Sprego, End-user Programming in Spreadsheets

Maria Csernoch
Faculty of Informatics
University of Debrecen
csernoch.maria@inf.unideb.hu

Piroska Biró
Faculty of Informatics
University of Debrecen
biro.piroska@inf.unideb.hu

Abstract
The non-acceptance of end-user computing, especially end-user programming leads to misconceptions and consequently to the underdevelopment of the wide public in computer-supported problem solving. To find methods in connection with end-user computing, the available sources and their application and usage had to be analysed, from which a selection is presented in this paper. On the contrary to the surface navigation approaches, we provide the essence of Sprego programming, which is a high mathability, computer-supported real world problem solving approach in spreadsheet environments, along with its theoretical background and tools. Sprego heavily relies on the previously published results which proved that functional languages serve novice programmers better than imperative languages, the functional data flow modelling, Technological Pedagogical Content Knowledge in spreadsheets, the functional language built into spreadsheets, and their simple interface which lessens the coding burden. We have found and proved that Sprego is an effective programming approach in end-user computing and beyond that it supports knowledge transfer between the subfields of computers sciences and other traditional sciences.

1. The state of art

1.1. The acceptance of end-user computing

> "This paper presents a model for recognition of errors in documents … This is a shame: I like the paper's subject matter; but this does not mean I believe that it belongs in a computing education conference." (private collection)

The above citation clearly describes the non-acceptance of end-user computing within computer science education, which is in accordance with the statement of Panko & Port (2013), who found that end-user computing seems invisible to IT professionals, corporate managers, and information system researchers. Beyond that we are faced with another misconception, namely that computer science is identified with computer driving license (Freiermuth et al., 2008; Hromkovič, 2009; Csernoch, 2017). Consequently, end-user computing is led by profit-oriented software companies, mostly proposing their belief in the fix nature of sciences (Wolfram, 2010, 2015; Chen et al., 2015; Csernoch, 2017), focusing on marketable software interfaces, which leads to secluded and mindless usage of tools, and ultimately to an extremely high number of error prone documents (Ben-Ari, 2011; Csernoch & Biró, 2014b, 2015e; Bewig, 2005; Burnett, 2009; EuSpRIG horror stories, 2015; Panko, 2008; Pemberton & Robson, 2000; Spreadshite, 2015; Thorne & Ball, 2008). In general, knowledge-transfer between the subfields of informatics/computer sciences and other traditional sciences is barely detectable. The wide public – end-users, not rarely over confident end-users (Panko 2015; SCF, 2016) – are not trained to use computers in real world problem solving but they are guided and specialized – if not self-taught – to pass exams and to carry out limited, problem oriented tasks.

1.2. Functions: a link between end-user programming, programming, and mathematics

In the course of analysing the recommendations of the different national curricula and the practices, we have come to the conclusion that much less attention is paid to the notion of function in spreadsheet environments. The functional language of spreadsheets, the concept of function and the general behaviour of functions are not emphasized. This negligence of function has a noticeable consequence in that spreadsheets are not considered as a practicing field of calling, applying functions when dealing with real world problems. We can find sources which claim that functional modelling and functional languages can better serve as introductory languages than imperative languages, but these findings have not reached the wider public (Booth, 1992; Lovászová & Hvorecký, 2003; Hubwieser, 2004, Schneider, 2004, 2005, Sestoft, 2011; Warren, 2004, Kadijevic, 2013).

Spreadsheet functions in general are n-ary, and in special cases variadic functions. Teaching them both in maths and in ICT/CS classes would be a great opportunity to introduce n-ary functions in practice, to give examples of how the arguments can be filled in, what role the order of the arguments plays, how the different data types can be taken care of, how composite functions can be created and evaluated, and how the arguments and the values of the embedded functions are connected. This approach is in accordance with long expressed results of research in developing functional thinking "Research, including early algebra research, suggests that students' flexibility with multiple representations both reflects and promotes deeper mathematical insights (Behr et al., 1983; Brizuela & Earnest, 2008; Goldin & Shteingold, 2001). Brizuela & Earnest note that 'the connections between different representations help to resolve some of the ambiguity of isolated representations, [so] in order for concepts to be fully developed, children will need to represent them in various ways'." (Blanton & Kaput, 2011).

