Available online at www.sciencedirect.com

ScienceDirect

Procedia - Social and Behavioral Sciences 174 (2015) 550 – 559

Procedia
Social and Behavioral Sciences

INTE 2014

The power in digital literacy and algorithmic skill

Mária Csernoch[a], Piroska Biró[b]*

[a,b]University of Debrecen, Kassai út 26. Debrecen 4028, Hungary

Abstract

We argue that in educational contexts ICT (Information and Communications Technology) and CS (Computer Sciences) should not be separated. To support our ideas, we present methods related to the minimalist principle with which students with different interests would develop algorithmic skills even in an ICT environment, and this introductory phase would lead the students on to more serious CS studies. The core of these methods is that from the very beginning of CSI education algorithms should be looked for in every computer-related problem. Deep-approach metacognitive methods should be applied, instead of uncontrolled sequences of surface-approach metacognitive activities such as aimless clicking, unplanned wandering, and relying on the newest features in graphical user interfaces (GUI). Our team has developed a deep-approach metacognitive method for teaching spreadsheet to novices, which is in accordance with the concept of building algorithms to solve computer-related problems. The three cornerstones of the method are (1) introducing as simple and as few functions as possible, (2) building multilevel formulas based on these functions, and (3) focusing on the problem instead of the features of the software. As the students make progress, the number of functions would be increased, but general purpose functions would still be focused on. Testing our deep-approach method has proved that it is a lot more effective in teaching spreadsheet than the classical surface-approach, wizard-based metacognitive methods, since all the basic elements are in accordance with the minimalist theory, which advises teaching as simple a language as possible for beginners to develop basic algorithmic skills. Beyond the direct advantages of the method, spreadsheet would serve as an introductory language to high level programming languages, which is our ultimate goal.

© 2015 The Authors. Published by Elsevier Ltd. This is an open access article under the CC BY-NC-ND license (http://creativecommons.org/licenses/by-nc-nd/4.0/).

Peer-review under responsibility of the Sakarya University

Keywords: deep- and surface-approach metacognitive processes; developing algorithmic skill; teaching spreadsheet

* Corresponding author. Tel.: +36-52-512-900/75128; fax: +36-52-512-996.
E-mail address: csernoch.maria@inf.unideb.hu

1877-0428 © 2015 The Authors. Published by Elsevier Ltd. This is an open access article under the CC BY-NC-ND license (http://creativecommons.org/licenses/by-nc-nd/4.0/).

Peer-review under responsibility of the Sakarya University
doi:10.1016/j.sbspro.2015.01.705

1. Introduction

Digital competency and digital literacy have been among the most popular expressions featuring in the curricula of the last few years, and the effect of both of these on different generations – the Y and Z generations of digital natives (the bit-generations) and the X generation of digital immigrants (Dani, 2013; Jukes, McCain & Crockett, 2010) – is one of societies' main concerns. We must give some consideration to these newly coined expressions and some background information related to them. First of all, the most important point is to emphasize that in order to develop digital competency effectively and efficiently, and to achieve digital literacy, formal education is needed. For formal education, teachers are needed, and for teachers, teacher education is needed. At this point the loop is closed, and we face the chicken and the egg problem: who teaches the teachers if there are no teachers? In Computer Science/Informatics (CSI) education this is one of the most crucial questions and for an answer we have to look back in time to the emergence of the subject. The contradictions, both of the science itself, and of the developing commercialized world as it interacted with the science, have affected both teachers and teacher education and consequently the development of digital literacy.

The pioneer teachers were self-educated, in most cases not supervised, and if so, certainly not by experts in the methodology of the subject, because it did not exist. These first teachers mainly taught programming languages, algorithms, binary arithmetic, and computer architecture. Over time they became accepted in their local environment, whether they were qualified or not; they used methods they had developed themselves, without proving their efficiency or effectiveness, due to a lack of time and methods.

In the meantime, computer science developed at an incredible speed, and the new mouse-based graphical user interfaces (GUI) increased the number of users and changed the approach and attitude towards computers. Everyone started to use computers regardless of whether they had any background knowledge, and software developers encouraged them to do so. These companies claimed that by using the GUI and its accompanying wizards the users would be able to solve problems. Users need do nothing more than click here and there and they will find the solution.

