good teaching materials. The ideas of the NBC support the idea that computational thinking should be the fourth fundamental skill along with the 3Rs. Knowledge gained in classes of informatics (computer sciences, digital culture, etc.) should be used in other classes, for real-world problem-solving.

The analysis of the frame curricula and the available course books clearly reveals that an extremely high number of topics are presented with an extremely low number of classes, and not infrequently with

details which do not match the requirements and the background knowledge of the target population. Consequently, there is no time for practice, building up schemata, or solving problems. No connections are built up between the topics of the subject. All the chapters are taught separately, ignoring the contents of previous years, and often with repetition and verbatim copying. Primarily, it is the interface and the tools of the programs which are explained. The course books are rather glossaries than teacher-learning materials. If there are any, tasks are not accompanied by solutions. Both the teachers and the students must solve these tasks without any help from the course books.

When tasks and their solutions are presented, the tools are focused on instead of the problem-solving process. The solutions are primarily computer cooking without giving thought to the problems, their algorithms, and without discussion or debugging, and without teaching how to approach digital problems. It seems that the results of computing education research have not reached the authors of these course books.

One further problem is the lack of qualified teachers. There is no post-graduate teacher program which can support in-service teachers of informatics. With the increased number of classes of informatics, we do not have enough teachers. The same is true in the case of pre-service teachers which foreshadows a tragic outcome. There will not be qualified teachers to teach these classes. We must find solutions as soon as possible so that teachers can learn informatics. If any country is thinking of introducing informatics as a school subject, their teacher education should be prepared first. Pre-service teachers must be convinced to select this subject, professional computer scientists must be persuaded to learn pedagogy and didactics so as to be able to teach, and in-service teachers of various subjects should learn informatics. The tools are in the schools, but to be successful, we are in great need of qualified teachers and teaching materials.

6. References

[1] OFI: National Base Curriculum 2020, in Hungarian: 202/2007. (VII. 31.) Korm. rendelete a Nemzeti alaptanterv kiadásáról, bevezetéséről és alkalmazásáról (2020). URL: http://ofi.hu/sites/default/files/attachments/mk_nat_20121.pdf, last accessed 2022/04/11.

[2] OFI: Frame Curricula for Grades 1-4. Digital Culture Grades 3-4, in Hungarian: Kerettanterv az általános iskola 1–4. évfolyama számára. Digitális kultúra 3–4. évfolyam. URL: https://www.oktatas.hu/pub_bin/dload/kozoktatas/kerettanterv/Digitalis_kultura_F.docx, last accessed 2022/04/11.

[3] OFI: Frame Curricula for Grades 5-8. Digital Culture Grades 5-8, in Hungarian: Kerettanterv az általános iskola 5–8. évfolyama számára. Digitális kultúra 5–8. évfolyam. URL: https://www.oktatas.hu/pub_bin/dload/kozoktatas/kerettanterv/Digitalis_kultura_A.docx, last accessed 2022/04/11.

[4] OFI: Frame Curricula for Grades 9-12. Digital Culture Grades 9-11, in Hungarian: Kerettanterv az általános iskola 9–12. évfolyama számára. Digitális kultúra 9–11. évfolyam. URL: https://www.oktatas.hu/pub_bin/dload/kozoktatas/kerettanterv/Digitalis_kultura_K.docx, last accessed 2022/04/11.

[5] OFI: National Base Curriculum 2012, in Hungarian: 110/2012. (VI. 4.) Korm. rendelete a Nemzeti alaptanterv kiadásáról, bevezetéséről és alkalmazásáról (2012). URL: http://www.nefmi.gov.hu/letolt/kozokt/nat_070926.pdf, last accessed 2022/03/25.

[6] A. Lénárd, A. Abonyi-Tóth, N. Turzó-Sovák, P. Varga: Digital Culture 5, in Hungarian: Digitális kultúra 5. Oktatási Hivatal. (2020), URL: https://www.tankonyvkatalogus.hu/pdf/OH-DIG05TA__teljes.pdf, last accessed 2022/04/11

[7] P. Varga, K. Jeneiné Horváth, Z. Reményi, Cs. Farkas, I. Takács: Digital Culture 9, in Hungarian: Digitális kultúra 9. Oktatási Hivatal. (2020), URL: https://www.tankonyvkatalogus.hu/pdf/OH-DIG09TA__teljes.pdf, last accessed 2022/04/11.

[8] A. Abonyi-Tóth, Cs. Farkas, N. Turzó-Sovák, P. Varga: Digital Culture 6, in Hungarian: Digitális kultúra 6. Oktatási Hivatal. (2021), URL: https://www.tankonyvkatalogus.hu/pdf/OH-DIG06TA__teljes.pdf, last accessed 2022/04/11.

[9] A. Abonyi-Tóth, Cs. Farkas, K. Jeneiné Horváth, Z. Reményi, T. Tóth, P. Varga: Digital Culture 10, in Hungarian: Digitális kultúra 10. Oktatási Hivatal. (2021), URL: https://www.tankonyvkatalogus.hu/pdf/OH-DIG10TA__teljes.pdf, last accessed 2022/04/11.
[10] OH: The content of the maturation exams of May-June 2022, in Hungarian: A 2022. május-júniusi érettségi vizsgák nyilvánosságra hozott anyagai. URL: https://www.oktatas.hu/kozneveles/erettsegi/2022tavaszi_vizsgaidoszak/2022tavasz_nyilvanos_anyagok_1, last accessed 2022/04/14.
[11] Caspersen, M, E. (Chair), Diethelm, I., Gal-Ezer, J., McGettrick, A., Nardelli, E., Passey, D., Rovan, B., Webb, M.: Informatics Reference Framework for School. The Informatics for All coalition (2022). URL: https://www.informaticsforall.org/, last accessed 2022/03/27.
[12] OH: Digital Culture 9, in Hungarian: Digitális kultúra 9. Sources. In Hungarian: Forrás állományok. URL: https://www.tankonyvkatalogus.hu/csatolmanyok/digit-kult9.zip, last accessed 2022/04/11.
[13] P. A. Kirschner, J. Sweller and R. E. Clark: Why Minimal Guidance During Instruction Does Not Work: An Analysis of the Failure of Constructivist Discovery, Problem-Based, Experiential, and Inquiry-Based Teaching. Educational Psychologist, vol. 41, no. 2, (2006) 75–86.
[14] G. Polya: How To Solve It: A New Aspect of Mathematical Method. Princeton University Press, Princeton, New Jersey, 1957.
[15] P. Virágvölgyi: The Mastery of Typography with Computers. In Hungarian: A tipográfia mestersége számítógéppel, Osiris Kiadó Kft., 2004.
[16] D. Jury: About Face: Reviving The Rules Of Typography, RotoVision SA, Switzerland, 2004.
[17] D. Jury: What is Typography?, RotoVision SA, Switzerland, 2006.
[18] J. Sweller, P. Ayres and S. Kalyuga: Cognitive Load Theory. Berlin: Springer, 2011.
[19] D. Kahneman: Thinking, Fast and Slow. New York: Farrar, Straus; Giroux, 2011.
[20] C. Angeli, and N. Valanides: Technological Pedagogical Content Knowledge: Exploring, Developing, and Assessing TPCK. 2015th Edition. Springer, 2015.
[21] J. Wing: Computational thinking. In Communications of the ACM, vol. 49, no. 3, (2006) 33–35, doi: 10.1145/1118178.1118215.
[22] P. Baranyi and A. Gilányi: Mathability: Emulating and enhancing human mathematical capabilities. In: 2013 IEEE 4th International Conference on Cognitive Infocommunications (CogInfoCom), (2013) 555–558.
[23] P. Baranyi, A. Csapo A., & G. Sallai: Cognitive Infocommunications (CogInfoCom), Springer International Publishing Switzerland, 2015. p.191. (978-3-319-19607-7, URL: http://www.springer.com/us/book/9783319196077#aboutBook
[24] R. Lister: After the gold rush: Toward sustainable scholarship in computing. in ACE '08: Proceedings of the tenth conference on Australasian computing education. vol. 78, (2008) 3–17.
[25] J. Hattie: Visible Learning for Teachers: Maximizing Impact on Learning. Routledge, 2012.
[26] M. Csernoch: Thinking Fast in Computer Problem Solving. Journal of Software Engineering and Applications 10(01). (2017) 11–40.
[27] L. Malmi, J. Sheard, P. Kinnunen, Simon, J. Sinclair: Computing Education Theories: What Are They and How Are They Used?, ICER ‘19: Proceedings of the 2019 ACM Conference on International Computing Education Research July 2019, (2019) 187–197 https://doi.org/10.1145/3291279.3339409.

From Non-Existent to Mandatory in Five Years – The Journey of Digital Education in the Austrian School System

Corinna Hörmann[1], Eva Schmidthaler[1], Lisa Kuka[1], Marina Rottenhofer[1] and Barbara Sabitzer[1]

[1]*Johannes Kepler University Linz, Altenbergerstraße 69, 4040 Linz, Austria*

Abstract

The concept of life-long learning has become progressively important over the last few decades. As part of this development the European Digital Competence Framework for the Digital Competence of Educators (DigCompEdu), a scientifically sound framework, responds to the need that every European citizen should gain necessary competences for enhancing and using digital technologies in a critical, innovative, and creative way. As long ago as 1985, Austria introduced the subject "Computer Science" in grade 9. Quite a long time there was solely this one year of mandatory IT-education during school career. When Austria implemented the mandatory curriculum "Digital Education" (Digitale Grundbildung) in September 2018 for all students in lower secondary education, 21st century skills finally found their formal way into additional grades. School administration could determine if they offer "Digital Education" as a stand-alone subject or if they implement the curriculum in an integrative way in several other subjects. Finally, in the school year 2022/23, "Digital Education" will be installed as compulsory subject in regular Austrian timetables. This paper reports on the journey of digital education in the Austrian school system – from the beginning with solely computer science topics to a stand-alone subject covering digital competences, media competences, as well as civic education.

Keywords

Digital Education, Digital Literacy, STEM, Computer Science

1. Introduction

Digitization is changing our working world drastically. New professions are emerging and adults will need new skills and qualifications in the future. Therefore, there is an urgent demand to react in the educational world in order to optimally prepare today's children for the challenges of the digital future. In addition to the requirements of the forthcoming professional life, the COVID-19 pandemic has also shown the importance of being equipped with digital skills and the need to intensively promote them. The rapid global development of the pandemic has presented educators with major challenges in continuing to teach regularly. With no preparation time,

Informatics in Schools. A step beyond digital education. 15th International Conference on Informatics in Schools: Situation, Evolution, and Perspectives, ISSEP 2022, Vienna, Austria, September 26–28, 2022

corinna.hoermann@jku.at (C. Hörmann); eva.schmidthaler@jku.at (E. Schmidthaler); lisa.kuka@jku.at (L. Kuka); marina.rottenhofer@jku.at (M. Rottenhofer); barbara.sabitzer@jku.at (B. Sabitzer)

0000-0002-4770-6217 (C. Hörmann); 0000-0001-9633-8855 (E. Schmidthaler); 0000-0002-0000-5915 (L. Kuka); 0000-0001-5772-0672 (M. Rottenhofer); 0000-0002-1304-6863 (B. Sabitzer)

schools had to switch to Emergency Remote Teaching (ERT), meaning that lessons are taught online instead of face-to-face [1]. Besides technical challenges educators and students were faced with, ERT also carries the stigma that it is less valuable for academic success [1].

However, the elimination of fixed structures in everyday school life due to the pandemic has not only brought disadvantages, but also led to an enormous boost in innovation and digitization and confirmed the importance of digital education.

After the subject of computer science was introduced for the 9th grade in 1985, the next big change only came in 2018 with the introduction of "Digital Education" as an independent subject or integrated into other subjects of the lower secondary school. In the school year 2022/23, the subject "Digital Education" should finally be anchored in 5th - 7th grade as a compulsory subject in the timetable and teachers should receive intensive further training. Grade 8 will follow in the consecutive year.

This paper presents an overview of the journey of digital education in the Austrian school system and provides an insight into the most important topics covered in this subject. The remainder of the paper is structured as follows: Section 2 provides an overview of the subject computer science in the Austrian school system, whereas sections 3.1 and 3.2 give an insight into the background and first integration of the subject "Digital Education" in lower secondary schools from 2018-2022. Finally, section 3.3 presents an outlook for the implementation of "Digital Education" as a compulsory subject starting in autumn 2022.

2. Computer Science in Austrian Schools

In 1985, computer science was introduced as a separate subject in the Austrian school system. Up to now, the Austrian curriculum only implements two hours computer science lessons a week in the 9th school level of Academic Secondary Schools. Even though Austria can look back on 37 years of computer science education in school, it has merely changed over the centuries. As a "newly" introduced subject, it had to compete with long-established subjects and justify the considerable costs for the necessary infrastructure and additional space requirements. The continuing rapid progress of information and communication technologies repeatedly tempted teachers to focus on technologies and products rather than on the underlying educational content and concepts [2]. In addition, computer science has neither a long tradition nor a lobby in the form of a long-established professional association [2, 3].

Another aspect that still is part of a heated debate is about the content of the subject. Although there is a curriculum, the content is described vaguely and leaves space for individual interpretation. Accordingly, Academic Secondary Schools, focusing on general education, had to decide what constitutes as general education content. Focusing on a device, the computer, typing skills or operating a computer are rather skills than education. Nevertheless, compulsory computer science classes at Academic Secondary Schools are largely reduced to the basics of computer-use as described in the European Computer Driving Licence (ECDL) curriculum, for example. However, they present young people with a distorted picture of the subject of computer science and can hardly convey higher educational values for an information society. Computer science as a school subject should do both – convey fascination of the subject, but also show that general educational content can be taught in this subject that has nothing to do

with a later career perspective as a computer scientist, but rather prepares for life in a modern society in which information plays a dominant role [4].

3. Digital Education in Austria

As long ago as 1985, Austria introduced the subject "Computer Science" in grade 9. Quite a long time there was solely this one year of mandatory IT-education during school career. When Austria implemented the mandatory curriculum "Digital Education" (Digitale Grundbildung) in September 2018 for all students in lower secondary education, 21st century skills finally found their formal way into additional grades. Over the past two decades the idea of life-long learning has become increasingly important. Back in 2006 "digi.komp", a model for digital competences, was introduced in Austria. It is divided into four groups depending on the school grade. *Digi.komp4* describes the model until grade four, *digi.komp8* states examples from grade five until grade eight, *digi.komp12* puts forward competences for grades nine to twelve, and further *digi.kompP* characterizes the model for teachers [5]. The Austrian "digi.kompP" competence model strongly relates to the European DigCompEdu framework.

The Joint Research Centre of the European Union defines "Digital Competence" as the following [6]:

> Digital Competence is the set of knowledge, skills, attitudes (thus including abilities, strategies, values, and awareness) that are required when using ICT and digital media to perform tasks; solve problems; communicate; manage information; collaborate; create and share content; and build knowledge effectively, efficiently, appropriately, critically, creatively, autonomously, flexibly, ethically, reflectively for work, leisure, participation, learning, socialising, consuming, and empowerment.

The European Framework for the Digital Competence of Educators (DigCompEdu) responds to the – due to the COVID-19 pandemic especially required – need that every European citizen should gain these necessary competences (as defined in [6]) to use digital technologies in a critical, innovative, and creative way. DigCompEdu provides a structure to understand what it means to be digitally competent in particular for an European educator. The scientific framework provides a sound background that can guide policies in different countries [7].

3.1. Masterplan for Digitalization

In 2018 the Austrian government published a *master-plan for digitalization* in which three sub-projects were presented. The first partial project "teaching and learning content" concentrates on revision of existing curricula, whereas digital content must be integrated. Moreover, it introduces the new subject "Digital Education" and furthermore demands the development and purchase of digital teaching and learning tools and materials for classrooms. Sub-project two defines "teacher training and teacher education". "Infrastructure and modern school administration" forms the third part of the Austrian master-plan and focuses on the expansion of technical infrastructure, installation of digital devices (technical & administrative), and simplifying school administration throughout the deployment of practice-oriented programs and tools [8].

3.2. 8-Point-Concept

The master-plan also presented an 8-point-concept to foster digital education. With its thematic setting, the concept includes all central areas of the education system that are inevitable for high-quality, future-oriented school operations.

Firstly, the objective of *"Portal Digital School" (PoDS)* was installed to bundle existing applications in order to provide consolidated and clear information that is easy to use, and additionally, support everyday school life in different ways. PoDS offers a uniformed online platform with essential applications for pedagogy and administration, for the technical support of the digitized learning process of students, and for support by educators. The portal has been available for federal schools since September 2020 and one year later for compulsory schools [9].

To support the implementation of the recommendations for standardization of the platforms, instructions and assistance are provided on the distance learning service portal of the BMBWF (Bundesministerium für Bildung, Wissenschaft und Forschung – Federal Ministry of Education, Science, and Research). School administration is supported in initiating and accompanying the process of standardizing the platforms at the school location [10].

Next, as part of a *"Massive Open Online Course" (MOOC)*, educators are well prepared for teaching in blended- and distance learning settings by implementing information and communication technologies. As a virtual format, MOOC is a supplement and extension of the offerings at teacher training colleges, where educators acquire basic knowledge for teaching in digital classrooms using mobile devices. The BMBWF recommended participation in preparation for the roll-out of digital devices at secondary level I. More than 11,000 Austrian teachers took part in the first accompanied round starting in August 2020 [11].

As a digital platform, the *"Eduthek"* provides in-depth materials for all types of schools and subjects from elementary up to secondary education. In the still ongoing expansion of the Eduthek, all digital teaching and learning resources are linked with the respective Austrian curricula [12].

Next, the installation of standardized *qualification marks* in the evaluation and certification of learning applications for mobile, blended, and distance learning is planned. The qualification mark is intended to provide orientation and assistance in the selection of innovative learning tools that are already available in Austria. The "Education Agency OeAD" developed and piloted the certification procedure, which started in September 2021 [13].

In order to be able to fully exhaust all possibilities of digitization in education and to promote the use of IT in schools, basic IT infrastructure (e.g. fiber optic broadband connection, powerful and sufficient WIFI coverage) is an absolute prerequisite. As part of the 8-point-concept the framework conditions for digitally supported teaching at federal schools will be significantly improved by 2023. All federal schools should provide a high-performance broadband connection in individual classrooms [14].

For the purpose of ensuring that each Austrian secondary school student and teacher has access to their own learning and teaching device, it was planned to fund digital devices in the 5^{th} and 6^{th} grades in the school year 2021/22. However, the COVID-19 pandemic caused considerable shipping problems, accordingly not all Austrian 5^{th} and 6^{th} graders have received a device yet [15]. Currently there are no devices for teachers available, so educators still have to (partially) finance their digital devices and other tools (e.g. learning applications and e-books)

on their own [16].

3.3. Introduction of the Subject "Digital Education"

The subject "Digital Education" was launched in September 2018 in lower secondary education (grades five to eight), implemented by two to four weekly lessons. There was a high level of flexibility and autonomy in the operational implementation – the content could be taught both in specially designed hours or in an integrated manner in existing subjects, whereas the organization and content distribution over the four school years of secondary level I could be determined autonomously at each school location [17]. In a survey conducted in Upper-Austria, 26% of the participating schools claimed that they introduced a stand-alone subject, 21% said they integrate the curriculum in other subjects, 45% picked "mixed", and 9 (8%) chose "other" [18]. Furthermore, from 2018 to 2022 students were solely graded with "successfully completed" or "not successfully completed".

"Digital Education" covers digital competences, media competences, as well as civic education. According to the curriculum, those three topics should not be taught separately but must be connected to other subject-specific fields. The *BGBL (Bundesgesetzblatt für die Republik Österreich — federal law gazette of Austria)* states that the main aim is to raise students who deal with media and technology responsibly and well-briefed [19].

The eight subject-specific topics are described as the following [19]:

1. Social aspects of digitalization: reflecting the usage of digital devices in everyday life as well as benefits and ethical boundaries
2. Information, data, and media: queries, evaluating sources, sharing information
3. Operating systems & standard software: basic knowledge of operating systems, text processing, presentation software, calculations
4. Media design: adopting, producing, and adapting media
5. Communication & social media: different communication platforms, creating digital identities, cloud-sharing
6. Data security & privacy: securing devices as well as private data
7. Technical problem solving: solving basic IT problems
8. Computational thinking: working with algorithms, creative usage of programming languages

The first topic *social aspects of digitalization* addresses digital devices and processes of daily life, as well as the possible impacts on the ongoing digitalization. Furthermore, students should discuss the merit, standards, and different interests in respect of the usage of digital media concerning ecological, religious, political, and cultural aspects [20].

Information, data, and media lists topics like searching, finding, comparing, and evaluating data. Therefore, it is of high importance that students learn how to identify reliable and valuable sources and how to cite those. Furthermore, students should know how to organize data and work with different file formats. Of course, they should also know how to cope with cloud-based file systems and how to share information with others [20].

The third section of the curriculum deals with *operating systems and standard software*, where students should learn how different operating systems work and how they vary from area to

area. Text processing, presentation, and spreadsheets are defined as the three main applications of *standard software* [20].

Media design covers the creation of various digital media formats, as well as learning to deal with information in a conscious way. Henceforth, it is crucial to differ between mass media and individual media, as well as the discussion of media manipulation, influence of media, and covert advertising [20].

In the fifth topic of the curriculum students get to know the terms *communication & social media*. They get acquainted with different digital communication tools and learn how to behave when using those ("netiquette"). Moreover, students cope with problematical messages or situations like "cyber-mobbing", hate speech on the net, "sexting", or "cyber-grooming", and get to know strategies how to evade and report those. Of course, it is also very important to create online identities carefully and to be aware of risks or misuse of personal information [20].

Data security and privacy describe the risks and dangers of digital environments, as well as the protection of digital devices against computer viruses and malware. Furthermore, it is essential that students learn how to set strong passwords and understand the basics of data encryption. Concerning *privacy*, students should be able to draw a distinction between direct, indirect, and non-personal data, as well as sensitive and non-sensitive data [20].

The seventh chapter of the curriculum of "Digital Education" in Austria covers *technical problem solving*. Hence, it is important that students know the basic components of a computer and a network. They should also be able to define and describe everyday bugs concerning digital devices, select appropriate solutions, or know how to apply help systems. Furthermore, students should be competent in the usage of data protection and backup systems [20].

The last section covers the world-wide popular competence of *Computational Thinking*. Jeanette Wing [21] defines the term as a fundamental 21st century skill for everyone. It is best described as a problem-solving process with distinctive problem-solving techniques and general intellectual practices [21]. Students should learn how to describe their everyday procedures, as well as postulate and understand well-defined, finite sequences (algorithms). Moreover, the curriculum states that students should develop simple programs and identify basic programming structures, like conditional statements, loops, or methods [20].

Concerning Oppl et al. (2021) the eight sub-areas depicted in the curriculum are perceived as having different levels of importance by both school administrations and teachers. In terms of budgeting with the available time resources, it can be assumed that the different sub-areas will also be given different amounts of attention by teachers. It is striking that the topics, which from the point of view of the learning outcomes can be assigned to the computer science perspective ("Computational Thinking" and "Technical Problem Solving"), are considered less important by the majority of teachers and school management than those areas that deal with media education and application competences [17].

3.4. Compulsory Subject Digital Education in Austria

In November 2021 the Austrian Minister of Education Heinz Faßmann presented the implementation of the compulsory subject "Digital Education" in the school year 2022/23. Surprisingly, one week after that Martin Polaschek took his seat – still he went on with the plan of his predecessor [22].

Besides, the major difference is that from 2018 to 2021 students were solely graded with "successfully completed" or "not successfully completed", whereas now they will receive traditional grades in five stages from "very good" to "inadequate" when completing the subject "Digital Education".

The revised model for the subject "Digital Education" plans to implement one annual hour peer week from grades 5 to 7 starting in the school year 2022/23. The new competence-oriented curriculum will be installed with the beginning of the school year 2023/24 in both primary level and secondary level I and to that end, "Digital Education" will be compulsory for all students. In addition, the new curriculum introduces the overarching topics "IT education" and "media education" starting with the 1st grade and their mandatory implementation in other lessons [23].

A group of experts from universities and teacher training colleges was entrusted with the creation of a draft for the new curriculum for "Digital Education". This draft was created with the help of national and international competency models [23]. In March 2022 the concepts of the new curriculum were presented by the Ministry of Education by implementing the 4C's of the 21st century: Critical Thinking, Creativity, Collaboration, and Communication [24]. Educators should help students to gain these skills to prepare for successful careers when entering the workforce [25].

A well established, two dimensional competence model forms the basis of the presented curriculum of "Digital Education" (see figure 1) [24]:

		orientation	information	communication	production	interaction
(T)	structures and functions of digital, IT, and media systems					
(G)	social interactions through the use of digital technologies					
(I)	interaction in the form of usage, action, and subjectification					

Figure 1: Competence Model of Austrian Curriculum "Digital Education" (adapted by the authors) [24]

The different areas of competences form the horizontal line: (1) orientation – analyzing and reflecting about social aspects of media change and digitization, (2) information – responsible handling of data, information, and information systems, (3) communication – communicating and cooperating using media systems, (4) production – creating and publishing digital content, designing algorithms, and creating software programs, (5) interaction – responsible use of offers and options of a digital world [24].

The vertical classification describes the subject-specific topics that are represented in the "Frankfurt Dreieck" (see figure 2) [24]. This theory can be seen as an extension of the famous "Dagstuhl-Dreieck" but concentrates on the aspects of digital education.

The three central concepts are based on the following perspectives: (T) technical-media – structures and features of digital, IT, and media systems, (G) social-cultural – social interactions

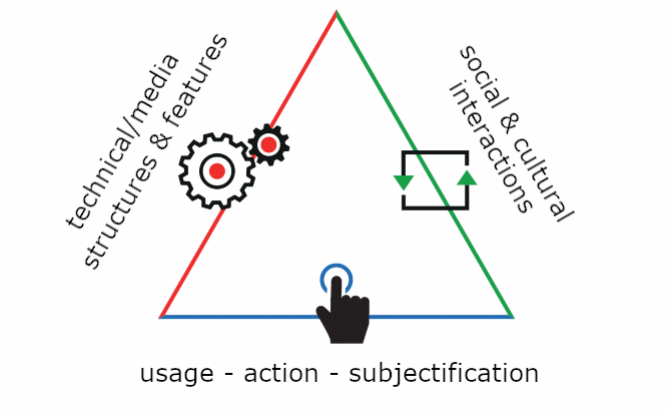


Figure 2: Frankfurt Dreieck (adapted by the authors) [26]

through the use of digital technologies, and (I) interaction-related – interaction in the form of usage, action, and subjectification [27].

The general plan of the curriculum of 2022 is very similar to the old one. Still, some differences in the exact formulation of competences exist that should be analyzed in further studies.

4. Conclusion and Outlook

The present paper displayed an overview of the journey of digital education in the Austrian school system and wanted to grant insight into the most important topics covered in this subject. When Austria implemented the mandatory curriculum "Digital Education" in September 2018 for all students in lower secondary education and with the compulsory subject starting in 2022, digital literacy finally found its official way into additional grades.

With the introduction of the compulsory subject another problem appeared, as currently no entire "Digital Education" studies in Austrian teacher education exist, as there is for other traditional subjects. In autumn 2022 postgraduate training for teachers is planned to tackle the lack of fully trained staff in "Digital Education". Besides, there still is no schoolbook that covers the current curriculum available yet.

To help educators dealing with the unfamiliar curriculum of "Digital Education", it is necessary to create a large collection of material for the specific topics, especially for the technical oriented ones. Henceforth, there is still a lot of effort that has to be put into the implementation of the curriculum to gain further motivation of the teachers.

References

[1] P. Bawa, Learning in the age of SARS-COV-2: A quantitative study of learners' performance in the age of emergency remote teaching, Computers and Education Open 1 (2020) 10. doi:10.1016/j.caeo.2020.100016.

