Education and Information Technologies (2021) 26:965–982
https://doi.org/10.1007/s10639-020-10295-0

Case study: Developing long-term knowledge with Sprego

Gábor Csapó, et al. *[full author details at the end of the article]*

Received: 28 March 2020 / Accepted: 29 July 2020 / Published online: 8 August 2020

Abstract

In Hungary, K-12 informatics/computer science education focuses on mostly surface-based methods. This approach can be observed in the teaching of several topics in the subject, of which we focus on spreadsheet management. This is further emphasized by regulatory documents – the Hungarian National Core Curriculum and Hungarian Curriculum Frameworks –, where handling algorithms, calling schemata, and problem-solving in general are only assigned to the programming topic. In the process of fulfilling the requirements of the school curricula and the various tool-centered exams, students become familiar with the software interfaces and how to navigate them, instead of developing computational thinking skills and learning how to approach and solve real-world problems.

Our educational system is based on a spiral teaching approach; therefore, spreadsheet management is taught throughout several grades in a small number of lessons. Prior research shows that students learning spreadsheet management with surface-approach methods do not build up a reliable knowledge structure. These students cannot solve problems in contexts differing to the ones in which they learned the topic and cannot use their surface navigation abilities in different software environments.

Our research group focuses on spreadsheeting with an algorithm-building and problem-solving method at the center of the teaching-learning process. For this purpose, we have developed and introduced the Sprego (Spreadsheet Lego) methodology. Sprego is based on Pólya's four-step concept-based problem-solving approach, and its efficiency has already been proved compared to traditional low-mathability surface-approach methods. In the comparison of the low- and high-mathability approaches, several further questions arise, and amongst them one crucial aspect is how the different methods support the schema-construction and knowledge built up in long-term memory. In this paper we discuss this question using a delayed post-test that was carried out one year after the treatment period. We focused on the students' achievement both in the experimental (Sprego) and control (traditional surface-approaches) groups based on the methods used one year prior to the administration of the delayed post-test. The results show that students who learned the spreadsheet management topic with Sprego achieved significantly better scores on the delayed tests than those students who used low-mathability approaches.

Keywords K-12 education · End-user computing · High-mathability computer problem-solving · Sprego

1 Introduction

Research dealing with spreadsheet documents has found that more than 90% of them are error-laden (Teo and Tan 1999; Tort et al. 2008; Panko 2008; Powell et al. 2008, 2009a, 2009b; Abraham and Erwig 2009; Kadijevich 2009, 2013; Panko and Aurigemma 2010; Tort 2010; W2 2012; Jorgensen 2013; Kwak 2013; Garrett 2015). These error-prone and bricolaged documents are the consequences of ineffective, low-mathability[1] teaching approaches conducted in schools and self-studying. Spreadsheet education primarily focuses on graphical interfaces, where the focus is on the browsing of menus, toolbars, buttons, and wizards. Consequently, students and end-users cannot solve real-word problems requiring deeper understanding in data management (Tort 2010). However, there are sources and research studies which have proved that spreadsheeting is functional programming (Booth 1992; Hubwieser 2004; Vandeput 2009; Sestoft 2011; Hermans 2019), and not simply aimless navigation around the interface. Our research group has developed a method for spreadsheeting with the goal of teaching spreadsheet management focusing on analyzing and understanding problems, wording and building algorithms, and completing tasks with discussion and abstraction - in general, developing computational thinking skills through problem-solving. We discuss this method in detail in Section 2. One of the reasons why our method was brought to life is the peculiarities and unfavorable changes in the Hungarian Public Educational System, which lowered reduced the number of informatics classes, and openly switched from programming to low-mathability end-user document handling, ignoring the need to develop students' computational thinking skills.

1.1 The Hungarian public education system (K-12)

Since the comparison of the low- and high-mathability approaches is one of the major points of the present paper, we cannot leave undiscussed the fact that the K-12 informatics education of the Hungarian school system, as mentioned above, plays a crucial role by defining the requirements and also providing suggestions for the methods to be applied. In the Hungarian K-12 Public Educational System there are four different patterns (the numbers indicate the years). Note, that these categories do not include the technical institutes.

1. 8 + 4: elementary school (1–8), high school (9–12)
2. 8 + 1 + 4: elementary school (1–8), high school with 1-year intensive language learning (0), high school (9–12)
3. 6 + 6: elementary school (1–6), combined junior and high school (7–12)
4. 4 + 8 elementary school (1–4), combined junior and high school (5–12)

Regardless of these options and students' selections of schools, at the end of K-12 education, students must take a final graduation exam involving multiple subjects. The

[1] This usage [mathability] has basically two forms: in some cases we use existing functions and methods provided by a system, and we apply these tools to solve the problems (low-mathability approach). Another possibility is, if we, based on existing means of the system, develop new programs and functions for solving new problems (high-mathability) (Baranyi and Gilányi 2013).

National Core Curriculum (NCC) is the theoretical and conceptual base of the learning content. With the autonomy of schools in mind, it defines the general goals, the main areas of competence to be transferred, the content division of public education, and the development tasks to be implemented in each content phase. The NCC identifies key competencies, priority development tasks and areas of skill development. Based on this, the Educational Authority prepares the Hungarian Curriculum Frameworks (HCF), which determines the compulsory subjects, the number of classes, the content of the school curricula and the requirements. 10% of the class time is allowed to be managed by the schools and their teaching staff. Considering all the requirements of the NCC and HCF, schools create their local curricula which determine the logical structure of the knowledge material.

In 1995, computer science studies appeared as an independent knowledge area in the NCC (NAT 1995). However, there was no informatics teacher education at that time. The result was that many currently in-service informatics teachers, both through post-graduate courses of various levels and quality and self-study, re-educated themselves from being teachers of maths, physics, or similar subjects without any background knowledge in computer sciences and informatics.

The introduction of the computer science knowledge area had two goals in the HCC (NAT 1995):

- to improve analytical thinking skills,
- to develop conscious use of computer applications and tools.