Even teachers fell for this, and giving up the teaching of algorithms, switched to aimless clicking, not looking for the algorithms in these new programs; consequently they stopped developing their own and the students' algorithmic skills.

Nomenclature

A	ICT: Information Communication Technologies
B	CS: Computer Sciences
C	CSI: Computer Science/Informatics, including both ICT and CS
D	CAAD-based (Computer-Algorithmic And Debugging-based): deep-approach metacognitive processes of computer related activities with an emphasis on building algorithms and debugging results
E	TAEW-based (Trial-And-Error Wizard-based): surface-approach metacognitive processes of computer related activities highly dependent on the graphical interface

1.1. Fiasco

1.1.1. Facts and results

Recent studies have found that more than 90% of e-documents have errors (Panko & Aurigemma, 2010; Tort, Blondel & Bruillard, 2008; Tort, 2010, Csernoch & Bujdosó, 2009; Csernoch 2010), and uneducated computer users cause serious financial losses by providing unreliable data and by taking much more time than problems require (van Deursen & van Dijk, 2012). Along with these findings, other publications have provided evidence that these mistakes are due to a lack of algorithmic skills and thinking (Angeli 2013; Panko & Aurigemma, 2010; Biró & Csernoch, 2013a, 2013b). However, the Students On Line session of the PISA 2009 survey proved that computer usage in schools does not necessarily increase the level of digital competency (OECD, 2011). This ambiguity clearly indicates that we have serious problems with the methods employed to teach CSI in most countries. Faced with these problems, countries have reacted in different ways. In the United Kingdom in 2012 the number of computer

classes was increased, in order to focus on the development of the algorithmic skill, while in Hungary the number of CSI classes was reduced in the 2013 Core Curricula. In 2012 France introduced Informatique et sciences du numerique (ISN) as an optional subject in the science track for the 12th grade. Despite this, no more than one sixth of French students study CSI only for one year. Who is right? Which path should be followed? Which are the most effective and efficient ways to improve?

The answer lies mainly in the characteristics of the science. The problems of CSI education emerge from the particularities and contradictions of the science, namely that it is:

- a new science without any direct predecessors,
- a science developing at a speed previously unknown in any other science,
- a science with its own commercialized word developing around it, and
- a science operating in the context of the pressures and the needs for computer usage and for information.

In this heavily industrialized science, researches and teachers have to find methods to develop basic algorithmic skills which are effective, efficient, and last but not least, acceptable to the different generations of computer users.

1.1.2. ICT is blamed

Recent reports indicate that office packages are to blame for this fiasco, and they should be banished from CSI courses. One of the most extreme views is Gove's, who states that "Instead of children bored out of their minds being taught how to use Word and Excel by bored teachers, we could have 11 year-olds able to write simple 2D computer animations using an MIT tool called Scratch. By 16, they could have an understanding of formal logic previously covered only in University courses and be writing their own Apps for smartphones." (Gove, 2012).

In our opinion, even these 11 and 16 year-olds should be able to create error-free e-documents. Gove is mistaken when he thinks that CSI classes should not teach how word processing and spreadsheet. These classes should teach the structure of documents and formulas, and the algorithms behind them. Actually, these programs are quite challenging. Typing and aimlessly clicking on the surface should be banished, not the programs themselves. In general, office packages are harmless. They are programs which operate on algorithms, and we have to find and teach the algorithms behind these programs. Those teachers who find these programs boring and make their students think the same are effectively saying that they themselves never have looked for these algorithms.

1.1.3. Surface-approach metacognitive processes – reasons

Our team has launched the Testing Algorithmic and Application Skills (TAaAS) project in the 2011/2012 academic year. In the TAaAS project we tested the Informatics knowledge and the usage of terminology of freshmen students at the Faculty of Informatics of the University of Debrecen, Hungary on the first week of their arrival.

In the TAaAS project it has been proved that most of the computer related activities are metacognitive processes (Biró & Csernoch, 2013a), because users are supposed to read and understand the signs and messages of the GUI and the wizards, and they must make their decisions on the basis of the information offered. However, the results of the TAaAS project clearly indicate that in most cases users adopt trial-and-error driven solutions (TAEW, Trial-and-error wizard-based) (Csernoch & Biró, 2013a), mainly because they do not understand and are not interested in the messages provided by the software. Consequently, these methods of creating e-documents can be categorized as surface approach methods, and such as they are not sufficient for problem solving, and consequently, developing digital competency, digital literacy, and algorithmic skills.