[2] S. Friedrich, W. Hartmann, Informatikunterricht im Spannungsfeld zwischen Tastendruck und UML, in: G. Brandhofer, G. Futschek, P. Micheuz, A. Reiter, K. Schoder (Eds.), 25 Jahre Schulinformatik in Österreich. Zukunft mit Herkunft, 2010, p. 27–28.
[3] B. Döbeli, ICT im Hosensack – Informatik im Kopf?, in: G. Brandhofer, G. Futschek, P. Micheuz, A. Reiter, K. Schoder (Eds.), 25 Jahre Schulinformatik in Österreich. Zukunft mit Herkunft, 2010, p. 35–44.
[4] R. Mittermeir, Informatikunterricht zur Vermittlung allgemeiner Bildungswerte, in: G. Brandhofer, G. Futschek, P. Micheuz, A. Reiter, K. Schoder (Eds.), 25 Jahre Schulinformatik in Österreich. Zukunft mit Herkunft, 2010, p. 54–73.
[5] Bundesministerium Digitalisierung und Wirtschaftsstandort, Digitales Kompetenzmodell für Österreich DigComp 2.2 AT, Technical Report, Bundesministerium Digitalisierung und Wirtschaftsstandort, 2018.
[6] A. Ferrari, Digital Competence in Practice: An Analysis of Frameworks, Joint Research Centre of the European Commission. (2013) 91. URL: http://dx.doi.org/10.1007/978-3-642-33263-0{_}7. doi:10.2791/82116.
[7] C. Redecker, Y. Punie, European framework for the digital competence of educators: DigCompEdu, Publications Office of the European Union, 2017. URL: https://moodle.ktu.edu/pluginfile.php/428841/mod_resource/content/1/pdf_digcomedu_a4_final.pdf. doi:10.2760/159770.
[8] W. u. F. Bundesministerium für Bildung, Masterplan für die Digitalisierung im Bildungswesen, 2018. URL: https://www.bmbwf.gv.at/Themen/schule/zrp/dibi/mp.html.
[9] BMBWF, PODS – Digitale Schule – Bundesministerium für Bildung, Wissenschaft und Forschung, 2020. URL: https://digitaleschule.gv.at/portal-digitale-schule/.
[10] BMBWF, Uniform communication process – Digitale Schule – Bundesministerium für Bildungswissenschaft und Forschung, 2020. URL: https://digitaleschule.gv.at/vereinheitlichung-der-plattformen/.
[11] BMBWF, MOOC – Digitale Schule – Bundesministerium für Bildung, Wissenschaft und Forschung, 2020. URL: https://digitaleschule.gv.at/lehrenden-fortbildung/.
[12] BMBWF, Eduthek – Digitale Schule – Bundesministerium für Bildung, Wissenschaft und Forschung, 2020. URL: https://digitaleschule.gv.at/ausrichtung-der-eduthek-nach-lehrplanen/.
[13] BMBWF, Learning Apps – Digitale Schule – Bundesministerium für Bildung, Wissenschaft und Forschung, 2020. URL: https://digitaleschule.gv.at/gutesiegel-lernapps/.
[14] BMBWF, Expansion of the school's basic IT infrastructure – Digitale Schule – Bundesministerium für Bildung, Wissenschaft und Forschung, 2020. URL: https://digitaleschule.gv.at/ausbau-der-schulischen-basis-it-infrastruktur/.
[15] BMBWF, Digital devices for students – Digitale Schule – Bundesministerium für Bildung, Wissenschaft und Forschung, 2020. URL: https://digitaleschule.gv.at/digitale-endgerate-fur-schulerinnen-und-schuler/.
[16] BMBWF, Digital devices for teachers – Digitale Schule – Bundesministerium für Bildung, Wissenschaft und Forschung, 2020. URL: https://digitaleschule.gv.at/digitale-endgerate-fur-lehrerinnen-und-lehrer/.
[17] S. Oppl, W. Fuchs, M. Dobiasch, Zur inhaltlichen Schwerpunktsetzung im Rahmen der verbindlichen Übung Digitale Grundbildung an österreichischen Mittelschulen, Online

Journal for Research and Education 16 (2021). URL: https://journal.ph-noe.ac.at/index.php/resource/article/view/990/988.

[18] Author, Digital literacy in lower secondary education – a first evaluation of the situation in austria, International Conference on Informatics in Schools (2020).

[19] BGBLA, Bundesgesetzblatt der Republik Österreich: 71. Verordnung, 2018. URL: https://www.ris.bka.gv.at/Dokumente/BgblAuth/BGBLA{_}2018{_}II{_}71/BGBLA{_}2018{_}II{_}71.html.

[20] L. Sponring, Digitale Grundbildung für digi.komp8 – Handbuch für Lehrerinnen und Lehrer, E. Dorner, 2018.

[21] J. M. Wing, Computational Thinking, Communications of the ACM (2006). URL: https://www.cs.cmu.edu/{~}15110-s13/Wing06-ct.pdf.

[22] S. Rosner, V. Schiretz, Digitale Grundschulung – Wiener Zeitung Online, 2022. URL: https://www.wienerzeitung.at/nachrichten/politik/oesterreich/2143354-Digitale-Grundschulung.html.

[23] M. Polaschek, Erledigung BMBWF - Österreichisches Parlament - 9338, 2022. URL: www.parlament.gv.at.

[24] BMBWF, Projekt Lehrpläne NEU - Fachlehrplan Digitale Grundbildung, 2022. URL: https://www.ahs-informatik.com/app/download/10356599585/22-03-14_Fachlehrplan_Digitale+Grundbildung_sek+I.pdf?t=1651008169.

[25] Connections Academy, How online education builds career readiness with the 4 cs, 2013. URL: https://www.connectionsacademy.com/support/resources/article/how-online-education-builds-career-readiness-with-the-4-cs.

[26] Pädagogische Hochschule Schwyz, Medien und Informatik - PHSZ - Dagstuhl, ???? URL: https://mia.phsz.ch/Dagstuhl/.

[27] T. Brinda, N. Brüggen, I. Diethelm, T. Knaus, S. Kommer, C. Kopf, P. Missomelius, R. Leschke, F. Tilemann, A. Weich, Frankfurt-Dreieck zur Bildung in der digital vernetzten Welt – Ein interdisziplinäres Modell, 2019.

Teaching fundamental programming concepts with the BBC micro:bit

Nina Lobnig, Markus Wieser, Stefan Pasterk and Andreas Bollin

Department of Informatics Didactics, University of Klagenfurt, Universitätsstraße 65-67, 9020 Klagenfurt, Austria

Abstract

Learning to program is essential for computer science and computer science education. It helps in structured, at best computational thinking and in understanding how computers and algorithms work. We view this as a critically important skill these days. Programming can be taught in different ways using different tools and technology. In this contribution, we present materials for the BBC micro:bit specially designed for learning and practicing fundamental programming concepts such as sequential program flow, branching, variables, and loops, as well as event control. The learning materials consist of several parts and are structured step-by-step with detailed explanations and examples. Furthermore, they are supplemented by motivating, playful exercises. We describe the learning materials in detail, why they were structured in this way, and what teaching ideas and thoughts underlie their development.

Keywords

learn programming, micro:bit, block based programming, computer science education, programming concepts

1. Introduction

With the impact of computer science in society and education, also the interest of people to get in touch with this area's concepts rises. Besides related subjects at schools, other institutions offer workshops or laboratories to get in touch with topics of computer science. The University of Klagenfurt offers the *Informatics Lab* [11] (supported by the RFDZ, the regional center for subject-related didactics) since 2014, which develops teaching materials for computer science education. These materials are tested in repeatedly offered thematic workshops on a variety of computer science topics. On the website[1], there is also a platform where the developed materials are available for teachers and all other interested people. The lab does not only search for good examples, it also tests existing learning materials and continuously develops new ones.

Learning to program is part of most international curricula in different stages of education starting at early ages. Programming is often connected to computational thinking and to the understanding how computers and algorithms work. To support the process of learning, different programming environments and hardware kits have been developed and are used in

Informatics in Schools. A step beyond digital education. 15th International Conference on Informatics in Schools: Situation, Evolution, and Perspectives, ISSEP 2022, Vienna, Austria, September 26–28, 2022

nina.lobnig@aau.at (N. Lobnig); markus.wieser@aau.at (M. Wieser); stefan.pasterk@aau.at (S. Pasterk); andreas.bollin@aau.at (A. Bollin)

[1] https://www.rfdz-informatik.at/ (note: site in German)

practice. One very popular tool is the *BBC micro:bit*, a small single board computer developed for the use in school classes [1].

In the *Informatics Lab*, workshops with the BBC micro:bit are offered as it is one of the most popular tools in many (local) schools. This, along with the fun and easy part of block-based programming, was one of the reasons for choosing the micro:bit. Another reason were the many possibilities in using the micro:bit later on and the available extensions, from working with the pins and breadboard to controlling robots (e.g. Bit:Bot).

In a first version, these workshops followed an exploratory approach. After a short explanation of the programming interface and the handling of the micro:bit, the students tried out their own ideas as well as given broad tasks (such as programming 'rock-paper-scissors'). These tasks were broad in the fact that their solution required a combination of several programming concepts (in the case of the example above: branching, variables and random numbers in combination with each other). However, these concepts were not explicitly taught, the intuitive and self-explanatory usage of block-based programming was relied upon. This often led to the situation that students who had a certain way of structured (computational) thinking managed the tasks quite well and others (often the majority) needed a lot of help and so solutions were often worked out together with the teacher.

This contribution presents revised and improved learning materials for the micro:bit which are currently used in the *Informatics Lab* and follow a more structured approach with focus on teaching fundamental programming concepts.

2. Related Work and Development

2.1. Related Work

A first step in the revision process was searching for comparable materials for teaching programming with the micro:bit. We could find some promising resources, but they were quite similar to the first version of materials in the *Informatics Lab* and inherited the same difficulties. There are a lot of learning resources that have the same or similar exploratory approach and focus on the fun part of working with the micro:bit and rely on the easy to use programming environment. One example is the open educational resource textbook from Bachinger and Teufel [1] (or the web version [8]), which is mentioned on many websites for using the micro:bit in school. They already write in the preface that the book is in fact not a pure computer science nor a programming textbook, but instead 'a collection of exciting, creative and practical tasks that help students to develop an awareness of everyday problems, for which they can use their microcomputer to solve'. They also state that the fun aspect is one of the core goals ('The joy of doing is always at the center!').

The fun and motivating part was one of our goals too, but we also wanted to teach the most important, fundamental ideas of computer science and programming. The exploratory approach did not fulfill this like we wanted to. These requirements were also not met by any found resources (e.g. [1, 9, 7, 6, 13, 5] and many others) and seemed to face the same difficulties like we had at first.

Therefore, more than four years ago, two of the authors started developing a new concept for the micro:bit workshops and for learning how to program with it. The goal was to focus on

programming concepts, combined with the playful approach offered by the micro:bit. Those are basic materials and provide a good basis for going further. It can be followed by working with the hardware components of the micro:bit or going deeper into the world of programming. Furthermore, with the acquired knowledge, it should also be more easy to learn textual programming later on, because the concepts are already known and have been practiced. This was the reason for developing the materials presented here and what goal they pursue.

2.2. Fundamental Programming Concepts

Probably every computer scientist can list the most important, fundamental programming concepts or program structures as variables, branches and loops (and maybe some more). A look into different programming books shows these as well, as in 'Sprechen Sie Java?' (translated to 'Do you speak Java?') [10], which is frequently used in German-speaking countries (and especially at universities), and many other: independently whether that is Java or Python as a programming language, or you program in VBA, and whether it is object-oriented or not (see [2, 4, 3]).

Now, we look at which fundamental programming concepts can be easily realized with the micro:bit, all object-oriented elements are discarded and what remains is: writing simple, sequential programs, working with variables (with the assignment of values to them), perhaps dealing with input and output, as well as the two fundamental control structures branch and loop.

In our material, the programming concepts are first learned and understood separately and used in combination in later tasks. In this way, we hope to improve the understanding of the concepts and that students do not create code on a guessing approach, but with specific considerations. For this reason, the materials deal with branches first, followed by variables, and then the combination of both. We hope to reduce the complexity by covering only one concept at a time. The loops then complete the material, although they could theoretically also be placed in between, but since the common textbooks (see references above) deal with them in this order (first branches, then loops), this was also chosen here. To make branching and arithmetic comparisons possible without the variable concept (in a meaningful way), random numbers are necessary. This is one of the reasons why these are taught quite early (in comparison to other approaches for learning to program). Another reason is that randomization is used quite often with the micro:bit as well as in programming - it occurs in dice or random games and similar programs (for example the popular oracle or rock-paper-scissors games), that are often implemented with the micro:bit.

The concept of subroutines, such as methods and functions can be implemented with the micro:bit as well, but due to the limited programming possibilities this does not actually make sense and rather seems to be a forced approach. Working with arrays is indeed possible and micro:bit offers suitable programming blocks, but this concept is somewhat more complex. In our opinion it is not one of the basic programming concepts and therefore is not covered in the (current) basic materials. For working with arrays in a meaningful way, prior knowledge about variables and especially about branches and loops is necessary. Hence, this can be used for an extension of the (basic) materials presented in this paper.

2.3. Didactical Considerations

The reason for the materials to be structured this way is based on educational and didactic considerations collected in the *COOL Informatics* approach [12]. It fosters a fun and motivating way of learning and follows the four principles *discovery*, *cooperation*, *individuality*, and *activity*. The fun factor was a basic consideration as the possibly first contact with programming should be exciting to increase the interest. But this factor is supported by the micro:bit as it is easy to use and makes a lot of fun.

The material is divided into several parts - each with an input part that explains the particular concept and its blocks and shows an example, as well as an exercise part with several different tasks. The explaining parts always contains a practical example, which describes and illustrates the introduced concept in detail, as it is promoted in the *COOL Informatics* principle *discovery*. Indeed, the materials are also designed for *self-study* and can be used and understood without a supervising teacher and can also serve as a reference guide for learners (this is done quite often, see later in chapter 4). As such, they also offer the possibility of being used in inverted or flipped classroom settings. Due to this, the materials can be seen as a textbook and allow flexibility, especially with the students' heterogeneous backgrounds and interests, following the principle *individuality*. The material has been designed to suit the abilities of all learners, which can vary widely. Furthermore, the tasks for the different topics are mixed in terms of difficulty and include so-called expert tasks (challenging tasks - for high-achieving students or those with previous experience or deeper interest). Additionally, according to the principle *activity*, learners should be encouraged and given the chance to implement their own ideas and have room to do so. The principle *cooperation* is considered within the workshops as students are encouraged to work together on tasks and to help each other if possible.

In addition, it was a matter of concern to make the materials appealing to all genders and to ensure that the tasks do not reflect any stereotypes or anything similar. Therefore, the tasks are kept very neutral (in terms of gender) and also the shown figures and graphics, as well as the used colors are not gender-specific. In addition, the examples and exercises are not overly technical or mathematical, as is often the case in programming books and courses. This should make programming more interesting for everyone, regardless of other (technical) interests and abilities.

Finally, the materials are *Open Educational Resources*, short OER, and thus available for use, free for modification by others (under the CC-BY-NC-SA license), and open for improvements and revisions.

3. Overview of the Learning Material

The material consists of six parts and every part is a combination of an explanatory and an exercise part. In total it has more than 30 pages and another nearly 30 pages for solutions to the exercises. Therefore, we will not present all the material here and instead only take a look what the material is about and give an overview of the content and the structure as well as the type of explanations and examples given. The documents are available on the website of the RFDZ[2].

[2] https://www.rfdz-informatik.at/grundlagenmicrobit/ (German only)

The materials are aimed for children from nine years and above. It was used in workshops with children from the upper elementary school and lower secondary school. There are no prerequisite skills, other than text comprehension and reading, the presented learning materials were designed for the first encounter to programming and programming environments. Although it is not a disadvantage if experience (with Scratch, code.org or other) is already brought along.

3.1. Introduction of the micro:bit and the Programming Environment

The first part starts with a general explanation of the micro:bit and answers the questions what the micro:bit is, what features it has and what you can do with it. Also, the advantages of block-based programming are briefly mentioned. It continues with the overview of the programming environment, where to find what and how to work with it. After all the introductory bits, the first blocks - the basic ones - are presented. It starts with the two blocks 'on start' and 'forever', which are already in the programming environment at the very beginning, and then the other basic blocks are listed with short explanations. These blocks are output blocks and display the given output, like text or symbols, with the 25 LEDs at the micro:bit's front side. It concludes with a first sample program (displaying a 'Hello' on start and then showing a smiley face forever), followed by a reference to the exercises.

The exercises start with the invitation to try implementing own ideas with the presented blocks and then give some example exercises for those, who have no idea or prefer predefined tasks.

3.2. Event control (Working with Input) and Random Numbers

This part consists of two concepts and the first proceeds with simple, easy to understand blocks. Those are the ones controlling the inputs recognized by the micro:bit. This is considered easy as the micro:bit was designed for this (in contrast to many other programming environments).

The learning material for this concept again starts with a short introduction and introduces the involved blocks. The given sample code for working with this type of blocks is an extension of the first sample program (the one from the part before) and adds the code for showing an arrow to the left, whenever button A is pressed. The provided exercises (apart from the suggestion to try out your own ideas) are producing output depending on certain movements or input via buttons. For example greet the persons next to you, when the button on their side is pressed and greeting yourself for the A+B input.

The second concept to be taught in this part of the learning material is the concept of *randomization* and therefore introduces the block (from the mathematics block category) for getting a random number within a specified range. This enables to write programs that represent a dice (sample program) and other fun possibilities, like an oracle that 'predicts' how many presents someone will get for their next birthday, or to challenge your seatmate who is better at guessing the micro:bits next shown number, or to develop an a instructions generator (with an extra description list, which activity to do when a certain number pops up). The given examples already indicate, why we decided to introduce the randomization quite early in the learning materials. It enables creating fun programs with just a few blocks and a bit of creativity.

Additionally, it makes it possible to introduce the branching concept in the next step without the need for variables.

3.3. First Contact with Branching

In this part, a first contact with branching is provided for the students. It again starts with a short introduction and then goes over to the explanation of the corresponding blocks, which are the branching block itself as well as the block for conditions (like the comparison of two numbers). The concept of branching is explained as a distinction of cases and 'distinguishes between a special event or not'. We had the event handling before, but now we can define by ourselves, what a special event should be. And for each of the two cases, special event or no special event, programming blocks can be provided. The block for the (branching) condition is introduced as a so called 'question box', that can tell yes or no. With it, we can for example check if two given numbers are the same or one is smaller than the other and so on.

The two blocks are then shown in a sample program. This program is the improved version of the oracle from the exercises to *randomization* and displays a tick or cross depending on the random number given. This sample program is explained in great detail and supported by the illustration shown in Figure 1.

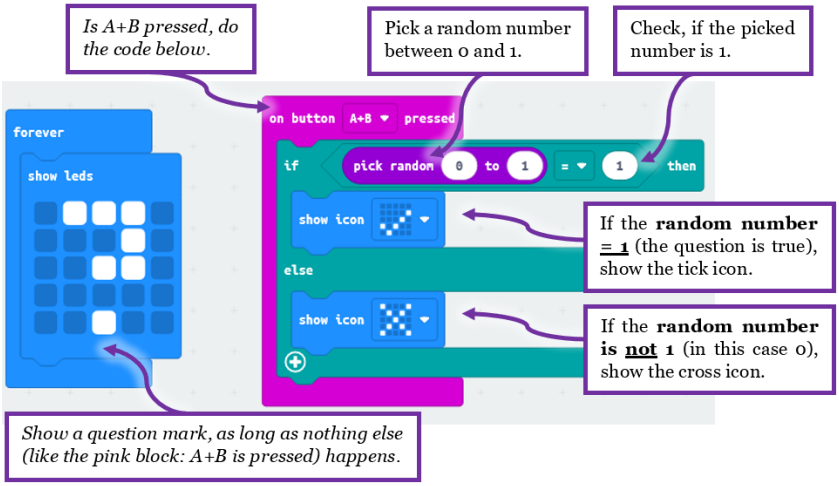


Figure 1: Explained sample code for branching

The associated exercises are comprehension questions about branching and conditions in combination with random numbers. It displays code snippets, but with a changed range for the random numbers and the students have to assign the possible random numbers to the corresponding code sections. This type of exercise is a pen and paper task, an excerpt is shown in Figure 2. But there are exercises for programming too, like implementing drawing lots with matching output text or randomly showing one of two possible icons when pressing a button or a rain and sunshine simulator (two sunny days for one rainy day) with matching icons.

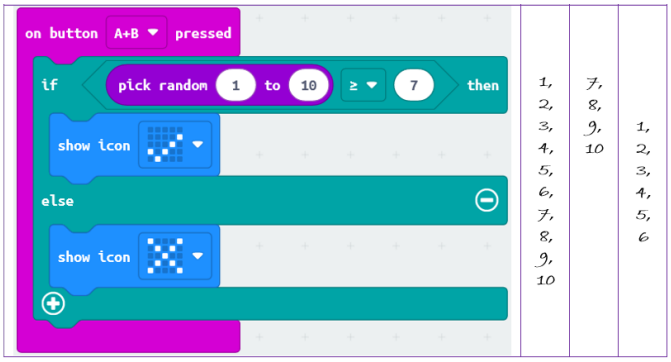


Figure 2: Excerpt of a exercise on branching and conditions with columns to write down: all possible numbers, the numbers for tick and those for cross

3.4. Start Working with Variables

After a first contact with branching and conditions and understanding and practicing this concept, the concept of variables is introduced meanwhile without combining it with branching. It is presented as the way to store values. It is explained as a kind of a cell, into which a single number can be written. This stored number can be retrieved and also be changed. The changing of a variable value is, by the way, explained like erase and newly write a value into the cell. Following this introduction, it is shown how to create a (new) variable and the related programming blocks - for retrieving the value, changing it *to* a certain number and changing it *by* a certain number.

The given sample code is a implemented counter, that goes up by one or down by two depending on the pressed button. This program is expanded with a starting value of 5. Then the question is raised, how to make the program start with a random number and how to expand the code to show a heart for reaching the target value of ten (instead of the shown number in any other case). The corresponding solution with explanations is also part of the learning material.

There are corresponding exercises for programming with variables too. From tasks for exploring the function and effect of the blocks, to various simple counter programs as well as a bit more complex tasks with negative variable values and events for reaching the target value. The task for 'experts' is one with counting how many times the micro:bit has been shook, but with a randomly chosen target value (that can be displayed). For this, two variables are needed.

3.5. Combination of Branching and Variables

In the next-to-last part of the learning material, the concepts of branching and variables are combined and we show the possibility of having more than two branching cases and how to handle that. What follows is the usual introduction and short explanation to expanding the branching block and where which code belongs. In this part, there are no new blocks. The given sample code together with detailed explanations is from the familiar oracle program, which is expanded by one more case. The added case represents the neutral oracle answer 'I don't know' and is displayed with a certain smiley icon. The corresponding code is shown in Figure 3.

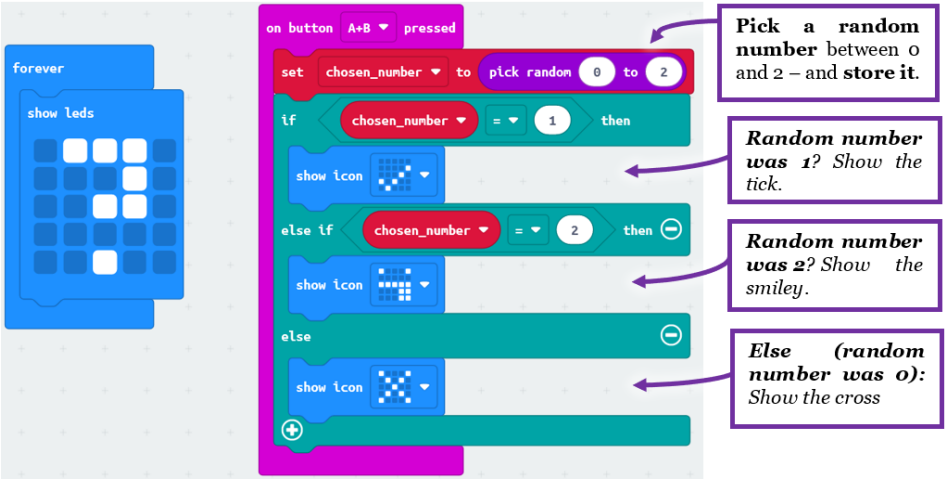


Figure 3: Explained sample code for a oracle with three cases (branching in combination with variables)

With this knowledge, it is now possible to have even more fun exercises, starting with expanded oracle games to the implementation of rock-paper-scissors. As noted before, this is part of many micro:bit materials and is also suggested as one of the most prominent games on microbit.makecode.org. Here it might become clear, why we take a rather structured approach with our materials, rather than a solely experimental approach (the problems that arise in doing so have already been briefly addressed in the mentioned section).

Other exercises in our learning material are the slightly modified game 'Rock-Paper-Scissors-Lizard-Spock' (known from a popular tv series), as well as a version of using the micro:bit as a dice with the related dice-icons. There is also a task for 'experts', which is is about implementing an instructions generator that even checks if the instructed activity was performed correctly (like pressing a certain button or moving the micro:bit in a certain way).

3.6. Repeating Code with Loops

The last part is about repeating code with loops (going beyond the already known forever loop). The three main types of loops - counting, for and while - are introduced. Each of these is explained with an example, such as repeating code exactly nine times with the counting loop or using the for loop for the additional output of the current number of repeats. The while loop is demonstrated with the example of a startup screen, which is displayed till shaking the micro:bit.

For practising the loop concept and the use of the three different loop types, there are associated exercises too. They are about reproducing a very similar code to the ones explained before (like repeating code for a specified number of times or adding a startup screen to an already existing program). Another exercise is about counting backwards and another one about implementing a counter (like in the exercises related to variables), that flashes an icon depending on the set number. The next-to-last exercise is implementing an adjustable countdown and has a follow-up question regarding the use of negative numbers. Expanding the program to work with these too is then the goal of the last, the 'expert' task (requires branching).

4. Experience with the Material

The presented learning material for the micro:bit has been around since the end of 2018 (first concept) or mid-2019 (first elaborated version) and therefore, we already worked with it multiple times as well as teachers did. We are now interested in experiences with the materials and therefore, we started a first pre-study asking three teachers about their opinions (through free text answers to questions regarding their perception and experience with the learning material).

Moreover, we wanted to learn more from the students' point of view, so we did a second pre-study (n=18) and asked those of our last micro:bit workshop to fill out a survey on their perception. The findings are very promising and show a trend (students' interest in computer science as well as in programming increased quite well), but further research is needed.

4.1. Experiences from the Teacher's Point of View

In the *Informatics Lab*, we use the materials in larger workshops when the learners are with us for several days (e.g. in summer) or the workshop consists of several units (e.g. once a week over a few weeks). Typically, we introduce and explain the concepts and corresponding blocks in front of the class and afterwards it is the students' turn for practicing the concepts and writing programs by themselves. This teaching setting is also used by the three teachers - out of one told us, that she does not use the explanatory parts often with students, but for preparation purposes or for looking things up. This teacher also wrote, she uses the exercise solutions quite often. She for example cuts them into pieces and has the students reassemble the correct code or places the solutions on the side of the classroom and the students can go and take a look when they need help or need a hint how the code should be structured.

There are quite big differences in the knowledge and experience as well as in the performance of the students, but the materials were designed for handling this heterogeneity. The materials are often printed for all the students and given to them step-by-step, so low performing students can reread things and following parts can be given ahead of time - for everyone to do challenging things and practice newly known concepts. In addition, it is possible for everyone to work at their own pace. All this (well working with heterogeneity) was also broadly mentioned by all of the teachers. The large amount of tasks (and its clear separation) was also noted very positively.

Moreover, the asked teachers view the developed materials as a good approach for learning fundamental programming concepts (using the micro:bit). And students are seemingly programming code more purposeful and less chaotic (mentioned by one teacher), but this has to be investigated further to put this as a general statement. Besides this, it is observed that the associated exercises encourage students to work with the micro:bit and practice the concepts (in a playful way), as mentioned by the teachers. Later on, it is also possible to do group projects with the micro:bit and even interdisciplinary approaches like the multiple examples found on the internet (for example the cited references from section 2.1).

4.2. Survey Results on the Student Perception

We now know quite well what the experience and opinions look like from the teachers point of view. But what about the student side? How do they perceive the lessons and the materials?

And do they - positively or negatively - influence their perception of computer science and programming? These were our central questions in a short survey the students from our last workshop filled out. The lessons (referred as 'workshop' later on) were held in April/May 2022 as part of the computer science education in a local school and 22 students (lower secondary school) took part, of which 18 completed the survey (two weeks after the last lesson). The majority of the students had no prior experience with programming and very few with computer science. The existing knowledge was only about the basic handling of the computer (turning it on, navigating, opening programs) and in software for text-editing and creating slideshow presentations.

