Informatics in Education, 2015, Vol. 14, No. 2, 175–197

DOI: 10.15388/infedu.2015.11

Testing Algorithmic Skills in Traditional and Non-Traditional Programming Environments

Mária CSERNOCH[1], Piroska BIRÓ[1],
János MÁTH[2], Kálmán ABARI[2]

[1] *University of Debrecen, Faculty of Informatics, Hungary*
[2] *University of Debrecen, Faculty of Arts and Humanities, Hungary*
e-mail: {csernoch.maria, biro.piroska}@inf.unideb.hu,
{math.janos, abari.kalman}@arts.unideb.hu

Received: February 2015

Abstract. The Testing Algorithmic and Application Skills (TAaAS) project was launched in the 2011/2012 academic year to test first year students of Informatics, focusing on their algorithmic skills in traditional and non-traditional programming environments, and on the transference of their knowledge of Informatics from secondary to tertiary education. The results of the tests clearly show that students start their studies in Informatics with underdeveloped algorithmic skills, only a very few of them reaching the level of extended abstract. To find reasons for these figures we have analyzed the students' problem solving approaches. It was found that the students, almost exclusively, only consider traditional programming environments appropriate for developing computational thinking, algorithmic skills. Furthermore, they do not apply concept and algorithmic based methods in non-traditional computer related activities, and as such, mainly carry out ineffective surface approach methods, as practiced in primary and secondary education. This would explain the gap between the expectations of tertiary education, the students' results in the school leaving exams, and their overestimation of their knowledge, all of which lead to the extremely high attrition rates in Informatics.

Keywords: algorithmic skills, spreadsheet, deep and surface metacognitive approaches, self-assessment, school leaving exams.

1. Introduction

That the computer has become ubiquitous is not in question. The question is how effectively we can use it. This simple question, however, starts an avalanche of other questions. Wing stated that "Computational thinking is a fundamental skill for everyone, not just for computer scientists. To reading, writing, and arithmetic, we should add computational thinking to every child's analytical ability." (Wing, 2006). Most school curricula have been changed in the last two decades to support this approach by emphasizing the

importance of the development of digital literacy and competency. In most countries these competences have been integrated into traditional school subjects and/or a new subject was introduced. This latter alternative is effectively the equivalent of formal education in CSI. However, formal education requires teachers, and teachers require teacher education. At this point the loop is closed, and we are faced with the chicken and the egg problem: who teaches the teachers if there are no teachers? In 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 in the science itself and in the developing commercial world as it has interacted with the science, affect teachers, teacher education and consequently the development of digital competency and literacy. What should we teach and how should we teach CSI in and outside of formal education?

The very first question is: "*Should we teach students to program?*" In 1993 Soloway claimed that "…those students who wish to major in computer science should learn to program in school, but for them, learning to program is simply vocational training." (Soloway, 1993). However, he did not make it clear which school(s) should be responsible for teaching programming. He and his colleagues went further, claiming that everyone should develop algorithmic skills, but non-traditional programming environments would fit non-professionals better. Twenty years later, however, we are faced with the problem that non-professional end-users do mainly unplanned, aimless clicking in the GUI and are satisfied with unchecked, non-bugged results, (Ben-Ari, 1999; Csernoch and Biró, 2014a, 2014b; EuSprig, 2015; OECD, 2011; Panko and Aurigemma, 2010; Powell *et al.*, 2008; Tort *et al.*, 2008; Van Deursen and Van Dijk, 2012). This behavior leads to the extremely high number of documents and programs carrying mistakes and errors, causing serious financial losses (Panko and Aurigemma, 2010). The other source of losses is the human factor; the time and the number of participants needed to produce this questionable result (Van Deursen and Van Dijk, 2012). However, what do we know about the professionals? About those who – according to Soloway (1993) – are supposed to learn programming as their vocational training. Is their level of digital competency higher than the level of the non-professionals? Are they prepared for high level programming when they start their tertiary education in Computer Sciences? Do they know what Computer Sciences are when they enter universities and colleges? We doubt it. We have come to realize that the terminology usage and the algorithmic skills of students arriving at the Faculty of Informatics – especially since the number of students in Informatics has increased – have not developed to match the requirements of courses in higher education.

2. Sample

2.1. *The Tests of the TAaAS Project*

However, neither the realization of the students' underdeveloped algorithmic skills, nor the high percentage of dropout CSI students, nor the high number of semesters the students spend in their CSI studies (Csernoch and Biró, 2013b; Tan and Venables, 2010) can

indicate exactly what the students do or do not know when they start their tertiary education. To see clearly what knowledge the students have brought with them, we launched the TAaAS project (Testing Algorithmic and Application Skills) in the 2011/2012 academic year at the Faculty of Informatics of the University of Debrecen, Hungary (Biró *et al.*, 2014, 2015, Biró and Csernoch, 2013a, 2013b, 2013c, 2014; Csernoch *et al.*, 2014; Csernoch and Biró, 2013a, 2013b, 2013c, 2014a, 2014b, 2014c).

The TAaAS project is a paper based testing process, which focuses on the level of the participants' algorithmic skills, their usage of terminology, and their problem-solving abilities in different software environments. It runs on different levels. Each level has its own purposes, but they meet at the end.

- Introductory test: Testing BSc, BA, MSc, MS students and student teachers of Informatics on the first week of their first semester.
- CAAD test: Testing students' spreadsheet knowledge after covering spreadsheet with a deep approach metacognitive method, entitled the Computer Algorithmic and Debugging based approach (CAAD) (Csernoch and Biró, 2015b).

The primary aims of the testing are to reveal the students' level of understanding, the connection between the similar tasks presented in the different software environments, the students' level of terminology usage, and their level of computational thinking.

2.1.1. *Introductory Test*

The date of the introductory test is strictly set to the first week of the students' tertiary course in order to test the knowledge they have gained in their previous studies, in such a way as not to be influenced by the curriculum of the new institute. The introductory test includes two questionnaires:

- General information, attitude and self-assessment questionnaire. This takes about 10–15 minutes, and then the papers are collected.
- Informatics questionnaire, which is the primary test. This takes about 45 minutes, with eleven task of traditional programming, spreadsheet programming, word processing, handling files, and calculation in different numeral systems.

The general paper includes questions relating to the students' computer usage habits, their results in the school leaving exams in Mathematics and Informatics (SLE, 2014), in competitions testing Informatics, and ECDL (2014), if they have taken part in any. The other sets of questions focus on the students' self-assessment: how they evaluate their knowledge in the different subfields of Informatics, and what further studies they think they need. With the third group of questions we are testing the students' approach towards spreadsheet programming, which is closely related to our CAAD test.

2.1.2. *CAAD Test*

It was realized that the metacognitive problem solving approaches involved in computer related activities have never been mapped. We have found the problem solving approaches used in other sciences (Case and Gunstone, 2002) and in programming (Booth, 1992), but not a complete typology to cover all computer related activities. To fill this gap, we have merged the previously published problem solving typologies and made

the necessary amendments (Csernoch and Biró, 2015c). The result is a typology which consists of two deep approach methods – concept based (Polya, 1954; Booth, 1992; Case and Gunstone, 2002) and Computer Algorithmic and Debugging (CAAD) based (Booth, 1992, Csernoch and Biró, 2015b) – and three surface approach methods – algorithmic based (Booth, 1992; Case and Gunstone, 2002), information based (Booth, 1992; Case and Gunstone, 2002), and Trial-And-Error Wizard (TAEW) based (Csernoch and Biró, 2015b). The latest covers all the unplanned sequences of activities which are carried out in the GUI (Graphical User Interface), without the users necessarily knowing whether the output is the result of the original problem or not.