Several other consequences of the teachers' choice of the TAEW-approach could be reported, but we must emphasize one which has far-reaching consequences: the method is not suitable for measuring the students' knowledge (Csernoch & Biró, 2013b). The tests of the TAaAS project have also revealed that the teachers hardly know any more than the students, and more disconcertingly, they are not able to judge the students' knowledge (Csernoch & Biró, 2013b).

Similar results were found by testing the students' and the teachers' programming skills (Biró & Csernoch 2013a). The results of the test clearly show that both the students, and more unfortunately, the teachers, have problems with tracking codes both in programming and spreadsheet. After seeing the teachers' results it is no wonder that in general they are not able to develop the algorithmic skills of the students. The other surprising and

disappointing result of the test is that those who passed the middle level graduation exam – where knowledge of office packages is tested on computer – have not been helped in developing their algorithmic skills. The results of the test are in accordance with our findings, considering the different metacognitive approaches in teaching computer applications and programming (Csernoch & Biró, 2013a, 2013b). Based on these tests, it is clear that in order to develop algorithm skills, digital thinking programming and applications should not be separated from each other, because computer related problems – the problems of the digital world – are based on algorithms.

2. Results

2.1. Deep-approach metacognitive processes

To solve the problems of the digital world algorithms have to be built and these algorithms have to be coded. The methods which support this approach are entitled Computer-algorithmic And Debugging-based methods (CAAD) (Csernoch & Biró, 2013a, 2013b), and are classified as deep-approach metacognitive processes (Case & Gunstone, 2002, 2003; Case, Gunstone & Lewis, 2001). The CAAD approach can be applied to any computer related problem. We have developed and tested CAAD-based methods for solving spreadsheet (Csernoch, 2012; Csernoch & Biró 2013a) and word processing problems (Csernoch, 2009). In both cases the primary aims of the methods are that in advance of the actualization of the problem the algorithms have to be created, and, based on the algorithm, the coding process follows (Biró & Csernoch, 2013a, 2013b, 2014). With spreadsheet, since it is classed as a functional language the coding process is more closely related to other programming languages, while in word processing the coding in the planned order is carried out by clicking on the command buttons, filling in the fields in GUI.

2.2. Theoretical background to the CAAD-based approach

The core of our CAAD-based method is based on the following theories:

- the Minimalist theory (Carroll, 1990; Nielsen, 1993; Warren, 2004),
- the Guided-instructions theory (Kirschner, Sweller & Clark, 2006),
- Phenomenography (Booth, 2001),
- Constructionism (Papert & Harel, 1991).

In the following chapters the implementation of these four theories are discussed in detail in the context of the spreadsheet environment. We argue that based on these theories efficient and effective methods for developing algorithmic skills would be created in the ICT environment.

1.1.4. Minimalist theory in the spreadsheet environment

From time to time, the most popular software publishers release new packages on the market boasting new features, which are of course more powerful than their predecessors. Spreadsheet programs, in addition to these new features, introduce new functions, specialized to solve specific problems. However, on one hand, these new functions make the program more difficult to use and understand for novices, while on the other hand, there is no way to prepare a general purpose spreadsheet program to solve all the world's problems, so there can never be enough upgrades. All these efforts are in vain.

According to the minimalist theory, we can teach spreadsheet to novices with as few and as simple functions as possible (Table 1, Group 1), and along with these functions we can teach how to build algorithms to solve more demanding problems, focusing on the problem, not on the software. Simple functions have two characteristics; (1) they are general purpose functions, and (2) they have only a few arguments. The other tool to solve problems using these simple general purpose functions is the creation of multilevel functions. Building multilevel functions can be well demonstrated by the popular Russian (matrjoska) dolls (Fig. 1). The encapsulated functions behave in exactly the same way as the encapsulated dolls do:

- we start building the multilevel function with the innermost function,
- outside it comes its hypernym, whose one argument – the input – is the output of the innermost function,
- this relationship between hyponyms and hypernyms continues until we reach the outermost function,
- the output of the outermost function is the output of the multilevel function.