However, the original aims of the NCC seem to be misunderstood, and an undesired and unforeseen development can be identified. Computer problem-solving is restricted to programming and teaching programming in an extremely low number of classes, while the teaching of end-user knowledge is carried out exclusively by applying surface-based, low-mathability approaches (Baranyi and Gilányi 2013; Chmielewska et al. 2016). A similar phenomenon can be observed and experienced in several other education systems as well, and can be proved by the existence of error-laden and error-prone documents and the financial losses derived from them (Teo and Tan 1999; Tort et al. 2008; Panko 2008; Powell et al. 2008, 2009a, 2009b; Abraham and Erwig 2009; Kadijevich 2009, 2013; Panko and Aurigemma 2010; Tort 2010; W2 2012; Jorgensen 2013; Kwak 2013; Garrett 2015).

In the HCF released in 2012 (OFI 2012), which was based on HCC 2012 (NAT 2012), the indicated number of informatics classes was dramatically reduced (Table 1) compared to the HCF released in 2008 (OFI 2008). However, the knowledge contents required were not reduced. In addition, the requirements of the HCF are not expressed clearly, being too general and ambiguous, which has led to various free interpretations in practice (OFI 2012; Nagy 2018). At the end of 2019, a new NCC (NAT 2020) was published, followed by a new HCF in the spring of 2020 (OFI 2020). The authors of these documents have recognized the importance of problem-solving, and increased the compulsory number of classes in the hope of a more successful informatics education based on international experience. This may sound like a progressive step, but at the time of the present paper, postgraduate courses for teachers are not prepared to adapt to such changes. Therefore, in practice, informatics teachers are expected to continue teaching with traditional methods with low efficiency, and uneducated teachers fill in

Table 1 The indicated number of classes in each grade based on the Frame Curricula released in 2008, 2012 and 2020 (OFI 2008, 2012, 2020)

	Grades											
	1	2	3	4	5	6	7	8	9	10	11	12
2008			0.5	0.5	0.5	1	1	1	1.5	1	1.5	1.5
2012–2024						1	1	1	1	1		
2020–			1	1	1	1	1	1	2	1	2	

the empty positions, due to the lack of teachers of informatics. In the past few years, informatics education has been ineffective, failing to meet the needs of the industry and higher education. This tendency is also observable in the results of the latest PISA-measurement (OECD 2011). It shows that in Hungary the majority of the X, Y and Z generations are digital illiterates. To facilitate a change, teaching methods that focus on problem-solving and the development of the students' computational thinking skills are essential in today's informatics education.

1.2 Concept-based problem-solving approaches

Teachers decide independently on the contents to be taught and on the methodology they apply. As a result of the extremely strict time limit (from the 2012 NCC) and the insufficient teacher training programs – lack of postgraduate training courses and a low level of programming knowledge –, programming as a topic area is usually omitted. Consequently, students' algorithmic skills do not develop as expected. In accordance with this, teaching end-user computing focuses on tools, and on navigating the interface. However, recently, more and more teachers have been recognizing the importance of real-world computer problem-solving and of building and recalling algorithms and schemata in end-user computing, as well (Pólya 1954; Hubwieser 2004; Kahneman 2011; Sestoft 2011; Sweller et al. 2011; Csernoch et al. 2015; Csernoch and Dani 2017; Hermans 2019; Swidan and Hermans 2019). We are convinced that concept-based problem-solving approaches, which have been proved effective and efficient in other school subjects, should be introduced in all topics of informatics education, and not exclusively in programming.

1.2.1 Pólya's concept-based problem-solving approach

Pólya's concept-based approach clearly defines the steps involved in solving real-world problems, declared in his well-known book, How to Solve It (Pólya 1954).

1. You have to understand the problem. Find the connection between the data and the unknown. You may be obliged to consider auxiliary problems if an immediate connection cannot be found.
2. You should eventually obtain a plan of the solution.
3. Carry out your plan of the solution, check each step.
4. Examine the solution obtained.

1.2.2 ACM & IEEE levels of mastery

Pólya's problem-solving approach can be recognized in the IEEE & ACM report (IEEE and ACM Report 2013, which defines three levels of mastery. The levels of mastery are equivalent to Pólya's first, third and fourth steps of problem-solving. Unfortunately, planning does not appear as a separate level, in spite of the fact that it plays a crucial role in computer problem-solving.

1. familiarity: understanding the problem
2. usage: carrying out
3. assessment: conscious choice, discussion

Hungarian informatics education mostly focuses on the second level of mastery of the ACM & IEEE report. Understanding, analyzing the problem and building the algorithm are omitted; the "doing" starts at level two (Fig. 1), and consequently, there is no way to reach level three. Students are required to work on their own – self-studies and -teaching – or follow step-by-step instructions – computer cooking[2] –, without appropriate guidance. However, this approach has been proved insufficient and ineffective (Kirschner et al. 2006).

The biggest disadvantage of computer cooking is that students are only able to solve tasks in familiar interfaces, and only if they are provided with a list of instructions. As soon as a new software version is released or the use of another software family is required, the students' surface knowledge become outdated and/or inadequate.

Our research group developed and tested a method for teaching spreadsheet data management, based on problem-solving and algorithm building (detailed in Section 2). Our aim was to provide a more effective methodology, compared to the traditional surface-approaches, to develop students' computational thinking skills through the creation and recalling of schemata.

It has been proved that students' logical skills can be improved by doing spreadsheeting (Kruck et al. 2003). However, Kruck failed to prove that sequential and spatial skills can also be developed. On the contrary, our research group managed to prove that with concept-based, high-mathability spreadsheet teaching, students' spatial and sequential skills can also be developed. It was also found that high-mathability teaching approaches support students' ability in building and recalling schemata, recognizing connections between problems and applying interface-independent knowledge (Csapó et al. 2020).

2 Sprego

In Hungary, the informatics textbooks list more than 100 problem-specific functions in the topic of spreadsheet management. In addition to learning the names and content of

[2] The expression 'computer-cooking' is borrowed from cookbooks, where students are required to follow strict step-by-step instructions, without giving any thought to design, computer-problem solving, or discussion. They are not requested to carry out any cognitive work, and not allowed to check the correctness of the tasks provided (Figure 1). Only the correctness of strictly defined steps are evaluated.