1.3. Technological Pedagogical Content Knowledge (TCPK)

Our arguing considering the acceptance and utilization of end-user programming is in close connection with TCPK which clearly states that "…it is not sufficient for teachers to be knowledgeable about technology or pedagogy in order to use technology efficiently in the classroom. … teachers need also to know how technology can be integrated with specific content in meaningful ways." (Mishra & Koehler, 2006). Angeli went one step further and detailed the conditions in which TCPK can be used effectively in spreadsheet environments: " teachers need to (1) develop educational rationale about why spreadsheet are important to teach, (2) understand the educational affordances of spreadsheet in teaching particular content domain, (3) identify content domains that can benefit from the use of spreadsheets, (4) be knowledgeable of students' learning difficulties with spreadsheets, and (5) teach spreadsheets within the context of a meaningful curriculum topic" (Angeli, 2013).

2. Programming vs. "user-friendly" spreadsheet management

"I think computers are the greatest tool for conceptually understanding math. … they liberate you from calculating to think at a higher level. But like all tools, they can be used completely mindlessly…" (Wolfram, 2010).

What we primarily experience in spreadsheet environments is the mindless usage of tools. Spreadsheet programs in general do not support functionality and programming. The software companies prefer to communicate the "user friendly" aspect of these programs and environments. They declare that there is no need for any background knowledge in order to use these programs, since the available software tools can fulfil the users' aims perfectly. Unfortunately, this approach is almost unconditionally accepted in education, where most teachers and teaching materials communicate the software companies' profit oriented slogans, instead of focusing on the algorithmic approach to problem solving in spreadsheet environments. The software companies' user friendly slogans emphasize the role of the environments, and they introduce more and more novel surface tools which enchant end-users, without giving any further thought to the problems. Unfortunately, this approach is highly supported by the widely accepted ECDL exams (Csernoch, 2017), the recently published Spreadsheet Competency Framework (SCF, 2016), teaching and learning materials – including printed and online coursebooks, built-in wizards and helps, and various ICT/CS curricula.

However, within spreadsheets there are tools available, although they are not emphasized, which serve high mathability problem solving, for short end-user programming. In this framework we have introduced Sprego – Spreadsheet Lego –, which fulfils all the requirements of high mathability, computer-supported real world problem solving in spreadsheet environments (Csernoch & Biró, 2013, 2014a, 2015a, 2015b, 2015c, 2015d, 2015e; Biró & Csernoch, 2014, Biró et al., 2015a, 2015b). Sprego matches the requirements of functional data flow modelling, takes further advantage of functional languages (Booth, 1992, Sestoft 2011; Hubwieser, 2004; Warren, 2004; Scheider, 2004, 2005), relies heavily on real world artefacts – authentic tables in the present environment –, the concept of function, Boolean expressions, n-dimensional vectors, and discussion and debugging.

We claim that these concepts can be introduced as early as primary maths and ICT/CS classes, and show, on one hand, how teaching mathematics and ICT/CS can be reformed, in accordance with both Wolfram (2010) and Gove (2012), on the other hand, how we can provide various examples which can develop the students' rule-recognition and rule-following skills, help building their concept of function (Blanton & Kaput, 2011; Skemp, 1987, Vuorikari et al., 2016), and teach them programming in end-user environments. These are the tools which end-users need to make fast but reliable decisions (Csernoch, 2017).

3. Sprego: end-user programming in functional languages

3.1. Problem solving in Sprego

Sprego (Csernoch, 2014; Csernoch & Biró, 2015b, 2015c) focuses on the programming and data management aspects of spreadsheets, relies heavily on the concept of function, emphasizes the role of multilevel and multivariable – n-ary – functions, and applies them intensively. The concept and algorithmic aspects are at the centre of attention when using Sprego, similar to mathematics (Pólya, 1954) and to other programming and data management environments (Vuorikari et al., 2016).

3.2. Sprego functions

Sprego declares that instead of the 500+ "user friendly" functions of spreadsheets, only a dozen general purpose functions – Sprego12 – would serve as the basis for an effective high mathability problem solving method (Csernoch, 2014; Biró & Csernoch 2015a, 2015b, Csernoch, 2017). This predicted number is in accordance with findings in programming (Hromkovič, 2014; Mayer, 1981) and in general spreadsheet use (Walkenbach, 2010).

Sprego12 contains the following functions:

- handling strings: LEFT(), RIGHT(), LEN(), SEARCH() (in which students were found better than handling numbers (Szanyi, 2015)
- handling numbers: SUM(), AVERAGE(), MAX(), MIN()
- making decisions based on yes/no question(s)conditions, handling vectors, and handling errors: IF(), MATCH() and INDEX(), and ISERROR().

We have to emphasize here that the set of Sprego functions is an open set; consequently, any general purpose function can be added, according to the problems which emerge (Csernoch, 2014).