Table 1. Three steps of introducing general purpose spreadsheet functions

Step 1	Step 2	Step 3
SUM()	MATCH()	SMALL()
AVERAGE()	INDEX()	LARGE()
MIN()	ISERROR()	AND()
MAX()		OR()
LEFT()		NOT()
RIGHT()		ROW()
LEN()		COLUMN()
SEARCH()		OFFSET()
IF()		TRANSPOSE()
		ROUND()

Fig. 1. Russian (matrojska) dolls simulating the encapsulation of functions in a multilevel function

Nielsen in 1991 and Warren in 2004, claimed that spreadsheet programs fulfill the expectations of the minimalist theory but neither of them provided further details or methods. As has been mentioned in section 1.1.3, the mere existence of a spreadsheet program does not in itself fulfill the requirements of the minimalist theory. However, a CAAD-based approach to spreadsheet, as outlined in this chapter, would be the right tool to do this.

2.2.1. Guided-instructions theory

Kirschner, Sweller & Clark's (2006) paper gives details regarding why it is that minimal guidance during instruction does not work. The reason is simple: very little is stored in long term memory. This finding is in accordance with our CAAD-based approach.

With spreadsheet the CAAD-based approach means that algorithms have to be built both at workbook and formula levels. For novices we focus on the formula level. At this level we teach a small number of general purpose functions (Table 1) and how to create multilevel functions to solve problems (Fig. 1). Again, with these two tools and heavily guided instructions at the beginning we can develop the basic algorithmic skills of the students, and later on, based on this knowledge the students will be able to free themselves from the teachers' strict guidelines and solve problems on their own. However, we have to emphasize here that creating spreadsheet formulas requires algorithms and the methods for creating formulas have to be learned. Clicking and wandering aimlessly in a huge program will result not in any knowledge entering long term memory but rather frustration, boredom, and error-filled documents. With guided instructions for novice spreadsheet end-user programmers, beyond the direct advantages of developing algorithmic and debugging skills, we can teach our students how to reduce the tendency to produce errors in spreadsheet documents.

The other importance of guided instructions at the beginning is that spreadsheet languages, since they belong to the group of functional languages, would serve as introductory programming languages which lead on to high level programming languages. This is one of the reasons that in teaching spreadsheet we have to use the same already proven methods of teaching programming languages.

2.2.2. Phenomenography

Marton and Booth (1998) proved that learning has two aspects that are inextricably intertwined: (1) the "what" aspect, which refers to the content of learning, and (2) the "how" aspect, which refers to the way in which learning takes place. It has been empirically established that whatever it is that learners learn (the "what" aspect), a qualitative variation is to be found in the outcome, and, correspondingly, whatever learning tasks are undertaken (the "how" aspect), there is a qualitative variation in the ways the learners approach them, or go about them. There is a degree of consistency in the nature of the variation in the "how" aspect in that there is an overall pattern of deep and surface approaches. A deep approach to a learning task is characterized by the learner's intention of finding meaning in the content through tackling the task, whereas in a surface approach focus is rather on meeting the demands of the task as such.

The learner's experience is always one of learning something, in some way, and in some context; by holding the learner's experience of learning as the focus of study throughout – and not studying the learning of the content and the acts and the context as separate and distinct focuses – the content, the act, and the context remain united as constituents of the learner's experience.

In 2001 Booth detailed and provided examples of their phenomenon. She claims that the first form of learning might be exemplified, in computer related activities, by acquiring further aspects of a partly-known programming language, or learning to write programs in yet another imperative programming language, or by learning to work in another new operating environment, or acquiring mastery of a particular set of techniques. The second form of learning, characterized rather by seeing things in a new way and bringing new perspectives to bear on things, is more like learning functional programming or object-oriented programming when hitherto only imperative programming has been encountered, and having to approach problems in quite a new way, having to consider new structures, and to develop new ways of understanding.