While building her typology for programming approaches, Booth proved as early as the early 90's that functional languages are effective as introductory languages (Booth, 1992). The simplicity of these languages allows us to focus on the problem, instead of the coding details. However, these findings have never reached the wider public. In the meantime, based on similar theoretical backgrounds, a novel tool, spreadsheet programs appeared on the market. However, the programmability of spreadsheets has never reached the wider public, either. These two ends – programming in functional languages and a user-friendly tool for coding – have met in Sprego (Csernoch, 2012, 2014; Csernoch and Biró; 2015a, 2015c; Csernoch and Balogh, 2011).

We have introduced Sprego, which is a deep approach metacognitive method to teach building algorithms in spreadsheet environment. The core of the method is that spreadsheet is taught and handled as a programming language (Sestoft, 2010), which would serve either as an introductory language for professionals or as the ultimate language for end-user programmers (Biró and Csernoch, 2013a; Csernoch, 2012, Csernoch and Balogh, 2011; Warren, 2004). The method focuses on the development of algorithmic skills. As such it breaks with the traditional, but ineffective, TAEW-based (Trial-And-Error Wizard-based) methods in spreadsheets (Csernoch and Biró, 2014b). Its main idea is that for novices we introduce as few and as simple functions as possible, and teach how to create multilevel functions using these simple functions to solve problems. With the development of the students' skills the number of functions can be increased, but the focus is still on simple, not software specified, functions.

With the CAAD-test our goal was to reveal how the students' deep approach problem solving abilities are developed in non-traditional programming environments.

2.1.3. *Data Sources of the TAaAS Project*

In the evaluation phase of the TAaAS project we focus on comparing the results of the tasks presented in different programming environments, and on the results of the different sources and testing methods. The statistical analyses of the data deriving from the different sources might provide explanations for the unsatisfactory level of our students, and guidelines for improving the effectiveness and efficiency of our primary and secondary CSI education.

Our further aim is to widen the project, and to test students from different institutes and countries to see the similarities and differences, to reveal which education systems best support the development of algorithmic skills and computational thinking and which approaches should be adapted to make the other systems work effectively.

We do not share those extreme opinions which blame birotical (computer-related office tools) software (Gove, 2012, 2014) for any distracting effect, and consequently, for making CSI education ineffective. We argue that these programs are harmless; the commercialized world developed on the basis of these programs and teachers' unconditional acceptance of the TAEW-based approaches has led us to our current fiasco. We further argue that since programs are algorithm driven, if we therefore teach any software from an algorithmic point of view, we would be much more successful, and students' computational thinking would develop, which is the ultimate goal.

2.2. *Participating Students*

The TAaAS project was launched with three major programming BSc courses at the Faculty of Informatics of the University of Debrecen (DE): Software Engineering (SOE), System Engineering (SYE), and Business Information Management (BIM), and was repeated in the following years under similar conditions, with altogether 950 DE students taking part (Table 1). The project was extended in the 2013/2014 academic year, when three more Hungarian institutes joined: Eötvös Loránd University (ELTE, Budapest), Eszterházy Károly College (EKF, Eger), and the College of Nyíregyháza (NYF, Nyíregyháza) (Biró *et al.*, 2014; Csernoch *et al.*, 2014; Csernoch and Biró, 2013b).

The school leaving exams are held at intermediate and advanced levels, and to enter tertiary CSI education students can choose which level they take (SLE, n.d.). Mathematics is compulsory, while Informatics is not, which explains the lower number of exams in Informatics. Due to some students from foreign countries and some uncompleted questionnaires, the number of students taking the Mathematics exams is lower than the expected 950 (Table 2).

We have to note here that in Hungary there has been formal CSI education both in primary and secondary education since the 1995 National Curricula was launched (NAT, 1995, 2003, 2007, 2012; European Schoolnet, n.d.). According to these documents, CSI studies should start as early as the 1st grade and continue until the 12th grade. However, the realization of the National Curricula, in the form of the frame curricula (Kerettanterv 2000, 2009, 2013), shows different results, and these CSI classes do not provide enough space and time for the development of algorithmic skills in traditional programming environments.

Table 1

The number of students at the Faculty of Informatics of the University of Debrecen participating in the TAaAS project in the three testing years

	SOE	SYE	BIM	Sum
2011/2012	115	86	109	310
2012/2013	108	111	101	320
2013/2014	115	115	90	320
Sum	338	312	300	950

Table 2
The students' results in the school leaving exams in Informatics and in Mathematics

	SOE	SYE	BIM	Average
	School leaving exam (SLE) – *intermediate level*			
Informatics	84.1 (N=175)	82.2 (N=237)	80.8 (N=193)	82.3 (N=605)
Mathematics	74.1 (N=277)	71.4 (N=265)	74.4 (N=252)	73.3 (N=794)
	School leaving exam (SLE) – *advanced level*			
Informatics	72.5 (N=127)	66.4 (N=37)	55.7 (N=16)	69.7 (N=180)
Mathematics	68.1 (N=22)	70.3 (N=17)	68.9 (N=24)	69.0 (N=63)

The results of the school leaving exams in Informatics were found significantly higher than in Mathematics in all the three majors (Wilcoxon signed rank tests: $p < 0.001$), both at intermediate and advanced levels. However, the results of the school leaving exams do not show significant differences between the three majors in Mathematics (Kruskal-Wallis probe: $\chi^2(2) = 5.802$, $p = 0.055$). In Informatics significant differences were found between the three majors (intermediate: $\chi^2(2) = 8.451$, $p = 0.015$, advanced: $\chi^2(2) = 14.41$, $p < 0.001$), while the comparison of the pairs found only the SOE students' results differed from those of the BIM students. Given the low number of students taking the advanced level Mathematics exam, we cannot say anything about the differences, but their results seem very similar (Table 2).

We can conclude that according to the results in the school leaving exams, the students start their tertiary education in CSI with similar knowledge, both in Informatics and in Mathematics, while their problem solving skills are on a higher level in Informatics than in Mathematics. Based on these data, we can assume that students would produce similar good results in the test.

2.3. *The Tasks of the TAaAS Project*

In the present article we compare the results of three of the algorithmic tasks of the TAaAS tests. The tasks are different in nature.

- **Task 1:** An algorithm with a multilevel IF structure, which has an X and Y pair for input, with three possible input values – A, B and 0, and with only four possible output values – 3, 2, 1 and 0. A table is presented with 9 pairs of inputs, and the output cells must be filled in. Previous analyses proved that this is the easiest task among the three (Fig. 1) (Biró *et al.*, 2014, 2015).
- **Task 2:** Three pseudo codes and an accompanying picture with the input values presented. The task is to tell what the codes do (Fig. 2).

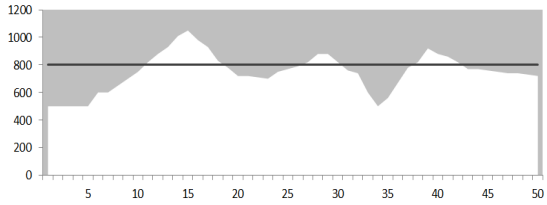
You draw two cards (X, Y) from two packs. In both packs you can find cards with the letter A, cards with the letter B and cards with zero. Give the points in the last column of the table according to the algorithm given below!

```
V:=X="A" or Y="A"
W:=X="B" or Y="B"
If V and W then Point:=0
else If V then Point:=1
else If W then Point:=2
else Point:=3
```

X	Y	Point
A	A	
A	B	
A	0	
B	A	
B	B	
B	0	
0	A	
0	B	
0	0	

Fig. 1. The code, the input values and the empty cells for the outputs of Task 1.

What do the following programs do? What is stored in variable DB and M?



N=50, measured values:
500,500,500,500,500,600,
600,650,700,750,820,880,
930,1010,1050,980,930,830,
780,720,720,710,700,750,
770,790,820,880,880,820,
760,740,600,500,560,670,
780,820,920,880,860,820,
770,770,760,750,740,740,
730,720.