1. Open the spreadsheet application and open the file called improvements.xlsx from your candidate drive. Save the file as costings.xlsx to your candidate drive.
2. On the projection worksheet, zoom the display to 100%.
3. Widen column A so that the content of the column is fully visible.
4. Enter a function in cell B11 to calculate the sum of the cell range B5:B10.
5. Copy the sum function in cell B11 to the cell range C11:F11.
6. Enter a formula in cell B13 that subtracts cell B11 from cell B3. Copy the formula in cell B13 to the cell range C13:E13.
7. Enter a formula in cell F5 with an absolute cell reference for one cell only that divides the content of cell E5 by the content of cell E11. Copy the formula in cell F5 to the cell range F6:F10.
8. Format the cell range F5:F11 as a percentage with no decimal places and save.

Fig. 1 Part of a sample task from an ICDL (International Computer Driving License) Spreadsheets module (ICDL 2020), where even the first sentence carries an error (applications are run, not opened). In the example there is no real task to solve, and students are required to complete Steps 1–8 like slaves

these functions, the students must also learn their arguments, the order of the arguments, the domains, the ranges of the functions, and their specific use-cases. Moreover, several problem-specific functions follow illogical approaches in some arguments. For example, solving an inequality problem with a variable using the wildly used *IF?() functions,[3] end-users are required to code concatenated strings, instead of simple yes/no questions (Fig. 2) (Csernoch 2014). Consequently, students have to memorize a great amount of information about problem-specific functions, as several parts of this knowledge are tied to specific situations. With such a low number of classes prescribed in the NCC, there is no time to teach and practice so many functions, not to speak of developing a solid, schemata-based knowledge.

The Sprego abbreviation originates from the playful word combination of Spreadsheet and Lego, referring to the small blocks (building blocks) of problem-solving, which can be built into complex structures. Our Sprego methodology focuses on teaching spreadsheet data management through the analysis of authentic datatables, structuring problems, selecting strategies, coding, and discussion. In contrast with the traditional low-mathability problem-specific approaches, the Sprego method uses only general-purpose spreadsheet functions. These functions can be sorted into three subcategories, presented in Table 2. In the coding process students build these functions into multilevel array formulas. Compared to the large number of problem-specific functions presented in textbooks, Sprego covers the requirements of both the HCF, the final exams, and also the ICDL/ECDL with only twelve general-purpose spreadsheet functions. Note, that this set of functions is an open set, to which further functions can be added, according to the requirements of novel problems.

[3] COUNTIF(), COUNTIFS(), SUMIF(), SUMIFS(), AVERAGEIF(), AVERAGEIFS(), MINIFS(), MAXIFS()

```
=COUNTIF(range,criteria)
=COUNTIF(A1:A4,10), equality with a constant
=COUNTIF(A1:A4,A4), equality with a variable
=COUNTIF(A1:A4,">55"), inequality with a constant
=COUNTIF(A1:A4,"<"&B4), inequality with a variable
```

Fig. 2 Examples of using the COUNTIF() function for solving equality and inequality problems (Microsoft 2020)

Due to the low number of functions, in the analytical phase, Sprego supports the algorithm-building process and, in coding, the introduction and the application of multi-level, n-ary functions and formulas, along with the concept of one- and two-dimensional arrays.

Problem-solving in Sprego follows Pólya's four-step concept-based problem-solving approach (Pólya 1954):

1. Analyzing the data and world-world problem(s) at hand. Formulating the desired output based on the content of the table and requirements of the task.
2. Planning the algorithm(s) and highlighting the input and output values of each step of the algorithm.
3. Coding the algorithm using general-purpose spreadsheet functions.
4. Discussing, analyzing, testing and evaluating the results.

To increase the effectiveness of the methodology, we have developed unplugged and semi-unplugged tools based on previous studies which consider multisensorial perceptions (Shams and Seitz 2008; Bell and Newton 2013; Kátai et al. 2014). Our collection of unplugged tools includes 3D printed matryoshka dolls, toy barrels, and origami boat sets (objects of different sizes and colors), team vests and additional students' tools – e.g. scissors, painter's tapes, exercise books, and teachers' white board tools. Our semi-unplugged tools – 2D and 3D animations – are developed to help understand algorithms, multilevel functions, and how data exchange is carried out between the different levels (Csapó and Sebestyén 2017, 2020; Gulácsi and Dienes 2018; Sebestyén et al. 2018). The other set of semi-unplugged tools is a selection of authentic datatables originating on the Internet to make students interested and motivated, both in the content and the programming approach.

Table 2 The twelve general-purpose Sprego functions grouped into three categories

Sprego text	Sprego number	Sprego pro
LEFT()	MIN()	IF()
RIGHT()	MAX()	INDEX()
LEN()	SUM()	MATCH
SEARCH()	AVERAGE()	ISERROR()

One further advantage of the Sprego approach is that it gives space to computer- and software-independent creative activities, expressed in the students' mother tongue. Furthermore, the methodology develops schemata (Skemp 1971; van Merriënboer and Sweller 2005, which can be effectively and efficiently recalled and applied in tasks that require similar algorithms in a different context (Pólya 1954; Skemp 1971; Kahneman 2011; Sweller et al. 2011; Kirschner and De Bruyckere 2017).

3 Developing long-lasting knowledge with Sprego

In our previous research, we measured the effectiveness of Sprego in teaching spreadsheet management in experimental and control groups. We found that those students who studied the topic with Sprego obtained significantly higher results (65.42%) compared to the students who used interface-based, traditional, low-mathability methods (38.00%) ($p = 0.0013$) (Csapó et al. 2020). In this paper, our aim was to examine how Sprego affects students' long-lasting spreadsheet knowledge. Our research group measured students' knowledge in experimental and control groups one year after their last spreadsheet management lesson. We selected groups who did not have any spreadsheet management lessons in the intervening period.