In a simplified way, we can say that Marton and Booth found that those methods which focus only on content and in which students cannot see the outcome of the learning experience are not effective enough. It is like not seeing the wood for the trees. Students are lost in the technical details and cannot grab the main idea. In Warren's wording (2004) this concept is expressed in the following form: "There is a need for an approach that allows the Computer Science issues to be in the foreground and the language issues in the background." In a human computer interaction (HCI) the problem, the programmer should be the focus of the activity, not the language, the software, the environment, or the hardware.

Marton and Booth's theory can easily be adapted to spreadsheet environment. However, using TAEW-based surface approach methods in spreadsheets the primary aim of the processes is to find a suitable function. But it should not work this way: first the algorithm has to be built and then we have to apply tools to code our algorithm. The tool is already provided in section 2.2.1. This method is in accordance with Booth's second form of learning: students gain experience in a functional language with a perspective towards other high level programming languages.

Booth's first form of learning is present in at least two different forms in spreadsheet. First of all, in the direct environment, where despite being familiar with only a few functions, demanding problems could be solved. Furthermore, the ability to solve problems in a spreadsheet environment might lead to similar problem-solving abilities in other programming languages.

In general, we state that the goal of learning spreadsheet is not to learn functions but to solve problems. The functions are only tools which are needed for coding; consequently, they should not be the primary target of the learning process. In most Informatics course books long lists of spreadsheet functions are presented, like in a reference book, suggesting that students have to learn these functions. The whole process should be the other way around: problems should be presented in these course books and the functions should only play a minor role.

If we teach only a few simple and general purpose functions along with the methods needed to build multilevel functions

- the functions can be easily remembered and
- both the functions and the building of multilevel functions can be adapted to other programming languages.

In this concept both spreadsheet course books and spreadsheet classes should use authentic tables instead of typing and creating artificial tables during the class. The hindrance caused by typing tables in a learning spreadsheet environment has far reaching consequences. Here, we find the connection to the next theory: artificial tables will not

provide data and consequently there will be no need to construct them. Again, interest is lost. In a spreadsheet environment we have to convince the students that they "desperately" need this piece of software; they have to see the power in this program from the very beginning.

2.2.3. Constructionism

Neither phenomenography nor constructionism support typing data sources in introductory application classes. The unexpected consequences of typing tables in introductory spreadsheet classes are listed below.

- Typing is boring. Students get bored at the very beginning.
- Typing is not Informatics. Consequently, typing steals time from real Informatics.
- Typed data is uncontrollable. Students type at different speeds and with different abilities. Consequently, the results are hard to use in a classroom environment.
- The amount of data is not sufficient. Consequently, students will find that creating spreadsheet for such a small amount of data is a waste of time.
- Manually typed tables are not challenging enough. No one is interested in information retrieval in a short and boring table.
- Students cannot see the power of a spreadsheet through typing. Consequently, they lose interest in it before they become familiar with it.

Learning and using applications should be a creative occupation. That is where the constructionism theory has its role. If we use authentic source documents for information retrieval and if we create documents which would communicate the desired content, these processes are nothing else but a construction of information or sources of information. Although these documents do not necessarily reach the wide public, the students in their micro community – even in a class – do construct something which has a value, the value of information.

2.2.4. The actualization of CAAD in spreadsheet

Based on the theories outlined in the previous chapters we have developed a deep-approach CAAD-based metacognitive method on the formula-level of spreadsheet programs. The method is applied to a rarely used but powerful feature of spreadsheet, namely the Conditional Single Result Array Formula (CSRAF). This feature of spreadsheet programs requires

- knowledge of a few simple and general purpose functions – minimalism,
- knowledge of how to build multilevel formulas – constructionism,
- complete guidelines from teachers at the beginning – guided instructions,
- meaningful, usually authentic tables with a huge amount of data – phenomenography,
- the teachers' choice of demanding problems at the students' level – constructionism, phenomenography,
- students' awareness of problems – constructionism, phenomenography.

Through the course of the application of CSRAFs, spreadsheet programs

- serve as a tool for building algorithmic skills – phenomenography,
- serve as an introductory language to high level programming languages – phenomenography,
- serve as an environment for solving demanding data retrieval problems – constructionism, phenomenography,
- serve as an environment for debugging formulas – guided instructions, constructionism, phenomenography,
- serve as a tool for lowering the number of error-filled documents – guided instructions, constructionism, phenomenography,
- increase students' confidence – constructionism, phenomenography,
- increase students' skills in handling problems – constructionism, phenomenography.