The first four questions aimed at the perception of computer science and programming, the students had before the workshop and now have since the workshop. The precise questions were Q1: 'If you can remember, what did you think of computer science *before* the workshop (with the micro:bit)?' and Q2: 'What do you think of computer science *since* the workshop with the micro:bit?' (both translated from German). And the same for programming too (Q3 and Q4). The results are promising, although more (also comparative) data has to be collected on this in the future. But it shows a strong trend, as we can see in Figures 4 and 5. The interest in both has increased quite a lot, which are very good news (regarding our goals and the promises of the micro:bit).

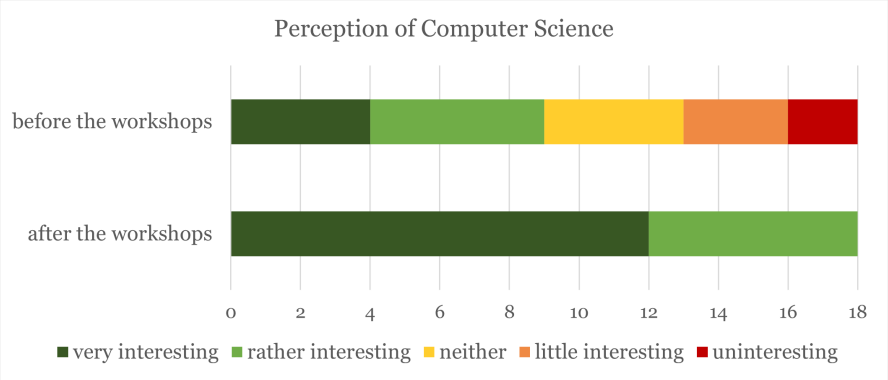


Figure 4: Results on the change of students perception to computer science (n=18)

Another part of the survey (Q5) was to tick all the words the students would describe the workshop with. There were eight predefined words ('was fun', interesting, time-consuming, exciting, motivation, confusing, difficult and boring) as well as room for other words. This was used a few times, but this can be omitted (as synonym words were used and ticked additionally to the original ones). All 18 students described the workshop to be fun and nearly all of them (16) ticked 'interesting' too. But learning to program (and programming itself) is also time-consuming, which was expressed by seven students. 'Exciting' and 'motivating' were ticked six times each, confusing four and difficult three times and boring never.

Q6 and Q7 focused on how interesting the workshop was perceived as well as the given tasks/exercises (overall). We are happy with the result, displayed in Figure 6, as it gives a good feedback to the workshop. However, the first may include side parameters (e.g. sympathy).

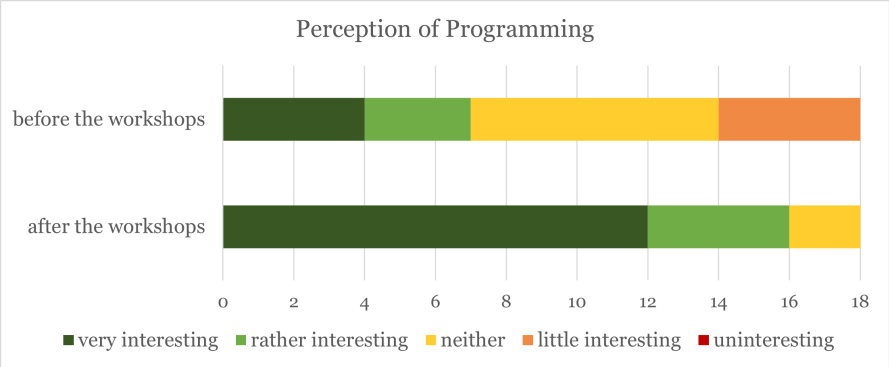


Figure 5: Results on the change of students perception to programming (n=18)

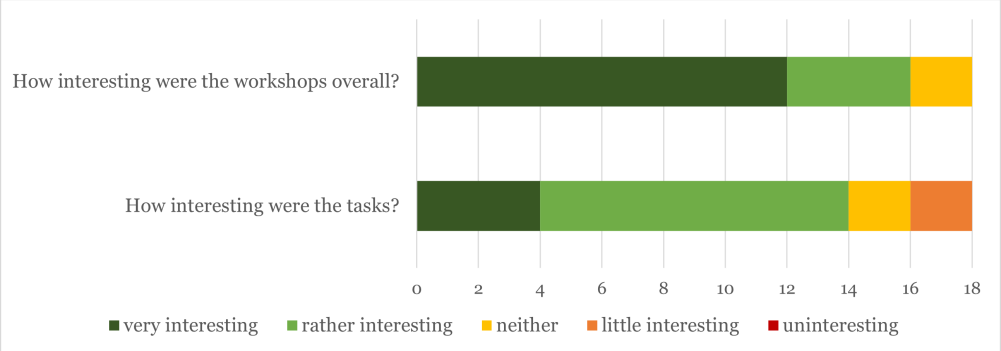


Figure 6: Interest in the programming course and the provided exercises (n=18)

Q8 was on the difficulty of the exercises. Exactly half of the students rated them as appropriate (9), a few as rather easy (3), a few more as somewhat difficult (5) and one as very difficult. Because of this, we will consider improving the tasks a little bit (for example with reassembling code snippets and other task types) in the future. The last question (Q10) was an open question (regarding further feedback), where two students reported they would have enjoyed more time and lessons with the micro:bit.

So overall, the workshop and the developed learning materials are not only perceived well by teachers, but also by the students, who seem to have a lot of fun using (and learning with) the micro:bit. The survey results are promising and we are curious on more data and research regarding the materials.

5. Conclusion and Future Work

As shown above, the developed learning materials are suitable for learning programming (using the BBC micro:bit) and at the same time increase the students' perception on computer science and programming. Even out of a teachers point of view, the materials are very practical, especially in heterogeneous classes (regarding performance and pre-knowledge in programming).

It is planned to produce videos and interactive (H5P) elements for this learning material and do further research - with more students and teachers and also on other aspects of the material. In the future, we also want to look at whether the concepts taught are also remembered and mastered in the long term. Another issue is translating the learning materials into English, but there is currently limited time and capacity for all of this. We also wish to improve the materials further in expanding the materials to more concepts (e.g. arrays) as well as the given exercises.

References

[1] Bachinger, A., Teufel M.: Digitale Bildung in der Sekundarstufe - Computational Thinking mit BBC micro:bit. 1st edn. Austro.tec, Grieskirchen (2018). Available under: https://micro-bit.eeducation.at/images/f/f2/Buch-microbit _20180729.pdf. Last accessed 19 May 2022

[2] Inden, M.: Einfach Java: gleich richtig programmieren lernen. 1st edn. dpunkt.verlag, Heidelberg (2021).

[3] Kellner, F., Brabänder, C.: Programmieren in VBA. In: VBA mit Excel. Springer, Berlin, Heidelberg (2020).

[4] Klein, B.: Einführung in Python 3: in einer Woche programmieren lernen. 2nd edn. Hanser, München (2014).

[5] LAG Informatik: micro:bit - Beispiele für den Unterricht. 2018. https://doc-player.org/140654151-Micro-bit-beispiele-fuer-den-unterricht.html. Last accessed: 20 May 2022

[6] Learning Lab (TU Graz): BBC micro:bit Werkstattbeispiele. https://learninglab.tugraz.at/in-formatischegrundbildung/bbc-microbit-werkstattbeispiele/. Last accessed 20 May 2022

[7] makecode.org: Tutorials, Games. https://makecode.microbit.org/. Last accessed 19 August 2022

[8] microbit eEducation: micro:bit - Das Schulbuch. https://microbit.eeducation.at/wiki/Haupt-seite. Last accessed 20 May 2022

[9] microbit.org: Projects. https://microbit.org/projects/. Last accessed 19 August 2022

[10] Mössenböck, H.: Sprechen Sie Java?: eine Einführung in das systematische Programmieren.3rd edn. dpunkt.lehrbuch, Heidelberg (2005).

[11] Pasterk, S., Sabitzer, B., Demarle-Meusel, H., and Bollin, A.: Informatics-Lab: Attracting Primary School Pupils for Computer Science. In proceeding of the 14th LACCEI International Multi-Conference for Engineering, Education, and Technology, San Jose, Costa Rica (2016).

[12] Sabitzer, B.: A Neurodidactical Approach to Cooperative and Crosscurricular Open Learning: COOL Informatics, Habilitation thesis, University of Klagenfurt, (2014).

[13] SchulArena.com: micro:bit. https://www.schularena.com/ict/informatik/make-it/micro-bit. Last accessed: 20 May 2022

Short Paper Session

Analysis of the educational potential of attributes of physical computing systems for the development of computational thinking

Eric Schätz[1], Alke Martens[1] and Lutz Hellmig[1]

[1]*Institute of Computer Science, University of Rostock Albert Einstein Straße 22, D-18059 Rostock, Germany*

Abstract

Physical computing is becoming more and more popular in computer science education. On the one hand, products like the BBC Micro:Bit or the Calliope Mini are designed to offer students an easy and highly motivating access to computer science education. On the other hand, aspects of physical computing systems also become relevant in computer science curricula. Next to the named candidates, there is a big variety of educational computer systems on the market with very different focuses, concepts and possibilities. They allow students to do many different activities which summed up can be called "physical computing". Physical computing compromises for example programming, technique, sensors and actors, algorithms, evaluation, bug detection and fixing, software project thinking, design thinking and so forth. Additionally, the field itself is constantly developing: new physical computing systems appear in an at least annual cycle. Since the educational potentials of those systems are partly overlapping and partly disjunct, it is hard to find a clearly arrangeable structure of those systems. In this work, we want to describe our approach of structuring this field by finding attributes which can specify physical computing systems for educational use. In the following, we want to observe, of how those attributes effect on the potential of developing computational thinking in computer science education, using physical computing systems. We are going to specify our observations by pointing out the relations between the found attributes and the different components of computational thinking.

Keywords

Physical Computing, Structure, Computational Thinking, STEM

1. Introduction

Over the last years we have observed a growing number of computer systems on the market which are designed to bring physical computing to class. The variety of those computer systems, we are going to call them *Physical Computing Systems (PhCS)* in the following, is big. They address students starting from elementary school up to 12th grade and in some cases even university students as well. The number of those systems is still growing and with new and

Informatics in Schools. A step beyond digital education. 15th International Conference on Informatics in Schools: Situation, Evolution, and Perspectives, ISSEP 2022, Vienna, Austria, September 26–28, 2022

eric.schaetz@uni-rostock.de (E. Schätz); alke.martens@uni-rostock.de (A. Martens); lutz.hellmig@uni-rostock.de (L. Hellmig)

https://pidi.informatik.uni-rostock.de/eric-schaetz/ (E. Schätz);
https://pidi.informatik.uni-rostock.de/alke-martens/ (A. Martens);
https://pidi.informatik.uni-rostock.de/lutz-hellmig/ (L. Hellmig)

0000-0002-0877-7063 (E. Schätz); 0000-0001-7116-9338 (A. Martens); 0000-0002-9421-8566 (L. Hellmig)

improved versions of already available devices (such as the BBC Micro:Bit V2 [1]), the provided possibilities increased as well which leads to a growing educational potential of physical computing.
The topic also got recognized by curriculum committees so that physical computing got integrated for example in the curriculum of computer science in 8th grade in the German federal state Mecklenburg Western Pomerania [2].
The variety of available PhCSs on the market is big, and even though they all provide physical computing for students, they address different groups of students, train different competences and support different activities. For example, if students get told to bring a robot to follow a line, students can just get started writing the software if they use an OzoBot [3], while they first need to physically construct a car, if the teacher hands out a Lego Mindstorms EV3 Educator Box [4]. The given example illustrates the impact of the choice of a PhCS on the student activities and the educational setting. The focus of a lecture is also influenced by the way a used PhCS is given to the students. To continue the example of the Lego Mindstorms EV3 Educator, the students could already get a completed car out of Lego or at least a preselection of parts, they are going to need for the given task. So the Lego Mindstorms EV3 Educator provides a range of activities which are all part of *physical computing*.
This leads to different use case scenarios of PhCSs in class which can be separated into two groups:

1. **There is *a direct need* of a use of PhCSs provided by the lectures content.**
   This can be for example topics like analog-digital-convertion or the development of sensor based application such as in the curriculum of computer science in 8th grade in the federal state of Mecklenburg Western Pommerania [2] claimed.
2. **There is *no direct need* of a use of a PhCSs by the lectures content.**
   However, in some cases it can still be advantageous to use a PhCS in class:
   a) The use of PhCSs can increase the motivation of the class by extending the diversity of methods. Interviews we did with teachers at workshops have shown, that many of the asked teachers see this as a main reason to use PhCSs in class for topics which do not directly require the use of PhCSs.
   b) We claim in this paper, that the use of PhCSs in class can also have a positive impact on the development of computational thinking and other competences beyond the assumption that it just increases the motivation of a class (which itself already would have a positive impact on the development of those competences).

In conclusion, there is a big variaty of PhCSs availsable which can be used in diverse educational settings with wide range of focuses and different aims on improvement of lectures. In order to analyze the educational potential of a concrete PhCS, there is a need for a structure.

2. Creation of a structure for PhCSs

There are already approaches of structuring the field for example by defining categories for systems [5]. Those categories focus on the end-products which come out of projects which can be done with specific PhCSs. Our aim is to create a structure for PhCS for educational use.

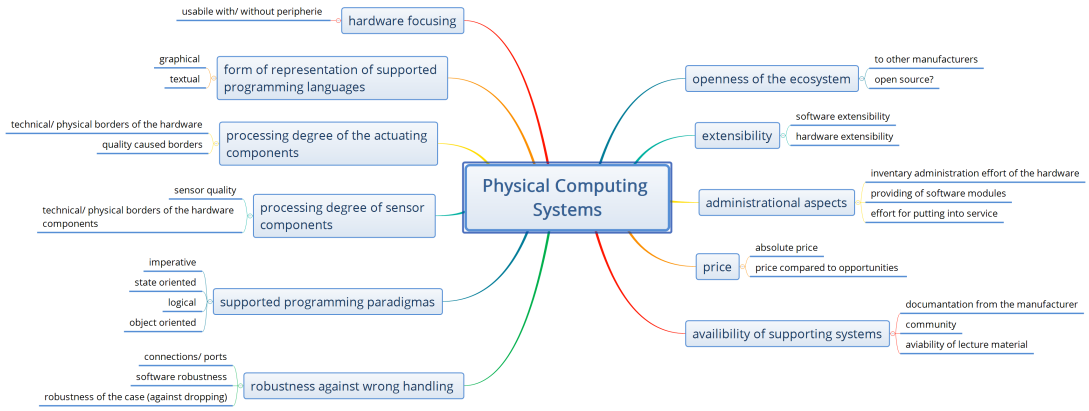


Figure 1: Attributes of physical computing systems for an educational setting [6]

Since many of these systems provide a range of projects and fundamental characteristics can variate through available extensions, it is not possible to find a set of closed classes or categories where PhCSs can be sorted in. This led us to define attributes of PhCSs, to characterize concrete devices for an educational setting. We have discussed those attributes in our workgroup and interviewed computer science teachers as well in order to create a set attributes. This set still needs to get verified by interviewing a larger number of teachers and by analyzing more concrete PhCSs and fitting them into our structure. We have put our results into a mindmap (Fig. 1), showing head- and sub-attributes [6] .

3. Connection of the attributes to computational thinking

3.1. Preselection of the attributes

As mentioned in the introduction, we claim that the use of physical computing systems in class can have a positive effect on the development of computational thinking. To reduce the connections to analyze, we are doing a preselection while connecting the attributes to the disciplines of computational thinking (Decomposition, Pattern Recognition, Abstraction, Algorithm Design) [7] [8] as shown in table 1. In the table, "+" stands for "*there is most likely an impact*", "0" for "*there might be a little impact*" and "-" for "*there is most likely no impact*".

As a result, there are attributes which can likely have a big impact on the development of computational thinking, while many attributes, which are also important for the use of PhCSs in an educational setting, do most likely not relate on the development of those thinking structures.

3.2. Illustration of the impact on a concrete example

The preselection in table 1 shows, that especially the attributes of the processing degree of actuating and sensor components can have a big impact on the development of computational

Table 1
Preselection of attributes and disciplines of computational thinking.

Head-Attribute	Decomposition	Pattern Recognition	Abstraction	Algorithm design
Hardware focusing	0	-	-	-
Form of the representation of supported programming languages	0	0	0	0
Processing degree of the actuating components	+	+	+	+
Processing degree of sensor components	+	+	+	+
Supported programming paradigms	0	0	0	0
Robustness against wrong handling	0	-	-	-
Openness of the ecosystem	0	-	0	-
Extensibility	0	-	0	-
Administrational aspects	-	-	-	-
Price	-	-	-	-
Availability of supporting systems	-	-	-	-

thinking.
By processing degree of sensor components we mean how much given sensor information is purged of false measurements. Due physical boarders, it is usual that a sensor generates false outputs once a while. This can be caused by many reasons and be easily observed by connecting a display and an ultrasonic sensor to a BBC Micro:Bit for example. Kids would notice that the distance shown on the display sometimes says 0 cm and just a second later the correct distance. This is can be a problem in many use cases, like self-driving cars which would break immediately so it get's obvious that those values need to be cleared. In computer-networks connection errors appear frequently as well, so computers use protocols to detect and repair those errors [9]. A PhCS with a higher processing degree is for example the Lego Mindstorms EV3 and one with a lower processing degree is the BBC Micro:Bit (at least related to some sensors, like the ultrasonic sensor). In conclusion this attribute provides a scale starting from raw value-sensors up to already cleared sensor values.

The processing degree of actuating components is analog to the sensor components. As a result this attribute provides a fluid scale from "*actuating part is doing likely what it is supposed to to*", via "*is it likely doing what it is supposed to do and gives a feedback if the action was successful*" until "*it is doing exactly what is supposed to do*".

At this point, it leads to the question: Why and how can this attributes have a big impact on the development of computational thinking? PhCSs with a higher processing degree of actuating and sensor components hide those processing algorithms in software modules which are typically black boxes. On the one hand, this way they allow programmers to focus on higher located problems and to create more complex software, on the other hand it is not always the

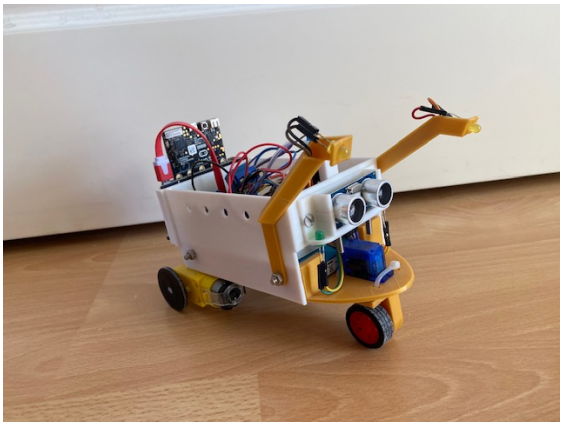


Figure 2: Simple robot car with DC Motor

goal in computer science education to develop really powerful and complex programs. PhCSs with a lower processing degree of actuating and sensor components allow a look into those blackboxes which can lead to a better understanding of the problem itself by the student because the problems are closer to the technical root and more tangible.

In the following, we are going to sketch an example how the processing degree of actuating components effects on computational thinking:
Students are getting a BBC Micro:Bit which is placed on a little robot car with DC-Motors (Fig. 2).

The task for the student is to let the car move exactly 100 cm. Even though the task sounds quite simple, students will notice really soon it is not that easy to solve with this primitive robot and that the problem consists out of many part-problems:

1. There is no possibility for the car to break other then turning of the motor off, which does not lead to an immediate stop. The car is most likely going to keep moving for at least some cm after the motor was turned off.
2. If the robot has moved more than 100 cm, there is no possibility for the robot to move backwards.
3. The Motor can just run with one speed, there is no way to adjust the speed.
4. There is no possibility for the Micro:Bit to recognize how far the car has driven yet other than stopping the time.
5. ...

As a result, this tasks leads inevitably to **decomposition**, which is the first discipline of computational thinking we are looking at. The starting problem (moving the car exactly 100 cm) turns out to be harder than expected but is still easy to understand. This makes it easier for students finding part-problems.
Analysing those part problems lead students to **pattern recognition**: If they find a way to enable the robot moving backwards, the robot can also stop this way. In conclusion part-problems 1 and 2 can be solved together:

Problems 1 and 2 can be solved by building an H-Bridge which switches the poles of the DC-Motors by using just 2 relays. Problem 3 is used in common by pulse width modulation (PWM), which can also be solved with a relay which turns the motor on and off in an high frequency. Since the clicking of the relay is hearable, other than the switching of a transistor (which is usually used in higher level systems), students directly connect the technical principle to the noticeable effect. Problem 4 can only be solved by using a sensor, for example an ultrasonic sensor which can be attached to this car. In order to do design the algorithm (as a last step), students have to do **abstraction** and decide which (physical) effects are important for their solution and which are not. By putting all part solutions together, students are doing **algorithm design**, which is the fourth part of computational thinking we have been looking at.
Related work, such as Brandhofer [10] have shown as well, how computational thinking can be developed by using the BBC Micro:Bit.

4. Discussion

In the stated example we have shown how students have to use thinking-structures of computational thinking by solving a physical computing problem. In this example students use a PhCS with a very low processing degree of actuating components. So far, we have shown how such a PhCS can have a positive impact on the development of computation thinking. However, we have not shown yet, if low level processing degree has a *better* impact than a high level degree. At this point we want to mention that a high level processing degree PhCS can also have a positive impact of the development of computational thinking. However, this example shows, that the development of computational thinking can take place on really simple problems which almost seem to be trivial. With a PhCS with higher processing degrees of actuating parts, there are far more complex problems able to be solved. However, it is always necessary to use given software and hardware modules which usually appear as black boxes. This way, students always need to ”trust” given black boxes and most likely need to read their specifications. With the use of a PhCS which is as simple as possible, problems like the sketched one get complex enough to disaggregate them into part problems. Because the generated part problems are so close to the technical root, students get clear, that those problems can not be disaggregated and do not need to be disaggregated anymore. In conclusion, the usage of low level processing degree PhCSs can simplify the process solving a given problem by training the structures of computational thinking because appearing part problems are more tangible for students, while the single disciplines of computational thinking do not get simplified so that the training-effect does not suffer by the simplification.
If a PhCS provides actuating components with a wide range of processing degree (like the BBC Micro:Bit does by supporting simple DC-Motors as well as servo-motors and step-motors), teachers still have the opportunity to increase the hardware level and generate new possibilies for class projects this way.

5. Conclusion

In this paper we sketched our approach of a possible structure of PhCSs and showed how the educational potential of a concrete PhCS can be discussed by analyzing specific attributes. We did that on the example of the processing degree of actuating components. Our work is still in progress, as we still need to interview a bigger group of teachers in order to verify our set of attributes. We do not claim our preselection of attributes (Table 1) to be final yet and are likely going to update it in the future. We already did pretests with a group of 20 student from grades 7 to 10 on the sketched example of developing computational thinking by using low level PhCSs. This fall, we are going to test this and other examples on another group of students.

References

[1] Microbit Foundation, The new BBC micro:bit with speaker and microphone, 2020. URL: https://microbit.org/news/2020-10-13/meet-the-new-bbc-microbit/.

[2] Ministerium für Bildung, Wissenschaft und Kultur Mecklenburg Vorpommern, Rahmenplan Informatik und Medienbildung, 2019. URL: https://www.bildung-mv.de/schueler/schule-und-unterricht/faecher-und-rahmenplaene/rahmenplaene-an-allgemeinbildenden-schulen/informatik/.

[3] Ozo Edu Inc., Ozobot | Robots to code and create with, 2022. URL: https://ozobot.com/.

[4] LEGO Group, MINDSTORMS EV3 | MINT ab der neunten Klasse, 2022. URL: https://education.lego.com/de-de/products/lego-mindstorms-education-ev3-set/5003400#ev3-set.

[5] M. Przybylla, A. Grillenberger, Fundamentals of Physical Computing: Determining Key Concepts in Embedded Systems and Hardware/Software Co-Design, in: The 16th Workshop in Primary and Secondary Computing Education, ACM, Virtual Event Germany, 2021, pp. 1–10. URL: https://dl.acm.org/doi/10.1145/3481312.3481352. doi:10.1145/3481312.3481352.

[6] E. Schätz, A. Martens, Attributes of physical computing systems – the perspective of pedagogy, in: Edulearn22 Proceedings, Palma, Spain, 2022, pp. 5561–5565. URL: https://library.iated.org/view/SCHATZ2022ATT. doi:10.21125/edulearn.2022.1309.

[7] N. Ruh, D. Koch, G. Philips, M. Stangl, Computational Thinking, 2022. URL: https://oinf.ch/konzept/computational-thinking/.

[8] P. Curzon, P. W. McOwan, Computational Thinking: Die Welt des algorithmischen Denkens – in Spielen, Zaubertricks und Rätseln, Springer Berlin Heidelberg, Berlin, Heidelberg, 2018. URL: http://link.springer.com/10.1007/978-3-662-56774-6. doi:10.1007/978-3-662-56774-6.

[9] H. Sack, C. Meinel, Digitale Kommunikation, X.media.press, Springer Berlin Heidelberg, Berlin, Heidelberg, 2009. URL: http://link.springer.com/10.1007/978-3-540-92923-9. doi:10.1007/978-3-540-92923-9.

[10] G. Brandhofer, O. Kastner-Hauler, Der micro:bit und Computational Thinking, Lehr-Lernprozesse gestalten, analysieren und evaluieren (2020) (2020) 20. URL: https://journal.ph-noe.ac.at/index.php/resource/article/view/914.

Poster Session

Integration of a Informatics Teaching-Learning Laboratory into Pre-Service Informatics Teacher Education

Frauke Ritter, Nadine Schlomske-Bodenstein and Bernhard Standl

Karlsruhe University of Education, Bismarckstr. 10, 76133 Karlsruhe, Germany

Abstract

In order to understand the digital world and the underlying concepts of digital devices and applications, one of the skills required is algorithmic thinking. Therefore, it is important that not only the students learn algorithmic thinking in school, but also future informatics teachers. To enable them to experience this as effectively as possible, factors such as TPACK competencies and self-efficacy are important. This poster presents a study design on how a teaching-learning lab (TLL) can be integrated into pre-service informatics teacher education. The goal is, on the one hand, to increase the future teachers TPACK competencies and self-efficacy, and, on the other hand, to develop two different didactic approaches that foster pupils' algorithmic thinking. Considering this, a Robotic workshop with a block-based programming language was developed integrating these didactic approaches, both of which are designed to promote pupils' algorithmic thinking. The students conducted these two workshops in our TLL and reflected on their didactic activities. Using a mixed-methods approach, we analyzed how students' TPACK skills and self-efficacy develop over time and how two different didactic approaches differ in terms of promoting algorithmic thinking in pupils. In the next step further data will be collected, evaluated and analyzed to achieve more insights to this preliminary results.

Keywords

Algorithmic Thinking, Computational Thinking, Block-based programming

1. Introduction

It is undoubtedly important that today's students learn to use digital tools and devices in a self-directed way. Moreover, in our digital-driven world, knowledge of computer science and digital orientation is required to understand its complexity. Therefore, competencies in computational thinking (CT) are required. Among various definitions, CT is also the ability to design algorithms and interact with the digitized world based on computer science skills [1, 2, 3, 4]. In [5] algorithmic thinking is defined as the ability to create and understand algorithms. In practice this means to be able to analyze and specify a problem, to know the rules of solving a problem as well as to design an algorithm and further improve it to be more

Informatics in Schools. A step beyond digital education. 15th International Conference on Informatics in Schools: Situation, Evolution, and Perspectives, ISSEP 2022, Vienna, Austria, September 26–28, 2022

frauke.ritter@ph-karlsruhe.de (F. Ritter); nadine.schlomske-bodenstein@ph-karlsruhe.de (N. Schlomske-Bodenstein); bernhard.standl@ph-karlsruhe.de (B. Standl)

0000-0001-5744-9456 (F. Ritter); 0000-0003-0835-165X (N. Schlomske-Bodenstein); 0000-0002-8849-2980 (B. Standl)

efficient [5]. The instructional learning environment needs to be aligned with the learners' needs. They need to be supported in such a way through the learning environment that they can acquire these computational skills. At the same time, pre-service teachers need training in self-reflection, self-efficacy, and TPACK competencies [6] to teach algorithmic thinking to future pupils. This poster presents the study design in which, on the one hand, the development of TPACK competencies (Technological Pedagogical Ccontent Knowledge) [6] and the self-efficacy of students attending a TLL-seminar are investigated. Students conduct various workshops at school classes in the TLL during the semester and repeatedly reflect on their didactic activities. On the other hand, two different didactic approaches (see Figure 1) of a Robotic workshop with a block-based programming language, both of which are intended to promote algorithmic thinking of the pupils, are evaluated within the study. The study follows the mixed-method design.