3.1 Sample

The testing process was carried out in seven groups at a primary and a secondary school in Debrecen, Hungary. The selection of the groups was based on the students' previous official spreadsheet education, according to the local curricula. The students needed to have completed spreadsheet studies one year in advance of our testing. Based on the methodology applied in classes in the previous academic year, two major groups were formed: an experimental and a control group, studying with Sprego and traditional approaches, respectively. Within these groups we can distinguish between 8th and 10th grade students (Table 3).

In the experimental groups, the 8th grade students (groups e8_1 and e8_2) studied spreadsheeting in eight lessons, with one lesson per week. The 10th grade groups (e10_1 and e10_2) studied it in six lessons, with one lesson per week (Table 3). In two of the control groups (c10_1 and c10_2) spreadsheeting was taught in 12 lessons, with two lessons per week through six weeks, while the third group (c10_3) had six lessons, with one lesson per week. The differences between the classes in the same grades depended on the educational program in which the students studied (Table 4).

Table 3 The number of students who participated in the delayed post-test in the experimental and control groups

	Experimental groups				Control groups		
group	e8_1	e8_2	e10_1	e10_2	c10_1	c10_2	c10_3
N	18	18	17	15	7	13	7

Table 4 The types of the education (total grades divided between education levels) of the groups and their weekly and total number of lessons that focused on spreadsheeting in the year prior to the test

	Experimental groups				Control groups		
Group	e8_1	e8_2	e10_1	e10_2	c10_1	c10_2	c10_3
Type of education	6 + 6	6 + 6	8 + 1 + 4	8 + 1 + 4	6 + 6	6 + 6	8 + 1 + 4
Lesson number/week	1	1	1	1	2	2	1
Total lesson number	8	8	6	6	12	12	6

We must note that the number of lessons which are dedicated to spreadsheeting in the selected grades is extremely low, according to the spiral HCF. From our point of view, this low number of lessons is not enough to develop students' basic computational thinking skills. One further advantage of Sprego must be mentioned in this context, namely that Sprego can be utilized to cover both spreadsheeting and programming topics. Therefore, it can save valuable class time and it also prepares students for the database management topic.

Collecting data for our experiment was carried out with delayed post-tests in printed form. According to the schedule of the school, the tests were completed during the final period of the academic year 2018/2019, at the end of May and the beginning of June. The overall size of the sample is $N = 95$. The distribution of the students between the experimental and control groups is detailed in Table 5. The smaller size of the control groups can be explained by both the extra-curricular occupation of students at the end of the academic year and the involvement of teachers who did not belong to our research group, which meant that we had no control over their schedules or classroom activities.

3.2 Validation of the test

The test measures the students' problem-solving strategies and gathers information on the algorithms they apply in the problem-solving process. The knowledge items included in the test are in accordance with the knowledge items, output requirements and knowledge transfer elements which appear in the Hungarian Curriculum Framework in grades 5–6 and grades 7–8 (Table 5).

The Hungarian Curriculum Framework for grades 5–6 contains the "Problem-solving with informatics tools and methods" topic, which guides students through the steps of algorithm building: planning, studying different methods leading to the solution, analyzing error possibilities, efficiency and the decision-making process.

In the topic of "Algorithmization and data modeling", students are introduced to relational data structures. The goal of this topic is to give students the ability to gather relevant information for solving problems, following the steps of the problem-solving process they have learned before. By the end of the topic, students, officially, are able to build algorithms of simple problems and code them in the programming language(s) provided.

In grades 7–8, the conceptual system of the students is further expanded; they learn the concept of spreadsheets (cell, row, column, reference, formula), different data

Table 5 Knowledge items and output requirements from the Hungarian Curriculum Framework included in our test

	grades 5–6	grades 7–8
building algorithms	x	
data formats	x	
data classification and interpretation	x	
using functions to calculate the sum, average, maximum and minimum of values		x
writing algorithms in natural language	x	
writing algorithms in pseudo code		x
coding, programming	x	
using parameters and arguments		x
conditional statements		x
building solutions from the bottom-up, and refining them step-by-step		x
defining outputs/results based on input data		x

formats (text, number, currency) and data-illustration tools (diagram). In addition, the topic includes classifying and interpreting data, drawing conclusions from data, and functions which are required for basic data analysis (sum, average, minimum, maximum).

In the "Problem-solving with informatics tools and methods" topic, grade 7 and 8 students learn about algorithm building with pseudo code and they build programs based on algorithms. Subsequently, in "Algorithmization and data modeling", students become familiar with the coding of algorithms in a programming language – in our case this programming language is the functional language of spreadsheet environments. In this topic the following concepts are introduced: parameterization, conditional statements, loops and recursions. The Hungarian Curriculum Framework also incorporates the principle of bottom-up construction of solutions and step-by-step refinement of complex algorithms.

Our test focuses on tasks that measure the knowledge elements connected to the topics described above. According to the specifics of the Hungarian spiral education system, these topics are revisited in grades 9–10. Considering the requirements of the curricula, the test administered in the research was found to be valid using face validity.

3.3 Tasks of the delayed post-test

To measure the students' knowledge, we used the spreadsheet-related questions that were presented in the TAaAS project (Testing Algorithmic and Application Skills) (Csernoch et al. 2015). The tasks (Appendix, Fig. 3.) gathered data on the students' data analyzing, algorithm and formula building, and schema and function recalling skills (Tasks *a*–*e*). Besides these, their formula evaluating skill were also tested in Task *f*. In order to solve the tasks, the students had to collect data from the table accompanying the tasks. The topic of the table is the countries of the world, adapted from the mock graduation test of informatics in 2004 (OFI 2004). The table consists of 235 data records from row 2 to 236, including the name, the continent, the population (in

thousands), and the area of the countries. A variable with an unknown value was also indicated in cell G2.

Task *a* requires students to use the algorithm of linear search in order to name the capital city of the largest country. In Task *b* the population density of the countries must be calculated. In order to solve this task, beyond spreadsheeting knowledge, students must also know how to calculate the population density, which requires some mathematical and geographical background knowledge. In addition to this, they must take into consideration that population values are given in thousands – this data can be collected from the table provided. In Tasks *c* and *e* conditional counting had to be carried out, using a constant value in Task *c* and a variable in Task *e*. Task *d* is a conditional average calculation with also a variable. In Task *f* students had to analyze a complex array formula and explain what it does using natural language.