It has been proved with the tests conducted on the TAaAS project that one of the main reasons for error prone documents is the almost exclusive usage of TAEW-based methods, especially in the learning stage. Based on these theories we switched to a CAAD-based method whose focus is the building of CSRAFs. The method was first applied to gifted high schools students participating in application competitions. The students' results in these competitions improved a great deal. Encouraged by the results of the competitions we applied the method to a couple of normal Informatics classes, and it also worked.

2.2.5. Students' development in a CAAD-based environment

The first tests of the method were carried out with university students in the academic year 2011/2012 and were repeated in the following two years. The students were tested three times.

- The first stage of the test was carried out at the beginning of the students' tertiary studies, with students who had previously studied spreadsheet with TAEW-based methods in elementary and high schools. Consequently, in terms of the methods with which the students were familiar, at this stage there was no difference between Group1 and Group2.
- The second stage of the test took place immediately after students had covered spreadsheet with a CAAD-based method (Group1).
- The third test was applied one year later, testing students who had covered spreadsheets with CAAD-based (Group1) or TAEW-based methods (Group2).

In all three tests we were focusing on the methods the students used to create spreadsheet formulas. In the first test the students' results were extremely low, below 5% on average (Fig. 2.). In the second test the results of those students who covered spreadsheet with the CAAD-based method increased to above 60% (Fig. 3, left). In the third test the question was what knowledge is stored in long term memory. It was found that those students who learned with the CAAD-based method had results around 40%, while those who used the TAEW-based method were still unable to solve these problems (Fig. 3, right).

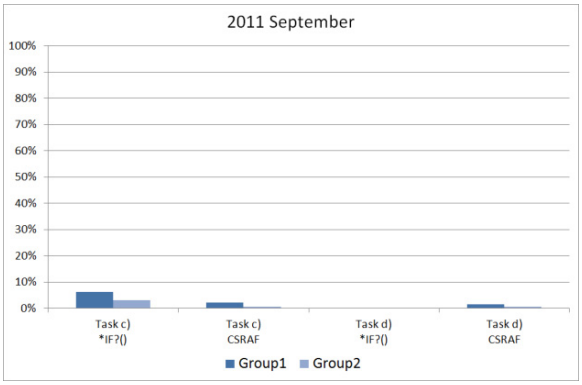

Fig. 2. The students' results in the first TAaAS test in September, after covering spreadsheet with TAEW-based methods in elementary and high schools

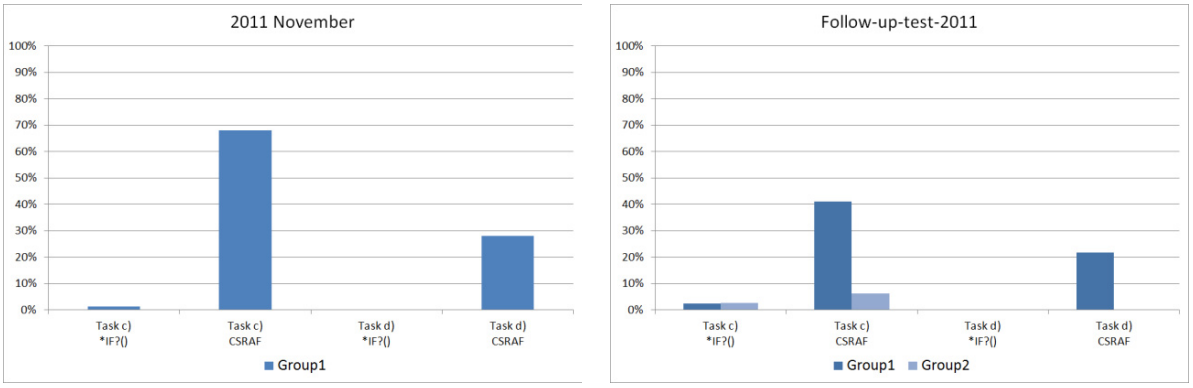

Fig. 3. The students' results in the second test, after covering spreadsheet with the CAAD-based method (left), and in the third test, in next November, one year after covering spreadsheet with CAAD- or TAEW-based methods (right)

The other goal of the test was to detect whether the CAAD-based method made the students switch to the multilevel formulas instead of using the error prone built-in functions or, based on their previous 6–8 years of studies, they still stuck to the built-in functions.