2. Study Design

2.1. Integration of the Teaching Learning Laboratory in a seminar

N = 4 students of informatics attended a didactics seminar in which they gave Robotic workshops with the teaching-learning laboratory (TLL) visiting school classes in which they were videotaped. During the semester, the students reflected on their teaching skills based on these video recordings. At the end of the semester, they were surveyed regarding their development of TPACK competencies [6] as well as self-efficacy using Post-then-Pre-questionnaires.

2.2. Robotic Workshop Design

We developed one workshop with two didactical structures that are intended to promote pupils algorithmic thinking, see Figure 1:

A: SWAT concept [7]: a guided problem-oriented approach, following a semantic wave [8]
B: CPB concept: a constructionist problem-based approach [9, 10]

In both didactical structures pupils work on problems using Robots and a block-based programming language.

Figure 1 shows the almost identical but structured differently in terms of methodology topics of the workshops: The didactical structure A, which is based on our developed guided problem-based SWAT concept [7], methodically follows a semantic wave in each module and is described here in more detail [11]. The order of the topics is also shown in Figure 1 and the problems are guided in the sense that they are broken down into sub-problems which the pupils then have to solve independently. The second structure is an constructionist problem-based concept [9, 10] in which the first module provides the pupils with all the relevant knowledge information (knowledge base) they need to be able to solve modules 2-4 independently and can chose the order of the modules freely. The workshop is conducted with pupils who are in 8th or 9th grade of a secondary school.

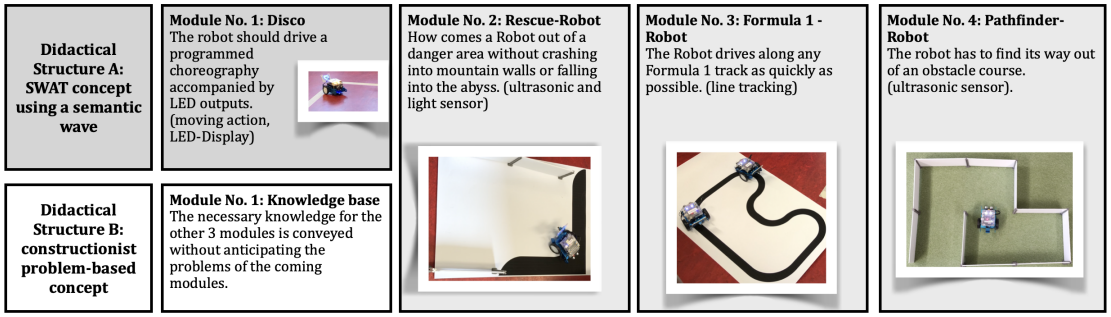


Figure 1: Robotic workshop with two didactical structures in comparison: SWAT concept [7] using a semantic wave [8] and Constructionist problem-based concept [9, 10].

2.3. Method and Instruments

In order to assess students' TPACK competences and self-efficacy on the one hand and, on the other hand pupils' competence in algorithmic thinking, we investigate the following research questions (RQ):

RQ1: How can pre-service teacher's TPACK competencies and self-efficacy be developed?

RQ2: To what extent can pupils' algorithmic thinking be developed through different workshop settings?

The research questions are addressed in a mixed-method approach and will be examined with the following instruments (IRQ): *IRQ1:* Questionnaire for the TPACK-competencies and the self-efficacy in a retrospective Post-Then-Pre-questionnaire design and a qualitative descriptive analysis. *IRQ2:* Pupils' worksheets and created programs evaluating with the help of a reductive, qualitative content analysis according to the interpretative paradigm of [12] using a developed category system (qualitative analysis); valid Algorithmic Thinking Test [13] (pre-posttest with control group) analyzing by using the approximate adjusted fractional Bayes factors for testing informative hypotheses [14] (quantitative analysis).

3. Summary

In this poster, we present our integration of the Informatics TLL into the education of pre-service teachers. It is a novel concept to integrate a TLL into teaching, where on the one hand pupils come to learn the core competence algorithmic thinking and, on the other hand, students can gain didactical hands-on experience already during the studies and not only in the traineeships. Feedback analyses allow for extensive student self-reflection during the semester. In addition, the development of TPACK competencies related to informatics teaching as well as students' self-efficacy are evaluated and assessed through self-assessment. Finally, the two didactical structures are also investigated in terms of teaching algorithmic thinking to pupils. In a next step, more data will now be collected, as N = 4 students is a small number of students so far, and then be evaluated.

Acknowledgments

This work was supported by the Vector Foundation.

References

[1] V. J. Shute, C. Sun, J. Asbell-Clarke, Demystifying computational thinking, Educational Research Review 22 (2017) 142–158.
[2] J. M. Wing, Computational thinking, Communications of the ACM 49 (2006) 33–35.
[3] P. J. Denning, M. Tedre, Computational Thinking: A Disciplinary Perspective, Informatics in Education 20 (2021) 361–390. doi:10.15388/infedu.2021.21.
[4] J. M. Wing, Computational thinking and thinking about computing, Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences 366 (2008) 3717–3725. doi:10.1098/rsta.2008.0118.
[5] G. Futschek, Algorithmic Thinking: The Key for Understanding Computer Science, Lecture Notes in Computer Science (2006) 159–168.
[6] M. J. Koehler, P. Mishra, W. Cain, What is Technological Pedagogical Content Knowledge (TPACK)?, Journal of Education 193 (2013) 13–19. doi:10.1177/002205741319300303.
[7] F. RITTER, B. STANDL, Promoting Student Competencies in Informatics Education by Combining Semantic Waves and Algorithmic Thinkingv, Informatics in Education 00 (2022). doi:10.15388/infedu.2023.07.
[8] K. Maton, Making semantic waves: A key to cumulative knowledge-building, Linguistics and Education 24 (2013) 8–22. URL: http://dx.doi.org/10.1016/j.linged.2012.11.005. doi:10.1016/j.linged.2012.11.005.
[9] A. Csizmadia, B. Standl, J. Waite, Integrating the constructionist learning theory with computational thinking classroom activities, Informatics in Education 18 (2019) 41–67.
[10] B. Standl, Conceptual modeling and innovative implementation of person-centered computer science education at secondary school level, University of Vienna, Vienna, Austria (2013).
[11] F. Ritter, B. Standl, Promoting Computational Thinking in Teacher Education - Combining Semantic Waves and Algorithmic Thinking, in: Proceedings of the 2022 ACM Conference on International Computing Education Research - Volume 2, ICER '22, Association for Computing Machinery, New York, NY, USA, 2022, pp. 48–49. URL: https://doi.org/10.1145/3501709.3544285. doi:10.1145/3501709.3544285.
[12] P. Mayring, T. Fenzl, Handbuch Methoden der empirischen Sozialforschung, Springer Fachmedien Wiesbaden, 2019. URL: http://dx.doi.org/10.1007/978-3-658-21308-4_42. doi:10.1007/978-3-658-21308-4.
[13] M. Román-González, Computational Thinking Test: Design Guidelines and Content Validation, Proceedings of the 7th Annual International Conference on Education and New Learning Technologies (EDULEARN 2015) (2015) 2436–2444. doi:10.13140/RG.2.1.4203.4329.
[14] H. Hoijtink, J. Mulder, C. van Lissa, X. Gu, A Tutorial on Testing Hypotheses Using the Bayes Factor, Psychological Methods (2019). doi:10.1037/met0000201.

Exploring Pre-service Informatics Teachers' Ability to Assess Their Own Performance in Supporting Learners Through Algorithmic Problem Solving Processes

Nadine Schlomske-Bodenstein, Bernhard Standl

Karlsruhe University of Education, Bismarckstr. 10, 76133 Karlsruhe, Germany

Abstract

This pilot study explores N = 4 pre-service informatics teachers who were video-recorded when guiding a group of learners in solving algorithmic problem-solving tasks. Before and after the process, they were asked to assess their ability to guide the learners in a real, accepting and emphatic way when addressing and formulating questions or giving feedback. The group of learners also evaluated the competencies of their learning guides.

Keywords

Video-Vignettes, Empathy, Algorithmic Problem-Solving

1. Introduction

The skill to solve problems of algorithmic tasks depends on the amount and quality of knowledge in a domain [1] - in this case in informatics/algorithmic thinking. Educating future teachers in informatics addresses the acquisition of algorithmic skills in a problem-based context. The teaching methods need to be aligned with intended learning outcomes [2]. In accordance with the features of instructional quality, classroom management, structuring of the learning context as well as cognitive activation and student-centeredness impacts learning [3]. In accordance with the student-centered approach, students acquire their intended learning outcomes in a supportive atmosphere which is influenced by the teacher's ability to be authenticity, unconditional positive regard and empathic understanding [4].

1.1. Video Vignettes

Pre-service informatics teachers' professional vision - the ability to notice and interpret various classroom settings via video - has been well examined in particular in the educational context [5, 6, 7, 8]; in particular the following features of instructional quality as *1) making the learning*

Informatics in Schools. A step beyond digital education. 15th International Conference on Informatics in Schools: Situation, Evolution, and Perspectives, ISSEP 2022, Vienna, Austria, September 26–28, 2022

nadine-schlomske-bodenstein@ph-karlsruhe.de (N. Schlomske-Bodenstein); bernhard.standl@ph-karlsruhe.de (B. Standl)

0000-0003-0835-165X (N. Schlomske-Bodenstein); 0000-0002-8849-2980 (B. Standl)

goals transparent to the learners, 2) activating the learners, 3) supporting and guiding the learning-processes as well as 4) evaluating it [8]. Previous studies investigated pre-service informatics teachers' perception of empathy in feedback settings where they support learners in their learning processes [9]. Vignettes enable a standardized assessment using videos in different classroom settings [10]. It is applied in education research with becoming teacher [11, 12].

The goal of this explorative pilot-study is to further develop a video-vignette instrument as a professional reflection tool for informatics students to reflect and further develop their ability to guide learning processes in the context of algorithmic thinking in a student-centered way. The following research question is driving this study: *To what extent can the pre-service informatics teachers assess their performance to guide learners in a student-centered way?*

2. Method

The sample consists of N = 4 (1 female and 3 male at the average age of 25) pre-service computer science teachers. They participate in a didactics seminar in informatics. During the semester they run workshops with pupils and are videotaped. During the semester, they reflect their teaching competencies based on the video recordings from them. For this experiment they were trained to guide learners in student-centered teaching (authenticity, unconditional positive regard and empathy) way in accordance with [4]. The pre-service teachers each chose an algorithmic task they wanted to use in a student-centered way. The task for the learning group included problem-oriented tasks where they had to work together to find an algorithm to solve a problem. The tasks were based on algorithms in [13]. Fig. 1 shows an example of an algorithmic task where students were asked to develop an algorithm to search for an object in a sorted CD collection and describe it in natural language.

Figure 1: Example of an task on searching a CD in a sorted collection: "Describe an algorithm in natural language to find the selected CD" [13]

Before each setting of guiding the group of learners in their algorithmic problem-solving tasks, they were asked to predict their degree of student-centeredness before guiding the group of learners and estimating their student-centeredness afterwards on on a 5-point Likert scale (1 = not at all true, 2 = rather not true, 3 = neither false nor true, 4 = rather true, 5 = totally true). We used an instrument, we developed and piloted before with N = 20 pre-service teachers in informatics [9].

2.1. Reliability

The instrument showed a Cronbach's Alpha of 0.94 [9]. We further shortened it, deleting 6 items which had an internal consistency of less than 0.62 and reformulating 2 items. Another item was deleted due to similarity. In this study, the assessment instrument consisted of 18 items. The scales realness and acceptance consisted of 2 items each and the empathy scale consisted of 14 items. Empathy included the subscales response behaviour, accuracy of assessment and atmosphere. The scales are based on instruments which were developed in the therapeutic context [14, 15, 16], in educational context [17] as well as in computer science education [18, 19].

3. Preliminary Results and Discussion

This section shows the preliminary findings of our empirical study that aimed: (1) to further develop a vignette-instrument on teacher's professionalization and (2) to assess pre-service teachers' ability and perception of student-centeredness when put into concrete settings where they guide learning processes in algorithmic thinking. First insights are shown in Table 1. The pre-service informatics teachers underestimated their ability to guide learners in a real, accepting and emphatic way. Further results will be discussed in the poster presentation.

Table 1
Description of the self-estimation regarding student-centeredness before and after

Algorithmic Task	**Realness**		**Acceptance**		**Empathy**	
	AM	SD	AM	SD	AM	SD
self-estimation before guiding a group of learners	4.13	0.48	4.13	0.75	3.63	0.62
self-estimation after guiduing a group of learners	4.25	0.65	4.38	0.48	3.68	0.24

Therefore, this explorative case study gives first insights into pre-service teachers in informatics have difficulties in accurately predicting their expected performance when guiding learners in problem-solving tasks of algorithmic thinking. In a next step, further examples will be videotaped and assessed and N = 30 informatic students will be asked to rate the different settings with regard to student-centeredness as well.

4. Conclusion

Professionalizing future informatics teachers in guiding learners through the learning processes of solving algorithmic task in a problem-based context is a central issue in teacher education in informatics. This vignette-instrument investigates the reflection as well as professionalization. However, this study is limited in various points which must be addressed in future work. The sample includes only a very limited amount of pre-service teachers in informatics. In future work, more learners will be included who will be videotaped when guiding learners in problem -based algorithmic thinking tasks. Future work needs to include more videos as well as more students to rate the vignettes and thus use this instrument for reflection.

References

[1] N. M. Seel, Psychologie des Lernens: Lehrbuch für Pädagogen und Psychologen; mit 17 Tabellen und zahlreichen Übungsaufgaben, Reinhardt, 2000.

[2] J. Biggs, C. Tang, Teaching for quality learning at university, McGraw-hill education (UK), 2011.

[3] M. Kleinknecht, Unterrichtsqualität, in: E. Kiel, K. Zierer (Eds.), Unterrichtsgestaltung als Gegenstand der Wissenschaft, volume 2, Baltmannsweiler, 2011, pp. 65–76.

[4] C. R. Rogers, Freedom to learn for the 80's (1983).

[5] K. Stürmer, T. Seidel, S. Schäfer, Changes in professional vision in the context of practice, Gruppendynamik und Organisationsberatung 44 (2013) 339–355.

[6] C. Goodwin, Professional vision, in: Aufmerksamkeit, Springer, 2015, pp. 387–425.

[7] M. G. Sherin, S. Y. Han, Teacher learning in the context of a video club, Teaching and Teacher education 20 (2004) 163–183.

[8] K. Stürmer, T. Seidel, A standardized approach for measuring teachers' professional vision: The observer research tool, in: Teacher noticing: Bridging and broadening perspectives, contexts, and frameworks, Springer, 2017, pp. 359–380.

[9] B. Standl, N. Schlomske-Bodenstein, Development of a vignette-based instrument to assess students' perception of the teacher's empathy during collaborative algorithmic problem-solving tasks, in: 2022 IEEE Global Engineering Education Conference (EDUCON), IEEE, 2022, pp. 1154–1160.

[10] I. Knievel, A. M. Lindmeier, A. Heinze, Beyond knowledge: Measuring primary teachers' subject-specific competences in and for teaching mathematics with items based on video vignettes, International Journal of Science and Mathematics Education 13 (2015) 309–329.

[11] S. Schäfer, T. Seidel, Noticing and reasoning of teaching and learning components by pre-service teachers, Journal for educational research online 7 (2015) 34–58.

[12] T. Seidel, K. Stürmer, Modeling and measuring the structure of professional vision in preservice teachers, American educational research journal 51 (2014) 739–771.

[13] B. Vöcking, H. Alt, M. Dietzfelbinger, R. Reischuk, C. Scheideler, H. Vollmer, D. Wagner, Taschenbuch der Algorithmen, Springer, 2008.

[14] G. J. Pine, A. V. Boy, Learner centered teaching: A humanistic view, Love, 1977.

[15] G. T. Barrett-Lennard, Dimensions of therapist response as causal factors in therapeutic change., Psychological monographs: General and applied 76 (1962) 1.

[16] S. E. Decker, C. Nich, K. M. Carroll, S. Martino, Development of the therapist empathy scale, Behavioural and cognitive psychotherapy 42 (2014) 339–354.

[17] R. Colwell, Kids don't learn from people they don't like, Journal of Research in Music Education 27 (1979) 46–48.

[18] B. Standl, Conceptual modeling and innovative implementation of person-centered computer science education at secondary school level, University of Vienna, Vienna, Austria (2013).

[19] R. Motschnig-Pitrik, A. Holzinger, Student-centered teaching meets new media: Concept and case study, Journal of Educational Technology & Society 5 (2002) 160–172.

More than a substitute? Digital exams with Tablets in Computer Science

Erbil Yilmaz, Vincenzo Fragapane and Bernhard Standl

Karlsruhe University of Education, Bismarckstr. 10, 76133 Karlsruhe, Germany

Abstract

As a result of the Covid-19 pandemic, not only did digital teaching get a massive boost, but also the delivery of digital exams was the subject of new developments. Although the design of digital exams offers the potential for new breakthrough opportunities, exam regulations and the need for short-term adjustments often stand in the way. Consequently, methods are being considered to transfer existing exam formats into the digital format. In computer science studies, paper-and-pencil exams are used because they are very suitable for many areas such as theoretical computer science. They often consist of a mixture of a selection of answer options, making sketches for automata and free text answers. The question, however, is how to translate this format to digital while still meeting the framework and still allowing students the same requirements as in the regular not digital approach. In this poster we will first discuss the requirements for an iPad exam in theoretical computer science and what possibilities there are to transfer this exam to the digital level. Second, we will describe an empirical approach, how to investigate the effects of the digital transformation on the examination process itself.

Keywords

Digital Exams, Tablets, Pre-Service Computer Science Teacher

1. Introduction

The Covid-19 pandemic has led to an increased implementation of digital exams at universities. This also applies to computer science. The focus here was on finding a replacement for paper-and-pencil exams as quickly as possible. Accordingly, digital exams were aligned with their analog counterparts. However, digital exams are accompanied by new possibilities as well as effects of testing, which will be addressed in this poster in the context of pre-service computer science teacher education and is driven by two questions of interest: *1. What possibilities does the use of iPads provide in exams on theoretical computer science in pre-service informatics teacher education? 2. How do these possibilities influence the examination process itself?* The aim of this poster is to show what framework conditions must be in place and what possibilities there are in terms of implementation. Furthermore we want to shed some light on a future study, that focusses on the effects of tablet exams from the student's perspective as well as on the exam results. Therefore the poster does not describe empirical results but intends to initiate a

Informatics in Schools. A step beyond digital education. 15th International Conference on Informatics in Schools: Situation, Evolution, and Perspectives, ISSEP 2022, Vienna, Austria, September 26–28, 2022

erbil.yilmaz@ph-karlsruhe.de (E. Yilmaz); fragapane@ph-karlsruhe.de (V. Fragapane); bernhard.standl@ph-karlsruhe.de (B. Standl)

0000-0002-8849-2980 (B. Standl)

discussion and provide possibilities.

2. Exams in Theoretical Computer Science

In this section we describe tasks, implemented in a paper-and-pencil exam in informatics as we used it in the subject of theoretical computer science. The given tasks are classical exam questions from the topics ”Finite Automata” and ”Formal Languages”. They are taken from an exam on theoretical computer science, which is written as part of the pre-service teacher training (Undergraduate level) at the Karlsruhe University of Education. The competences required for the solution refer to central elements of theoretical computer science: automaton construction and transformation, transformation of transition tables into transition diagrams, connections between regular grammars and finite automata, and regular expressions as an alternative form of representation to languages accepted by finite automata. In the examples shown below in Fig. 1, subtask A1(a) requires a transition diagram as the solution. In subtask A2, on the other hand, a regular expression is sought. Both tasks make special demands on the student regarding the input of the solutions. In the first task the solution is sketched as a graph, in the second task it is given as a mathematical expression.

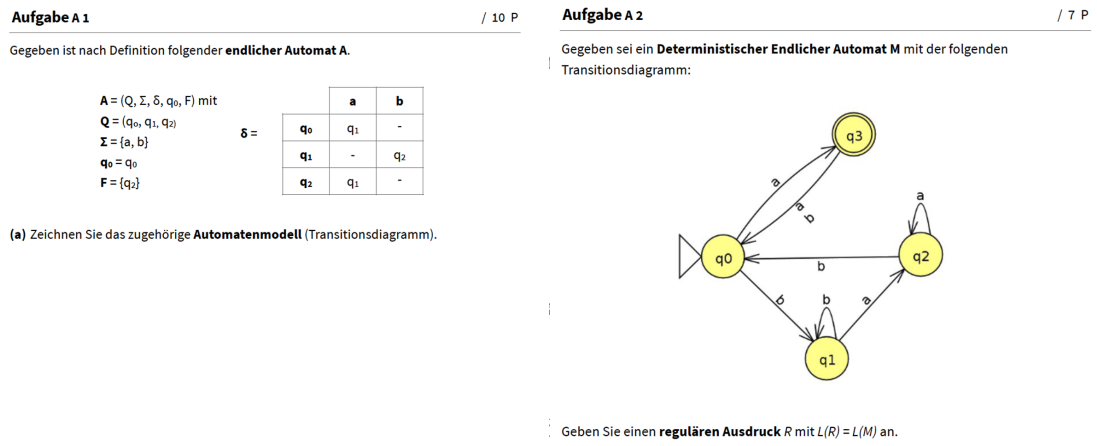


Figure 1: Example tasks from an exam on theoretical computer science.

3. Implementation of Digital Exams with Tablets

In this section, we briefly present the prerequisites for using tablets from the perspective of hardware and software.

3.1. Hardware Requirements

Although there is a wide range of manufacturers for tablets, we focus here on the iPad, which is used at our university. The use of iPads for exams is not new and has been addressed in the

literature such as in [1, 2]. In addition, numerous references to implementation possibilities can be found on the websites of universities[1]. In order to use the iPad for exams, certain requirements can be considered in advance: (1) The use of a protective film on the tablet to create the same writing feeling as on paper [3]. (2) The use of an additional keyboard (not the screen keyboard) to support input [2].

3.2. Software Requirements

With regard to software requirements the following recommendations for the use of iPads in exams can be made: (1) In our case Moodle can be used. The Safe Exam Browser of Moodle [1] and the question type of Moodle ”freehand drawing” allow a mixed input via fields or selection options and additionally the possibility to make sketches in the Moodle exam environment. (2) For easier administration, a central management of the iPads via MDM/Jamf can be implemented.

4. Options for an iPad Exam in Theoretical Computer Science

Considering this, the following conclusions can be drawn for an iPad exam in theoretical computer science: (1) The combination of sketches and text input is an important requirement. For example, a sketch of a finite automate can be made first and then the regular expression can be written down. (2) Dynamic elements can be included in exams, e.g., finite automata animated. (3) Applying learning analytics and capturing the solution process in iPad exams can support a better unterstanding of the examination process itself. (4) It could also be supported by audio input: students describe ”thinking aloud” their solution path in formative assessments, but also describe open questions and problems.

5. Further Steps

Paper-and-pencil exams and their digital counterparts have been compared in several studies in the past decades [4]. Although the conclusions are not consistent, the results indicate at better student performances with digital formats [4, 5, 6]. Moreover, according to psychological learning research, the way information is presented, for example as a text, picture or animation, has an influence on the cognitive representation [7]. Therefore we want to investigate in a future study, how the way the tasks are illustrated – e.g. in an interactive way on the iPad – influences the exam results and the understanding of the students. This intention comprises two perspectives: First, a quantitative analysis of exams with iPads, including learning analytics. Second, a qualitative access to the students’ experiences in computer science exams.

6. Conclusions

Based on the interest of this poster ”What possibilities does the use of iPads provide in exams on theoretical computer science?”, the following conclusions can be drawn.

[1] https://www.uni-rostock.de/universitaet/kommunikation-und-aktuelles/medieninformationen/detailansicht/n/mit-dem-fingertippen-durch-die-klausur

- With a suitable combination of iPad, keyboard, pen and protective film, an appropriate basis for the implementation of digital exams can be created.
- However, this also involves additional effort and expense that should not be underestimated.
- Nevertheless, if the integration of iPads into the theoretical computer science exams can be seen as more than just a substitution for paper-and-pencil exams, the additional effort may be worthwhile as it also provides new opportunities.
- If this technology is used to enrich a written exam with additional possibilities, iPads could enable new perspectives.
- In future work, we will compare the differences between traditional paper-and-pencil exams with the use of iPads in theoretical computer science and how these differences are interpreted in the students' perspective.

7. Acknowledgements

This research was funded by Stiftung Innovation in der Hochschullehre grant number FBM2020-EA-2820-03680.

References

[1] D. Sitzmann, K. Kruse, D. Gallaun, N. Kubick, B. Reinhold, M. Schnabel, L. Thoms, H. Barbas, S. Meiling, Aufbau eines mobilen testcenters für die hamburger hochschulen im rahmen des projekts mintfit e-assessment, Die Hochschullehre 8 (2022) 113–129.

[2] M. J. Cheesman, P. Chunduri, M.-L. Manchadi, K. Colthorpe, B. Matthews, Student interaction with a computer tablet exam application replicating the traditional paper exam, Mobile Computing 4 (2015) 10–21.

[3] R. Goshima, F. Harada, H. Shimakawa, Classifying solving behavior by handwriting on tablets, in: 2021 13th International Conference on Computer and Automation Engineering (ICCAE), IEEE, 2021, pp. 54–58.

[4] S. Nikou, A. A. Exonomides, Student achievement in paper, computer/web and mobile based assessment, CEUR Workshop Proceedings 1036 (2013) 107–114.

[5] A. Nardi, M. Ranieri, Comparing paper-based and electronic multiple-choice examinations with personal devices: Impact on students' performance, self-efficacy and satisfaction, British Journal of Educational Technology 50 (2019) 1495–1506.

[6] N. Kalogeropoulos, I. Tzigounakis, E. A. Pavlatou, A. G. Boudouvis, Computer-based assessment of student performance in programing courses, Computer Applications in Engineering Education 21 (2013) 671–683.

[7] W. Schnotz, Commentary: Towards an integrated view of learning from text and visual displays, Educational psychology review 14 (2002) 101–120.

Teachers' Views on Preliminary Design Principles for Computational Thinking Integration into Non-Computing Subjects

Jacqueline Nijenhuis-Voogt[1], Sabiha Yeni[2] and Mara Saeli[1]

[1]*Radboud University, Nijmegen, The Netherlands*
[2]*Leiden University, Leiden, The Netherlands*

Abstract

There is international effort towards integrating Computational Thinking in non-computing subjects, motivated by the learning outcomes widely prescribed in the literature. However, *how* should teachers embed computational thinking practices and concepts in their subjects? Based on a review of the current state of affairs, we compiled a list of preliminary design principles for the integration of CT in subject content. In this poster we present the first results of a study in which teachers' views on those preliminary design principles are examined. The participants were three K–12 non-computing teachers who developed a lesson in which they integrated CT into the subject content of their lesson. After teaching their lesson, they were individually interviewed on their views on our preliminary design principles in light of their recent experience. Our preliminary findings reveal that certain design principles are viewed as important by all teachers.

Keywords

Design Principles, Computational Thinking, Non-computing Subjects, CT in context

1. Introduction

The last decade has seen a shift of focus towards Computational Thinking (CT) and its integration across international curricula. Scientific literature as well as documents to support its practice in the educational settings have proliferated at fast pace. Particular attention is devoted towards the integration of CT in subjects other than Computer Science (CS), also defined as CT in context. CT is regarded as "procedural thinking" [1] for problem-solving, sourcing its approaches from computer science, and where solutions can be carried out by any information processing agent, whether human, computer, or a combination of both [2]. CT combines the attitudes and the skill-sets essential not only for computer scientists, but for any 21st century citizens.

When Papert introduced the concept of CT [1] he believed that by only learning to program, children were bound to develop computational problem-solving skills that they could directly apply within other contexts. However, there is no empirical evidence on this transdisciplinary transfer, also defined as high-road transfer. Voogt and colleagues [3] suggest that to facilitate transfer attention is needed to ways of incorporating CT skills within existing school subjects,

Informatics in Schools. A step beyond digital education. 15th International Conference on Informatics in Schools: Situation, Evolution, and Perspectives, ISSEP 2022, Vienna, Austria, September 26-28, 2022
jacqueline.nijenhuis@ru.nl (J. Nijenhuis-Voogt); s.yeni@liacs.leidenuniv.nl (S. Yeni); mara.saeli@ru.nl (M. Saeli)

so that students can develop creative ways to apply CT within the respective disciplines. Integrating CT in non-computing subjects provides further potential benefits to the pedagogies (and pedagogical tools) of both CS and non-CS subjects [4]. For example, non-CS subjects can deploy concepts (e.g. sequences, loops, conditionals, operators), practices (e.g. abstraction, decomposition, algorithmic thinking, evaluation and generalization) as pedagogical tools to strengthen and widen their repertoires [5].