3.3. Authentic tables

It is obvious that spreadsheet environments and tables work as sources of both functional modelling and problem solving; however we have to be aware that setting school tasks in the context of 'real world' situations, for example through the use of word problems, is not sufficient to make them meaningful for pupils (Angeli, 2013;. Ainley & Pratt claimed (2005) that "…there is considerable evidence of the problematic nature of pedagogic materials which contextualize mathematics in supposedly real-world settings, but fail to provide a purpose that makes sense to pupils. … We see the purposeful nature of the activities as a key feature of out-of-school contexts which can be brought into the classroom through the creation of well-designed tasks." However, badly-designed 'realistic' test items and tasks do not serve the original purpose (Cooper & Dunne, 2000) for various reasons. One of the main reasons is that students' lack of background knowledge does not allow them to build the concept, and from that point on, solve the problem (Csernoch et al., 2015; Csernoch & Biró, 2016).

Burnett (2009) discussed similar findings focusing on end-user programming and programmers. She claimed that end-user programmers tend to focus on the content and the problems which they are interested in. The common ground of these different approaches is the use of authentic tables from real world contexts. Furthermore, research has clearly proved that one of the reasons for failure when teaching spreadsheets is the decontextualized and technocentric teaching methods (Angeli, 2013; Csernoch & Biró, 2016; Mireault, 2016; Csernoch, 2017).

Authentic tables allow students to carry out real data analyses: here the focus is not on knowledge brought into the class from outside, but on knowledge offered by the table. By becoming familiar with the table at the beginning of the process of problem solving, students can grasp the characteristics of the data and can also reveal connections between the various items. This explains the students' preference of text-based problems.

3.4. Array formulas

Beyond the Sprego functions, another hardly known spreadsheet tool is applied in Sprego, namely the array formula (Sestoft, 2011; Walkenbach, 2010; Wilcox & Walkenbach, 2003). Software companies recommend copying the formulas, even though this method has been recognized as one of the main sources of spreadsheet errors (Panko, 2013) along with the different reference types and their learning difficulties (Angeli, 2013), which are unavoidable when formulas are copied.

At the beginning of the learning process, array formulas rules out copying formulas and cell references, and leaving space for these concepts in advanced studies. On the other hand, we can introduce one-dimensional arrays, vectors, and use them intensively in problem solving. With this tool, we can introduce the n-dimensional vector in maths classes also, which is considered higher mathematics in most curricula, but essential in high level programming languages.

3.5. Data types

"Serious" programming and data management blame spreadsheets for negligent data-type-management (Panko & Port, 2013; Vágner & Zsakó, 2015). However, we argue that automated type recognition would serve as a convenient tool for novices. In this environment students do not have to handle the different data types manually, consequently, they can focus on the problem, the model, and the content of arguments instead of the coding details. The immediate output would also help students recognizing the data type of the output of the formulas (Data Sets, 2008). Beyond this, consciously pre-prepared tables – depending on the level of students and the goal(s) of the tasks and classes – would allow teachers to apply them with various purposes.

According to Schneider (2005), the subject of Boolean data types and Boolean functions has disappeared from some curricula in mathematics. However, in informatics it is a fundamental data type. In Sprego, Boolen expressions are introduced in the form of yes/no questions – the phenomenon transferred from language studies – and primarily used to as the 'inside' formula for making decisions based on the answers to the questions. Students at a very early age would form yes/no questions and decide on the output based on the answers.

Consequently, introducing Boolean data type and functions in a computer environment would also change the maths curricula. It is not only the output of formulas which can be considered in relation to data types but also the arguments and operands of these formulas. To make these formulas work the students have to be aware of the data types which the functions and the operators can accept. In spreadsheets we do this without the tiresome direct definition and declaration of variables and arrays.

3.6. Discussing, debugging

Spreadsheets are also blamed for not supporting discussion and debugging. It is true that built-in functions and the algorithms behind them are undetectable (Csernoch, 2014). However, by applying Sprego we are able to evaluate the formulas step-by-step, just as we do in "serious" programming (Csernoch, 2014, 2015; Biró & Csernoch, 2013, 2014, 2015a, 2015b; Csernoch & Biró, 2014a, 2014b, 2015b, 2015c).

However, the discussion and debugging of solutions involves far more than checking the syntactic correctness of the formulas. In spreadsheets, due to automatic type recognition, the data types always have to be thoroughly checked by the user. Automatic type recognition is both a blessing and a curse. On one hand, it helps beginners to recognize the assigned data types along with the types of the output values of the functions and reduces the coding burden, which plays a crucial role in end-user programming (Section 3.5). On the other hand, automated data type recognition would be irreversible. Undesired conversions have to be corrected or techniques have to be found to prevent automated data type recognition (Csernoch, 2014; Csernoch, 2015). Beyond these errors, similar to programming, semantic errors are the most demanding in spreadsheets.