3.4 Analyzing and processing the data

The students' results were recorded and stored in a database designed for our purposes, where every task was divided into the smallest possible meaningful knowledge items. Since Tasks *c*–*e* can be solved either by using array formulas – with the Sprego methodology – or by using problem-specific functions built into spreadsheet programs, several different solutions were accepted and itemized.

We must also note that regardless of the language, both the Hungarian and the English versions of the functions were accepted. Moreover, if the name of the functions and the order of the arguments were correct, but only the parenthesizes did not match, we accepted the answers as correct. We must also note that the 8th grade students did

	A	B	C	D	E	F	G
1	Country	Continent	Capital	Area	Population (thousand)		
2	Afghanistan	Asia	Kabul	647500	27756		
3	Albania	Europe	Tirana	28748	3545		
4	Algeria	Africa	Algiers	2381740	32278		
5	American Samoa	Oceania	Pago Pago	199	69		
6	Andorra	Europe	Andorra la Vella	468	68		
7	Angola	Africa	Luanda	1246700	10593		
8	Anguilla	America	The Valley	102	12		
233	Yemen	Asia	Sanaa	527970	18701		
234	Yugoslavia	Europe	Belgrade	102350	10657		
235	Zambia	Africa	Lusaka	752614	9959		
236	Zimbabwe	Africa	Harare	390580	11377		

a. What is the capital city of the largest country?
b. What is the population density of each country?
c. How many African countries are in the table?
d. What is the average population of those countries whose area is smaller than G2?
e. How many countries have a surface area greater than G2?
f. What is the result of the following formula?
{=SUM(IF(B2:B236="Europe",IF(LEFT(A2:A236)="A",1)))}

Fig. 3 The tasks of the delayed post-test

not study the linear search algorithm, so it was not expected that they would solve Task *a* completely.

The data analyzing process was carried out with the R software package (The R Foundation 2019). The normalities of the data were examined with the Shaphiro-Wilk normality test, while the significance between the group differences was analyzed with Mann-Whitney tests.

4 Analyzing the long-term effectiveness of Sprego

The results of the tasks in the delayed post-test are presented in Table 6. In the data analyzing process the different problem-solving strategies were handled independently, and the final results are calculated in percentages.

The table shows that the results of the experimental groups are higher than those of the control groups in all but one task – in Task *b* group c10_2 achieved a higher percentage than groups e8_1 and e8_2.

Considering the overall results of the tests, the group with the highest percentage is the experimental group e10_1 with 55.08%, and the weakest is c10_3 with 7.20%. Note that the overall result of group e8_2 is approximately the same as the results of experimental groups e10_1 and e10_2, in spite of the fact that there is a 2-year age difference between the students. Another important finding is that the control groups, despite the 2-year difference and having double the number of classes per week in groups c10_1 and c10_2, were not able to achieve such a high result as the Sprego 8th grade groups.

The effectiveness of the algorithm- and schema construction approach of the Sprego methodology is clearly represented by the result achieved in Tasks *c*-*e*, since these three tasks are based upon the same algorithm. The students in the experimental group achieved significantly better results in these tasks (Table 5). The control group, however, faced difficulties when it came to the use of problem-specific functions; built-in function fragments not following the syntactical rules were encountered frequently. These findings clearly show that the students of the control group were not able to choose the appropriate tools to solve the tasks. In the formula analyzing task

Table 6 The results (%) of the delayed post-test in the experimental and control groups by classes and tasks

	Experimental groups (%)				Control groups (%)		
Task	e8_1	e8_2	e10_1	e10_2	c10_1	c10_2	c10_3
a	5.19	6.67	12.94	18.22	3.81	3.08	0.00
b	20.83	19.44	50.00	23.33	14.29	32.69	0.00
c	65.00	73.89	62.94	67.33	21.43	38.72	11.43
d	35.19	59.26	64.71	61.48	25.40	19.74	0.00
e	48.77	59.26	65.36	56.30	7.94	20.62	7.94
f	57.41	77.78	74.51	75.56	14.29	56.41	23.81
Total	38.73	49.38	55.08	50.37	14.52	28.54	7.20

(Task *f*) three Sprego groups achieved results higher than 74.00%, and one group achieved 57.41%. In contrast, in the control groups, the highest result – the c10_2 group with 56.41% – is lower than the weakest result in the experimental groups. Note that in Task *f* the control groups c10_3 and c10_1 could not even achieve 15.00% and 25.00%, respectively. In general, we can conclude that the lowest result of the Sprego groups is higher than the highest result of the low-mathability, traditional groups (Fig. 4). Note that while discussing the results, we must keep in mind that these students had a one-year delay after their last spreadsheet classes. Therefore, lower scores are expected in the delayed post-test than the scores which students would obtain right after their treatment period.

In general, we can conclude that the experimental groups achieved significantly better results than the control groups (Table 7). The only exception was the previously mentioned Task *b*, where the Mann-Whitney test did not show a significant difference ($p = 0.0841$). This can be explained by the fact that the task required knowledge-transfer items from other school subjects. A thorough analysis of the answers revealed that most of the students – both from the experimental and control groups – achieved lower points not because they lack knowledge of formula-creation, but because they did not know how to calculate population density.

Based on the results presented above, the groups who studied with the Sprego methodology achieved significantly better results than the control groups studying with traditional, surface approach, low-mathability methods. This allows us to conclude that the Sprego methodology is significantly more effective in developing long-term knowledge compared to the traditional approaches.