However, integrating CT in non-computing subjects means that non-CS teachers need to be guided on "how" CT can be embedded, assessed and taught in their subject. In this poster we present the first results of a study into teachers' views on preliminary design principles for computational thinking integration into non-computing subjects. The guiding research question is: What are the teachers' ideas about the preliminary design principles for Computational Thinking integration in non-computing subjects?

2. Method

For this study, we collaborated with three teachers in K–12 education. The teachers, with support of the researchers, developed a lesson in which they integrated CT into their subject content. After teaching the lesson, each teacher has been individually interviewed and asked about the design principles they used during the development of their lesson. Afterwards we presented them with a list of preliminary design principles and asked for their feedback and views. This list has been compiled based on prior studies regarding the embedding of CT in non-computing school subjects [6, 7, 8, 9, 10].

An overview of the participating teachers is given in Table 1. A qualitative data analysis approach was used to analyze the transcripts of the interviews.

Table 1
Participating teachers.

Teacher's name (pseudonym)	Class / Students	Lesson topic
Robin	9th grade Most students: 14–15 years old	Biology lesson regarding determination of cell types. [4] Students were asked to construct a decision tree for this determination in as few steps as possible.
Simon	6th grade Most students: 11–12 years old	Primary education. Develop an app for the determination of flora and fauna. Select a few plants/animals and construct a decision tree for the determination of your plant/animal. Make a design for the user interface of your app.
Tanya	4th, 5th, and 6th grade Students aged 9–12 years old	Primary education, project-based lessons outside regular curriculum. The lesson was part of a lesson series about starting your own business. In this specific lesson, students were asked to determine the price of their own product/business.

3. First Findings

Preliminary findings suggest that knowledge regarding design principles might be part of teachers' tacit knowledge: teachers know how to design a lesson but it seemed hard for teachers to describe which design principles they used during the development of their lesson. Only one of the three teachers (Robin) formulated some design principles, before being presented with our list. Teacher Robin described how he linked a cognitive and a social goal to his lesson, and emphasized student collaboration. Robin commented: *"The social goal is linked to collaboration . . . and also to teaching students to persevere"* (quotes have been translated from Dutch into English).

After our initial open question about design principles, the teachers were presented with our preliminary design principles list and they gave their views and feedback. Our findings reveal that there is agreement among the teachers regarding certain design principles. All three teachers think it is important to **Support low threshold high ceiling learning activities [10]**; because *"it has to be clear to all students what we are talking about . . . no thresholds, they should all be able to participate"* (Robin). Another teacher added that this also concerns the teacher's strategy: *"how do I approach a group, do I lead them step-by-step or do I let them figure it out by themselves?"* (Tanya). Teachers also agree on the importance of **Employ different active learning strategies [6]**; and **Provide opportunities for students to collaborate**.

The teachers differ in their opinion on **Incorporate CT terminology within the subject content and promote students' express problems in CT terminology**. A teacher stated: *"the integration [of CT into his biology lesson] is very important, and I think you accomplish more this way than by just teaching them the flat terms . . . these terms are difficult to grasp for students"* (Robin). Another teacher, however, described using the CT terms: *"recently, in a lesson I used these terms [variables] deliberately, because I know that in Scratch, we also use the word 'variable' and I hope they will then [when working on a Scratch project] recognize it"* (Tanya).

Teachers also had different opinions on **Form the content of lesson based on clear, measurable subject related and CT related learning objectives**. One of the teachers agreed with the need for clear learning objectives but picked up on the word 'measurable': *"does everything need to be measurable? For computational thinking, I think creativity is a very important aspect"* (Simon). The other teachers described how they started with subject related learning objectives and then used CT as a means to accomplish these goals (Robin) or selected CT goals that can be linked to the lesson (Tanya).

4. Discussion and Future work

This poster presented teachers' first views and feedback on the preliminary design principles for computational thinking integration in non-computing subjects. The first preliminary findings suggest teachers do agree on some design principles, however additional data analysis is needed to understand reasons for disagreement on the other design principles. Also, only one teacher was able to describe which design principles he used in developing his lesson, while the other two were not able to single out any. Therefore, further research should investigate teachers' tacit knowledge on design principles and ways to make it explicit. Finally, future work should

also include a wider number of participants, from both primary and secondary schools, and from a wider spectrum of subjects.

References

[1] S. Papert, Mindstorms: Computers, children, and powerful ideas, NY: Basic Books (1980).

[2] J. M. Wing, Research Notebook: Computational Thinking–What and Why?, The Link Magazine, Carnegie Mellon University's School of Computer Science (2011).

[3] J. Voogt, P. Fisser, J. Good, P. Mishra, A. Yadav, Computational thinking in compulsory education: Towards an agenda for research and practice, Education and Information Technologies 20 (2015) 715–728. doi:10.1007/s10639-015-9412-6.

[4] J. Nijenhuis-Voogt, S. Yeni, E. Barendsen, Integrating CT into biology: Using decision tree models to classify cell types, in: CTE-STEM 2022 conference, 2022. doi:https://doi.org/10.34641/ctestem.2022.460.

[5] J. Malyn-Smith, I. A. Lee, F. Martin, S. Grover, M. A. Evans, S. Pillai, Developing a framework for computational thinking from a disciplinary perspective, in: Proceedings of the International Conference on Computational Thinking Education, 2018, pp. 182–186.

[6] V. Barr, C. Stephenson, Bringing computational thinking to K–12: what is involved and what is the role of the computer science education community?, ACM Inroads 2 (2011) 48–54. URL: https://doi.org/10.1145/1929887.1929905.

[7] T.-C. Hsu, S.-C. Chang, Y.-T. Hung, How to learn and how to teach computational thinking: Suggestions based on a review of the literature, Computers & Education 126 (2018) 296–310. doi:https://doi.org/10.1016/j.compedu.2018.07.004.

[8] S.-C. Kong, Components and methods of evaluating computational thinking for fostering creative problem-solvers in senior primary school education, in: Computational thinking education, Springer Singapore, 2019, pp. 119–141.

[9] I. Lee, J. Malyn-Smith, Computational thinking integration patterns along the framework defining computational thinking from a disciplinary perspective, Journal of Science Education and Technology 29 (2020) 9–18. URL: https://doi.org/10.1007/s10956-019-09802-x. doi:10.1007/s10956-019-09802-x.

[10] P. Sengupta, J. S. Kinnebrew, S. Basu, G. Biswas, D. Clark, Integrating computational thinking with K-12 science education using agent-based computation: A theoretical framework, Education and Information Technologies 18 (2013) 351–380. URL: https://doi.org/10.1007/s10639-012-9240-x. doi:10.1007/s10639-012-9240-x.

Easy Programming Tasks and Informatics for Everybody

Michael Weigend [1]

[1] *University of Münster, Schlossplatz 2, 48149 Münster, Germany*

Abstract

Programming tasks are an important element of informatics education in schools and universities. The difficulty to solve such tasks results from several facets, including: the size of the search space for finding an algorithmic solution, the number of problem-solving steps while developing an algorithm, the size of the code (lines of source code) that must be written and the degree of novelty regarding the learners' knowledge when they try to solve the task. The results of a survey with participants of a university course "Python programming for everybody" suggest that easy tasks are appreciated by all students including high-performing enthusiasts.

Keywords

Programming task, difficulty, Python

1. The Difficulty of Programming Tasks

Informatics education that aims to be appealing to a wide audience (including less talented and less interested students) requires a sufficient reservoir of easy tasks, which every participant can solve.

1.1. Programming Tasks for Learning

This contribution focuses on a special type of programming tasks that are supposed to be solved in the computer lab during a lesson. They are characterized by these properties:

Constructive. Programming tasks require writing formal code. They may challenge creativity and give opportunity construct a relevant digital product. This is the basic idea of Constructionism [1].

Material-related. The tasks are closely related to a limited area of knowledge, which is presented in learning material (textbook sections, code examples, illustrations etc.) and motivates to elaborate this material. A "starter project" may be in-volved. This is a short runnable program that can be tried out and changed [2].

Voluntary. The tasks are not part of an exam. Course participants choose for themselves which tasks they want to solve, in which order they solve them, in which social constellation (alone or in pairs) they solve them and which material they use to solve them.

Self-assessable. Learners can check solutions themselves. The task has been solved correctly if the program works. The IDE provides both assessment and immediate feedback.

Self-scalable. Learners can expand the goal or discard parts of it and thus change the difficulty of the task.

Quickly solvable. Easy programming tasks can be solved and checked in a few minutes or seconds.

1.2. Facets of Difficulty

Informatics in Schools. A step beyond digital education. 15th International Conference on Informatics in Schools: Situation, Evolution, and Perspectives, ISSEP 2022, Vienna, Austria, September 26–28, 2022
EMAIL: mw@creative-informatics.de
ORCID: 0000-0003-3246-5147


Estimating the difficulty of programming tasks is important for classroom teaching, for the design of textbooks and for online teaching systems. This section presents some criteria for defining the difficulty of a programming task with emphasis on the lower end of the scale. In contrast to other approaches to measure difficulty [3, 4], not only the task, but the entire learning situation in which it is embedded is considered.

Size of Problem Space. Programming is a problem-solving process. An easy programming task has a small problem space [5, 6]. There are only a few code elements that are suitable for a solution, and which can easily be found in the related learning material.

Distance to an Algorithmic Solution. An easy programming task requires only a few reasoning steps to find the idea of an algorithmic solution. Example: The following task descriptions are all related to the same given starter project but induce increasing distances:

A: Make the font on the button larger.

B: Make sure that the text is better readable.

C: Make the application accessible to the visually impaired.

Familiarity of Problem Representation. An easy programming tasks is not an insight problem that requires an unusual representation [7], but a problem that can be solved without going far beyond practiced thinking habits.

Effort. An easy programming task can be solved with little effort in a short time. However, learners decide themselves on the effort they want to invest in a task (typing a given example program instead of copying it, extending the task, experimenting etc.).

Diversity and the Degree of Novelty. Programming tasks can be used to practice programming concepts that are not yet familiar. In an easy programming task, only a single novel programming concept is practiced. It has a low degree of novelty.

Comprehensibility. In an easy task, the goal is easy to grasp.

2. Observations in a University Class

This section presents the results of surveys conducted in an introductory course at the University of Münster, Germany on programming with Python, which was aimed at students of all faculties and did not require any previous knowledge. Each 180-min event was divided into thematic blocks, each consisting of a lecture with discussion and a subsequent exercise phase (each about 20 minutes). For each of these exercise phases, the students got a set of tasks from which they could choose. The students were encouraged to have a look at all tasks and then pick some to work on. Participation was anonymous and voluntary. One of the questionnaires in winter semester 2021/22 was on processing easy tasks. It was filled by 14 persons. Of these, 6 stated that they would write the final exam, 4 people did not want to take part and 4 people had not yet decided.

Tasks designed as easy tasks were marked as “easy”. However, the students’ subjective perception of the difficulty level might have been different. For this reason, the introductory text of the questionnaire contained a passage characterizing easy programming tasks like this: “You have a solution idea within 5 seconds and are convinced that you can solve the task - even if writing the program text may take longer than 5 seconds.”

The questionnaire asked for self-observations on three aspects:

- Strategies for selecting tasks (S)
- Strategies for processing easy tasks (P)
- Emotional effects (E)

The participants were asked to tick appropriate statements and had the opportunity to add observations in words. Table 1 shows some results.

Table 1
Self-observations on processing easy tasks (n = 14)

Statement	Count
S1: I skip easy tasks and focus on more difficult tasks.	1 (7%)
S2: When I read an easy task, I first check whether I have already mastered the related content before I solve the task.	1 (7%)
S3: I solve easy tasks, even if I feel that I am underchallenged.	3 (21%)
S4: I start with the easy tasks and then solve the difficult ones.	11 (79%)
S5: During an exercise, I first solve tasks that interest me. Only then do I deal with the easy tasks.	2 (14%)
S6: I deal with easy tasks when I can't cope with more difficult tasks.	1 (7%)
P1: I spend a few seconds thinking about the easy programming task. When I am sure that I can solve it correctly, I stop working on the task and do not write any code.	3 (21%)
P2: Even if I can solve an easy programming task in my mind, I write down program text.	10 (71%)
P3: Even though I am quite sure that my solution of the easy programming task is correct, I run the program and check if it works correctly.	13 (92%)
P4: An easy programming task inspires me to explore programming on my own. This means: After solving the task, I try out other program constructs	4 (29%)
E1: When working on an easy programming task, I get aggressive, for example because the task seems trivial or irrelevant to me.	0 (0%)
E2: When solving an easy programming task, I realize that I understand programming better.	10 (71%)
E3: When working on an easy programming task, I realize that I have not yet understood one thing correctly.	10 (71%)
E4: When solving an easy programming task, I feel satisfaction.	11 (79%)
E5: When working on an easy programming task, I feel motivated to ask a question about the topic addressed in the task.	5 (36%)

3. Discussion

The students' responses suggest that easy programming tasks are highly appreciated. They are rarely skipped (S1) and often used to get started with a topic (S3, S4). In contrast to overexplaining (explaining things that are already known), easy tasks do not trigger aggressive feelings (E1) which can be the result of being bored. They are important for self-diagnosing when the solution is implemented (E2, E3), but have no diagnostic value to students if they are only read (S2).

Easy programming tasks (with a low risk of failure) are highly attractive. Learners run self-written programs, even if they are sure that they work and thus do not expect additional cognitive learning (P1, P2, P3). Easy tasks cause positive feelings (E1, E4) and sometimes encourage students to ask questions (E5) or to explore programming on their own (P4).

4. Designing Easy Tasks

Here are some patterns for easy tasks:

- Try. Copy source code and run it.
- Change. Change names or values in a starter project.
- Expand. Add some code to a starter project.

- Transfer. Use the starter project as a pattern for a project on a different topic.
- Improve. The starter project contains a clear weakness that needs to be improved.
- Experiment. Plan and conduct an experiment on a question regarding semantics or syntax.

(Partly) based on the survey results, I suggest some general design principles for easy tasks:

- Since most leaners start with easy tasks, these should be solvable very quickly, so there remains enough time to do more challenging exercises.
- Easy tasks should focus on a narrow area of the learning material and activate it's elaboration. The task description should contain explicit references ("Use table 1.2") or supply the needed information directly.
- Easy tasks should not be easy variants of difficult tasks. Each task should stand for itself and cover some interesting aspect. All tasks of a task set should have the same value. Ideally, when a student explains the solution of an easy task, he or she should make a genuine contribution that has not already been made by someone else in the context of a more difficult task.
- Inclusive lessons offer easy programming tasks to all topics, including difficult ones. For a programming technique that might overwhelm part of the audience, the teacher can create an easy task that relates to a more fundamental aspect. This "substitute" task may create awareness of a difficult technique without requiring mastering it.

5. Conclusion

Inclusive computer science lessons are a challenge for computer science teachers to develop simple programming tasks. The effort is worth it, as it seems that easy programming tasks are an enrichment for all learners – even the talented and interested ones.

6. References

[1] I. E. Harel and S. E. Papert, Constructionism. Ablex Publishing, Norwood, NJ, 1991.
[2] M. Weigend. Starter Projects in Python Programming Classes, in: Open Conference on Computers in Education, Springer, Cham, 2021, pp. 104–115.
[3] T. Effenberger, J. Čechák, and R. Pelánek, Measuring difficulty of introductory program-ming tasks. In Proceedings of the Sixth ACM Conference on Learning@ Scale, 2019, pp. 1–4.
[4] J. Sheard, A. Carbone, D. Chinn, T. Clear, M. Corney, D. D'Souza, J. Fenwick, J. Harland, M.-J. Laakso, D. Teague, How Difficult Are Exams? A Framework for Assessing the Complexity of Introductory Programming Exams, in: Proceedings of the 15th Australasian Computing Education Conference (ACE '13), 2013, pp. 145–154.
[5] J. R. Anderson, Problem solving and learning, in: American psychologist, 48(1), 35, 1993.
[6] A. Newel and H. A. Simon, Human problem solving, Englewood Giffs, NJ: Prentice-Hall, 1972.
[7] S. A. Kaplan and H. A. Simon, In search of insight, in: Cognitive Psychology, 22(3), 1990, pp. 374-419.

Identifying Teaching Patterns in Pre-service Informatics Teacher Education

Bernhard Standl, Nico Hillah and Nadine Schlomske-Bodenstein

Karlsruhe University of Education, Bismarckstr. 10, 76133 Karlsruhe, Germany

Abstract

This paper presents a case study of N = 3 seminars in teacher education in informatics, in which the overall goal is to identify and describe effective teaching patterns for easy reuse. Therefore, this case study first examines learning and instructional designs and their implications with (N = 9) pre-service informatics teachers and (N = 3) university teachers as instructional scenarios with associated evaluation data, and then uses them to identify best practices. It is argued in this poster that best examples of instructional settings in pre-service informatics teacher education can be assessed and finally reused. First results show insights in the impact of giving feedback in seminars. We conclude this poster with some implications for learning and instructional design in informatics.

Keywords

Teaching patterns, Taxonomy, Technology-enhanced learning and instruction

1. Introduction

When designing teaching for pre-service informatics teachers, the choice of digital tools and teaching methods needs to be aligned with the intended learning outcomes [1]. However, teachers design their lectures based on their epistemological assumptions on good teaching and learning [2]. In informatics in particular, the seminars need to address the acquisition of algorithmic and computational thinking. Computational thinking includes areas such as abstraction, decomposition, but is also the ability to design algorithms and interpret the world based on computer science knowledge [3, 4, 5, 6]. To assess instructional quality, several features of instructional quality have been identified and categorized as distinctive predictors of students' learning [7, 8]. Considering the pattern approach [9, 10, 11], it is aimed to identify teaching patterns. Hence, the following research question is driving this study: *To what extent can teaching examples in informatics be conceptualized, and teaching patterns be identified in teacher education?*

Informatics in Schools. A step beyond digital education. 15th International Conference on Informatics in Schools: Situation, Evolution, and Perspectives, ISSEP 2022, Vienna, Austria, September 26–28, 2022

bernhard.standl@ph-karlsruhe.de (B. Standl); nico.hillah@ph-karlsruhe.de (N. Hillah); nadine-schlomske-bodenstein@ph-karlsruhe.de (N. Schlomske-Bodenstein)

0000-0002-8849-2980 (B. Standl); 0000-0001-7818-6042 (N. Hillah); 0000-0003-0835-165X (N. Schlomske-Bodenstein)

2. Method

In order to identify the patterns, the collected data needs to be empirically analyzed. Therefore, data is structured according to computer science content, methodological approaches, and key characteristics of teaching quality. Considering this, to identify initial evidence for patterns, our research efforts are based on the case study approach as described by [12] in the mixed methods paradigm [13, 14] and are embedded in the design-based research approach [15] [16]. We quantitatively assess the instructional quality with an instrument which was already piloted and showed a Cronbach's Alpha of 0.95 [17]. Both, the teachers and the learners evaluated the instructional quality. Every item is rated on a 5-point Likert scale (1 = not at all true, 2 = rather not true, 3 = neither false nor true, 4 = rather true, 5 totally true). Moreover, the teachers were interviewed afterwards and were asked to fill out the evaluation survey regarding the digital tools that were used and additional information on their instructional setting into our databases. Three courses in informatics for teacher education were assessed: two courses on didactics and one course on database: **Didactics of Informatics**: students learn how to apply and integrate different didactic models when designing teaching sessions in informatics. **Seminar on Didactics**: Students run workshops with pupils and are videotaped. During the semester, they reflect their teaching competencies based on these video recordings. **Databases**: The course introduces the concept of relational database technology. Students work in groups to design and implement their own databases. N = 24 students in informatics participated in three different courses. However, not all of them filled out the evaluation surveys. The sample consists of N = 9 pre-service teachers in informatics. N = 3 in the didactics of informatics (1 female and 2 males with an average age of 24), N = 4 in the seminar on didactics (1 female and 3 males with an average age of 25), N = 2 in databases (2 males with an average age of 27).

3. Preliminary Results

We analyzed the mean scores (M) and the standard derivations (SD) of each workshop to examine the instructional quality. Table 1 illustrates a descriptive overview of the means and standard deviations for all scales. The course where most teachers participated by giving feedback to the students and prompted them through questions was rated very high in all scales. However, the course size was very small and not all students filled out the evaluation in the end.

4. Discussion and Conclusion

In this paper, we explored three different courses in informatics, which included feedback on students' performances from the teachers, and from the learners. The first assumption which emerges from these preliminary descriptive results is that, receiving feedback from multiple teachers' perspectives positively impacts the student's learning process. Moreover, it indicates that becoming teachers need to be trained on how to provide feedback to the learners on their learning outcomes. In future steps, this assumption needs to be investigated with larger samples by evaluating whether providing students with multiple feedback and by bringing them to reflect on their learning outcomes from multiple perspectives impacts their learning process.

Table 1
Description of the Instructional Quality

Scale	Didactics of Informatics N = 3		Seminar on Didactics N = 4		Databases N = 2	
	M	SD	M	SD	M	SD
Self-efficacy	3.67	0.58	3.75	1.50	4.50	0.71
Structuring	3.73	0.70	4.25	0.64	5.00	0.00
Activating	3.75	0.90	4.69	0.63	5.00	0.00
Motivating	4.17	0.29	4.63	0.60	4.50	0.35
Learning strategies	4.00	0.25	4.44	0.72	4.75	0.00
Feedback	4.00	1.00	4.00	0.41	4.25	0.35
Interest	3.92	0.58	4.17	1.44	5.00	0.00
Offer variants	4.67	0.00	3.75	0.96	5.00	0.00
Constructive alignment	4.67	0.00	4.33	0.47	4.67	0.00
Level of difficulty	4.00	0.00	4.75	0.71	4.00	0.00
Teacher	4.42	0.38	4.44	0.38	4.63	0.18
Total	3.82	0.43	4.32	0.73	4.73	0.09

At this stage, our case study is limited to various points that must be addressed in future stages. Data mining is normally based on a huge set of data [18]. A constraint of our case study is the limited amount of empirical data used to run the pattern processing. Once these limitations and future research aims have been addressed, structuring data in a database will offer new data mining approaches. The identification of best teaching practice patterns then can further result in innovative changes in teacher education in informatics aimed at easy reuse and transfer of informatics teaching best practices.

Acknowledgments

This project (grant number 01JA2027) is part of the “Qualitätsoffensive Lehrerbildung”, a joint initiative of the Federal Government and the Länder which aims to improve the quality of teacher training. The programme is funded by the Federal Ministry of Education and Research. The authors are responsible for the content of this publication.

References

[1] J. Biggs, C. Tang, Teaching for quality learning at university, McGraw-hill education (UK), 2011.
[2] I. Hähnlein, Erfassung epistemologischer Überzeugungen von Lehramtsstudierenden: Entwicklung und Validierung des StEB Inventars, Ph.D. thesis, Universität Passau, 2018.
[3] V. J. Shute, C. Sun, J. Asbell-Clarke, Demystifying computational thinking, Educational Research Review 22 (2017) 142–158.
[4] J. M. Wing, Computational thinking: What and why, 2017.
[5] J. M. Wing, Computational thinking, Communications of the ACM 49 (2006) 33–35.

[6] P. J. Denning, M. Tedre, Computational thinking, Mit Press, 2019.
[7] A. Helmke, Unterrichtsqualität und lehrerprofessionalität, Diagnose, Evaluation und Verbesserung des Unterrichts 2 (2009).
[8] T. Seidel, R. J. Shavelson, Teaching effectiveness research in the past decade: The role of theory and research design in disentangling meta-analysis results, Review of educational research 77 (2007) 454–499.
[9] C. Alexander, A pattern language: towns, buildings, construction, Oxford university press, 1977.
[10] B. Standl, N. Schlomske-Bodenstein, A pattern mining method for teaching practices, Future Internet 13 (2021) 106.
[11] R. Bauer, Didaktische Entwurfsmuster: Der Muster-Ansatz von Christopher Alexander und Implikationen für die Unterrichtsgestaltung, Waxmann Verlag, 2015.
[12] R. K. Yin, Case study research and applications, Sage, 2018.
[13] A. Tashakkori, C. Teddlie, C. B. Teddlie, Mixed methodology: Combining qualitative and quantitative approaches, volume 46, sage, 1998.
[14] C. Teddlie, A. Tashakkori, Overview of contemporary issues in mixed methods research, Sage handbook of mixed methods in social and behavioral research 2 (2010) 1–44.
[15] S. Barab, Design-based research: A methodological toolkit for engineering change, in: The Cambridge Handbook of the Learning Sciences, Second Edition, Cambridge University Press, 2014, pp. 151–170.
[16] D.-B. R. Collective, Design-based research: An emerging paradigm for educational inquiry, Educational researcher 32 (2003) 5–8.
[17] N. Schlomske-Bodenstein, B. Standl, A pilot study to identify evidence-based technology-enhanced teaching patterns in pre-service teacher education, in: Society for Information Technology & Teacher Education International Conference, Association for the Advancement of Computing in Education (AACE), 2022, pp. 571–576.
[18] S.-J. Yen, Y.-S. Lee, C.-W. Wu, C.-L. Lin, An efficient algorithm for maintaining frequent closed itemsets over data stream, in: International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems, Springer, 2009, pp. 767–776.

Learning through Teaching - External Control of Computer Science Learning Applications

Thiemo Leonhardt[1,*,†], Nadine Bergner[1,†]

[1]TU Dresden, Nöthnitzer Str. 46, 01187 Dresden, Germany

Abstract

The competence of computer science teachers is crucial for the learning process of the students. To improve this, a project is presented here in which (prospective) teachers learn to better understand the learning process of students. To do this, they learn to use digital learning applications and to accompany the students as learning guides. For this purpose, learning data of the students is collected in the applications, aggregated and presented on a dashboard. Furthermore the dashboard offers the possibility for the learning guide to influence the learning application. This procedure is expected to lead to an increase in competence with regard to the learning content for the students as well as an increase in competence with regard to the teaching of the learning content for the (prospective) teachers. The goal is to impart knowledge about misconceptions and difficulties of technical content to (prospective) computer science teachers and to use this knowledge adequately in learning situations, especially with digital learning applications.

Keywords

Learning through teaching, teacher training, teacher education, computer science education, competence development

1. Introduction

Digital learning applications are particularly widespread in computer science education. On the one hand, this is due to the technical competence of the computer science teachers and on the other hand to a technical affinity resulting from the school subject itself. Digital learning applications for computer science education include for example development environments for programming that have been specially adapted for schools, simulations that make it possible to understand processes within networks, and learning games and quizzes that present computer science content [4, 8].

In addition to the pure use of learning applications in the classroom, the learning design of the applications is an essential topic for student teachers in computer science, as questions about learning objectives, learning content, and learning taxonomies can be discussed. The

Informatics in Schools. A step beyond digital education. 15th International Conference on Informatics in Schools: Situation, Evolution, and Perspectives, ISSEP 2022, Vienna, Austria, September 26–28, 2022

*Corresponding author.

†These authors contributed equally.

thiemo.leonhardt@tu-dresden.de (T. Leonhardt); nadine.bergner@tu-dresden.de (N. Bergner)

https://tu-dresden.de/ing/informatik/smt/ddi/ (T. Leonhardt); https://tu-dresden.de/ing/informatik/smt/ddi/ (N. Bergner)

0000-0003-4725-9776 (T. Leonhardt); 0000-0003-3527-3204 (N. Bergner)

possibility of data collection through learning analytics techniques creates new insights into the learning processes of students and design difficulties of learning applications.

In this project, learning data from established computer science learning applications are modeled, aggregated, and then visualized. It is possible to analyze the data via a dashboard and then to intervene in the learning process. Our thesis is, that this leads to a better understanding of learners' barriers and possible misconceptions which helps to increase the competence of (prospective) teachers in the creation of learning materials. Furthermore, an optimization of the learning applications is an additional effect.