In general, we can conclude that it is primarily the user's responsibility to handle the errors in their work. In a teaching-learning situation, discussion and debugging, both as pre- and in-class activities, play a crucial role.

3.7. Sprego coding

Once the model and the algorithm (Csernoch, 2014; Csernoch & Biró, 2015b, 2015c; Schneider, 2005) are clarified to a problem, the coding is carried out in a spreadsheet environment (Csernoch, 2014; Csernoch & Biró, 2015b, 2015c; Schneider, 2005). Sprego supports the building of composite functions, however, it is always the user's decision whether to introduce additional variables or arrays for displaying partial results or not – similar to traditional programming environments.

Building composite or embedded functions, we start with the innermost function. The output of the function is one of the arguments of the function outside the first one. This second function has an output again, and so on, until we reach the outermost function, whose output is the solution of the problem. At present, composite functions are barely taught in general education, however, with Sprego this concept can also be introduced at a very early age (for further details see Sections 3.8).

In Sprego the process of problem solving does not focus on browsing through the 500+ functions, but rather on the algorithm, and on calling with fast thinking a couple of familiar functions in the coding process (Kahneman, 2011). In the case of building the multilevel functions we advance from inside out, all the steps can be evaluated, the results can be displayed step-by-step, and as such, discussed and debugged. Consequently, the other advantage of this coding method is that it reduces the risk of creating erroneous spreadsheet documents. The application of array formulas allows us avoiding the repeated copying of the extracted formulas, so one frequent source of errors in spreadsheet documents.

Using additional variables and arrays and displaying all the outputs of the algorithms step-by-step has its advantages and disadvantages. Its advantages are that formulas are simple – holding only one step – and the output of each step is clearly presented. The disadvantages are that a vector or a variable has to be created for each step and the spreadsheet would be loaded with unnecessary data.

3.8. Unplugged and semi-unplugged Sprego tools

The unplugged tools invented for supporting Sprego programming would help youngsters and novice end-user programmers in building functional models and composite structures.

The simplest unplugged tools are shortened but enlarged text-based printed samples which students can cut into pieces according to the requirements of the tasks. The printed or semi-printed forms – prepared for printing but displayed on digital tools – of tables serve students as playgrounds where they can carry out the algorithms manually.

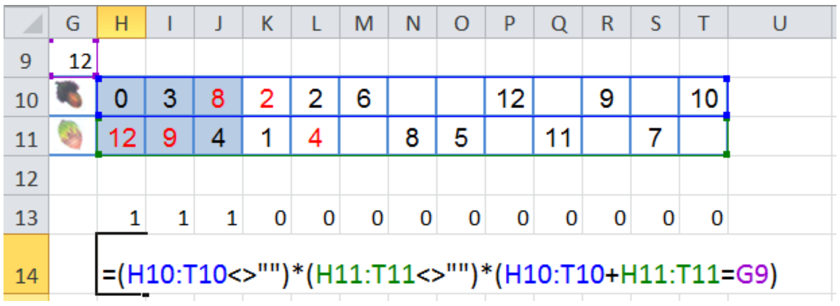
However, the main attraction of these tools is the hand-made or pre-prepared matrjoska dolls with all their accessories. A set of matrjoska dolls is a handy tool for building composite functions. The students can work with their own dolls and there is one additional set for the teacher to work on the board. Even the uncompleted sets can be used – pieces mysteriously disappear –, e.g. for displaying the mismatched pairs of parentheses.

In our experience the set of dolls are accompanied with paper balls, post-its, and stickers. The pieces of paper serve as the input variables or arrays, which can be inserted into the dolls. On the stickers the code, the steps of the algorithm, the output data, and/or the output data type can be written and placed around the doll. These stickers also serve as a tool to "close" a doll (a function), showing that whatever happens inside the doll, by closing it we only see the output, and this value serves as an input for the outside doll (function). For further specification, e.g. for distinguishing the different data types we can use different colours of pens or tapes. When students disassemble their dolls, there is an opportunity to go through again the functional model, the algorithm. Beyond that the tapes can be stickered into the students' notebook in the proper order, without additional writing exercises.

However, the original matrjoska dolls are too valuable for frequent usage in classes. Consequently, we had to develop methods for creating dolls. We can buy readymade sets of barrels in stores. They would serve well, but still expensive. Beyond that we have experienced that the order of the colours of the sets is not necessarily the same, and usually does not match the colour of the teacher's set. We found that our own 3D-printed sets serve us the best. There are no colour problems, the students can use their own sets from classes to classes, and they are available at a reasonable price. On further solution for creating dolls is origami. We have found that origami boats can be folded into closed objects, which can hold the smaller ones. The advantages of the paper boats are that students can create them at the beginning of the class – it takes only a couple of minutes –, the number of boats needed depends on the number of steps of the algorithm, there is no need for tapes and stickers – we can write on the side of the boats –, and finally, the sets can be inserted into the notebooks, as a complete task.