Task 2.1

```
DB:=0
Loop from i=1 to N
  If X(i)>800 then DB:=DB+1
End loop
```

Task 2.2

```
DB:=0
Loop from i=2 to N−1
  If X(i)<X(i−1) and X(i)<X(i+1)
     then DB:=DB+1
End loop
```

Task 2.3

```
M:=0
Loop from i=2 to N
  If X(i)−X(i−1)>M then M:=X(i)−X(i−1)
End loop
```

Fig. 2. The accompanying picture, the input data and the pseudo codes of Task 2.

- **Task 3:** Programmable spreadsheet problems accompanied with a sample table (Fig. 3). The three tasks to solve with formulas – four in 2013/2014 –, and a multilevel spreadsheet formula to decode, are similar to Task 2.

Both Tasks 1 and 2 are borrowed from a Hungarian programming competition for students of 5–8th grades (NT, 2009).

To evaluate Task 2 the different categories of understanding of the SOLO taxonomy were adapted to the special circumstances of the test:

- Ignored (1).
- Prestructural (2).
- Unistructural (3).
- Multistructural (4).
- Relational (5).

The Extended abstract category of the original classification is left out, since it has no relevance in the test (Biggs and Collis, 1982; Biró and Csernoch, 2014; Clear *et al.*, 2008; Lister *et al.*, 2006; Sheard *et al.,* 2008).

When handling spreadsheet as an environment for programming in functional languages, we were able to adapt the SOLO categories to the spreadsheet solutions (Biró and Csernoch, 2014). Using this method we can compare traditional and non-traditional programming solutions, the students' level of understanding, and their approaches to problem-solving in different environments.

	A	B	C	D	E
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

Task 3.1 How many African countries are in the table?

Task 3.2 What is the average population of those countries whose surface area is smaller than G1?

Task 3.3 How many countries have a surface area greater than G1?

Task 3.4 {=SUM(IF(B2:B236="Europe",IF(LEFT(A2:A236)="A",1)))}

Fig. 3. The sample table and problems of Task 3.

3. Hypotheses

H1: *Students of Informatics start their tertiary education with well-developed algorithmic skills and computational thinking, with a firm concept of computers, and with correct terminology usage.*

H2: *The results of the school leaving exams are able to tell apart the different levels of knowledge, and serve as a correct measure of algorithmic skills.*

H3: *The students of Informatics have the knowledge to provide correct self-assessment values, indicating their knowledge in the different fields of Informatics.*

H4: *The students' algorithmic skills are software independent.*

4. Results

4.1. *Classification of Students*

4.1.1. *Preliminary Groups*

The test results and the self-assessment values of the three majors at the University of Debrecen (SOE, SYE, BIM – preliminary groups) are presented in Fig. 4–Fig. 6.

The students' self-assessment values are higher in spreadsheet than in programming in all the three preliminary groups; however, their results in the test proved exactly the

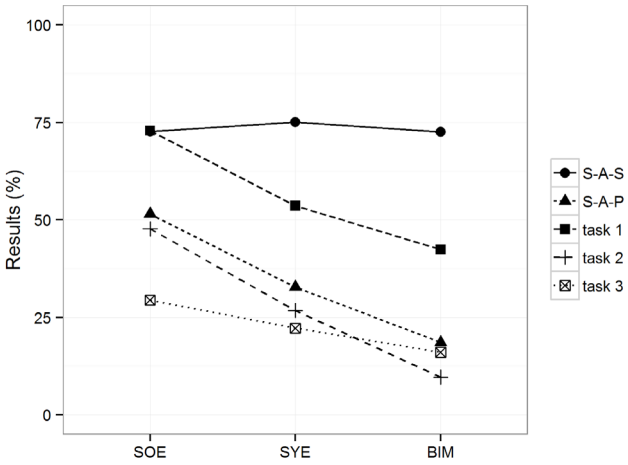


Fig. 4. The students' results in the three tasks compared to their self-assessment values in programming (S-A-P) and spreadsheet (S-A-S).

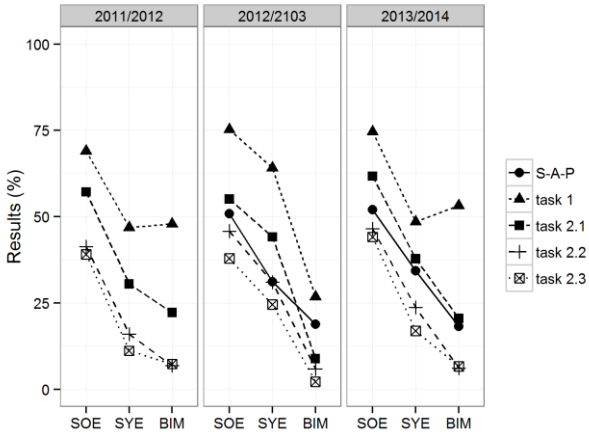


Fig. 5. The students' results in Tasks 1 and 2 compared to their programming self-assessment values (S-A-P).

opposite (Fig. 4). Task 1 is found to be the easiest among the three different kinds of tasks, and Task 3 the most difficult. BIM is the only group whose spreadsheet result is slightly above the decoding tasks; in the other groups the results of both Tasks 1 and 2 are better than Task 3.

It is clear from figures Fig. 4 and Fig. 5 that the students' results in Task 1 are above both their programming self-assessment values and the results of the other two tasks. The dominance of Task 1 over Task 2 is the straightforward consequence of the requirements of the tasks. In Task 1 the students had to do nothing other than pick one out of the four numbers, while in Task 2 semantically correct natural language sentences had to be written after decoding the pseudo codes. The spreadsheet problems were found more difficult than the students predicted (Fig. 4). However, if we study the most recently published

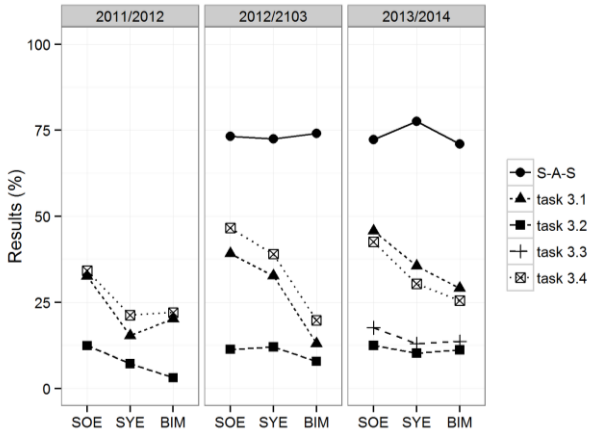


Fig. 6. The students' results in the spreadsheet task compared to their spreadsheet self-assessment values (S-A-S).

papers dealing with the high number of spreadsheet documents with errors (EuSprig, 205; Panko and Aurigemma, 2010; Powell *et al.*, 2008; Tort *et al.*, 2008), and the undue length of time spent creating birotical documents (Van Deursen and Van Dijk, 2012), we can conclude that the results of the test are not surprising at all. Beyond finding evidence in these papers for the incorrectness of the spreadsheet documents, it is also covertly evident that end-users either felt so comfortable about their knowledge or are so ignorant of their non-existing knowledge that it does not occur to them to check their documents, which is the well-known Dunning-Kruger effect (Kruger and Dunning, 1999).

Fig. 6 clearly shows that the students' spreadsheet self-assessment values (S-A-S) greatly exceed their results in the test. Beyond this fact, Task 3.1 shows similar results to 3.4, while Tasks 3.2 is similar to 3.3 in the three majors (Wilcoxon signed rank tests: Tasks 3.1 and 3.4, SOE, SYE, BIM: $p = 0.3168$, $p = 0.4699$, $p = 0.4635$; Tasks 3.2 and 3.3: $p = 0.02531$, $p = 0.2642$, $p = 0.3363$, respectively). The only significant difference was found in the SOE group between Tasks 3.2 and 3.3. The differences between the pairs are significant (Wilcoxon signed rank tests: pairs Tasks 3.1–3.4 and Tasks 3.2–3.3: SOE, SYE, BIM: $p < 0.001$). The differences between these two pairs can be explained by the nature of the tasks. Task 3.1 is a simplified task using constant and checking equality in the condition. Task 3.4 is a complete array formula for decoding, which is easier than creating formulas. On the other hand, both Tasks 3.2 and 3.3 are generalizations of Task 3.1.