2. Related Work

The fact that guided teaching can also lead to an increase in professional competence among teachers has been known since the 1980s [2, 5]. These results were transferred by Jean-Pol Martin into the theory learning by teaching and were developed further since then. In addition, studies such as a meta-analysis of school-based tutoring programs [3] also show positive results. Reflective knowledge is said to be constructed through interactive processes of knowledge construction when teaching others [7]. Koh et. al. argue, the teacher's success depends in part on building reflective knowledge [6]. This can be stimulated by teachers reflecting on their own understanding of the material and connecting it to their own prior knowledge as they teach. In contrast, the learning process of the teachers is less likely to occur if they merely summarize the material without connecting it to their prior knowledge. Our approach starts at this point and tries to trigger these effects in (perspective) teachers by presenting learning data and playing an active role as a learning guide. In a recent study, Aslan shows that learning through teaching also can be used successfully in the classroom specifically with biology and chemistry content [1]. However, transfer of her findings to the entire STEM (science, technology, engineering, and mathematics) field is likely.

3. Project and Research Plan

In our research plan, for the three learning applications (fig. 1) Struktog[1], RegEx[2], and Inside the Router[3] learning data are modeled and make them accessible in a learning record store. Struktog is a structure diagram editor, where imperative algorithms can be modeled [anonymized]. This is primarily designed as a tablet and desktop variant. RegEx is a collaborative matching game where words have to be matched to the appropriate regular expressions [anonymized]. The game is optimized for a multi-touch table to allow collaborative learning and simultaneous input. In the virtual reality (VR) learning application Inside the Router, the player takes on the role of a home router in allocating network packets. This is optimized for VR glasses. Through the broad selection of these applications, we hope to gain good insights into the practicality of our approach to computer science education.

[1] https://dditools.inf.tu-dresden.de/struktog/
[2] https://ddigames.inf.tu-dresden.de/zuordnunginf/
[3] https://ddigames.inf.tu-dresden.de/sites/vr/router/

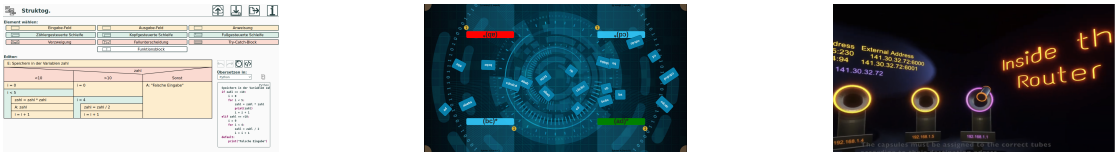


Figure 1: learning applications left: Struktog - center: RegEx - right: Inside the Router

The data from each of the applications is aggregated and presented on a application-specific dashboard. The dashboard is extended in such a way that it is possible to intervene in the learning process. As a research design, within a specialized didactic seminar, student teachers in computer science will take partly the role of the learner and partly the role of the teacher. The increase in competence will then be assessed by an A/B testing.

References

[1] Aslan, S.: Is Learning by Teaching Effective in Gaining 21st Century Skills? The Views of Pre-Service Science Teachers. Educational Sciences: Theory and Practice **15**(6), 1441–1457 (Dec 2015)

[2] Bargh, J.A., Schul, Y.: On the cognitive benefits of teaching. Journal of Educational Psychology **72**(5), 593 (1980). https://doi.org/10/fpwkbq

[3] Cohen, P.A., Kulik, J.A., Kulik, C.L.C.: Educational Outcomes of Tutoring: A Meta-analysis of Findings. American Educational Research Journal **19**(2), 237–248 (Jan 1982). https://doi.org/10/dchpj5

[4] García-Peñalvo, F.J., Rees, A.M., Hughes, J., Jormanainen, I., Toivonen, T., Vermeersch, J.: A survey of resources for introducing coding into schools. In: Proceedings of the Fourth International Conference on Technological Ecosystems for Enhancing Multiculturality. pp. 19–26. TEEM '16, Association for Computing Machinery, New York, NY, USA (Nov 2016). https://doi.org/10/gqn6pv, https://doi.org/10.1145/3012430.3012491

[5] Grzega, J., Schöner, M.: The didactic model LdL (Lernen durch Lehren) as a way of preparing students for communication in a knowledge society. Journal of Education for Teaching **34**(3), 167–175 (Aug 2008). https://doi.org/10/fdzrdh

[6] Koh, A.W.L., Lee, S.C., Lim, S.W.H.: The learning benefits of teaching: A retrieval practice hypothesis. Applied Cognitive Psychology **32**(3), 401–410 (2018). https://doi.org/10/gdjrn2

[7] Roscoe, R.D., Chi, M.T.H.: Understanding Tutor Learning: Knowledge-Building and Knowledge-Telling in Peer Tutors' Explanations and Questions. Review of Educational Research **77**(4), 534–574 (Dec 2007). https://doi.org/10/cjz6ph

[8] Szabo, C., Sheard, J., Luxton-Reilly, A., Simon, Becker, B.A., Ott, L.: Fifteen Years of Introductory Programming in Schools: A Global Overview of K-12 Initiatives. In: Proceedings of the 19th Koli Calling International Conference on Computing Education Research. pp. 1–9. Koli Calling '19, Association for Computing Machinery, New York, NY, USA (Nov 2019). https://doi.org/10/gp76jf, https://doi.org/10.1145/3364510.3364513

Workshops

A Logic-System Simulator Designed for Teaching

Jean-Philippe Pellet

University of Teacher Education, Lausanne, Switzerland

Abstract

This workshop introduces the participants to an open-source logic simulator the author has been developing, highlighing where it differs from existing solutions and showing how its features make pegagogical sense according to a PRIMM approach (*Predict-Run-Investigate-Modify-Make* [1]). Building on this, we show hands-on activities and exercises that enrich the traditional pedagogical approach to logic gates and logic systems.

Keywords

computer science education, logic-system simulator, PRIMM

1. Workshop Contents

One of the benefits of teaching of computer science is that the teachers can get students to build small computer systems that can be interacted with. Examples include programming a graphical interface, a small robot, or an electronic board system like a Microbit or Arduino. In this workshop, we want to go down one step in the hardware layers while staying in the same state of mind: we demonstrate a tool for simulating logic gates and other basic computer components. This tool is free, open source, mostly serverless, and available at https://logic.modulo-info.ch).

We show during the workshop how this simulator differs from other solutions (quickly showing them and how they are used [2, 3]). We highlight its pedagogical possibilities well suited to a PRIMM approach [1], an extension of the well known *Use-Modify-Create* model [4]. We show and discuss how the proposed tool makes it easy not only to design and simulate circuits (which all such simulators do), but also to distribute and share circuits, to tune simulation parameters, to hide parts of the circuits, to create investigation tasks, to include faulty components, and so on. All these features enrich the pedagocial possibilities of the simulator and make it more useful for teachers.

In the workshop, we propose some pedagogical scenarios adapted to early high school students (full adder, ALU and flip-flops, but also a completely math-free scenario). We also show how teachers can create their own exercises and demonstrate the teacher settings.

Informatics in Schools. A step beyond digital education. 15th International Conference on Informatics in Schools: Situation, Evolution, and Perspectives, ISSEP 2022, Vienna, Austria, September 26–28, 2022

jean-philippe.pellet@hepl.ch (J. Pellet)

0000-0001-7559-397X (J. Pellet)

References

[1] S. Sentance, J. Waite, Primm: Exploring pedagogical approaches for teaching text-based programming in school, in: Proceedings of the 12th Workshop on Primary and Secondary Computing Education, 2017, pp. 113–114.

[2] V. Luković, R. Krneta, A. Vulović, C. Dimopoulos, K. Katzis, M. Meletiou-Mavrotheris, Using Logisim educational software in learning digital circuits design, in: Proceedings of 3rd International Conference on Electrical, Electronic and Computing Engineering IcETRAN, 2016, pp. 5–1.

[3] R. A. S. Rueda, J. A. S. Silis, Logic.ly simulator. Technological tool to facilitate the teaching-learning process about mathematics?, Dilemas Contemporáneos: Educación, Política y Valore (2018).

[4] I. Lee, F. Martin, J. Denner, B. Coulter, W. Allan, J. Erickson, J. Malyn-Smith, L. Werner, Computational thinking for youth in practice, ACM Inroads 2 (2011) 32–37.

Maker-Education: Interdisciplinary Computer Science Activities

Bernadette Spieler[1,*,†], Tobias M. Schifferle[1,*,†]

[1]Zurich University of Teacher Education, Lagerstrasse 2, 8090 Zurich, Switzerland

Abstract

Typical characteristics of Making are creative and novel solutions and open, problem-based learning environments. Thereby, Making facilitates interdisciplinary connections and cross-cutting competences, for example, technical understanding, creativity, craft skills, or concepts of sustainability and entrepreneurship. Traditional crafts can be combined with modern technologies such as programmable embroidery machines, 3D printers, laser cutters, or microcomputers. In this way, Making can be an inspiration to further develop a mutual understanding of teaching in tandem with the promotion of creative project activities, simultaneously combining analog and digital tools across disciplines. However, Making is still in its early stages in schools and education. Therefore, much is still undefined and open.

Keywords

Maker-Education, Making, Open Learning Spaces, Interdisciplinary, Teacher Training, STEAM

1. Making, Makerspaces & Maker-Education

In recent years, the Maker movement has surged in popularity and has become attractive for didactic research [1]. With the introduction of the Swiss Curriculum 21 (https://lehrplan.ch/) in German-speaking Switzerland in 2014, a strong focus on general competencies and future skills such as creativity can be observed in schools. In connection with Making, familiar characteristics from the constructionism theory, such as "learning-by-doing", "objects-to-think-with" or principles such as student-centered, project-oriented, individualised, and self-organised learning are emphasised. Making has many dimensions, and especially in informatics, it can cover topics such as programming, robotics, virtual reality, or data literacy (e.g., building measuring instruments). It also allows for interdisciplinary projects, e.g., explaining biological phenomena or creating musical instruments.

Informatics in Schools. A step beyond digital education. 15th International Conference on Informatics in Schools: Situation, Evolution, and Perspectives, ISSEP 2022, Vienna, Austria, September 26–28, 2022

*Corresponding author.

†These authors contributed equally.

bernadette.spieler@phzh.ch (B. Spieler); tobias.schifferle@phzh.ch (T. M. Schifferle)

https://phzh.ch (B. Spieler); https://phzh.ch (T. M. Schifferle)

0000-0002-0877-7063 (B. Spieler); 0000-0003-1018-7158 (T. M. Schifferle)

2. Making & Informatics

In informatics, students should develop critical thinking and problem-solving skills, especially in the context of Computational Thinking (CT) [2]. CT is generally understood as processes that strengthen logical, algorithmic and abstract thinking. Thus, Making can be an anchor to creatively introduce students to informatics principles, e.g., in programming or design projects [3]. However, the purpose of Making is not just to reproduce projects, but to encourage active participation, knowledge sharing and collaboration through open exploration and creative use of technology. Making can thus be seen not only as an opportunity to experiment with new forms of teaching informatics, but also as a driving force for a new learning culture that opens up broader perspectives for school development.

3. The Making at School project

Makerspaces can be publicly accessible creative spaces at fixed locations, temporary "popup" makerspaces [4], or mobile versions such as experimental kits ("Workshop in a Box") or integrated into classrooms as Maker-Education [5]. To permanently establish Making and the corresponding "Maker Mindset" in schools, exchange and further teacher training formats are needed in addition to the supply of infrastructure and high-quality interdisciplinary teaching materials. During the project "Making at School" as part of the digitisation initiative of the Zurich Universities of Applied Sciences (DIZH), modules for Making are collaboratively designed by the Zurich University of Teacher Education (PHZH), together with the University of Zurich, as well as in cooperation with the School Authority of the City of Zurich [6].

References

[1] S. Becker, M. Jacobsen, Becoming a Maker Teacher: Designing Making Curricula That Promotes Pedagogical Change, Frontiers in Education (2020). doi:10.3389/feduc.2020.00083.

[2] J. Wing, Computational Thinking, Communications of the ACM (2006) 33–35.

[3] J. Hughes, J. A. Robb, M. S. Hagerman, J. Laffier, M. Cotnam-Kappel, What makes a maker teacher? Examining key characteristics of two maker educators, International Journal of Educational Research Open 3 (2022) 100–118. doi:10.1016/j.ijedro.2021.100118.

[4] B. Spieler, M. Grandl, V. Krnjic, The hAPPy-Lab: A Gender-Conscious Way To Learn Coding Basics in an Open Makerspace Setting, in: Proceedings of the 13th International Conference on Informatics in School: Situation, Evaluation and Perspectives, ISSEP 2755 (2020) 64–75.

[5] S. L. Sydow, A. Åkerfeldt, P. Falk, Becoming a Maker Pedagogue: Exploring Practices of Making and Developing a Maker Mindset for Preschools, FabLearn Europe / MakeEd 2021, ACM, New York, NY, USA, 2021, pp. 1–10. doi:10.1145/3466725.3466756.

[6] B. Spieler, T. M. Schifferle, M. Dahinden, The "Making at School" Project: Planning Interdisciplinary Activities, in: Proceedings of the 27th ACM Conference on Innovation and Technology in Computer Science Education Vol. 2, ITiCSE '22, ACM, New York, NY, USA, 2022, p. 624. doi:10.1145/3502717.3532150.

Promoting Algorithmic Thinking: Students Using mBots and Block-Based Programming Practical Insights into Two Different Problem-Based Approaches

Frauke Ritter, Anette Bentz and Bernhard Standl

Karlsruhe University of Education, Bismarckstr. 10, 76133 Karlsruhe, Germany

Abstract

In our Teaching-Learning-Laboratory for Informatics (TLL), students from the Karlsruhe University of Education develop workshops and try them out on school classes. The workshops aim to teach the pupils algorithmic thinking - here mBot-Robots are used, which are programmed with a block-based programming language in 4 modules (disco animation, rescue scenario, Formula 1 - race and maze). Methodologically, two different problem-based approaches are evaluated: a guided problem-oriented approach, which didactically follows a semantic wave, and a constructivist problem-based approach. We will test both approaches in the workshop and then discuss them.

Keywords

Algorithmic Thinking, block-based programming, Robotics, Semantic Wave

Description of the Workshop

In our digitized world, it is important that students learn the core competency of computational thinking at school in a sustainable way in order to understand the technologies and processes behind the tools and end devices they use every day. This is achieved on the one hand with the help of motivating tasks or gadgets (computational action [1]) and on the other hand with the help of suitable didactic approaches that go beyond a mere programming course. To promote school students' algorithmic thinking competencies, we elaborated two teaching approaches: a guided problem-based [2, 3] and a constructionist problem-based one [4, 5]. In this workshop, we will show how we implement these two approaches with mBots in workshops at our computer science teaching-learning lab. The workshop will also provide the opportunity to experiment directly with the mBot robots. mBot robots are easy-to-build robots with light and ultrasonic sensors as input options and an LED display as a further output option in addition to driving the robot itself. This simple handling with basic robot functions and the option of block-based

Informatics in Schools. A step beyond digital education. 15th International Conference on Informatics in Schools: Situation, Evolution, and Perspectives, ISSEP 2022, Vienna, Austria, September 26–28, 2022

frauke.ritter@ph-karlsruhe.de (F. Ritter); anette.bentz@ph-karlsruhe.de (A. Bentz); bernhard.standl@ph-karlsruhe.de (B. Standl)

https://www.ph-karlsruhe.de (F. Ritter); https://www.ph-karlsruhe.de (A. Bentz); https://www.ph-karlsruhe.de (B. Standl)

0000-0001-5744-9456 (F. Ritter); 0000-0002-8849-2980 (B. Standl)

programming using the software mBlock make the mBot robot a motivating gadget for students. Therefore, the aim of the workshop is to get to know, carry out and discuss the two different teaching approaches (a guided problem-based and a constructionist problem-based) in one workshop with the same content (see Figure 1): first a disco animation is programmed, then a rescue scenario in a crisis area, a Formula 1 race and finally a labyrinth. After trying both approaches, the advantages and disadvantages of each will be discussed in order to further develop our concepts.

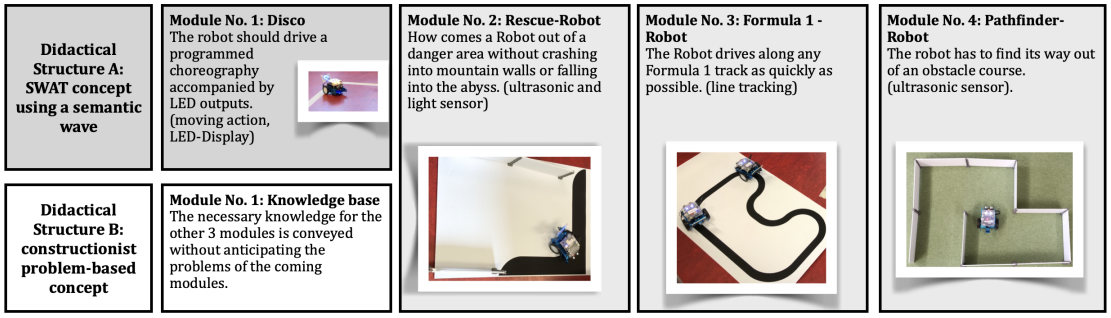


Figure 1: mBot workshop with two didactical approaches in comparison: SWAT concept using a semantic wave and a constructionist problem-based concept.

Acknowledgments

This work was supported by the Vector Foundation.

References

[1] M. Tissenbaum, J. Sheldon, H. Abelson, Viewpoint from computational thinking to computational action, Communications of the ACM 62 (2019) 34–36. doi:10.1145/3265747.

[2] F. Ritter, B. Standl, Promoting Student Competencies in Informatics Education by Combining Semantic Waves and Algorithmic Thinkingv, Informatics in Education 00 (2022). doi:10.15388/infedu.2023.07.

[3] F. Ritter, B. Standl, Promoting Computational Thinking in Teacher Education - Combining Semantic Waves and Algorithmic Thinking, in: Proceedings of the 2022 ACM Conference on International Computing Education Research - Volume 2, ICER '22, Association for Computing Machinery, New York, NY, USA, 2022, pp. 48–49. URL: https://doi.org/10.1145/3501709.3544285. doi:10.1145/3501709.3544285.

[4] A. Csizmadia, B. Standl, J. Waite, Integrating the constructionist learning theory with computational thinking classroom activities, Informatics in Education 18 (2019) 41–67.

[5] B. Standl, Conceptual modeling and innovative implementation of person-centered computer science education at secondary school level, University of Vienna, Vienna, Austria (2013).

Creating Personalized Online Courses based on Competencies and Open Educational Resources

Stefan Pasterk, Nina Lobnig, Lukas Pagitz and Kevin Wiltschnig

University of Klagenfurt, Department of Informatics Didactics, Universitaetsstrasse 65-67, 9020 Klagenfurt, Austria

Abstract

Digital technologies take over a central role in today's society and with that also in education. They enable new hybrid teaching and learning formats such as *blended learning*, full online environments such as *massive open online courses (MOOCs)*, or even *personalized open online courses (POOCs)*. The project *DigiFit4All* aims to generate individualized online courses in computer science and digital education based on competency models together with pre-testing the learners. For this purpose, a platform supporting the creation of online courses and enabling the automatic import into a learning management system is developed. In this workshop, the teachers' and learners' workflows are presented and can be tested on the platform by the participants.

Keywords

personalized learning, online course, open educational resources, competencies.

1. Introduction

Digital technologies have a deep impact on today's society and with that also on education. Over the last years, the number of online teaching materials in the form of *massive open online courses (MOOCs)*, which offer courses on various topics, arose. *Personalized open online courses (POOCs)* are the next step to individual learning in online settings with customized content. The Austrian project *DigiFit4All* aims to generate POOCs for computer science and digital education for pupils, university students, teachers, and administrative staff, to support educators and lecturers. It is a four-year cooperative project of the *University of Klagenfurt*, the *Johannes Kepler University Linz*, the *University for Continuing Education Krems*, and the *Vienna University of Technology*, which started in May 2020.

For the purposes of the project, an online platform is developed, which is based on several components [1]. It supports the organization of competencies and competency models, the collection of learning resources in the form of *open educational resources (OER)*, pre- and post-tests for individualization and assessment, and the direct export to a learning management system. The *GECKO (Graph-based Environment for Competency and Knowledge-Item Organization - https://gecko.aau.at)* platform enables the organization of competencies and the connection to learning resources and test questions, which are also stored on the platform.

Informatics in Schools. A step beyond digital education. 15th International Conference on Informatics in Schools: Situation, Evolution, and Perspectives, ISSEP 2022, Vienna, Austria, September 26–28, 2022

stefan.pasterk@aau.at (S. Pasterk); nina.lobnig@aau.at (N. Lobnig); lukas.pagitz@aau.at (L. Pagitz); kevin.wiltschnig@aau.at (K. Wiltschnig)

https://www.aau.at/en/informatics-didactics/team/pasterk-stefan/ (S. Pasterk)

Assessment and pre-tests are performed via *KAUA (Košice and Alpen-Adria University Assessment - https://kaua.aau.at)* [2], and a plugin regulates the import of learning material into a *Moodle (https://moodle.org/)* course.

After two years of development, the platform and the materials are prepared to start a pilot study in autumn 2022. In the workshop, the participants will be able to test the functionality of the platform and take a look at created resources. They learn how to operate the system, how it supports the creation of online courses, and how it can be used for their own purposes.

2. Creating Courses by Competency Selection

In the role of teachers or lecturers, the workshop participants can use the *GECKO* platform to follow the workflow for a course creation. After they defined a course, teachers select the first competency they want to reach in their course from a predefined pool of international competencies. As the system stores dependencies of the competencies as well [3], it suggests necessary foreknowledge in the form of competencies which should be reached before. Teachers decide if they want to add these suggestions to the course or not. Each of the competencies is connected to one or more learning resources prepared to reach the competency. So, as the next step, the teachers select the learning materials for their course. They repeat the steps until they have integrated all the competencies for their course. After that, the course can be imported into *Moodle*. Overall, the participants learn about a platform that can help them create their online courses.

3. Testing the Learning Resources

After the participants created their own courses, they change their role and become learners. They can enrol in their own courses and participate in a pre-test to find out which competencies they already have acquired. The test items are also connected to competencies and with that to the learning resources of the course. Following the results of the pre-test, the course only shows the material of those competencies the learner has not yet reached. In this role, the participants can take a look at the developed materials, which are mainly in German but will include translations or subtitles, and provide feedback.

References

[1] S. Pasterk, L. Pagitz, A. Weiß, A. Bollin, DigiFit4All – Conceptualisation of a Platform to Generate Personalised Open Online Courses (POOCs), in: D. Passey, D. Leahy, L. Williams, J. Holvikivi, M. Ruohonen (Eds.), Digital Transformation of Education and Learning - Past, Present and Future, Springer International Publishing, Cham, 2022, pp. 247–258.

[2] A. Bollin, M. Kesselbacher, C. Mößlacher, Ready for Computing Science? A Closer Look at Personality, Interests and Self-concept of Girls and Boys at Secondary Level., Springer International Publishing, 2020.

[3] S. Pasterk, A. Bollin, Graph-based Analysis of Computer Science Curricula for Primary Education, in: IEEE Frontiers in Education Conference (FIE). IEEE., 2017.

Learn to query databases in an interactive and engaging way

Nicole Burgstaller[1], Claudia van der Rijst[1], Nina Lobnig[2] and Claudia Steinberger[3]

[1] *Universität Klagenfurt, D!ARC, Klagenfurt, AUSTRIA*
[2] *Universität Klagenfurt, Informatics Didactics, Klagenfurt, AUSTRIA*
[3] *Universität Klagenfurt, AICS, Klagenfurt, AUSTRIA*

Abstract

Databases in general and relational database queries in particular are increasingly a topic in schools. However, teachers often lack suitable, open, high-quality, modular, interactive, and reusable database learning materials and data. Also, existing SQL editors are often too complex for students and do not provide enough helpful feedback for solving specific queries. In this workshop, we present an approach to teaching and learning SQL in an interactive and engaging way. Our approach is based on our experience from two projects in which we developed a database MOOC and the responsive web tool 'aDBenture' that allows students to solve adventures with SQL game based in different domains.

Keywords

Relational Databases, SQL, MOOC, Active Learning, Gamification

1. Introduction

The subject area of databases and thus the handling of relational databases is anchored in some curricula, at least in Austria [1]. However, teachers often find a lack of suitable, high-quality and available learning materials and sample data. For students, existing SQL editors are also often too complex and do not provide enough feedback when solving concrete tasks. We are currently working on two projects that focus on training students in databases: In the eInformatics@Austria project [2], we are developing an interactive and engaging MOOC on the topic of "databases". In the aDBenture project we are working on a web-based and responsive tool that allows learners to work with SQL in a game-based way.

The goal of the workshop is to learn in the discussion with workshop participants how databases in general and SQL in particular are currently taught in schools, and to get feedback on our interactive SQL learning environment aDBenture as well as our learning materials under development.

Participants will be able to play a prepared adventure with aDBenture and provide us with feedback on the design and their own user experiences. In the role of teacher, participants will view analyses of the game results. They can also create a small adventure themselves and test each other's adventures. The motivation of the workshop is to support the teaching and learning of relational databases in an interactive and engaging way. We would also like to find some beta testers for aDBenture.

2. aDBenture – A game-based approach to learn SQL

aDBenture can help teachers motivate their students to learn SQL - anytime, anywhere. In aDBenture, adventures can be solved by writing SQL queries. An adventure consists of several tasks that need to be solved using SQL. Each task has predefined sample queries that are used to evaluate the

Informatics in Schools. A step beyond digital education. 15th International Conference on Informatics in Schools: Situation, Evolution, and Perspectives, ISSEP 2022, Vienna, Austria, September 26–28, 2022
EMAIL: {firstname.lastname}@aau.at
ORCID: 0000-0002-6936-2750 (A. 1); 0000-0003-1616-7051 (A. 2); 0000-0002-7097-8317 (A. 3); 0000-0002-5111-2286 (A. 4)


correctness of the applied queries and to give intelligent feedback in case of errors. There are some predefined adventures in aDBenture, which are set in different domains. For example, there is a database of criminal cases to make adventures interesting for students as players. In addition, teachers can flexibly create their own adventures in the role of the author, they can reuse and adapt existing adventures and also upload their own database on which their adventures are based. Multiple relational databases are supported. In addition, teachers can also analyze the learning progress of their students.

In the role of a player, students have the opportunity to solve adventures in a browser responsible and without any installation effort. As there are stored sample queries defined behind the tasks, learners can immediately recognize the correctness of their solutions and differences from the expected result. They can choose to play 'anonymously' or make their results analyzable for their teacher as logged-in users. Two master theses are currently focusing intelligent error messages and solution recommendations for players and using aDBenture for exams.

3. A Database MOOC meets aDBenture

In the project eInformatics@Austria [2] we are working on the development of a German database-MOOC. The aim in this project is to develop reusable open and motivating learning resources on various levels of Bloom's taxonomy [3]. The special feature of our database-MOOC are animated learning videos that embed database knowledge in the context of a police station where criminal cases are solved using a database (for a Bloom level 2 example see [4]; for a Bloom level 1 example see [6]). The videos are also supplemented in the MOOC with interactive H5P elements [7]. So, the design of the database-MOOC considers proven didactic methods. Interactivity, multimodality and the opportunity to become active as a learner, play an important part in the MOOC. This is where aDBenture is used also in the database-MOOC context.

But aDBenture can be also used completely independent of the MOOC. Sometime it makes sense to use it together with learning materials available on the web via a URI. Thus, lessons from the database MOOC (made available via iMooX [5] in summer 2023) can also be integrated and used in topic-specific adventures of aDBenture. These database MOOC videos will also be available on Youtube (with English subtitles) and can then be embedded via links in aDBenture as well. In this way, SQL fundamentals can be consumed by students in the context of solving adventures.

4. References

[1] A. Kornfellner, Databases in Schools, Master's thesis (German), Johannes Kepler Universität (JKU), Linz, Austria, 2019, URL: https://epub.jku.at/obvulihs/download/pdf/4556161.

[2] eInformatics@Austria, BMBWF - Digitale und soziale Transformation in der Hochschulbildung, URL: https://www.tuwien.at/einformatics.

[3] M. Forehand, Bloom's taxonomy, Emerging perspectives on learning, teaching, and technology 41(4) (2010) 47-56.

[4] eInformatics@Austria and a Crime Case using Outer Joins, Video, 2022. URL: https://www.youtube.com/watch?v=sxymgyLgTFk&t=44s.

[5] M. Ebner, iMooX- a MOOC platform for all (universities), 7th International Conference on Electrical, Electronics and Information Engineering (2021) 1-5, URL: https://www.academia.edu/63635980/iMooX_a_MOOC_platform_for_all_universities .