The pre-service teachers of our faculty are developing mobile applications with small animations for the fundamental Sprego problems (Csapó & Sebestyén, 2015). The animations display the steps of the algorithms in simple contexts, whose avatars are the matrjoska dolls. The application is under construction and development but the actual version is available and the students are open for further suggestions (Csapó & Sebestyén, 2017).

Functions can be introduced as early as the first grade maths classes (Blanton & Kaput 2004 cited in Blanton & Kaput, 2011). A typical example of dealing with functions is filling in the missing cells of a table, based on a word-task or the other way around, finding rule(s) based on the sample values of the table (Blanton & Kaput, 2011). Similar but interactive function machines can be created in spreadsheets. One example is presented in Figure 1.



	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
9	12														
10		0	3	8	2	2	6			12		9		10	
11		12	9	4	1	4		8	5		11		7		
12															
13		1	1	1	0	0	0	0	0	0	0	0	0	0	
14		=(H10:T10<>"")*(H11:T11<>"")*(H10:T10+H11:T11=G9)													

Figure 1 – A spreadsheet application and its accompanying formula to simulate a function machine.

4. Summary

We have presented Sprego as a tool for developing the students' notion of function and their programming skills in already existing spreadsheet environments.

In programming languages, handling functions is essential; however, in imperative languages, handling variables and arrays seems extremely demanding, especially for those who do not want to be professional programmers. In declarative languages the focus is much more on the problems than on the coding details. Being aware of the advantages of functional languages, the simplified environment of spreadsheets and the concept of functions introduced in maths classes, we developed a method which mathematics, ICT/CS, and end-user computing can benefit from.

5. Conclusions

It is generally understood that there is a great need for fundamental changes in both the maths and ICT/CS curricula and we are in great need of methods supporting computer-aided real world problem solving. However, it seems less obvious that the maths and ICT/CS subjects are interwoven and changes introduced in one should affect the other. It is claimed that programming would help students in developing their procedural thinking better than other previously accepted subjects, but further connections have hardly been made. Some of the approaches focus on imperative languages, while others search for less traditional programming tools. In spite of the obvious connection between the two subjects through functions, less attention has been paid to functional languages and the roles they would play in teaching functions in mathematics.

With Sprego we can revolutionize the teaching of composite and n-ary functions, can introduce vector and Boolean expressions in basic maths curricula, and offer a spreadsheet-based high mathability problem solving approach. Beyond supporting studies in mathematics, Sprego fulfils the need for an applicable method in functional languages; it takes advantage both of the functional languages and the familiar environments of spreadsheets.

Beyond the obvious connection to mathematics and programming, the extremely high number of available authentic tables would provide real world problems to solve in various classes. With this tool, Sprego is TCPK compatible, which emphasizes the importance of those contents which are interesting for the students, connected to other subjects and generate questions, and lead to real discussions and debugging.

6. References

Ainley, J., & Pratt, D. (2005). The Dolls' House Classroom. In Hlen L. Chick & Jill L. Vincent (Ed.), *Proceedings of the 29th Conference of the International Group for the Psychology of Mathematics Education*. PME29, Melbourne, Australia, 114–122.

Angeli, C. (2013). Teaching Spreadsheets: A TPCK Perspective. In *Improving Computer Science Education*. (Eds.) D. M. Kadijevich, C. Angeli, and C. Schulte. Routledge.

Ben-Ari, M. (2011). Non-myths about programming. *Communications of the ACM*, 54(7), 35. http://doi.org/10.1145/1965724.1965738.

Bewig, P. L. (2005). How do you know your spreadsheet is right? Principles, Techniques and Practice of Spreadsheet Style. arXiv:1301.5878 [cs]. Retrieved January 12, 2016 from http://arxiv.org/abs/1301.5878.

Biró, P., & Csernoch, M. (2013). Deep and surface structural metacognitive abilities of the first year students of Informatics. In *2013 IEEE 4th International Conference on Cognitive Infocommunications (CogInfoCom)* (pp. 521–526). http://doi.org/10.1109/CogInfoCom.2013.6719303.

Biró, P., & Csernoch, M. (2014). Deep and surface metacognitive processes in non-traditional programming tasks. In *2014 5th IEEE Conference on Cognitive Infocommunications (CogInfoCom)* (pp. 49–54). http://doi.org/10.1109/CogInfoCom.2014.7020507.