These results are in alignment with our previous findings regarding the efficacy of Sprego. However, we must keep in mind that the sample of our current research was

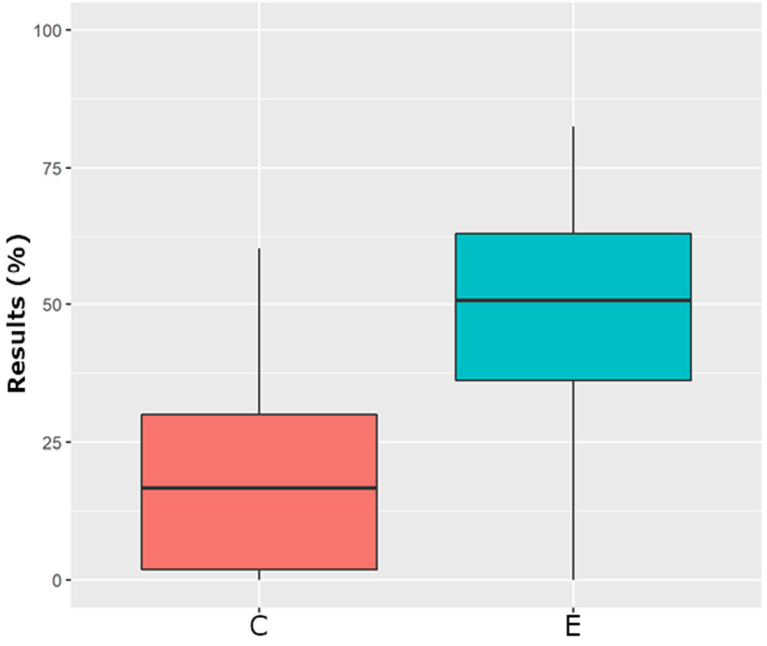

Fig. 4 The lowest and the highest results (%) in control groups (C) and the experimental groups (E)

Table 7 The results (%) of the delayed post-test of the experimental and control groups with the values of significance

Task	Experimental groups (%)	Control groups (%)	W	p
a	10.39	2.47	404	0.0000
b	28.31	19.44	720	0.0841
c	67.35	27.16	366	0.0000
d	54.74	16.09	302.5	0.0000
e	57.35	14.04	302.5	0.0000
f	71.08	37.04	517	0.0005
Total	48.20	19.37	264	0.0000

selected from a local institute in which we had the opportunity to teach students using Sprego. The selection of the groups was based on the appearance of the spreadsheet management topic in the local curriculum. While this approach provided us a sample of randomized students, further research is required in more institutes and with a larger sample to further support the findings in this paper.

5 Conclusion

The low-mathability, surface-approach methods are widely accepted and practiced in end-user computing and education, in spite of the fact that their low effectiveness and error-sensitivity has been revealed and proved (Ben-Ari 1999; Panko 2008; EuSpRIG 2020). Consequently, the time has arrived for the introduction of more effective, high-mathability, algorithm- and schema-based problem-solving approaches. This demand is clearly expressed by Wing (Wing 2006), who claims that computational thinking skills should be the fourth fundamental skill along with the 3Rs (Reading, wRiting, aRithmetic).

As an alternative approach to low-mathability end-user computer problem-solving, we offer Sprego, which is a programming methodology in spreadsheet interfaces, accompanied by numerous unplugged and semi-unplugged tools for better and deeper understanding of programming concepts (Szlávi et al. 2019).

In earlier studies we have found proof that Sprego is more effective than traditional approaches in immediate results. However, we were interested to see how Sprego affects the building up of schema and knowledge in long-term memory and how it develops sequential and spatial skills. The purpose of the present paper is to prove that Sprego is also more effective and efficient in this respect than traditional, surface-approach methods. To prove our hypothesis, we administered a delayed post-test, one year after the teaching-learning processes had taken place.

Our measurement proved that teaching students with a high-mathability, algorithm-focused, schemata building methodology develops long lasting knowledge more effectively than using the built-in problem-specific functions with interface browsing, without considering the algorithms behind them.

Future work includes testing the effectiveness and efficiency of the Sprego methodology with a larger sample of students to strengthen the results of our prior and current research. Furthermore, teacher training programs are required to educate informatics teachers in the use of Sprego to provide them with a methodological tool to aid the development of students' computational thinking skills.

Funding information Open access funding provided by University of Debrecen. This work was supported by the construction EFOP-3.6.3-VEKOP-16-2017-00002. The project was supported by the European Union, co-financed by the European Social Fund.

Appendix

The task sheet used for collecting data.

Answer tasks a) – e) using spreadsheet formulas and f) with an English sentence.

	A	B	C	D	E	F	G
1	**Country**	**Continent**	**Capital**	**Area**	**Population (thousand)**		
2	Afghanistan	Asia	Kabul	647500	27756		
3	Albania	Europe	Tirana	28748	3545		
4	Algeria	Africa	Algiers	2381740	32278		
5	American Samoa	Oceania	Pago Pago	199	69		
6	Andorra	Europe	Andorra la Vella	468	68		
7	Angola	Africa	Luanda	1246700	10593		
8	Anguilla	America	The Valley	102	12		
233	Yemen	Asia	Sanaa	527970	18701		
234	Yugoslavia	Europe	Belgrade	102350	10657		
235	Zambia	Africa	Lusaka	752614	9959		
236	Zimbabwe	Africa	Harare	390580	11377		

a) What is the capital city of the largest country?

.....

b) What is the population density of each country?

.....

c) How many African countries are in the table?

.....

d) What is the average population of those countries whose surface area is smaller than G2?

.....

e) How many countries have a surface area greater than G2?

.....

f) What is the result of the following formula?
{=SUM(IF(B2:B236="Europe",IF(LEFT(A2:A236)="A",1)))}

.....

References

Abraham, R., Erwig, M. (2009). Mutation operators for spreadsheets. Software engineering, IEEE transactions on volume: 35, issue: 1.

Baranyi, P., Gilányi, A. (2013). Mathability: Emulating and enhancing human mathematical capabilities. 2013 IEEE 4th international conference on cognitive Infocommunications (CogInfoCom), pp. 555–558.

Bell, T. & Newton, H. (2013) Unplugging computer science. In improving computer science education. (Eds.) Kadijevich, D. M. Angeli, C. and Schulte, C., Routledge.

Ben-Ari, M. (1999) Bricolage Forever! PPIG 1999. 11th Annual Workshop. 5–7 January 1999. Computer-based learning unit, University of Leeds, UK. Retrieved July 21, 2015, from http://www.ppig.org/papers/11th-benari.pdf.