[6] eInformatics@Austria and Simple Subqueries, Video, 2022. URL: https://www.youtube.com/watch?v=g3uFUQd0jcQ&t=4s.

[7] R. Singleton, A. Charlton, Creating H5P content for active learning. *Pacific Journal of Technology Enhanced Learning*, *2*(1), (2020) 13-14.

Tangible Computer Science

Martin, Cápay [1], Magdaléna, Bellayová [2]

[1] *Constantine the Philosopher University in Nitra, Tr. A. Hlinku 1, 949 01, Nitra, Slovakia*
[2] *eTeacher, o.z, Nitra, Štefánikova 57, 949 01, Nitra, Slovakia*

Abstract

Using of simple materials such as cards, strings, foam cubes, crayons, and a lot of moving around we can educate students about computer science fundamentals. This kind of popularization is called Computer science unplugged. Through well-designed activities and suitable questions, we can visualize even the abstract concepts of the main principles of the computer. The activities introduce students to underlying concepts in a way that is suitable to their age and level of comprehension, without technical details. The main purpose of the workshop is to describe a set of learning activities that are connected to each other's focused on how the data are going through the monitor, computer, and converter to the Internet. Activities were found suit-able for non-formal learning environments, but we use them even as a part of classes in formal education. Activities that look more like games than education motivate students actively create self-knowledge. Computer science unplugged is a good solution to reveal the computer science abstract even to the public. Most of our activities are new developed, one of them is taken from other source and updated to new concept of gaining the important information.

Keywords

Computer Science Unplugged. Constructivism. Engaged games.

1. Introduction

The visualization of abstract concepts through experiential teaching is an issue that could change the view on the not very popular parts of computer science. It has been shown that the elements of non-formal education are helpful [1]. Learning by playing the games is suitable for kids, students and even for adults, especially in STEM education [2]. Activities that look more like games than educating motivate students actively create a self-knowledge. Computer Science Un-plugged (CSU) is a collection of free experiential learning activities that teach Computer Science through engaging games and puzzles that use cards, string, crayons, and lots of running around. These activities rely on kinesthetic principles and involving teamwork [3]. Learning becomes more informal that is a vital part of instructional practice [4]. Manipulation with physical objects is typical for CSU.

2. Tangible activities

The workshop aims to present an interactive popularization constructivist activity, with a focus on computer science. The main purpose of our workshop is to describe a set of learning activities that connected to each other's focused on how the data are going through the monitor, computer, and converter to the Internet. The activities introduce underlying concepts in a way that is suitable to their age and level of comprehension, without technical details. The concepts are presented only using LEGO bricks, paper cards, foam cubes, wooden boxes strings, crayons, and lot of moving around (Figure 1.).

Informatics in Schools. A step beyond digital education. 15th International Conference on Informatics in Schools: Situation, Evolution, and Perspectives, ISSEP 2022, Vienna, Austria, September 26–28, 2022
EMAIL: mcapay@ukf.sk (A. 1); maggiebellayova@gmail.com (A. 2)
ORCID: 0000-0002-8352-0612 (A. 1)

Activities are effective as the first information about how computers see (store) different type of information, such as numbers or text. After that abstraction is more comprehended.

Fig. 1. Teachers aids for computer science unplugged activities

2.1. Set of learning activities

Through the well-designed activities and suitable questions, we can visualize even the abstract concepts of the main principles of the computer. We prepare activities focused on:
Communication – Activity aids: LEGO bricks. Activity is focused on communication rules. There are three players, the watcher, the builder, and the informer. Using LEGO bricks students need to build the construction. There are special rules for communication between the players.

- Binary digit - Activity is focused on explanation of binary coding of numbers. Understanding binary can lift a lot of the mystery from computers. By regrouping the foam blocks according to the paper cards with the predefined dots students create the knowledge how to convert the number from decimal to binary form.
- Binary counting - At a fundamental level the computers are just machines for flipping binary digits on and off.
- Encoding of characters – Understanding the representation of characters using coding standards. By using own coding table students need to encode the message for the next group. The receiver needs to be able to decode sender message.
- Converting from digital to analog form and vice versa – Using seven cards with number one on one side and zero on the other and tailor rubber (elastic string) we explain the transformation of the digital signal into the analog (pulse) signal.
- Furthermore, we have activities focused on searching in sorted and unsorted data.

3. References

[1] M. Cápay. Engaging games with the computer science underlying concept, in: International Conference on Interactive Collaborative Learning (ICL), 2015, pp. 975-979, doi: 10.1109/ICL.2015.7318160.
[2] M. Prensky. Digital game-based learning, 2001. New York: McGrawHill.
[3] T. Bell, I. Witten, M. Fellows. Computer Science without a computer. URL: http://csunplugged.org/
[4] S.C. Yanchar, and M. Hawkl. There's got to be a better way to do this: a qualitative investigation of informal learning among instructional designers, in: Educational Technology Research and Development, 2014.

Teaching programming concepts through gamification using the Sphero Bolt

Nicola Ottowitz[1], Claudio Ciesciutti[1] and Markus Wieser[1]

[1]*University of Klagenfurt, Universitätsstraße 65-67, 9020 Klagenfurt am Wörthersee, Austria*

Abstract

Learning programming is becoming increasingly important in schools, especially due to the steady advance of digitalization. However, teachers often lack motivating and interesting learning material. For students, learning to program is often associated with learning complicated syntax and getting a simple text output. But programming could be pretty fun and motivating, straight from the beginning. In this workshop we present an approach to teaching and learning programming in an interactive, motivating and playful way. Our approach is based on our experience in teaching programming skills to children of all ages at the Informatics Lab at the University of Klagenfurt.

Keywords

programming concepts, gamification, block-based programming, Sphero Bolt, Informatics Lab.

1. Introduction and Motivation

Teaching programming, or at least teaching the core concepts of programming, is anchored in the curriculum of the subject basic digital education in Austria[1], which is designed for students between 10 and 14. Especially in this age group, it is pretty hard to motivate the students to learn programming. For students, learning to program is often associated with learning complicated syntax and getting a simple text output. But programming could be pretty fun and motivating, straight from the beginning. The motivation could be achieved by using interesting equipment and motivating tasks. In our Informatics Lab at the University of Klagenfurt, we are teaching informatics and its key concepts to children. There we learned that it is not enough to teach things like programming correctly. It should also be motivating. And for children it is the most motivating if the first steps are easy to understand and they are getting results as soon as possible. The goal of our workshop is to support the learning and understanding of programming concepts like branches, variables, loops and functions through gamification, using the sphero bolt and its block-based programming environment.

Informatics in Schools. A step beyond digital education. 15th International Conference on Informatics in Schools: Situation, Evolution, and Perspectives, ISSEP 2022, Vienna, Austria, September 26–28, 2022

nicola.ottowitz@aau.at (N. Ottowitz); claudio.ciesciutti@aau.at (C. Ciesciutti); markus.wieser@aau.at (M. Wieser)

2. Learning programming concepts using the Sphero Bolt

Teaching Programming is a pretty complex matter, especially if you are using a text-based programming language. The students are focussing on syntax problems and the results are not really exciting at the beginning. In our Informatics Lab we are following a different approach on teaching programming, we use block-based programming at the beginnig. If you use languages like scratch, the novices could focus on the key concepts of programming, like branches, variables or loops and don't need to consider the syntax. These languages are also designed for quick results, so a program of three or four blocks could already have a visible result. The second part is the gamification. Gamification in programming courses has been recognized as a prospective technique that could improve student participation as well as impact learning in a positive way[2]. We use it in a way, so the result of the programming task is a playable game. Some of these could be programmed with a few lines of code and the students are highly motivated at the end for two reasons. First they can play a game they programmed on their own and second, they want to learn more techniques for writing new games. We found the Sphero Bolt[3] pretty useful for teaching programming concepts. Its environment is using block-based programming with optional switch to JavaScript and there are many possibilities developing motivating mini games.

References

[1] Curriculum of basic digital education,
https://www.ris.bka.gv.at/eli/bgbl/II/2022/267/20220706.
Last accessed 14.07.2022

[2] M. Venter, "Gamification in STEM programming courses: State of the art," 2020 IEEE Global Engineering Education Conference (EDUCON), 2020, pp. 859-866, doi: 10.1109/EDUCON45650.2020.9125395.

[3] Sphero Bolt,
https://sphero.com/products/sphero-bolt.
Last accessed 14.07.2022

Addressing Digital Literacy in a National Curriculum

Nataša Grgurina[1,2], Jos L. J. Tolboom[3]

[1]SLO, Amersfoort, The Netherlands
[2]University of Groningen, Groningen, The Netherlands
[3]SLO, Amersfoort, The Netherlands

Abstract
In this workshop, the participants discuss how to introduce the learning objectives of digital literacy into the K-12 curriculum. The integration of basic ICT skills, information literacy, media literacy, and computational thinking into various school subjects is of particular interest.

Keywords
Digital literacy, K-12, curriculum, integration of digital literacy

1. Extended Abstract

The introduction of digital literacy (DL) into national curricula is gaining attention worldwide [1][2]. Specifically, in the Netherlands, the entire K-12 curriculum. (i.e., standards) is about to be revised from November 2022 on and will include DL-related learning objectives.

Although there will always be room for debate, there seems to be a growing consensus about what is 'digital literacy'. In Europe, for instance, the DigComp frame-work [3] is gaining momentum [2]. So, the question arises: how do we introduce DL into the whole of the curriculum? One of the major questions in this respect is the following: is DL considered to be a separate knowledge domain or is it to be integrated into other disciplines? Or is this issue more subtle and does it require both a separate subject in its own right, as well as being integrated into existing subjects? The answer to this question will, of course, raise new questions: what parts (i.e., what learning goals) of the intended DL curriculum are to be integrated into what parts of the existing curriculum? In the current framework which is used in the Netherlands [4], there are three domains to be addressed: basic ICT skills, information literacy and media literacy.

In this workshop we will exchange ideas on how to introduce DL into a national curriculum, grades K-12, i.e., decide about the relevant learning objectives. We will present the ideas proposed in the Netherlands, and ask the workshop participants to critique them and present their ideas and good practices. By using a format for inter-national comparison, we will make differences and similarities in the way participating countries deal with the curricular challenges of the integration of DL explicit and comparable. We hope to be able to formulate several scenarios,

Informatics in Schools. A step beyond digital education. 15th International Conference on Informatics in Schools: Situation, Evolution, and Perspectives, ISSEP 2022, Vienna, Austria, September 26-28, 2022
n.grgurina@slo.nl (N. Grgurina); j.tolboom@slo.nl (J. L. J. Tolboom)
https://www.rug.nl/staff/n.grgurina (N. Grgurina); https://www.slo.nl/@1770/jos-tolboom (J. L. J. Tolboom)
0000-0003-2807-9550 (N. Grgurina); 0000-0002-8939-121 (J. L. J. Tolboom)

based on different strategies in different contexts, describing various sets of learning objectives. The results of the workshop will be shared with the committees responsible for the development of the new Dutch curricula. Besides this, we will share them with the ISSEP-community by utilizing appropriate online tools.

References

[1] S. Bocconi, A. Chioccariello, P. Kampylis, V. Dagienė, P. Wastiau, K. Engelhardt, J. Earp, M. A. Horvath, E. Jasutė, C. Malagoli, V. M.-D. V, G. Stupurienė, A. Inamorato Dos Santos, R. Cachia, N. Giannoutsou, Y. Punie, Reviewing Computational Thinking in Compulsory Education, Scientific analysis or review KJ-06-22-069-EN-N (online), Publications Office of the European Union, Luxembourg (Luxembourg), 2022. doi:10.2760/126955(online).

[2] J. Tolboom, L. van Rooyen (Eds.), Digital Literacy: Curriculum Development and Implementation in European Countries. CIDREE Yearbook 2021, SLO, Netherlands Institute for Curriculum Development, Amersfoort, 2021.

[3] R. Vuorikari, S. Kluzer, Y. Punie, DigComp 2.2, The Digital Competence framework for citizens: with new examples of knowledge, skills and attitudes, Publications Office of the European Union, https://doi.org/doi/10.2760/115376, 2022.

[4] A. Thijs, P. Fisser, M. van der Hoeven, 21e eeuw vaardigheden in het curriculum van het funderend onderwijs, SLO (nationaal expertisecentrum leerplanontwikkeling), Enschede, 2014.

Doctoral Consortium

Modeling of the System for Computational Thinking Automatic Assessment Doctoral Thesis Summary

Vaida Masiulionytė-Dagienė [1]

[1] *Vilnius University Institute of Data Science and Digital Technologies, Vilnius, LT-08412, Lithuania*

Abstract

One of the goals of computer science teaching is to develop students' computational thinking (CT) skills. However, proper assessment of CT skills still remains a challenge, as stated in various studies and observed in teaching practice. There are some tests created and tools developed for CT assessment, but all of them focus on one or two concepts of CT. There is still no unified system that could address all the main aspects of CT in CT assessment, especially when talking about automated CT assessment. The main aim of this thesis is to model a system for automated assessment of computation-al thinking skills based on learning analytics. So far, a literature review on the topic of computational thinking has been carried out and the main instruments for assessing computational thinking have been identified. Further work is being done on the analysis of the literature on learning analytics.

Keywords

Computational Thinking Assessment, Learning Analytics, Automatic Assessment, Assessment Instruments.

1. Problem and Planned Objectives

There are some tests created and tools developed for CT assessment, but all of them focus on one or two concepts of CT. There is still no unified system that could address all the main aspects of CT in CT assessment, especially when talking about automated CT assessment. The system of CT assessment will focus on the high school students, also it could be later on used in higher education or in IT sector to test computational thinking level of the employees. Because now in IT sector there is quite a problem to find skillful employees, and some companies are using IQ test or some other tests to test the logical thinking skills of future employees.

Planned objectives:

1. To analyze the existing methodologies and issues in the assessment of computational thinking;
2. To analyze and systematize learning analytics methods suitable for assessing computational thinking in e-learning systems;
3. To develop a model for an automated assessment system for computational thinking based on the analytics of learning data in problem solving;
4. Empirically evaluate the modelled assessment system for computational thinking.

2. Achieved results

Informatics in Schools. A step beyond digital education. 15th International Conference on Informatics in Schools: Situation, Evolution, and Perspectives, ISSEP 2022, Vienna, Austria, September 26–28, 2022
EMAIL: vaida.masiulionyte-dagiene@mif.vu.lt
ORCID: 0000-0002-7755-0428;

Literature review on Computational Thinking assessment tools have been done. As the results of the review the most popular method of assessing computational thinking still remain the various versions of tests. One of the most commonly used tools for assessing computational thinking without linking teaching and assessment to programming is the CTt [2] or Bebras Tasks. CT assessment review results also indicate that: more CT assessment tools are needed for high school, college students, and teacher professional development programs, more reliability and validity evidence needs to be collected and reported in future studies [4].

3. Related work

The most relevant to my research work is the in Zoombinis game developed assessment system for computational thinking. Zoombinis [1] is an award-winning learning game that was designed in the 1990s and re-released for current platforms. It is not only used for learning CT but also in recent years for CT assessment. In Zoombinis all the players actions are logged and then analyzed for the purpose of learning or the assessment. CT concepts that are assessed in Zoombinis [3]: problem decomposition, pattern recognition, abstraction, algorithm design.

4. References

[1] Asbell-Clarke, J., Rowe, E., Almeda, V., Edwards, T., Bardar, E., Gasca, S., Baker, R. S., & Scruggs, R. The development of students' computational thinking practices in elementary- and middle-school classes using the learning game, Zoombinis. Computers in Human Behavior (2021), 115. doi:10.1016/j.chb.2020.106587

[2] Román-González, M., Pérez-González, J.-C., & Jiménez-Fernández, C. Which cognitive abilities underlie computational thinking? Criterion validity of the Computational Thinking Test. Computers in Human Behavior (2017) 72, 678–691. doi:10.1016/j.chb.2016.08.047

[3] Rowe E., Almeda V., Asbell-Clarke J., Scruggs R., Baker R., Bardar E., Gasca S. Assessing implicit computational thinking in Zoombinis puzzle gameplay, Computers in Human Behavior (2021), Volume 120. doi:10.1016/j.chb.2021.106707.

[4] Tang, X., Yin, Y., Lin, Q., Hadad, R., & Zhai, X. Assessing computational thinking: A systematic review of empirical studies. Computers and Education (2020), 148. doi:10.1016/j.compedu.2019.103798

Integrated Digital Education: Computational Thinking for Everyone

Corinna Hörmann[1]

[1]Johannes Kepler University Linz, Altenbergerstraße 69, 4040 Linz, Austria

Abstract

As long ago as 1988, Austria introduced the subject "Computer Science" in grade 9. Quite a long time there was solely this one year of mandatory IT-education during school career. When Austria implemented the new mandatory subject "Digital Education" in September 2018 for all students in lower secondary education, computer science education finally found its way into additional grades. The curriculum covers digital competences, media competences, as well as civic education. Schools could decide if they offer stand-alone subjects or if they implement the curriculum in an integrative way in several other subjects. Finally, in the school year 2022/23, "Digital Education" will be installed as compulsory subject in regular Austrian timetables. Due to this change the main goal of the whole PhD thesis has to be rethought.

Keywords

Digital Education, 21st Century Skills, Computational Thinking, STEM

1. Research Questions

(1) Are there any advantages in the integrative implementation of "Digital Education" and if so which?
(2) What teaching materials do teachers at lower secondary schools need to better integrate "Digital Education" in their lessons?

2. Methodology

This PhD thesis is designed as mixed-methods approach, so both quantitative and qualitative data will be analyzed. The start formed a pilot study in order to gain better understanding if the initial implementation of the subject "Digital Education" caused any problems (n = 27). The next step formed a survey answered by 117 teachers approximately two years after the implementation of the subject. Both, the quantitative and qualitative data was investigated. The collected qualitative data was interpreted using the software MAXQDA by implementing a content analysis. During the emergency remote teaching phase in the course of the pandemic a

Informatics in Schools. A step beyond digital education. 15th International Conference on Informatics in Schools: Situation, Evolution, and Perspectives, ISSEP 2022, Vienna, Austria, September 26–28, 2022
corinna.hoermann@jku.at (C. Hörmann)
0000-0002-4770-6217 (C. Hörmann)

survey was conducted that was sent to all 133 local Upper-Austrian public schools, whereas 121 teachers were willing to complete the questionnaire.

3. Results

For the pilot study we designed a survey concerning experiences and challenges in the implementation of the subject. The majority of the sample group supported the idea of teaching "Digital Education" in an integrated way in their own subject. Moreover, the part that worried teachers most when new structures were introduced, is if they still have enough time to fulfill their own curriculum. In a survey in 2020 it was investigated how schools implement the new curriculum – stand-alone or integrated in other subjects. 26% claimed that they introduced a stand-alone subject, 21% said they integrate the curriculum in other subjects, 45% picked "mixed", and 9 (8%) chose "other". The qualitative data suggested that there are still people who fight with the new subject and that there are various problems that have to be tackled. Still, the attitude towards the curriculum was very positive. During the SARS-COVID-19 pandemic another survey was conducted. The following research questions lay out the basis for the evaluation of the data: (1) Is "Digital Education" still taught during emergency remote teaching? (2) If so, did students solely "automatically" or incidentally gather knowledge, as they had to use digital devices or were specific topics of the curriculum implemented? The results suggested that nearly every school continued with the lessons of "Digital Education" and teachers tried to impart numerous 21st century skills.

4. Related Work

Since the implementation of the new subject "Digital Education" in Austrian lower secondary schools was only four years ago with the COVID-19 pandemic hitting hard, it is difficult to find substantial research concerning the topic. Nevertheless, Christian Swertz (2018) from the university of Vienna conducted a survey concerning the pilot testing phase in 2017/2018. Results show that major challenges were the missing content and the lack of competences for teaching "Digital Education". This affected cross-subject integration as well as stand-alone lessons [1]. Currently, there are no long-term studies concerning digital literacy skills of students in Austria. In the Netherlands, on the other hand, Lazonder et al. [2] report on a three-year study that investigated the development of children's digital literacy skills. The study suggested that there is a linear increase in all skills, but nonetheless natural development of digital literacy skills is slow [2].

References

[1] C. Swertz, Digitale Grundbildung im Pilotversuch – Beobachtungen einer entstehenden Praxis, Medienimpulse 3 (2018).

[2] A. W. Lazonder, A. Walraven, H. Gijlers, N. Janssen, Longitudinal assessment of digital literacy in children: Findings from a large dutch single-school study, Computers & Education 143 (2020).

Enhancing Robotics in Early Secondary School

Alexandra Maximova[1]

[1]*Department of Computer Science, ETH Zürich*
Universitätstrasse 6, 8092 Zürich, Switzerland

Abstract

Computational thinking is a set of skills and processes that enable students to tackle complex problems, and which should therefore be fostered in school. Educational robotics initiatives are emerging all over the world as one motivating way develop these skills in classrooms. We would like to investigate whether educational robotics improve computational thinking skills in 12 years old students and how a corresponding curriculum and framework should be designed.

Keywords

educational robotics, computational thinking, novice programmers

1. Introduction

There are many efforts around the world to introduce robotics into classrooms, e.g., the *First Lego League*, the *World Robot Olympiad*, or the *Roberta*, *Robbo*, or *Robofest* initiatives and platforms. The corresponding hardware varies from moving robots like Lego models, Edison, or Thymio to programmable boards such as Calliope, Microbit, or the Oxocard.

Over the last 15 years our group developed a K-12 spiral curriculum to teach CS and programming. The curriculum starts with the Bluebot in Kindergarten and continues with Logo and Turtle Graphics at primary school. In secondary school students use the programming language Python and the TigerJython IDE (https://tigerjython.ch) or WebTigerJython (https://webtigerjython.ethz.ch) for advanced Turtle Graphics and robotics. Python was chosen due to its simple syntax and the possibility not to use or even mention advanced features (for example, object-oriented code) until the students feel confident with basic concepts. For this reason, we will stick with Python to nicely integrate our robotics course into this curriculum. We believe that teaching concepts is more important than teaching one concrete technology stack, which may become obsolete in 5–10 years. The TigerJython IDE was developed to have a sleek design without any features that could confuse novice programmers rather than help them. It offers improved error messages compared to standard Python, and the 'repeat' loop construct that allows to teach loops before students need to know about variables or conditionals.

We want to focus on supporting cheap hardware such as the educational boards Calliope Mini and Microbit that can be easily transformed in car-like robots. Thus the focus is on programming pre-built robots – not on engineering the robots themselves.

Informatics in Schools. A step beyond digital education. 15th International Conference on Informatics in Schools: Situation, Evolution, and Perspectives, ISSEP 2022, Vienna, Austria, September 26–28, 2022
alexandra.maximova@inf.ethz.ch (A. Maximova)
0000-0003-2598-0810 (A. Maximova)

2. Research Questions and Methodology

The overall research question is whether and how robotics can be used to foster computational thinking skills and competences in early secondary school – leading to several subquestions and subtasks that will be tackled along the way.

1. Develop a robotics curriculum for early secondary school combining programming concepts such as loops and modular design with a motivating application domain, i.e., robotics.
2. Design a framework to individually support novice programmers while they are practicing programming and robotics skills.
3. Adapt a computational thinking test for our needs. This step is crucial in order to be able to measure computational thinking gains in quantitative experiments on the field.

The main research question requires a quantitative study measuring the computational thinking skills before and after our intervention. The necessary teaching material will be developed using educational design research [1]. The curriculum will be thus refined during several pilot studies, allowing us to gain insights from practitioners and discover new knowledge.

3. Initial Results

At the time of writing, we developed a first version of a robotics curriculum for 12 years old students. During 6 lessons of 2.5 hours each, students program the Calliope Mini board and its car-like extension Calli:bot in Python in the Tiger Jython IDE. No previous programming knowledge is required, since the students learn basic programming concepts (loops, functions, variables, and conditionals) during the curriculum using motivating robotics tasks. Students learn how to program songs, how RGB colors work, what pixels are, how 2D-coordinates work, how to use sensors and buttons, and how to move the robot.

The first pilot study with two groups of 12 students each is running at the time of writing. At the end of the pilot study, student motivation will be collected using anonymous questionnaires. This curriculum will be refined and tested with another group of students during summer.

4. Related Work

Conducting educational research [1] defines the methodology we are following to develop the curriculum. We follow a constructionist approach as introduced by Papert [2]. Students have the possibility to experiment and *get things to work*. We also let us inspire by similar past projects, such as the hardware and software robotics framework developed by Assaf [3].

References

[1] S. McKenney, T. C. Reeves, Conducting Educational Design Research, Routledge, 2018.
[2] S. Papert, Mindstorms: children, computers and powerful ideas, Basic Books (1980).
[3] D. Assaf, R. Pfeifer, Embedit – an open robotic kit for education, in: International Conference on Research and Education in Robotics, Springer, 2011, pp. 29–39.

Pre-service Teachers' Experiences in Learning Programming for Basic Education

Megumi Iwata[1]

[1]University of Oulu, Pentti Kaiteran katu 1, 90570 Oulu, Finland

Abstract

Many countries have started introducing programming in the basic education. In the basic education, goal is not mastering the programming language but develop thinking skills and practices to understand the world where computing is embedded, such as computational thinking. Educating teachers is a key factor to enhance integration of programming and computational thinking into education. This thesis consisting of three sub-studies aims to explore pre-service teachers learning experience of programming. By applying theoretical frameworks of integrating digital technology into education, the first sub-study focuses on pre-service teachers' experience of designing learning materials for micro:bit programming in an introductory programming course. The second sub-study adds a scope of connecting learning programming with work contexts by investigating engineers' experience in using the learning materials. Based on these findings, the third sub-study aims to design learning activities in the introductory programming course.

Keywords

Teacher education, computational thinking, pedagogy,

1. Description

In Finland and many other countries, programming has been introduced in the basic education. In many cases, the purpose of programming in the basic education contexts is not mastering programming languages, but developing fundamental thinking skills and practice for programming, which are rooted in computing. Computational thinking (CT) is defined as "the thought processes involved in formulating problems and their solutions so that the solutions are represented in a form that can be effectively carried out by an information-processing agent" [Cuny, Snyder, and Wing, 2010, as cited in [1]]. Previous studies have identified needs to develop teacher education to enhance integration of programming and CT in education [2].

My doctoral thesis aims to explore learning experience of pre-service teachers (PSTs) who are novice in programming. Three sub-studies are conducted to understand the current practice of teacher education for integrating programming into basic education. In the first sub-study, I investigate PSTs' experience in a collaborative problem-solving project in an introductory programming course where they design the learning materials for micro:bit programming. I focus on their understanding of CT, as well as experiences in learning programming and

Informatics in Schools. A step beyond digital education. 15th International Conference on Informatics in Schools: Situation, Evolution, and Perspectives, ISSEP 2022, Vienna, Austria, September 26–28, 2022

megumi.iwata@oulu.fi (M. Iwata)

0000-0001-7944-5157 (M. Iwata)

practices of integrating programming into education. In the second sub-study I include a scope of learning programming associated with work. The Finnish National Core Curriculum for basic education instructs that education should be organized to allow the pupils to understand the significance of competence acquired in school for their future careers (Finnish National Board of Education, 2014). I explore how employees at a relevant company see the learning materials designed by the PSTs. In the third sub-study, I develop learning activities in the introductory programming course in teacher education based on the findings from the two sub-studies and examine how the learning activities enhance PSTs' learning of programming.

To investigate programming in education, the previous study [3] applies the frameworks for integration of digital technologies into education. Technology, pedagogy and content knowledge (TPACK) by [4] is one of the widely-used frameworks to demonstrate teacher's competence to integrate technologies into education.

My doctoral thesis is conducted as a case study. An introductory programming course for PSTs is the main context of the thesis. The first data collection took place in the course from March to May 2021. The data collected are 1) the recordings of the PSTs work in the remote sessions, and 2) the survey about the PSTs' experience collected after the course. The second data collection was conducted at the workshop at a multinational telecommunication company in December 2021. Participants are eight employees at the company. The data collected are 1) the observations during the workshop with the employees, and 2) the focus group interview after the workshop. Third data collection will take place in the spring 2023.

Preliminary results from the first data include: 1) PSTs' understanding on CT differs. 2) Technological and pedagogical knowledge is practiced in the project. 3) The PSTs demonstrate the challenges during the studies, such as lack of knowledge and experience, and unfamiliar ways of working (tinkering) in the ill-structured problem solving. 4) To overcome the challenges, different approaches are used, such as starting with suitable levels, utilize resources and working with others.