Biró, P., & Csernoch, M. (2015a). The mathability of computer problem solving approaches. In *2015 6th IEEE International Conference on Cognitive Infocommunications (CogInfoCom)* (pp. 111–114). http://doi.org/10.1109/CogInfoCom.2015.7390574.

Biró, P., & Csernoch, M. (2015b). The mathability of spreadsheet tools. In *2015 6th IEEE International Conference on Cognitive Infocommunications (CogInfoCom)* (pp. 105–110). http://doi.org/10.1109/CogInfoCom.2015.7390573.

Biró, P., Csernoch, M., Máth, J., & Abari, K. (2015a). Measuring the Level of Algorithmic Skills at the End of Secondary Education in Hungary. *Procedia – Social and Behavioral Sciences*, 176, 876–883. http://doi.org/10.1016/j.sbspro.2015.01.553.

Biró, P., Csernoch, M., Máth, J., & Abari, K. (2015b). Algorithmic Skills Transferred from Secondary CSI Studies into Tertiary Education. *International Journal of Social Education Economics and Management Engineering* 9(2), 426–432.

Blanton, M. L., & Kaput, J. J. (2011). Functional Thinking as a Route Into Algebra in the Elementary Grades. In J. Cai & E. Knuth (Eds.), *Early Algebraization* (pp. 5–23). Berlin Heidelberg, Germany: Springer. Retrieved January 25, 2016, from http://link.springer.com/chapter/10.1007/978-3-642-17735-4_2.

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

Burnett, M. (2009). What Is End-User Software Engineering and Why Does It Matter? In V. Pipek, M. B. Rosson, B. de Ruyter, & V. Wulf (Eds.), *End-User Development* (pp. 15–28). Berlin Heidelberg, Germany: Springer. Retrieved January 25, 2016, from http://link.springer.com/chapter/10.1007/978-3-642-00427-8_2.

Csapó, G. & Sebestyén, K. (2015) Teaching Application for Sprego Programming. In Hungarian: Oktatóprogram a Sprego táblázatkezelő módszerhez. In: Szlávi Péter, Zsakó László (szerk.) *INFODIDACT 2015.* Zamárdi, Hungary, Budapest: Webdidaktika Alapítvány, 2015. pp. 1-18.

Csapó, G. & Sebestyén, K. (2017). Sprego application. Retrieved April 25, 2017, from https://play.google.com/store/apps/details?id=hu.sprego.oktatoprogram

Chen, J. A., Morris, D. B., & Mansour, N. (2015). Science Teachers' Beliefs. Perceptions of Efficacy and the Nature of Scientific Knowledge and Knowing. In *International Handbook of Research on Teachers' Beliefs.* (Eds.) Fives, H. & Gill, M. G. Routledge, 370–386.

Cooper, B. & Dunne, M. (2000). Assessing Children's Mathematical Knowledge: Social Class, Sex, and Problem-solving. Buckingham, Philadelphia, PA: Open University Press.

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

Csernoch, M. (2015). *Algorithms and Schemata in Teaching Informatics.* Debreceni Egyetemi Kiadó, Debrecen. Retrieved January 25, 2016, from http://tanarkepzes.unideb.hu/szaktarnet/kiadvanyok/algoritmusok_es_semak_2.pdf.

Csernoch, M. (2017). Thinking Fast and Slow in Computer Problem Solving. *Journal of Software Engineering and Applications.* 10(1), Article ID:73749,30 pages 10.4236/jsea.2017.101002.

Csernoch, M., & Biró, P. (2013). Teachers' Assessment and Students' Self-Assessment on the Students' Spreadsheet Knowledge. *EDULEARN13 Proceedings*, 949–956. Retrieved from https://library.iated.org/view/CSERNOCH2013TEA.

Csernoch, M., & Biró, P. (2014a). Spreadsheet misconceptions, spreadsheet errors. (Eds.) Juhász, E., Kozma, T., *Oktatáskutatás határon innen és túl. HERA Évkönyvek I.*, Belvedere Meridionale, Szeged, 370–395.

Csernoch, M., & Biró, P. (2014b). Digital Competency and Digital Literacy is at Stake. In *ECER 2014, The Past, Present and Future of Educational Research in Europe*. University of Porto, Porto, 1–4.

Csernoch, M., & Biró, P. (2015a). Computer Problem Solving. In Hungarian: Számítógépes problémamegoldás, TMT, *Tudományos és Műszaki* Tájékoztatás, Könyvtár- és információtudományi szakfolyóirat, 62(3), 86–94.

Csernoch, M., & Biró, P. (2015b). Sprego Programming. *Spreadsheets in Education* (eJSiE), 8(1). Retrieved from http://epublications.bond.edu.au/ejsie/vol8/iss1/4.