Booth, S. (1992). *Learning to program: A phenomenographic perspective*. Gothenburg: Acta Universitatis Gothoburgensis.

Chmielewska, K., Gilányi, A. & Łukasiewicz, A. (2016). Mathability and mathematical cognition. 7th IEEE international conference on cognitive Infocommunications, CogInfoCom, 2016.

Csapó, G., Sebestyén, K. (2017). Educational software for the Sprego method. The Turkish online journal of educational technology, INTE 2017 October, 986–999. Retrieved 10/03/2020 from http://www.tojet.net/special/2017_10_1.pdf Accessed 11 December 2017.

Csapó, G., Sebestyén, K. (2020). Sprego - the official Sprego application. Retrieved 12/06/2020 from https://play.google.com/store/apps/details?id=hu.sprego.oktatoprogram

Csapó, G., Csernoch, M., Abari, K. (2020). Sprego: Case study on the effectiveness of teaching spreadsheet management with schema construction. Education and information technologies, Isssue 3/2020.

Csernoch, M. (2014). Programming with spreadsheet functions: Sprego. In Hungarian: Programozás táblázatkezelő függvényekkel – Sprego. Műszaki Könyvkiadó. Budapest.

Csernoch, M., & Dani, E. (2017). Data-structure validator: An application of the HY-DE model, 8th CogInfoCom, Debrecen, 2017. *IEEE Computer Society, 2017*, 197–202.

Csernoch, M., Biró, P., Máth, J., & Abari, K. (2015). Testing algorithmic skills in traditional and non-traditional programming environments. *Informatics in Education, 2015, 14*(2), 175–197.

EuSpRIG (2020).Horror Stories. Retrieved 02/01/2020 from http://eusprig.org/horror-stories.htm

Garrett, Nathan (2015). Textbooks for responsible data analysis in excel. Journal of Education for Business

Gulácsi, Á., Dienes, N. (2018). 3D software environment for educational Sprego programming. Turkish online journal of educational technology, INTE 2018 November, pp. 990-997. Retrieved 05/10/2019 from https://www.int-e.net/publication_folder/inte/inte_iticam_idec2018_v2.pdf

Hermans, F. (2019). How to teach programming (and other things)? Strange loop 12-14/09/2019, St. Louis, MO retrieved 07/10/2019 from https://www.youtube.com/watch?v=g1ib43q3uXQ

Hubwieser, P. (2004). Functional Modelling in secondary schools using spreadsheets. *Education and Information Technologies, 9*(2), 175–183.

ICDL (2020). Sample test: Word processing. Retrieved 25/05/2020 from https://www.icdleurope.org/workforce/sample-tests/

IEEE&ACM Report 2013 (2013). Computer Science Curricula 2013. Curriculum Guidelines for Undergraduate Degree Programs in Computer Science. December 20, 2013. The Joint Task Force on Computing Curricula Association for Computing Machinery (ACM) IEEE Computer Society. Retrieved 25/09/2019. https://www.acm.org/binaries/content/assets/education/cs2013_web_final.pdf

Jorgensen, H. (2013). How not to Excel in economics Retrieved 07/07/2018 from http://www.lowyinterpreter.org/post/2013/04/18/How-not-toExcel-in-economics.aspx

Kadijevich, D. (2009). Simple spreadsheet modeling by first-year business undergraduate students: Difficulties in the transition from real world problem statement to mathematical model. In M. Blomhřj & S. Carreira (Eds.), *Mathematical applications and modeling in the teaching and learning of mathematics: Proceedings the 11th international congress on mathematical education* (pp. 241–248). Mexico.

Kadijevich, D. (2013). Learning about spreadsheet. In K. Djordje, A. Charoula, & S. Carsten (Eds.), *Improving computer science education* (pp. 19–33). New York and London: Routledge.

Kahneman, D. (2011). *Thinking, fast and slow*. New York: Farrar, Straus and Giroux.

Kátai, Z., Tóth, L., & Adorjáni, A. K. (2014). Multi-sensory informatics education. *Informatics in Education, 2014, 13*(2), 225–240.

Kirschner, P. A., & De Bruyckere, P. (2017). The myths of the digital native and the multitasker. *Teaching and Teacher Education, 2017*(67), 135–142.

Kirschner, P. A., Sweller, J., & Clark, R. E. (2006). 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, 41*(2), 75–86.

Kruck, S. E., Maher, J. J., & Barkhi, R. (2003). A framework for cognitive skill acquisition and spreadsheet training. In M. A. Mahmood (Ed.), *Advanced topics in end user computing* (Vol. 2, pp. 212–233). El Paso: University of Texas at El Paso.

Kwak, J. (2013). The importance of excel. Retrieved 05/17/2019 from http://baselinescenario.com/2013/02/09/the-importance-of-excel

Microsoft (2020). COUNTIF function. Retrieved 20/05/2020 from https://support.office.com/en-gb/article/countif-function-e0de10c6-f885-4e71-abb4-1f464816df34

Nagy T. K. (2018) 3th International Interdisciplinary Conference 2018: Az informatika kerettanterv elemzése http://detep.unideb.hu/sites/default/files/upload_documents/kotet_interdisz_3.pdf

NAT 1995 (1995). Hungarian Core Curriculum. In Hungarian: 130/1995. (X.26). Korm. rendelet a Nemzeti alaptanterv kiadásáról. Retrieved 12/06/2018 from http://njt.hu/cgi_bin/njt_doc.cgi?docid=24382.38666

NAT 2012 (2012). Hungarian Core Curriculum. In Hungarian: 110/2012. (VI. 4.) Korm. rendelete a Nemzeti alaptanterv kiadásáról, bevezetéséről és alkalmazásáról. Retrieved 12/06/2018 from http://ofi.hu/sites/default/files/attachments/mk_nat_20121.pdf

NAT 2020 (2020). Hungarian Core Curriculum. In Hungarian: 5/2020. (I.31.) Korm. rendeelet: .A Nemzeti alaptanterv kiadásáról, bevezetéséről és alkalmazásáról szóló 110/2012. (VI.4.) Korm. rendelet módosításáról. Retrieved 03/05/2020 from https://magyarkozlony.hu/dokumentumok/3288b6548a740b9c8daf918a399a0bed1985db0f/megtekintes

OECD (2011): PISA 2009 results: Students on line: Digital technologies and performance (volume VI).