It was found that the students using the CAAD-based method almost exclusively switched to the multilevel CSRAFs, while in the other group some students preferred the CSRAFs formulas. Their solutions, due to the lack of guided instructions, were not correct, but the traces of multilevel formulas were recognizable.

As was mentioned in section 2.2.5, the method was first introduced to high school students, and their ability to solve spreadsheet problems improved significantly; however, their improvement was not tested directly, but observed through their results in competitions.

Similar improvements were registered by testing 8th grade students studying spreadsheet with a CAAD-based approach, in comparison with 8th grade students studying with the popular TAEW-based method (Majláth, 2013). All these findings mean that the method can be applied to the different levels of the education system.

3. Summary

It has been proved in the last couple of years that more than 90% of spreadsheet documents carry mistakes, and uneducated computer users cause serious financial losses by providing unreliable data and by taking much more time than problems require. It has also been proved that surface approach metacognitive methods are not effective and efficient enough to develop the skills required to solve spreadsheet problems. Even though the minimalist theory claimed in 1993 that spreadsheet languages would be perfect introductory languages, methods have not been developed. The reason was that the minimalist theory on its own was not enough to develop a method which would be used in education; three additional theories – guided instructions, phenomenography, and constructionism – had to emerge.

Based on this theoretical background we have developed a deep approach metacognitive method – the computer-algorithmic and debugging (CAAD) based method – for solving computer related problems, especially those within computer applications. In spreadsheet programs the tool which fulfills all the requirements of the CAAD-based method is the conditional single result array formula (CSRAF). The core of this method is that we use as few and as simple general purpose functions as possible, and develop the technique necessary to create multilevel functions.

By testing the method it was found that it is much more effective and efficient than the widely used and commercialized TAEW-based methods. The students using the CAAD-based method prove to be more confident and safer in creating spreadsheet formulas. Beyond this primary result, our tests also proved that with the CAAD-based method the students' algorithmic skills have developed, and that knowledge is stored in long term memory. Nielsen's prediction of the power hidden in spreadsheet and his suggestion to use spreadsheet as an introductory programming language has been proved with our CAAD-based method.

4. Acknowledgement

The publication was supported by the TÁMOP-4.2.2.C-11/1/KONV-2012-0001 project. The project has been supported by the European Union, co-financed by the European Social Fund and partly by OTKA (K-105262).

References

Angeli, C. (2013). Teaching spreadsheets: A TPCK perspective. In Kadijevich, Dj. M., Angeli, C., & Schulte, C. (Eds.). (2013). Improving Computer Science Education. New York and London: Routledge; p. 132–145.

Biró, P. & Csernoch, M. (2013a). Deep and surface structural metacognitive abilities of the first year students of Informatics. 4th IEEE International Conference on Cognitive Infocommunications, Proceedings, Budapest, 521–526.

Biró, P. & Csernoch, M. (2013b). Programming skills of the first year students of Informatics. XXIII. International Conference on Computer Science 2013, EMT, in Hungarian, pp. 154–159.

Biró, P. & Csernoch, M. (2014). Algorithmic approach to spreadsheets. Proceedings of VIII. Kiss Árpád Conference, in Hungarian. Accepted.

Booth, S. (2001). Learning Computer Science and Engineering in Context. Computer Science Education. 2001, Vol. 11, No. 3, pp. 169–188.

Carroll, J. M. (1990), The Nurnberg funnel: designing minimalist instruction for practical computer skill, M.I.T. Press, Cambridge, Mass.

Case, J. & Gunstone, R. (2002). Metacognitive development as a shift in approach to learning: an in-depth study. Studies in Higher Education, 27(4), 459–470.

Case, J. & Gunstone, R. (2003). Approaches to learning in a second year chemical engineering course. International Journal of Science Education, 25(7), 801–819.

Case, J., Gunstone, R. & Lewis, A. (2001). Students' metacognitive development in an innovative second year chemical engineering course, Research in Science Education, 31(3), pp. 331–355.