The other characteristic of Task 3.4 is that it is included with the purpose that it would serve as guideline for solving Tasks 3.1–3.3. However, the analyses of the solutions proved that the students did not realize the connection between these four tasks (Csernoch and Biró, 2014b). Students learn rules which work only for special problems, and this is, as such, a surface approach method, referred to as the algorithmic-based problem solving method in the Gase & Gunstone system (Case and Gunstone, 2002) and as expedient in Booth system (Booth, 1992). We must note here that the surface approach methods referred to as algorithmic-based and expedient – in Case & Gunstone's

and Booth's typologies, respectively – are different from our computer-algorithmic and debugging (CAAD) deep approach methods (Csernoch and Biró, 2014b; Csernoch and Biró, 2015c). Consequently, in this analysis we found that students are not able to do abstraction – their level of understanding is extremely low (for details see section 4.2.2).

4.1.2. *Knowledge-Based Clusters in Task 1*

Since Task 1 was found to be the most successful among the three tasks, it was used to define knowledge-based clusters. Four clusters were easily distinguishable; C1L, C2L, C3L and C4L, ranging from the best results to the worst, respectively. Those students who did not try or complete Task 1 are placed in C4L.

The result of C2L is closer to C1L, while C3L is closer to C4L. However, the comparison between the results from C2L and C3L revealed a remarkable difference between the two middle level clusters. Their knowledge is different in nature. As we show in Fig. 7, C2L indicates an average, uncertain knowledge, without any definable pattern. Their knowledge is quite arbitrary. However, C3L is different in nature. In certain cases they do almost as well as C1L, while in other cases their results are disastrous. This behavior of C3L led us to the conclusion that these students have limited knowledge. Until they reach their limit they are able to solve the problems almost perfectly; however beyond their limit they try to find escape routes, and this strategy leads them in completely false directions. In the case of Task 1 C3L did well when the input values of the X, Y pairs were A and B (Fig. 7, #1, 2, 4, 5), however, when one of the input values was 0 they concluded that the output should also be 0 (Fig. 7, #3, 6, 7, 8, 9).

Similar results were found when comparing the different Hungarian institutes in the 2013/2014 academic year (for details see Biró *et al.*, 2014). The same four clusters with the very same characteristics were found in the other institues, which clearly show that

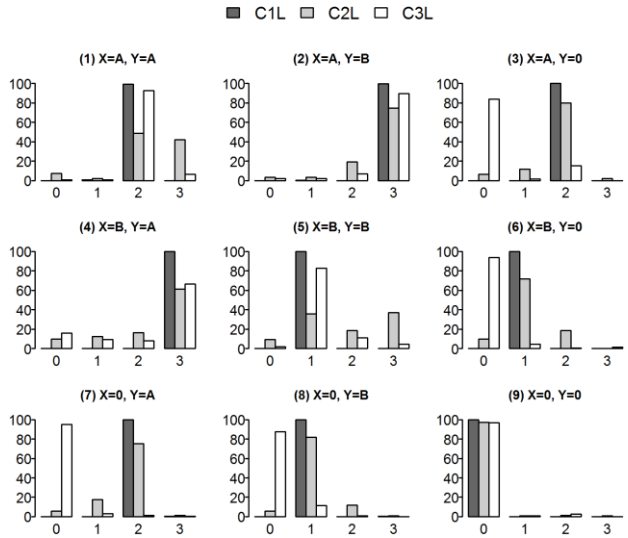


Fig. 7. The results of the three clusters in Task 1.

the results are not specific to the University of Debrecen, but rather a general tendency. To obtain further data we are planning to test other Hungarian and foreign institutes in the following years.

4.2. *Comparison of the Students' Results*

4.2.1. *Results of Task 2*

The results of the four knowledge based clusters in Task 2 are presented in Table 3, and the behavior of the different clusters can be tracked in Fig. 8. The results for the clusters, similar to the results in Task 1, follow a decreasing order in C1L, C2L, C3L and C4L, respectively. The behavior of C3L, which can be recognized in Task 1, is also detectable in Task 2; while C2L reaches Level 3 in all the three tasks with the highest percentage, until C3L stops at Level 1. This is the only level where C3L is above C2L. The knowledge of C3L in the decoding tasks of advanced primary school students is extremely limited.

4.2.2. *Results of Task 3*

It is clearly evident, both for the preliminary groups and for the C1L–C4L knowledge based clusters, that the students' results in spreadsheet are significantly lower than their self-evaluation values (Wilcoxon signed rank tests: $p < 0.001$).

The spreadsheet problems of the test proved to be more difficult than the decoding of the pseudo codes of Task 2. However, the result for the decoding of the array formula of Task 3.4 is similar to the decoding of the traditional pseudo codes (Table 3).

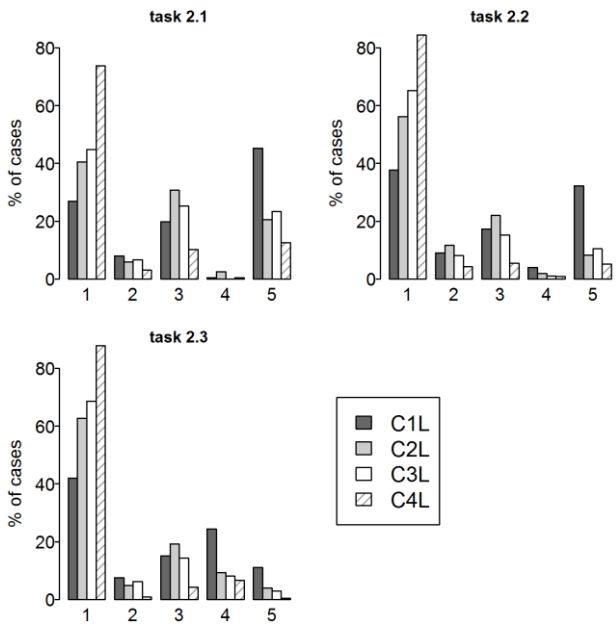


Fig. 8. The students' levels of understanding using the categories of the SOLO-taxonomy.

Table 3

The results of the four clusters in Tasks 2 and 3 based on the logical task (Task 1), and the self-assessment values in programming (S-A-P) and spreadsheet (S-A-S)

	C1L	C2L	C3L	C4L
N	279	205	210	256
S-A-P (%)	*46.9*	*35.9*	*32.4*	*23.0*
Task 1 (%)	**99.8**	**70.3**	**51.2**	0
Task 2.1 (%)	57.3	39.1	37.6	18.7
Task 2.2 (%)	46.1	23.7	20.8	9.5
Task 2.3 (%)	38.8	21.7	17.6	7.7
S-A-S (%)	*75.3*	*72.4*	*74.5*	*71.5*
Task 3.1 (%)	38.5	28.8	26.1	24.7
Task 3.2 (%)	13.5	8.9	9.1	7.4
Task 3.3 (%)	19.3	15.7	11.5	11.8
Task 3.4 (%)	41.0	29.3	32.4	23.4

Using the same knowledge based clusters, C1L–C4L, the results of Task 3 were analyzed (Table 3). The differences between the clusters are significant, except for Task 3.3 (Kruskal-Wallis: Task 3.1: $p < 0.001$, Task 3.2: $p = 0.034$, Task 3.3: $p = 0.3804$, Task 3.4: $p < 0.001$). However, when comparing the different pairs of clusters the differences are not significant. In fact, only a couple of pairs were found with significant differences: in Task 3.1 pairs C1L-C3L and C1L-C4L, in Tasks 3.2 and 3.3 none, in Task 3.4 pair C1L-C4L.