References

[1] J. M. Wing, Research notebook: Computational thinking–what and why?, ???? URL: https://www.cs.cmu.edu/link/research-notebook-computational-thinking-what-and-why.

[2] A. Yadav, S. Gretter, J. Good, T. McLean, Computational thinking in teacher education, in: Emerging research, practice, and policy on computational thinking, Springer, 2017, pp. 205–220.

[3] S. L. Mason, P. J. Rich, Preparing elementary school teachers to teach computing, coding, and computational thinking, Contemporary Issues in Technology and Teacher Education 19 (2019) 790–824.

[4] P. Mishra, M. J. Koehler, Technological pedagogical content knowledge: A framework for teacher knowledge, Teachers college record 108 (2006) 1017–1054.

Designing a Computational Thinking education framework

Martina Landman[1]

[1]TU Wien, Karlsplatz 13, 1040 Vienna, Austria

Abstract

The term computational thinking has already made it into many curricula, including the 2019 curriculum of Digital Basic Education in Austria. However, the importance of teaching computer science content and Computational Thinking (CT) in school classrooms is not reserved for the classroom alone, but can happen in many areas. The thesis will focus on how to design and develop CT tasks for use both in a school context as classroom tasks and as an outreach activity from universities to schools in form of workshops. With the help of Design Based Research, existing workshops for school classes of the TU Wien Informatics eduLAB will be evaluated and perfected, in order to find patterns in the later course, which provide effective tasks for school classes of lower secondary education level. From the patterns found, a model will be generated and tested that can be used to create CT and CS tasks from different contexts to include them into school classes and university outreach activities. Evaluation forms and qualitative interview analysis provide important insights into the needs of the teachers and students involved in the activities to fulfill the curriculum. The competencies that students fulfill as well as the skills that teachers need are examined. The results should also provide more teaching material for school teachers and a reusable template and framework for developing outreach activities in the area of computer science (CS) education and computational thinking for universities.

Keywords

computational thinking, secondary school, teaching computer science, designing tasks, education model.

1. Introduction

The importance of bringing computational thinking into people's minds has been undisputed since Janette Wing's [1] famous and often quoted article in 2006, where she said "Computational thinking is a fundamental skill for everyone, not just for computer scientists." In the last decade there where many definitions and descriptions on CT, for example the definition of Selby and Woollard [2] in 2013 that give us five thought processes: abstraction, algorithmic thinking, generalization, decomposition and evaluation.

Teachers and also lecturers can get confused on what to focus when they are trying to develop CT tasks for their students. Frameworks and models on how to create CT and CS tasks can help. For that, a deep research of existing frameworks, models and tasks is needed as well as an analysis and evaluation of existing workshops of TU Wien Informatics eduLAB, based on the empirical data.

Informatics in Schools. A step beyond digital education. 15th International Conference on Informatics in Schools: Situation, Evolution, and Perspectives, ISSEP 2022, Vienna, Austria, September 26–28, 2022

martina.landman@tuwien.ac.at (M. Landman)

2. Research Questions

One of the main research questions could be ”How to create good Computational Thinking tasks?”. This question is followed by a lot more: ”What competencies do students and teachers need?”, ”What frameworks, that exist, are effective?”, ”How to create a model for the creation of CT tasks?”, ”How can the future model be implemented in outreach activities of university as well as in school classes?”

3. Methodology in each step of the thesis

1. Literature study: Analyze existing definitions, frameworks, models and teaching material of CT to find a consent and an idea, how to build it together to a template.
2. Design Based Research, Observation: Analyze, evaluate and improve the existing workshops and tasks of TU Wien Informatics eduLAB with the things learned above.
3. Qualitative interviews and Evaluation forms: Get an idea what students and teachers need with evaluation forms after workshops and interviews.
4. Practical work: New CT tasks are created with newly gained knowledge. Also a framework for how to create those tasks is built.
5. Repeat steps 2-4 until the result is satisfying.
6. Testing the results with teachers in schools.

4. Related Works

There are people who already tried to formulate frameworks and models for developing CT. Romero and Lepage [3] describe how to develop CT through creative programming tasks. Another article that is linked to the topic of this work is the article from Palts and Pedaste [4] in 2020. They describe a Model in 3 stages with 10 competencies of CT. In 2014 Curzon et al. [5] did a 4 stage framework how to design CT lessons and an overview of which actions of students can be assigned to which of the five abilities to think according to Selby and Woollard [2].

References

[1] J. M. Wing, Computational thinking, Communications of the ACM 49 (2006) 33–35. doi:10.1145/1118178.1118215.

[2] C. Selby, J. Woollard, Computational thinking: the developing definition, University of Southampton (E-prints), 2013. URL: https://eprints.soton.ac.uk/356481.

[3] M. Romero, A. Lepage, B. Lille, Computational thinking development through creative programming in higher education, International Journal of Educational Technology in Higher Education 14 (2017) 1–15. doi:10.1186/s41239-017-0080-z.

[4] M. Palts, Tauno; Pedaste, A model for developing computational thinking skills, Informatics in Education - An International Journal 19 (2020) 113–128.

[5] P. Curzon, M. Dorling, T. Ng, C. Selby, J. Woollard, Developing computational thinking in the classroom: a framework (2014).

Research plan: A Recommender System for Autonomous Programming at Schools

Kevin Tang and Jacqueline Staub

University of Trier, Universitätsring 15, 54296 Trier, Germany

Abstract

Online learning environments for programming typically report only syntactic errors and not errors at the logical or semantic level. Incorrectly solved exercises can be caused by a number of reasons, ranging from misunderstanding its description to mistakes made due to misconceptions. The latter could prove detrimental to learning more advanced concepts and therefore should be corrected as early as possible. For this reason, we propose an extension of an existing learning environment for schools that places a special emphasis on providing students with meaningful feedback on their mistakes. This extension is intended to be implemented as a recommender system that offers learning opportunities (e.g., exercises) to students based on their current level of understanding as well as allowing them to progress independently without needing direct assistance from a teacher.

Keywords

Programming education, K-6 computer science education, recommender system, Turtle graphics, Logo,

1. Basic Information

Date of PhD start: 1st March 2022 at the University of Trier under the supervision of Jun.-Prof. Dr. Jacqueline Staub in the field of educational computer science.

2. Research Question

The overall research question consists of two parts. (1) Whether and how we can measure a student's misconception of a topic and (2) how we can use the data collected from students to improve their learning. To this end, we will implement a recommender system that allows us to analyze the collected data, including the mistakes they made, and suggest an adequate follow-up that matches their current level of understanding.

3. Methodologies

XlogoOnline is a block-based online programming environment for K-6 education developed at ETH Zurich and the University of Trier. This environment is currently used worldwide, which allows us to collect a large amount of data (anonymously) to conduct a quantitative study. We

Informatics in Schools. A step beyond digital education. 15th International Conference on Informatics in Schools: Situation, Evolution, and Perspectives, ISSEP 2022, Vienna, Austria, September 26–28, 2022

tang@uni-trier.de (K. Tang); staub@uni-trier.de (Jun.-Prof. Dr. J. Staub)

also plan to conduct on-site studies in German and Swiss schools to observe, on the one hand, what misconceptions students might have regarding a predefined set of topics, and whether our recommender system impacts student learning.

4. Achieved Results

We haven't started yet, so no results so far.

5. Related works

A literature review is still ongoing. To our current knowledge, there are several works in the research community on recommender systems in education. We have identified a few of them that take an approach similar to the one we could use in our research, namely, making recommenders based on the students learning progress [1, 2, 3]. We have also identified some literature on novice programmers' misconceptions [4, 5, 6].

References

[1] I. Brigui-Chtioui, P. Caillou, E. Negre, Intelligent digital learning: Agent-based recommender system, in: Proceedings of the 9th International Conference on Machine Learning and Computing, ICMLC 2017, Association for Computing Machinery, New York, NY, USA, 2017, p. 71–76. doi:10.1145/3055635.3056592.

[2] Q.-V. Dang, Implementing an individualized recommendation system using latent semantic analysis, in: Proceedings of the 6th International Conference on Information and Education Technology, ICIET '18, Association for Computing Machinery, New York, NY, USA, 2018, p. 239–243. doi:10.1145/3178158.3178163.

[3] P. Montuschi, F. Lamberti, V. Gatteschi, C. G. Demartini, A semantic recommender system for adaptive learning, IT Professional 17 (2015) 50–58.

[4] J. B. du Boulay, Some difficulties of learning to program, Journal of Educational Computing Research 2 (1986) 57–73.

[5] T. Kohn, Variable evaluation: An exploration of novice programmers' understanding and common misconceptions, in: Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education, SIGCSE '17, Association for Computing Machinery, New York, NY, USA, 2017, p. 345–350. doi:10.1145/3017680.3017724.

[6] P. Bayman, R. E. Mayer, A diagnosis of beginning programmers' misconceptions of basic programming statements, Commun. ACM 26 (1983) 677–679. doi:10.1145/358172.358408.

Development of Informatics Competencies among (prospective) Primary School Teachers

Christin Nenner

TU Dresden, 01062 Dresden, Germany

Abstract

The aim of the dissertation is to investigate the impact of developed teaching-learning courses for building up informatics-specific professional and teaching competencies among (prospective) primary school teachers. For this purpose, instruments will be developed to measure these competencies as well as the participants' self-assessment.

Keywords

Informatics education, Informatics competencies, Primary school, Teacher education

1. Related work

The 2017 CECE report called for consistent informatics education for all students, preferably from primary school, and outlined the state of integration [1]. In 2022, Germany's state was updated [2]. Primary school teachers' beliefs about informatics were investigated by Best [3].

Thematically based on the competency recommendations for the primary level of the German Informatics Society [4], informatics-specific introductory and in-depth courses were developed. **Introductory courses** [5] (half a day) have been offered since the summer semester of 2020 with the aim of providing a basic introduction to informatics content, so that the diversity of informatics is demonstrated. The **in-depth course** (one semester) has been offered since the summer semester of 2021, with a focus on teaching informatics content to primary school children. Participants develop their first informatics-specific lessons [6] and test them together with their fellow students to gain initial experience in teaching informatics-specific skills.

2. Research questions and methodology

How can (prospective) primary school teachers acquire informatics expertise within the framework of teaching-learning courses, so that:

- *RQ 1*: they can master them?
- *RQ 2*: they can teach them to children?
- *RQ 3*: they perceive themselves as having informatics-specific self-efficacy?

In order to investigate the courses, instruments are developed.

Informatics in Schools. A step beyond digital education. 15th International Conference on Informatics in Schools: Situation, Evolution, and Perspectives, ISSEP 2022, Vienna, Austria, September 26–28, 2022

0000-0002-5230-4343 (C. Nenner)

- **Self-assessment of informatics-specific professional and teaching competencies** *(RQ 3)*: The courses are examined using a pre-post test design for self-assessment of basic informatics competencies as well as teaching competencies on a 5-point Likert scale [5].
- **Informatics-specific teaching competencies** *(RQ 2)*: Within the in-depth course, the informatics-specific lessons developed by the participants themselves and tested with their fellow students are assessed by all participants and the lecturer using reflection sheets (version 1). The focus is on the didactic reduction, the fit to the target group and the action orientation. After the joint reflection, the participants revise their lesson plans (version 2). Versions 1 and 2 are compared in order to examine the teaching competencies.
- **Informatics-specific professional competencies** *(RQ 1)*: At the end of the in-depth course, the acquired informatics competencies are assessed by individually solving informatics-specific tasks (automatically evaluable test items) [7] closely related to the competencies facilitated in the course.

3. Results

Intermediate states of the instruments currently being developed are already being used to examine the courses [5, 7]. The application (pre and post) of the informatics-specific self-assessment items on competencies and teaching, in the context of an introductory course for primary school teacher education students in the summer semester of 2020, showed a significant increase in comparison to their self-assessment before their participation in the course [5].

References

[1] The Committee on European Computing Education (CECE), Informatics Education in Europe: Are We All In The Same Boat?, 2017.

[2] C. Nenner, N. Bergner, Informatics Education in German Primary School Curricula, in: A. Bollin, G. Futschek (Eds.), Informatics in Schools. A step beyond digital education., Springer International Publishing [in press], 2022.

[3] A. Best, Primary school teachers' beliefs on computer science as a discipline and as a school subject, in: Proceedings of the 15th Workshop on Primary and Secondary Computing Education, ACM, Virtual Event Germany, 2020, pp. 1–8.

[4] A. Best, C. Borowski, K. Büttner, R. Freudenberg, M. Fricke, K. Haselmeier, H. Herper, V. Hinz, L. Humbert, D. Müller, A. Schwill, M. Thomas, Kompetenzen für informatische Bildung im Primarbereich, 2019.

[5] C. Nenner, G. Damnik, N. Bergner, Integration informatischer Bildung ins Grundschullehramtsstudium, in: L. Humbert (Ed.), INFOS 2021 – 19. GI-Fachtagung Informatik und Schule, Gesellschaft für Informatik, Bonn, 2021, pp. 103–112.

[6] I. Diethelm, The Model of Educational Reconstruction for Computer Science as a Framework for Planning and Evaluating Lessons, 2015.

[7] D. Baberowski, C. Nenner, N. Bergner, Bereit für die Zukunft - Informatische Grundkompetenzen für alle Lehrkräfte, in: S. Ganguin, H. Tiemann, C. Glück (Eds.), Digitalisierung in der Lehrer:innenbildung, Springer Nature [in press], 2022.

Computational Thinking in Mathematics Education

Kristin Parve [1] and Mart Laanpere [1]

[1] *Tallinn University, Narva Road 25, Tallinn, Estonia, 10120*

Abstract

Computational thinking (CT) is a fast-expanding area of educational research since 2006. Although it is described as a mandatory part of future school curriculum in many countries, there is a list of challenges to overcome when implementing CT in schools. The aim of this PhD research is to design and validate a prototype of a pedagogical design model that can be used to integrate teaching and learning of CT in mathematics curricula in lower and upper secondary education.

Keywords

Computational thinking, mathematics education, STEAM education, design science, pedagogical design model.

1. Background

Computational Thinking has been an important part of educational innovation over the last years and already more than 20 countries in Europe integrate programming or computational thinking in their curricula [1]. The main motive to include CT into mathematics and science education is the growth of computational counterparts in related scientific. Including CT ideas to those subjects brings school education more in line with current professional practices in those fields [8]. CT and mathematical thinking are closely related, sharing similar traits like problem solving, modeling, analysing and interpreting data [6]. On the other hand, linking CT and mathematics could also have a more practical reason. A study concentrating on ICT education in Estonia [4] asserted that teaching digital skills is inconsistent all over the country, depending too much on the convenience of the teachers. It is alarming that only less than half of the Estonian basic schools have informatics or other similar courses held as a separate course to teach computing. For example, Bocconi, Chioccariello and Earp [2] argued that CT and programming have to be a compulsory part of the curriculum because teaching them as elective courses can result in limited local uptake and actual student participation. Finland and Sweden have already implemented teaching and learning programming and other CT related skills in their school curricula, mostly integrating it into mathematics courses, but also in crafts, technology, science and other subjects. Therefore, implementing teaching CT skills in mathematics is on the one hand a logical solution based on the similarities of these two fields. On the other hand, one may argue that it can be also driven by practical causes, including innovation and increase of relevance in mathematics as one of the most conservative subject areas in school.

2. Aims and Objectives

The aim of this research is to explore the possibilities of integrating CT ideas into mathematics education to design and validate a prototype of a pedagogical design model that can be used to integrate teaching and learning of CT in mathematics curricula and pedagogical strategies in lower and upper secondary education. To achieve this goal, three research questions were formulated. **RQ1**: What are the main barriers for integrating computational thinking into mathematics education and how to reduce

Informatics in Schools. A step beyond digital education. 15th International Conference on Informatics in Schools: Situation, Evolution, and Perspectives, ISSEP 2022, Vienna, Austria, September 26–28, 2022
EMAIL: kristin.parve@tlu.ee (K. Parve); mart.laanpere@tlu.ee (M. Laanpere)
ORCID: 0000-0002-8683-7017 (K. Parve); 0000-0002-9853-9965 (M. Laanpere)

them? **RQ2**: What are the effective pedagogical design principles and techniques for integrating computational thinking with mathematics education and how to prove the effectiveness of such intervention? **RQ3**: How to model the pedagogical design and implementation of embedded and evidence-centered assessment of CT in integrated STEAM learning projects?

The whole study will be divided into three consecutive cycles. First cycle (2021-2022) of the study is set up to understand the environment and context of the study, second cycle (2022-2023) will focus on the initial research-based design of the pedagogical design and the third cycle (2023-2024) of the study will focus on improvement and validation of the pedagogical design model.

3. Research Design and Methods

The study will combine a Design-Based Research approach that is common among educational researchers, and Design Science research methodology that has originated from the field of information systems research. Design-Based Research was introduced as a way of extending existing methods and linking theory and practice in educational research [5]. Its aim is to make educational research more relevant for classroom practices [5] through its iterative research [7]. Design Science, on the other hand, is a research paradigm that is meant to improve information systems [3]. It is seen as a process where *'designer answers questions relevant to human problems via the creation of innovative artifacts, thereby contributing new knowledge to the body of scientific evidence'* [3].

4. Achievements

The first conference paper called 'Symbiotic approach to Mathematical and Computational Thinking' has been accepted and presented in IFIP World Conference of Computers in Education in Hiroshima (August 23, 2022).

5. Acknowledgements

This research has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 856954, a Twinning project SEIS: Scaling up Educational Innovation in Schools.

6. References

[1] A. Balanskat, K. Engelhart, and A. Ferrari, The Integration of Computational Thinking (CT) across school curricula in Europe. European Schoolnet Perspective 2017, 2017.

[2] S. Bocconi, A. Chioccariello, and J. Earp, The Nordic approach to introducing Computational Thinking and programming in compulsory education. Report prepared for the Nordic@ BETT2018 Steering Group, 397-400, 2018.

[3] A. Hevner, and S. Chatterjee, Design science research in information systems. In Design research in information systems (pp. 9-22). Springer, Boston, MA, 2010.

[4] C. Leppik, H.-S. Haaristo, and E. Mägi, IKT-haridus: digioskuste õpetamine, hoiakud ja võimalused üldhariduskoolis ja lasteaias. Tallinn: Poliitikauuringute Keskus Praxis, 2017.

[5] P. Reimann, Design-based research. In Methodological choice and design (pp. 37-50). Springer, Dordrecht, 2011.

[6] C. Sneider, C. Stephenson, B. Schafer, and L. Flick, Exploring the science framework and NGSS: Computational thinking in the science classroom. Science Scope, 38(3), 10, 2014.

[7] F. Wang, and M. J. Hannafin, Design-based research and technology-enhanced learning environments. Educational technology research and development, 53(4), 5-23, 2005.

[8] D. Weintrop, E. Beheshti, M. Horn, K. Orton, K. Jona, L. Trouille, and U. Wilesnky, Defining computational thinking for mathematics and science classrooms. Journal of Science Education and Technology, 25(1), 127-147, 2016.

Training Computational Thinking: exploring approaches supported by Neuroscience

Cristiana Araújo[1]

[1]Centro ALGORITMI, University of Minho, Portugal

Abstract

Students present deficits in problem solving skills, reduced abstraction and logical reasoning skills. Several studies claim that Computational Thinking (CT) can help mitigate this problem, if acquired and trained from an early age. For training to be more effective, it is necessary to understand how the brain learns and the best way to promote learning. This Ph.D. work aims to use Neuroscience approaches and techniques to create Learning Resources that promote the training of CT. A web platform that will allow teachers to train their students in CT – Computational Thinking 4 All – is the final output of this Ph.D. project. This Ph.D. work is expected to contribute to the education of the 21st Century Citizen with a clearer understanding of the process of training the mind to reason logically and strategically, leading to the acquisition of CT.

Keywords

Computational Thinking, Computer Programming Education, Neuroscience

1. Research Questions

It is known that Higher Education Students, even in graduation programs where programming is an essential skill (such as Computer Science and Engineering) have a hard time learning to program [1, 2]. The support for this thesis is the belief that the key to overcome the computer programming education difficulties is to train Computational Thinking from an early age. To be able to train Computational Thinking (CT) and to teach Computer Programming effectively, it is important to research the cognitive neuroscience field to learn how the human brain works in terms of reasoning and to identify factors like motivation, attention, emotions that will impact on the knowledge acquisition process. Focusing in the K0-K12 universe of students, this Ph.D. work aims to help solving the problem of students' failure in learning Computer Programming, believing that learning to think computationally is one of the keys. The main research questions are:

RQ1. How to help students acquiring Computational Thinking, taking into account their background and profile?

RQ2. How to design Neuroscience-informed strategies to foster Computational Thinking training?

To answer these research questions the methodology adopted in this Ph.D. project is the Design Science Research Methodology (DSR) [3], composed of six steps: Identification of Problem and Motivation, Definition objectives of a solution, Design and Development, Demonstration, Evaluation, Communication. The answer to RQ1 will bring light on pedagogical approaches and on Learning Resources (LR) that are effective to train CT skills with different students of different ages and different socio-cultural profiles. To achieve that a theoretical and an experimental work will be done. The outcomes of this first step

Informatics in Schools. A step beyond digital education. 15th International Conference on Informatics in Schools: Situation, Evolution, and Perspectives, ISSEP 2022, Vienna, Austria, September 26–28, 2022

decristianaaraujo@hotmail.com (C. Araújo)

https://epl.di.uminho.pt/~cristiana.araujo/ (C. Araújo)

0000-0002-9656-3304 (C. Araújo)

will be upgraded and improved taking into consideration the knowledge captured from deeper readings of the Neuroscience literature. And again more experimental work will be conducted to support the proposals.

2. State of the Art

Within the scope of this Ph.D. a research work was carried out based on a literature review in three areas: Teaching/Learning Difficulties and Failure Rates in Computer Programming, Computational Thinking, and Neuroscience. In each of them, the most influential related works were identified. In the area of Teaching/Learning Difficulties I studied [1, 4, 5, 6] and Failure Rates in Computer Programming, the works read were: [7, 8, 9, 10]. The relevant work identified in the Computational Thinking are was: [11, 12, 13, 14]. In the area of Neuroscience, I have looked for the basics in following references: [15, 16]. Concerning the related work the next references standed out: [17, 18, 19]. This study originated 2 chapters of the pre-thesis document *Training Computational Thinking: exploring approaches supported by Neuroscience, Univ. of Minho, 2022. PhD - prethesis.*

3. Achieved Results

After the research and analysis that led to the writing of the SotA, an approach was designed that aims to fulfill the main objective: to train Computational Thinking (CT), using techniques supported on Neuroscience, to improve the teaching and learning of Computer Programming and also the skills for general problem-solving. Thus, the proposal is the creation of a web platform – Computational Thinking 4 All, that will be the workbench of any teacher who wants to prepare his students to acquire CT skills. The web platform will be built up from 3 modules.

Module 1: CT Learning Space. In this component the different concepts, approaches, techniques and instruments that describe and characterize CT and related matters are presented. **Module 2: OntoCnE** is composed of **Ontology for Computing at School** (this component describes OntoCnE – an ontology for Computing at School, structured in 3 layers, developed in context of this work – that aims at providing a detailed and rigorous description of two knowledge domains and their interception: Computational Thinking, and Computer Programming; OntoCnE is intended to support the design of education programs syllabus on CT as well as learning material) and **Computational Thinking Training Program** (curricula for the training of CT in K0-K12 formal education). **Module 3: Resources** is composed of **Learning Resource Repository** (providing created or reused Learning Resources (LR) to train CT), **Repository of Tests & Surveys** (a set of created or reused Tests that aim to measure the effectiveness of CT training) and **Guide for Assessing the Quality of Learning Resources** (containing a set of requirements to assess the quality of LR, mainly in the approaches suggested by Neuroscience). For more detailed information about the Computational Thinking 4 All platform see: https://computationalthinking4all.epl.di.uminho.pt/about.html.

Acknowledgments

This work has been supported by FCT – Fundação para a Ciência e Tecnologia within the R&D Units Project Scope: UIDB/00319/2020. Cristiana Ph.D. is supported by FCT, Research Grant, ref. 2020.09845.BD.

References

Due to lack of space, it was necessary to remove the bibliography referred to throughout the paper. The bibliography can be consulted at: https://computationalthinking4all.epl.di.uminho.pt/References/Issep2022_CT4ALL-DC.pdf

Computer science in primary schools and teacher education

Gabriel Parriaux

University of Teacher Education (HEP Vaud), Avenue de Cour 33, 1014 Lausanne, Switzerland

Abstract

In the French-speaking part of Switzerland, Computer Science (CS) is being introduced as a topic in obligatory school after a curriculum reform that happened in 2021. Teachers' education is key for such a reform to succeed, as most of the primary teachers have no prior CS knowledge. What knowledge do they need to properly teach CS? Using lexical data analysis of classroom recordings and interviews with teachers, we explore their vocabulary about programming. Along with analysis of activity and didactics, we investigate the relations between their way of using different representations of a program, their personal knowledge in informatics and professional experience.

Keywords

Computer science education, programming, primary teacher education, lexical analysis

1. Research Questions

In different contexts, we observe that representation of notions is a key concept for learning. In mathematics, the role of semiotic representation of notions has been investigated. It was observed that transformation of one register of representation to another is a source of cognitive complexity for pupils [1].

In CS education, researchers have shown the importance that the representation of a program had in terms of cognitive load for pupils [2]. The concept of notional machine as a pedagogical device that teachers employ to draw attention to some aspects of programming is also related to the question of representation, as notional machines adopt a specific representation for this purpose [3]. However, these questions are rarely discussed explicitly during teacher education.

One objective of the thesis is to address this problem by exploring the role of representation in relation with learning in the field of computer science at primary level. The focus is on teachers and on the way they use, understand and make their pupils use different representations of a program. It brings us to our research questions:

1. How do primary teachers use representations of a program? What do they say about those representations and about the way they use them?
2. Is there a relation we can observe between primary teachers' use of representations, their personal knowledge in informatics and professional experience? If there is, which one?

Informatics in Schools. A step beyond digital education. 15th International Conference on Informatics in Schools: Situation, Evolution, and Perspectives, ISSEP 2022, Vienna, Austria, September 26–28, 2022

gabriel.parriaux@hepl.ch (G. Parriaux)

0000-0002-8921-5459 (G. Parriaux)

2. Main research methodologies

The thesis articulates several approaches: activity analysis — in-class observations followed by self-confrontation interviews, didactics — misconceptions, notional machines, program representation and lexical data analysis — cluster analysis, correspondence analysis.

In a perspective of analysis of activity [4], we observe pre-service and experienced teachers involved in a project to experiment CS activities in primary classes. We collect their discourse as a corpus of data that we analyze using textual data analysis.

3. Already achieved results

We published two articles: one about knowledge of primary school teachers in France and Switzerland [5] and one about exploring resources using lexical analysis [6].

4. Most influential related works

Apart from the already mentioned articles, we want to mention this good example of the way textual data analysis can be applied to the field of teacher education [7].

References

[1] R. Duval, Transformations de représentations sémiotiques et démarches de pensée en mathématiques, in: Actes du XXXIIe colloque COPIRELEM, 2006, pp. 67–89.

[2] I. Kalas, A. Blaho, M. Moravcik, Exploring control in early computing education, in: International Conference on Informatics in Schools: Situation, Evolution, and Perspectives, Springer, 2018, pp. 3–16.

[3] S. Fincher, J. Jeuring, C. S. Miller, P. Donaldson, B. Du Boulay, M. Hauswirth, A. Hellas, F. Hermans, C. Lewis, A. Mühling, et al., Notional machines in computing education: The education of attention, in: Proceedings of the Working Group Reports on Innovation and Technology in Computer Science Education, 2020, pp. 21–50.

[4] M. Durand, J. Theureau, et al., The challenges of activity analysis for training objectives, Le travail humain 79 (2016) 233–258.

[5] B. Drot-Delange, G. Parriaux, C. Reffay, Futurs enseignants de l'école primaire: connaissances des stratégies d'enseignement, curriculaires et disciplinaires pour l'enseignement de la programmation, RDST. Recherches en didactique des sciences et des technologies (2021) 55–76.

[6] C. Reffay, G. Parriaux, B. Drot-Delange, M. Khaneboubi, Robotics in primary education: a lexical analysis of teachers' resources across robots, in: Proceedings of WCCE 2022, Springer, 2022. Accepted.

[7] M. Prieur, R. Monod-Ansaldi, V. Fontanieu, Réception des démarches d'investigation prescrites par les enseignants de sciences et de technologie, RDST. Recherches en didactique des sciences et des technologies (2013).