Csernoch, M., & Biró, P. (2015c). *Sprego programming.* LAP Lambert Academic Publishing. ISBN-13: 978-3-659-51689-4.

Csernoch, M., & Biró, P. (2015d). Wasting Human and Computer Resources. *International Journal of Social, Education, Economics and Management Engineering*, 9(2), 573–581.

Csernoch, M., & Biró, P. (2015e). The Power in Digital Literacy and Algorithmic Skill. *Procedia - Social and Behavioral Sciences,* 174, 550–559. http://doi.org/10.1016/j.sbspro.2015.01.705.

Csernoch, M. & Biró, P. (2016) Utilizing Sprego and Sprego contents. In: Jan Vahrenhold, Erik Barendsen WIPSCE '16: *Proceedings of the 11th Workshop in Primary and Secondary Computing Education.* 2 p. Münster, Germany, New York: ACM Press, 2016. pp. 102-103.

Csernoch, M., Biró, P., Máth, J., and Abari, K. (2015). Testing Algorithmic Skills in Traditional and Non-Traditional Programming Environments. *Informatics in Education,* 14(2), 175–197. http://doi.org/10.15388/infedu.2015.11.

Data Sets (2008). Updated: 19. 08. 2008. Retrieved May 25, 2015, from http://mathforum.org/workshops/sum96/data.collections/datalibrary/data.set6.html.

EuSpRIG Horror Stories (n.d.). Retrieved July 18, 2015, from http://www.eusprig.org/horror-stories.htm.

Freiermuth, K., Hromkovič, J., & Steffen, B. (2008). Creating and Testing Textbooks for Secondary Schools. In R. T. Mittermeir & M. M. Sysło (Eds.), *Informatics Education - Supporting Computational Thinking* (pp. 216–228). Berlin Heidelberg, Germany: Springer. Retrieved January 25, 2016, from http://link.springer.com/chapter/10.1007/978-3-540-69924-8_20.

Gove, M. (2012). Digital literacy campaign – Michael Gove's speech in full. BETT 2012. The Guardian. Retrieved January 25, 2016, from http://www.theguardian.com/education/2012/jan/11/digital-literacy-michael-gove-speech.

Hromkovic, J. (2009). *Algorithmic Adventures*. Berlin, Heidelberg, Germany: Springer Berlin Heidelberg.

Hromkovič, J. (2014). *Introduction to Programming with LOGO*. In German: Einführung in die Programmierung mit LOGO. Wiesbaden, Germany: Springer Fachmedien Wiesbaden.

Hubwieser, P. (2004). Functional Modelling in Secondary Schools Using Spreadsheets. *Education and Information Technologies*, 9(2), 175–183. http://doi.org/10.1023/B:EAIT.0000027929.91773.ab.

Kadijevich, D. (2013). Learning about spreadsheet. In Kadijevich, D., Angeli, C., and Schulte, C. (Eds.). (2013). *Improving Computer Science Education*. (pp. 19–33) New York: Routledge.

Kahneman, D. (2011). *Thinking, Fast and Slow*. New York: Farrar, Straus; Giroux.

Lovászová, G., & Hvorecký, J. (2003). On Programming and Spreadsheet Calculations. *Spreadsheets in Education* (eJSiE), 1(1) 44–51. Retrieved January 25, 2016, from http://epublications.bond.edu.au/ejsie/vol1/iss1/3.

Mayer, R. E. (1981). The Psychology of How Novices Learn Computer Programming. *ACM Computing Surveys* 13(1), 121–141. http://doi.org/10.1145/356835.356841

Mireault, P. (2016), Characteristics of Spreadsheets Developed with the SSMI Methodology. EuSpRIG2016. Retrieved July 21, 2016, from http://www.eusprig.org/pmireault-2016.pdf.

Mishra, P. & Koehler, M. J. (2006). Technological Pedagogical Content Knowledge: A new framework for teacher knowledge. *Teacher College Record*, 108(6), 1017-1054.

Panko, R. R. (2015). What We Don't Know About Spreadsheet Errors Today: The Facts, Why We Don't Believe Them, and What We Need to Do." arXiv:1602.02601 [cs]. Retrieved July 21, 2016, from http://arxiv.org/abs/1602.02601.

Panko, R. R. (2008). What We Know About Spreadsheet Errors. *Journal of End User Computing's. Special issue on Scaling up End User Development*. 10(2), 15–21.

Panko, R. R. (2013). The Cognitive Science of Spreadsheet Errors: Why Thinking is Bad. In *2013 46th Hawaii International Conference on System Sciences* (Vol. 0, pp. 4013–4022). Wailea, HI. http://doi.org/10.1109/HICSS.2013.513.