The students were not able to recognize the similarities between Tasks 3.1–3.3 and Task 3.4, and were not able to copy the solution of Task 3.4 to their formulas, but they were able to decode it more easily than the traditional loops. This finding strengthens our previously published results, which showed spreadsheet could be used as an introductory programming language using CAAD-based deep approach metacognitive methods (Biró and Csernoch, 2013a; Csernoch and Biró, 2014b).

We have to confront the unfortunate situation that the clusters which worked well for the categorization of the students in the programming task, do not work in the spreadsheet task; they are not able to distinguish between the different levels of the students' spreadsheet knowledge. This means that the students' spreadsheet knowledge is not connected to their ability to solve logical or programming tasks.

Considering these results, we selected another spreadsheet task in the test in order to create new knowledge-based clusters: 'What is the capital city of the largest country?' – Task 3.0. The characteristic of this spreadsheet task is that it requires both a knowledge of basic spreadsheet functions – INDEX(), MATCH() and MAX() – and the ability to handle multilevel functions, and as such, is related to Mathematics, to programming and to spreadsheet.

Three knowledge-based clusters were recognizable using Task 3.0: C1S, C2S and C3S, moving from the best to the worst, respectively (Table 4, Fig. 9). Those students who scored 0% in Task 3.0 make up C3S, and there is an extremely high number of these students (452 students, 47.58%), compared to those in C4L (256 students, 26.95%).

Table 4
The results of the three clusters based on Task 3.0 and their self-assessment values in spreadsheet (S-A-S)

	C1S	C2S	C3S	Average
N	185	313	452	950
S-A-S (%)	*79.6*	*74.5*	*70.5*	*73.5*
Task 3.0 (%)	**47.64**	**9.78**	**0.00**	**12.5**
Task 3.1 (%)	49.64	36.41	17.41	29.95
Task 3.2 (%)	17.67	15.22	2.98	9.87
Task 3.3 (%)	25.76	20.30	5.25	14.86
Task 3.4 (%)	47.57	37.91	21.17	31.82

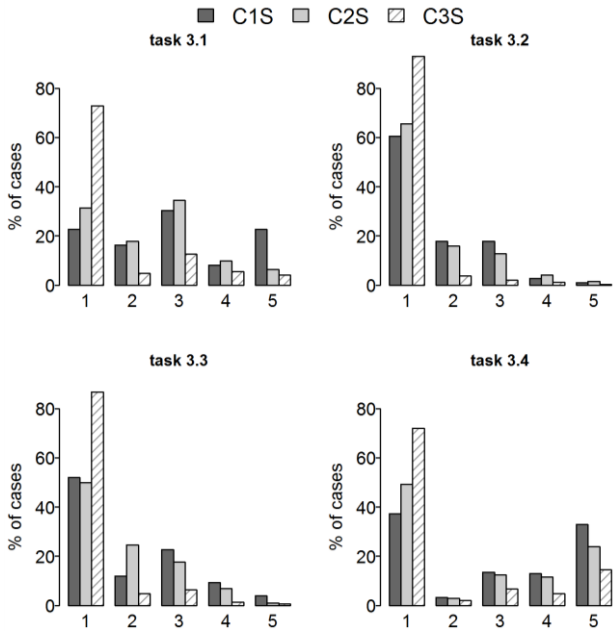


Fig. 9. The results of the three S clusters in solving the spreadsheet tasks, applying the SOLO categories of understanding.

This might explain why only two more clusters were distinguishable on the basis of Task 3.0.

Similar results were found using the C1S–C3S clusters as with the C1L–C4L clusters. There are significant differences between the results of the three clusters; however the comparison of the pairs hardly show any differences between the three clusters (Kruskal-Wallis: $p < 0.001$). We found the following pairs with significant differences: Task 3.1 is the only problem where all the pairs are significantly different; in Task 3.2, 3.3 and 3.4 pairs C1S-C3S and C1S-C3S, which means that only C3S is different from the others, while we cannot deduce anything from the two better groups. We can con-

clude that selecting a spreadsheet task as the basis for the knowledge based clusters proved no better than selecting a logical task (Task 1). These results further strengthen our previously published results which showed that students solve spreadsheet problems with surface approach methods, without supporting their solutions with algorithms. For better understanding and better results in solving spreadsheet problems CAAD-based deep approach methods should be applied. To obtain proof for this result we applied the SOLO categories of understanding to the spreadsheet solutions (Biró and Csernoch, 2014).

Fig. 9 presents the levels of understanding in the spreadsheet tasks of the different S clusters. Unfortunately, in all the three tasks the most frequent level is 1, Ignored. From now on we focus on the non-zero results. C3S stops at Level 1, in all the four spreadsheet tasks. A couple of students in the cluster solved Task 3.4 at Level 5, and some non-zero solutions were found for Task 3.1, but almost nothing for Tasks 3.2 and 3.3. In C2S and C3S in Task 3.1 the most frequent level of understanding is 3, Unistructural understanding. The difference between the two clusters is that C2S stops at this level, while C1S reaches Level 5. In Task 3.2, which is a three folded generalization of Task 3.1, both C2S and C3S stop at Level 3, so they are at the Prestructural and Unistructural levels. In C1S in Task 3.3, which is a two folded generalization of Task 3.1, the most frequent level is 3, but we can find solutions both at Levels 4 and 5. On the other hand, in C2S Level 2 is the most frequent, and the others follow in descending order. In Task 3.4 the pattern is simple: as the number of clusters increases the level of understanding decreases.

It is remarkable that Level 4 is the least frequent. This means that spreadsheet knowledge, unlike traditional programming, can hardly be categorized as Multistructural. Students either know the solution or they do not. We argue that it should not be like this, and that TAEW-based methods have led the students to this state. Beyond this, we have proved that when teaching spreadsheet with a CAAD-based method the importance of Level 4 increases (Csernoch and Biró, 2014b).

4.2.3. *School Leaving Exams*

With the comparison of the students' results in the test and in the school leaving exams we wanted to see what is gained or lost by not considering the Informatics school leaving exam as a compulsory requirement for entering tertiary studies in Informatics.

Both the L and S knowledge based clusters were used to compare the students' results in the test to their results in the school leaving exams. Table 3, Figures 10 and 11 show that there are differences between the clusters in terms of the results of the school leaving exams. The question was whether these differences are significant, and whether the results of the school leaving exams are able to distinguish between the levels of the students or not.

At advanced level Informatics only the C1L result was found to be significantly higher than that of the other clusters (Kruskal-Wallis probe: advanced: $\chi^2(3) = 14.53$, $p = 0.002$, intermediate: $\chi^2(3) = 8.7$, $p = 0.012$). In the comparison of the pairs at intermediate level C1L is different from the others, while at advanced level C1L can be distinguished both from C2L and C3L.