OFI (2004). Trial Graduation: informatics, advanced level. In Hungarian: Próbaérettségi 2004: informatika, emelt szint Retrieved 05/30/2020 from https://www.oktatas.hu/kozneveles/erettsegi/feladatsorok/probaerettsegi2004_valaszthato_targyak

OFI (2008). Hungarian Curriculum Framework. In Hungarian: Kerettanterv. 2/2008. (II.8.) OKM rendelete a kerettantervek kiadásának és jóváhagyásának rendjéről. Retrieved 06/01/2020 from http://www.nefmi.gov.hu/kozoktatas/tantervek/oktatasi-kulturalis

OFI (2012). Hungarian Curriculum Framework. In Hungarian: Kerettanterv. 51/2012. (XII. 21.) számú EMMI rendelet – a kerettantervek kiadásának és jóváhagyásának rendjéről. Retrieved 12/10/2018 from http://kerettanterv.ofi.hu/

OFI (2020). Hungarian Curriculum Framework. In Hungarian: Kerettanterv. Retrieved 06/06/2020 from https://www.oktatas.hu/kozneveles/kerettantervek/2020_nat

Panko, R, R. (2008). What we know about spreadsheet errors. Journal of End User Computing's Special issue on Scaling Up End User Development Volume 10, No 2. Spring 1998, pp. 15–21.

Panko, R. R., & Aurigemma, S. (2010). Revising the Panko-Halverson taxonomy of spreadsheet errors. *Decis. Support Syst., 49*(2), 235–244.

Pólya, G. (1954). How To Solve It. A New Aspect of Mathematical Method. Second edition (1957) Princeton University press, Princeton, New Jersey (1954).

Powell, S. G., Baker, K. R., & Lawson, B. (2008). A critical review of the literature on spreadsheet errors. *Decision Support Systems, 46*(1), 128–138.

Powell, S. G., Baker, K. R., & Lawson, B. (2009a). Errors in operational spreadsheets. *Journal of Organizational And End User Computing, 21*(3), 24–36.

Powell, S. G., Baker, K. R., & Lawson, B. (2009b). Impact of errors in operational spreadsheets. *Decision Support Systems, 47*(2), 126–132.

Sebestyén, K., Csapó, G., & Csernoch, M. (2018). Visualising Sprego inequality problems with 2D representations. The Turkish online journal of educational technology, INTE 2018 November (2), 888–898. Retrieved 22/04/2020 from http://www.tojet.net/special/2018_12_3.pdf.

Sestoft, P. (2011). *Spreadsheet technology* (p. 20). IT University of Copenhagen: IT University Technical Report Series.
Shams, L., & Seitz, A. R. (2008). Benefits of multisensory learning. *Trends in cognitive sciences, 2008, 12*(11), 411–417.
Skemp, R. (1971). *The psychology of learning mathematics*. New Jersey: Lawrence Erlbaum Associatives.
Sweller, J., Ayres, P. & Kalyuga, S. (2011). Cognitive load theory. Springer New York Dordrecht Heidelberg London.
Swidan, A., Hermans, F. (2019). The effect of Reading code aloud on comprehension: An empirical study with school students. CompEd '19 proceedings of the ACM conference on global computing education. Pp. 178-184.
Szlávi, P., Zsakó, L., & Törley, G. (2019). Programming theorems have the same origin. *Central-European Journal of New Technologies in Research, Education and Practice, 2019, 1*, 1–12.
Teo, T. S. H., & Tan, M. (1999). Spreadsheet development and 'what-if' analysis: Quantitative versus qualitative errors. *Accounting, Management and Information Technology, 9*, 141–160.
The R Foundation (2019). The R Project for Statistical Computing. Retrieved 12/06/2018 from https://www.r-project.org/
Tort, F. (2010). Teaching spreadsheets: Curriculum design principles. Proceedings of EuSpRIG 2010 conference, 99–110. Retrieved 10/10/2019 from http://arxiv.org/abs/1009.2787
Tort, F., Blondel, F.-M., & Bruillard, É. (2008). Spreadsheet Knowledge and Skills of French Secondary School Students. In R. T. Mittermeir & M. M. Sysło (Eds.), *LNCS 5090, 305–316, 2008*. Berlin Heidelberg: Springer-Verlag.
van Merriënboer, J. J. G., & Sweller, J. (2005). Cognitive load theory and complex learning: Recent developments and future directions. *Educational Psychology Review, 2005, 17*(2), 147–177.
Vandeput, E. (2009). Milestones for teaching the spreadsheet program. In proceedings of EuSpRIG 2009 conference, 133–143. Retrieved 10/10/2019 from http://arxiv.org/abs/0908.1189
W2 (2012): Report of JPMorgan Chase & Co. Management Task Force. Regarding (2012): CIO Losses. Retrieved 08/17/2019 from http://files.shareholder.com/downloads/ONE/2272984969x0x628656/4cb574a0-0bf5-4728-9582-625e4519b5ab/Task_Force_Report.pdf
Wing, J. M. (2006). Computational thinking. *Communications of the ACM, 49*(3), 33. https://doi.org/10.1145/1118178.1118215.

Affiliations

Gábor Csapó [1] · Katalin Sebestyén [1] · Mária Csernoch [2] · Kálmán Abari [3]

✉ Katalin Sebestyén
sebestyen.katalin@inf.unideb.hu

Gábor Csapó
csapo.gabor@inf.unideb.hu

Mária Csernoch
csernoch.maria@inf.unideb.hu

Kálmán Abari
abari.kalman@arts.unideb.hu

[1] Doctoral School of Informatics, University of Debrecen, 26 Kassai út, Debrecen 4028, Hungary

[2] Faculty of Informatics, University of Debrecen, 26 Kassai út, Debrecen 4028, Hungary

[3] Faculty of Arts and Humanities, University of Debrecen, 1 Egyetem tér, Debrecen 4032, Hungary