Panko, R. R., & Port, D. (2013). End User Computing: The Dark Matter (and Dark Energy) of Corporate It. *Journal of Organizational and End User Computing*, 25(3), 1–19.

Pemberton, J. D., & Robson, A. J. (2000). Spreadsheets in business. *Industrial Management & Data Systems*, 100(8), 379–388. http://doi.org/10.1108/02635570010353938

Pólya, G. (1954). *How to Solve It. A New Aspect of Mathematical Method*. (2nd Edition 1957), Princeton, NJ: Princeton University Press.

Schneider, M. (2004). An Empirical Study of Introductory Lectures in Informatics Based on Fundamental Concepts. In *Informatics and Students Assessment*, *Lecture Notes in Informatics* 1, 123–133.

Schneider, M. (2005). A Strategy to Introduce Functional Data Modeling at School Informatics. In R. T. Mittermeir (Ed.), *From Computer Literacy to Informatics Fundamentals* (pp. 130–144). Berlin Heidelberg, Germany: Springer. Retrieved from http://link.springer.com/chapter/10.1007/978-3-540-31958-0_16.

Sestoft, P. (2011). Spreadsheet technology. Version 0.12 of 2012-01-31. IT University Technical Report ITU-TR-2011-142. Copenhagen, Denmark: IT University of Copenhagen.

Skemp, R. (1971). *The Psychology of Learning Mathematics.* Lawrence Erlbaum Associatives, New Jersey, USA.

Spreadsheet competency framework (SCF). (2016), A structure for classifying spreadsheet ability in finance professionals. ICAEW. Retrieved November 21, 2016 from http://www.icaew.com/-/media/corporate/files/technical/information-technology/it-faculty/spreadsheet-competency-framework.ashx

Spreadshite. Fighting Spreadsheet Blight in Business Today (n.d.). Retrieved May 1, 2015, from http://spreadshite.com/home.html.

Szanyi, Gy. (2015). The investigation of students' skills in the process of function concept creation. *Teaching Mathematics and Computer Sciences*, 13(2), 249–266.

Thorne, S., & Ball, D. (2008). Considering Functional Spreadsheet Operator Usage Suggests the Value of Example Driven Modelling for Decision Support Systems. arXiv:0803.0164 [cs]. Retrieved January 25, 2016, from http://arxiv.org/abs/0803.0164.

Vágner, A., & Zsakó, L. (2015). Negative Effects of Learning Spreadsheet Management on Learning Database Management. *Acta Didactica Napocensia.* 8(2), 1-6. Retrieved January 25, 2016, from http://padi.psiedu.ubbcluj.ro/adn/article_8_2_1.pdf.

Vuorikari, R., Punie, Y., Carretero Gomez S. & Van den Brande, G. (2016). DigComp 2.0: The Digital Competence Framework for Citizens. Update Phase 1: The Conceptual Reference Model. Luxembourg Publication Office of the European Union. Retrieved April 13, 2017 from http://publications.jrc.ec.europa.eu/repository/bitstream/JRC101254/jrc101254_digcomp%20 2.0%20the%20digital%20competence%20framework%20for%20citizens.%20update%20phase%201.pdf

Walkenbach, J. (2010). Excel 2010 Bible. Retrieved December 13, 2015, from http://www.seu.ac.lk/cedpl/student%20download/Excel%202010%20Bible.pdf.

Warren, P. (2004). Learning to Program: Spreadsheets, Scripting and HCI. In *Proceedings of the Sixth Australasian Conference on Computing Education* 30 (pp. 327–333). Darlinghurst, Australia: Australian Computer Society, Inc. Retrieved January 25, 2016, from http://dl.acm.org/citation.cfm?id=979968.980012.

Wilcox, C., & Walkenbach, J. (2003). Guidelines and examples of array formulas. Retrieved December 13, 2015, from https://support.office.com/en-us/article/Guidelines-and-examples-of-array-formulas-3be0c791-3f89-4644-a062-8e6e9ecee523?CorrelationId=643cf62a-061f-461e-8008-d601efbad369&ui=en-US&rs=en-US&ad=US.

Wolfram, C. (2010, July 15). Stop Teaching Calculating, Start Teaching Math—Fundamentally Reforming the Math Curriculum. Transcript: Wolfram Technology Conference 2010 Talk. TED Global 2010. Retrieved October 12, 2015, from http://www.computerbasedmath.org/resources/Education_talk_transcript.pdf.

Wolfram, C. (2015). Evidence: Let's promote not stifle innovation in education. Retrieved October 12, 2015, from http://www.conradwolfram.com/home/2015/5/21/role-of-evidence-in-education-innovation.