Table 5
The L clusters' results in the school leaving exams

	C1L	C2L	C3L	C4L
	School leaving exam (SLE) – *intermediate level*			
Informatics	84.1 (N=154)	80.6 (N=135)	82.1 (N=147)	82.2 (N=169)
Mathematics	77.2 (N=225)	69.7 (N=177)	72.2 (N=179)	73.1 (N=213)
	School leaving exam (SLE) – *advanced level*			
Informatics	74.6 (N=85)	65.1 (N=40)	65.1 (N=28)	66.3 (N=27)
Mathematics	75.1 (N=24)	62.7 (N=10)	64.0 (N=10)	67.2 (N=19)

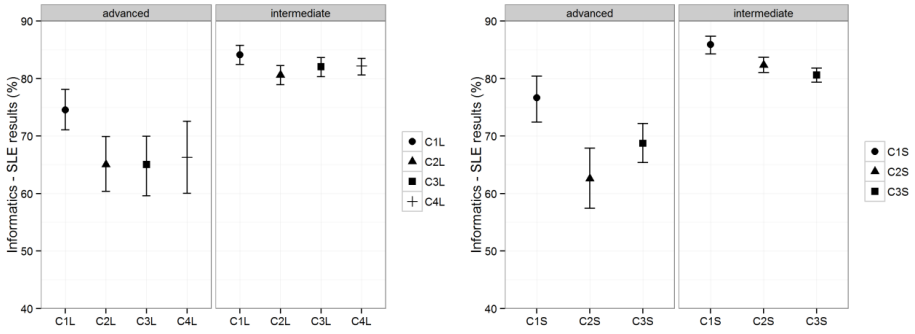


Fig. 10. The results of the C1L–C4L and the C1S–C2S knowledge-based clusters in the school leaving exams in Informatics at intermediate and advanced levels.

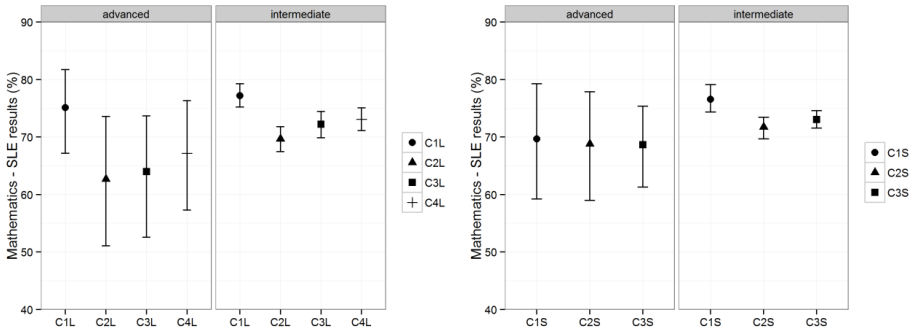


Fig. 11. The results of the C1L–C4L and the C1S–C2S knowledge-based clusters in the school leaving exams in Mathematics at intermediate and advanced levels.

Analyzing the connection between the S clusters and the results of the school leaving exams in Informatics, both at intermediate and advanced level, significant differences were found (Kruskal-Wallis probe: advanced: $\chi^2(2) = 17.88$, $p < 0.001$, intermediate: $\chi^2(2) = 10.16$, $p < 0.001$). Comparison of the pairs shows that C1S is different from both C2S and C3S at both levels but that there is no difference between the C2S and C3S pairs.

The comparison of the students' results in the test and in the Informatics school leaving exams clearly shows that the school leaving exams, especially at the intermediate level, are not able to distinguish between the different levels of knowledge. The best students do better in these exams, while there are no significant differences between the others' results. The connection between the school leaving exams and the S clusters tends to be stronger than with the L clusters, a result which led us to the conclusion that school leaving exams in Informatics, especially at intermediate level do not support the algorithmic approaches, and as such should not serve as a selector for entry to tertiary CSI education.

Consequently, not making the Informatics school leaving exam a requirement for enrolling on a course, nor as a measure to distinguish between students, is not a great loss. Although in itself this is bad news, since it questions the reliability of this exam, on the other hand, it seems a wise decision not to include it as a compulsory subject.

When comparing the results in the Mathematics school leaving exams and the clusters, we must first of all note that a very low number of students take the advanced level exams (Table 5). At intermediate level the school leaving exams, both in the L and S clusters, show significant differences (Kruskal-Wallis probe $\chi^2(3) = 25.79$, $p < 0.001$, $\chi^2(2) = 10.16$, $p < 0.001$). When comparing the pairs in Mathematics in the L clusters C1L was found to be different from the other three clusters, but no other pairs were found which differed from each other. In a similar way, in the S clusters C1S is different both from C2S and C3S, but there is no difference between C2S and C3S.

We can conclude that neither the Informatics nor the Mathematics school leaving exam at intermediate level are able to tell apart the different levels of the students' algorithmic skills and programming abilities. While the school leaving exams at intermediate level were not able to distinguish between the different majors, the advanced Informatics and our knowledge based clusters performed better.

4.2.4. *Clusters and the Majors*

We have seen that the school leaving exams were not able to distinguish between the different majors (Table 2). However, the knowledge based L clusters clearly show that the SOE students are present in the first clusters in the highest percentage, while BIM students appear in the lowest clusters in the highest percentage (Table 6). The BIM and the SOE students move in opposite directions. The SYE students in the L clusters are almost equally distributed (Pearson's Chi-squared test: $\chi^2(6) = 100.1$, $p < 0.001$).

It is difficult to discover anything more about the connection between the majors and the S clusters because of the extremely high number of Level 1 performances produced by the students in the spreadsheet problems (Table 7): in all the three majors the percentage of the students who were not able to solve Task 3.0 is between 45 and 50% (C3S).

Table 6
The number of students in the C1L–C4L clusters compared to the majors

	C1L	C2L	C3L	C4L	**Total**
BIM	47	58	71	124	**300**
	15.68%	19.33%	23.67%	41.33%	**31.58%**
SYE	78	74	74	86%	**312**
	25.00%	23.72%	23.72%	27.67%	**32.84%**
SOE	154	73	65	46	**338**
	45.56%	21.60%	19.23	13.61%	**35.58%**
Total	**279**	**205**	**210**	**2560**	**950**

Table 7
The number of students in the C1S–C3S clusters compared to the majors

	C1S	C2S	C3S	**Total**
BIM	40	111	149	**300**
	13.33%	37.00%	49.67%	**31.58%**
SYE	64	109	139	**312**
	20.51%	34.945	44.55%	**32.84%**
SOE	81	93	164	**338**
	23.96%	27.51	48.52%	**35.58%**
Total	**185**	**313**	**452**	**950**

The tendencies are the following: the percentage of the BIM students in the C1S cluster is the lowest, while their percentage in C3S is the highest. The percentage of the SOE students in C1S is the highest, while the SYE students lie between BIM and SOE. These results are similar to those found with the L clusters.

However, more SYE than SOE students are classified as C2S (Pearson's Chi-squared test: $\chi^2(4) = 15.32$, $p = 0.004$). The differences using the S clusters are statistically significant; however, these clusters are not able to distinguish the majors as well as the L clusters.

5. Conclusions

Testing the first year students of Informatics when starting their tertiary education in Hungary proved that students arrive from secondary education with underdeveloped algorithmic skills, and a low level of understanding programming tasks; consequently, the H1 hypothesis is not proved. Our high level of expectation was mainly based on the high results achieved in the school leaving exams in Informatics. The reason that H1 must be rejected is that we have proved by the analyses of the students' results in the test that

the school leaving exams in Informatics do not measure the students' algorithmic skills. These exams are able to distinguish the best students from the others, but are not able to indicate the differences between the different levels of weakness. Consequently, we also have to reject H2. Our results also suggest that spreadsheet problems are solved with TAEW-based methods, and not through a deep-structural algorithmic approach.

It was found that the students' self-evaluation in programming is quite acceptable. However, we have proved in our previous analyses that the best students are aware of their knowledge, while the worst evaluate their knowledge at approximately the same level as the best (Biró *et al.*, 2015), which is in accordance with the well-known Dunning-Kruger effect (Kruger and Dunning, 1999). However, the spreadsheet self-assessment values greatly exceed the students' real knowledge. Based on the Dunning-Kruger effect, we can conclude that in spreadsheet the students' low level of knowledge prevents them from forming a reliable self-assessment. The H3 hypothesis only works with the best students in programming, and not with any of the preliminary groups, nor the S the knowledge-based clusters.