Csernoch, M. & Biró, P. (2013a). The investigation of the effectiveness of bottom-up techniques in the spreadsheet education of students of Informatics. (Eds): Kozma T. and Perjés I., New Research in Education Studies. ELTE Eötvös Publisher, Budapest. pp. 369–392, in Hungarian.

Csernoch, M & Biró, P. (2013b). Teachers' Assessment and Students' Self-Assessment on the Students' Spreadsheet Knowledge. EDULEARN13 Proceedings July 1st-3rd, 2013 – Barcelona, Spain. Publisher: IATED. ISBN: 978-84-616-3822-2. pp. 949–956.

Csernoch, M. & Bujdosó, G. (2009). The mistakes occurring in exams and competition tasks and their consequences, in Hungarian. New Pedagogical Review. 2009/1.

Csernoch, M. (2009). Teaching word processing – the theory behind. Teaching Mathematics and Computer Science. 2009/1.

Csernoch, M. (2010). Teaching word processing – the practice. Teaching Mathematics and Computer Science. 8/2 (2010). pp. 247–262.

Csernoch, M. (2012). Introducing Conditional Array Formulas in Spreadsheet Classes. EDULEARN12 Proceedings. Barcelona, Spain. 2-4 July, 2012. Publisher: IATED, 7270–7279.

Dani, E. (2013). Bit-Generations and the Digital Environment. in Current Issues In Some Disciplines, Karlovitz János Tibor (ed). ISBN 978-80-89691-01-2.

Gove, M. (2012). Michael Gove speech at the BETT Show 2012. Retrieved from: https://www.gov.uk/government/speeches/michael-gove-speech-at-the-bett-show-2012, 10. 02. 2014.

Jukes, I., McCain T. & Crockett, L. (2010). Understanding the Digital Generation: Teaching and Learning in the New Digital Landscape. 21st Century Fluency Project Inc. ISBN-13: 978-1412938440.

Kirschner, A. P., Sweller, J. & Clark, E. R. (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), pp. 75–86.

Majláth, A. (2013). The application of conditional single result array formulas in primary school education, in Hungarian. Retrieved from: http://ganymedes.lib.unideb.hu:8080/dea/bitstream/2437/169848/4/szakdolgozat.pdf. 15. 07. 2013.

Marton, F. & Booth, S. (1998). The Learner's Experience of Learning. in The Handbook of Education and Human Development. Edited by: David R. Olson and Nancy Torrance.

Nielsen, J. (1993). Usability Engineering, Academic Press, Boston, MA.

OECD (2011). PISA 2009 Results: Students on Line: Digital Technologies and Performance (Volume VI). Retrieved from: http://browse.oecdbookshop.org/oecd/pdfs/free/9811031e.pdf. http://dx.doi.org/10.1787/9789264112995-en, 02.02.2014.

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

Papert, S. & Harel, I. (1991). Situating Constructionism. Constructionism, Ablex Publishing Corporation: 193-206. Retrieved from http://www.papert.org/articles/SituatingConstructionism.html, 02.02.2014.

Tort, F. (2010). Teaching Spreadsheets: Curriculum Design Principles. In S. Thorne (Ed.), Proceedings of the EuSpRIG 2010 conference: Practical steps to protect organisations from out-of-control spreadsheets, p 99–110.

Tort, F., Blondel, F. & Bruillard, É. (2008). Spreadsheet Knowledge and Skills of French Secondary School Students. R.T. Mittermeir and M.M. Sysło (Eds.): ISSEP 2008, LNCS 5090, 305–316, 2008. Springer-Verlag Berlin Heidelberg.

Van Deursen A. & Van Dijk J. (2012). CTRL ALT DELETE. Lost productivity due to IT problems and inadequate computer skills in the workplace. Enschede: Universiteit Twente. Retrieved from: http://www.ecdl.org/media/ControlAltDelete_LostProductivityLackofICTSkills_UniverstiyofTwente1.pdf. 02. 01. 2014.

Warren, P. (2004). Learning to program: spreadsheets, scripting and HCI, in Proceedings of the Sixth Australasian Conference on Computing Education – vol. 30, Darlinghurst, Australia, pp. 327–333.