Our analyses have proved that the students can only think in algorithms in traditional programming environments. They do not regard the recent developments in IT as being driven by algorithms; they solve problems in these non-traditional environments with surface approach methods and are not able to do generalizations and abstractions. Consequently, our H4 hypothesis must be rejected.

Considering the results of the analyses, if the universities rely heavily on the school leaving exams, their course structure and focus should be changed. On the other hand, methods must be developed and introduced in primary and secondary education which focus on the development of the students' algorithmic skills in different environments and in different computer related activities; the development of students' algorithmic skills should be independent of computer environments and computer related activities.

Beyond our local gains and losses, the state of CSI in primary and secondary education, its effectiveness, its efficiency, and its global methodological questions require more data collection and analyses. The problem is not Hungarian specific, it is border. Beyond considering the methodology of CSI, there is a great need for international and national standards of terminology, which should be CSI and methodology driven, instead of being commercially based.

Acknowledgements

The research 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.

The research was partly supported by the Hungarian Scientific Research Fund under Grant No. OTKA K-105262.

References

Ben-Ari, M. (1999). Bricolage forever! *PPIG 1999. 11th Annual Workshop. 5–7 January 1999. Computer-Based Learning Unit*. University of Leeds, UK. Retrieved April 12, 2014. http://www.ppig.org/papers/11th-benari.pdf

Biggs, J.B., Collis, K.E. (1982). *Evaluating the Quality of Learning: The SOLO Taxonomy*. Academic Press, New York.

Biró, P. Csernoch, M., Abari, K., Máth J. (2014). First year students' algorithmic skills in tertiary Computer Science education, In: Papadopoulos, G.A. (Ed.), *Proceedings of the 9th International Conference on Knowledge, Information and Creativity Support Systems, Limassol, Cyprus, November 6–8.* 301–306.

Biró, P. Csernoch, M., Máth, J., Abari, K. (2015). Algorithmic skills transferred from secondary CSI studies into tertiary education, World Academy of Science, Engineering and Technology *International Journal of Social, Education, Economics and Management Engineering*, 9(2), 292–298.

Biró, P., Csernoch, M. (2013a). An algorithmic approach to spreadsheets. In: András, B., Endre, K. (Eds.), *Interdiszciplináris Pedagógia és a Fenntartható Fejlődés*. DE Neveléstudományok Intézete, Debrecen, 310–321.

Biró, P., Csernoch, M. (2013b). Deep and surface structural metacognitive abilities of the first year students of Informatics. In: *4th IEEE International Conference on Cognitive Info-communications*, *Proceedings*. Budapest, 521–526.

Biró, P., Csernoch, M. (2013c). Programming skills of the first year students of Informatics. In: *XXIII. International Conference on Computer Science 2013*, EMT, 154–159. (In Hungarian).

Biró, P., Csernoch, M. (2014). Deep and surface metacognitive processes in non-traditional programming tasks. In: *5th IEEE International Conference on Cognitive Info-communications Conference, 4–5. Nov, 2014, Vietri sul Mare, Italy*. IEEE, 49–54. DOI: 10.1109/CogInfoCom.2014.7020507

Booth, S. (1992). *Learning to Program: A Phenomenographic Perspective*. Acta Universitatis Gothoburgensis, Gothenburg, Sweden.

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.

Clear, T., Whalley, J., Lister, R., Carbone, A., Hu, M., Sheard, J., Simon, B., Thompson, E. (2008). Reliably classifying novice programmer exam responses using the SOLO taxonomy. In: S. Mann, M. Lopez (Eds), *21st Annual Conference of the National Advisory Committee on Computing Qualifications (NACCQ 2008)*. Auckland, New Zealand.

Csernoch, M. (2012). Introducing conditional array formulas in spreadsheet classes. In: *EDULEARN12 Proceedings, Barcelona, Spain. 2–4 July, 2012*. IATED, 7270–7279.

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

Csernoch, M., Balogh, L. (2011). *Algorithms and Spreadsheets*. Magyar Tehetségsegítő Szervezetek Szövetsége, Budapest. (In Hungarian: *Algoritmusok és Táblázatkezelés – Tehetséggondozás a Közoktatásban az Informatika Területén*).

Csernoch, M., Biró, P. (2013a). Teachers' Assessment and Students' Self-Assessment on The Students' Spreadsheet Knowledge. In: L. Gómez Chova, A. López Martínez, I. Candel Torres (Eds.), *EDULEARN13: Proceedings of the 5th International Conference on Education and New Learning Technologies*, *July 1st–3rd, 2013 – Barcelona, Spain*. International Association of Technology, Education and Development, IATED, 949–956.

Csernoch, M., Biró, P. (2013b). Testing algorithmic and application skill. In: Szlávi, P., Zsakó, L. (Eds.), *INFODIDACT 2013*: *Informatika Szakmódszertani Konferencia, Zamárdi, Hungary, 21–22. November*. 1–20. (In Hungarian: Algoritmikus és alkalmazói készségek tesztelése).

Csernoch, M., Biró, P. (2013c). The investigation of the effectiveness of Bottom-up techniques in the spreadsheet education of students of informatics. In: Kozma, T., Perjés, I. (Eds.), *New Research in Education Studies*. ELTE Eötvös Publisher, Budapest. 369–392. (In Hungarian).

Csernoch, M., Biró, P. (2014a). Digital competency and digital literacy is at stake. *ECER 2014 Conference, 1–5. September, 2014, Porto, Portuga*l. The Past, Present and Future of Educational Research in Europe. University of Porto, 1–4. http://www.eera-ecer.de/ecer-programmes/conference/19/contribution/31885/

Csernoch, M., Biró, P. (2014b). Spreadsheet misconception, spreadsheet errors. In: Juhász, E., Kozma, T. (Eds.), *Oktatáskutatás Határon innen és túl. HERA Évkönyvek I.*, Belvedere Meridionale, Szeged, 370–395.

Csernoch, M., Biró, P. (2014c). The power in digital literacy and algorithmic skill. *INTE 2014 Conference, 25–27. June, 2014, Paris, Procedia – Social and Behavioral Sciences*. Accepted.

Csernoch, M., Biró, P. (2015a). Sprego programming. *Spreadsheets in Education (eJSiE)*. 8(1), art. 4. http://epublications.bond.edu.au/cgi/viewcontent.cgi?article=1175&context=ejsie

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

Csernoch, M., Biró, P. (2015c). *Sprego Programming*. LAP Lambert Academic Publishing. ECDL Foundation. Retrieved 10 June, 2014 from http://www.ecdl.com/

Csernoch, M., Biró, P., Máth, J., Abari, K. (2014). What do I know in Informatics? In: Kunkli, R., Papp, I., Rutkovszky, E. (Eds.), *IF2014 Conference, 27–29 August, 2014, Debrecen, Hungary.* University of Debrecen, Faculty of Informatics, Debrecen, 217–230. (In Hungarian: Mit tudok informatikából?).

van Deursen A., van Dijk J. (2012). *CTRL ALT DELETE. Lost productivity due to IT problems and inadequate computer skills in the workplace*. Universiteit Twente, Enschede. http://www.ecdl.org/media/ControlAltDelete_LostProductivityLackofICTSkills_UniverstiyofTwente1.pdf

ECDL Magyarország. Retrieved 10 June, 2014 from http://njszt.hu/ecdl

European Schoolnet: Transforming education in Europe. *Country Reports*. Retrieved 15 May, 2015 from http://www.eun.org/observatory/country-reports

EuSprig (2015). *EuSpRIG Horror Stories*. Retrieved 15 May, 2015 from http://www.eusprig.org/horror-stories.htm

Gove, M. (2012). *Michael Gove speech at the BETT Show 2012*. Published 13 January 2012. Retrieved 15 July, 2014 from https://www.gov.uk/government/speeches/michael-gove-speech-at-the-bett-show-2012

Gove, M. (2014). *Michael Gove speaks about computing and education technology*. Published 22 January 2014. Retrieved 15 July, 2014 from https://www.gov.uk/government/speeches/michael-gove-speaks-about-computing-and-education-technology

Kerettanterv 2000. http://www.nefmi.gov.hu/kozoktatas/tantervek/kerettanterv-2000

Kerettanterv 2009. Az oktatási és kulturális miniszter 2/2008. (II. 8.) OKM rendelete a kerettantervek kiadásának és jóváhagyásának rendjéről, valamint egyes oktatási jogszabályok módosításáról szóló 17/2004. (V. 20.) rendelet módosításáról. Magyar Közlöny, 2008. február 8., 20. szám II. kötet.

Kerettanterv 2013. Kerettantervek. A kerettantervek kiadásának és jóváhagyásának rendjéről szóló 51/2012. (XII. 21.) számú EMMI rendelet mellékletei. http://kerettanterv.ofi.hu/

Kruger, J., Dunning, D. (1999). Unskilled and unaware of it: how difficulties in recognizing one's own incompetence lead to inflated self-assessments. *Journal of Personality and Social Psychology,* 77(6), 1121–1134.

Lister, R., Simon, B., Thompson, E., Whalley, J.L., Prasad, C. (2006). Not seeing the forest for the trees: novice programmers and the SOLO taxonomy. In: *Proceedings of the 11th Annual SIGCSE Conference on Innovation and Technology in Computer Science Education*. New York, USA, 118–122.

NAT 1995. 130/1995. (X. 26.) Korm. rendelet a Nemzeti alaptanterv kiadásáról. Retrieved 15 July, 2010 from http://www.jogtap.hu/iskolai-jogkereso/130-1995-KORM.pdf. http://e-oktatas.barczi.hu/extra/tudasbazis/f-jk/130-1995-korm.html

NAT 2003. 243/2003. (XII. 17.) Korm. rendelet a Nemzeti alaptanterv kiadásáról, bevezetéséről és alkalmazásáról. Retrieved 15 July, 2014 from http://net.jogtar.hu/jr/gen/hjegy_doc.cgi?docid=a0300243.kor

NAT 2007. A Kormány 202/2007. (VII. 31.) rendelete a Nemzeti alaptanterv kiadásáról, bevezetéséről és alkalmazásáról szóló 243/2003. (XII. 17.) Korm. rendelet módosításáról. Retrieved 15 July, 2014 from http://www.zipernowsky.hu/letoltes/kerettanterv/nat_070815.pdf

NAT 2012. 110/2012. (VI. 4.) Korm. rendelet a Nemzeti alaptanterv kiadásáról, bevezetéséről és alkalmazásáról. Retrieved 15 July, 2014 from http://www.njt.hu/cgi_bin/njt_doc.cgi?docid=149257.218573

NT (2009). *Nemes Tihamér National Competition of Informatics 2008–2009 round I, 5–8th grade. Exercices*. (In Hungarian: *Nemes Tihamér Országos Informatikai Verseny 2008–2009 I. forduló, I. korcsoport, 5–8. osztály. Feladatok*). Retrieved May 17, 2013 from http://nemes.inf.elte.hu/2009/nt09-1f1.doc

OECD (2011). *PISA 2009 Results: Students on Line: Digital Technologies and Performance* (Volume VI). http://dx.doi.org/10.1787/9789264112995-en

Polya, G. (1954). *How To Solve It. A New Aspect of Mathematical Method.* (2d ed.) Princeton University Press, Princeton, New Jersey.

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

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.

SLE. *School leaving exams in Hungary – Tasks and correcting guides*. (In Hungarian: Központi írásbeli feladatsorok, javítási-értékelési útmutatók). Retrieved June 15, 2014 from http://www.oktatas.hu/kozneveles/erettsegi/feladatsorok

Sestoft, P. (2010). Spreadsheet technology. Version 0.12 of 2012-01-31. *IT University Technical Report ITU-TR-2011-142*. IT University of Copenhagen, December 2011.

Sheard, J., Carbone, A., Lister, R., Simon, B., Thompson, E., Whalley, J.L. (2008). Going SOLO to assess novice programmers. *SIGCSE Bull.*, 40(3), 209–213.

Soloway, E. (1993). Should we teach students to program? *Communications of the ACM*. 36(10), 21–24.

Tan, G., Venables, A. (2010). Wearing the assessment 'BRACElet'. *Journal of Information Technology Education*: *Innovations in Practice*, 9.

Tort, F. Blondel F.and Bruillard, É. (2008). spreadsheet knowledge and skills of French secondary school students. In: Mittermeir, R.T., Sysło, M.M. (Eds.), *ISSEP 2008*, (LNCS 5090). Springer-Verlag, Berlin, Heidelberg, 305–316.

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

Wing, J.M. (2006). Computational thinking. *Communications of the ACM*, 49(3), 33–35. DOI: 10.1145/1118178.1118215

M. Csernoch was born in 1963 in Szentes, Hungary. She received her MSc in 1986 at Kossuth Lajos University, PhD in 2006 and Habil. in 2012. Currently she works as associate professor at the University of Debrecen, Faculty of Informatics in Hungary.

Her research interests are didactics of Informatics – specialized in developing algorithmic skills, computational thinking, and teaching programming languages –, computational linguistics and computer aided language teaching and learning.

P. Biró was born in 1983 in Ditro, Romania. She received her M.Sc. degree in Computational Mathematics in University of Babes-Bolyai, Faculty of Mathematics-Informatics. Currently she works as assistant lecturer in the University of Debrecen, Faculty of Informatics in Hungary.

Her research interests are didactics of Informatics, algorithmic thinking, high level programming languages and computer aided education.

J. Máth was born in 1959 in Cegléd, Hungary. He received his PhD in 1997 at Kossuth Lajos University. Currently he works as assistant professor in the University of Debrecen, Faculty of Arts and Humanities in Hungary.

His research interests are statistics, knowledge space theory and cognitive load theory.

K. Abari was born in 1971 in Debrecen, Hungary. He received his PhD in 2013 at University of Debrecen. Currently he works as senior lecturer in the University of Debrecen, Faculty of Arts and Humanities in Hungary.

His research interests include Speech Research (acoustic characteristics of speech sounds), Statistical Analysis (functional data analysis) and Statistical Software (R).

Algoritminių įgūdžių testavimas tradicinėse ir netradicinėse programavimo aplinkose

Mária CSERNOCH, Piroska BIRÓ, János MÁTH, Kálmán ABARI

Projektas „Algoritminių ir taikomųjų įgūdžių testavimas“ buvo pradėtas 2011–2012 akademiniais mokslo metais, buvo tiriama pirmųjų metų informatikos studentų algoritmavimo įgūdžiais tradicinėse bei netradicinėse programavimo aplinkose bei kaip jų informatikos žinios persikėlia iš vidurinės mokyklos į universitetą. Tyrimo rezultatai aiškiai parodė, kad studentai pradeda informatikos studijas su labai neišvystytais algoritminiais įgūdžiais, tik keli studentai demonstruoja geresnius rezultatus. Norint rasti priežastis, buvo išanalizuoti studentų taikomi problemų sprendimo metodai. Nustatyta, kad studentai dažniausiai naudoja tik tradicines programavimo aplinkas, skirtas plėtoti informatinio mąstymo ir algoritminius įgūdžius. Be to, jie nenaudoja konceptų ir algoritmais grįstų metodų netradicinėse su kompiuteriu susietose veiklose. dažniausiai taiko neveiksmingus paviršinius metodus, kurie naudojami pradinėse ir vidurinėse mokyklose. Tai gali paaiškinti atotrūkį tarp aukštųjų mokyklų lūkesčių ir mokinių baigiamųjų egzaminų rezultatų bei to, kaip mokiniai patys įvertina (pervertina) savo žinias.