

Thinking Fast and Slow in Computer Problem Solving

Maria Csernoch

Faculty of Informatics, University of Debrecen, Debrecen, Hungary

Email: csernoch.maria@inf.unideb.hu

How to cite this paper: Csernoch, M. (2017) Thinking Fast and Slow in Computer Problem Solving. *Journal of Software Engineering and Applications*, 10, 11-40. <http://dx.doi.org/10.4236/jsea.2017.101002>

Received: November 30, 2016

Accepted: January 20, 2017

Published: January 23, 2017

Copyright © 2017 by author and Scientific Research Publishing Inc.

This work is licensed under the Creative Commons Attribution International License (CC BY 4.0).

<http://creativecommons.org/licenses/by/4.0/>



Open Access

Abstract

Research in spreadsheet management proved that the overuse of slow thinking, rather than fast thinking, is the primary source of erroneous end-user computing. However, we found that the reality is not that simple. To view end-user computing in its full complexity, we launched a project to investigate end-user education, training, support, activities, and computer problem solving. In this project we also set up the base and mathability-extended typology of computer problem solving approaches, where quantitative values are assigned to the different problem solving methods and activities. In this paper we present the results of our analyses of teaching materials collected in different languages from all over the world and our findings considering the different problem solving approaches, set in the frame of different thinking modes, the characteristics of expert teachers, and the meaning system model of teaching approaches. Based on our research, we argue that the proportions of fast and slow thinking and most importantly their manifestation are responsible for erroneous end-user activities. Applying the five-point mathability scale of computer problem solving, we recognized slow thinking activities on both tails and one fast thinking approach between them. The low mathability slow thinking activities, where surface navigation and language details are focused on, are widely accepted in end-user computing. The high mathability slow thinking problem solving activities, where the utilization of concept based approaches and schema construction take place, is hardly detectable in end-user activities. Instead of building up knowledge which requires slow thinking and then using the tools with fast thinking, end-users use up their slow thinking in aimless wandering in huge programs, making wrong decisions based on their untrained, clueless intuition, and distributing erroneous end-user documents. We also found that the dominance of low mathability slow thinking activities has its roots in the education system and through this we point out that we are in great need of expert teachers and institutions and their widely accepted approaches and methods.

Keywords

Computer Problem Solving, End-User Computing, Teaching Materials, Mathability, Meaning System Model, Expert Teachers

1. Introduction

1.1. The State of Art

It is mutually agreed that “real” computer problem solving is mostly related to professional programmers, so end-users do not have to carry out such activities, since they are highly supported by “user-friendly”, “full-proof” environments. Two of the most frequent phrases in end-user computing is “I can use it, that’s enough” and “I can learn it on my own”. This is in accordance with Panko’s finding, which claims that end-user computing is “invisible to IT professionals, corporate managers, and information systems (IS) researchers” [1]. It seems that most of the participants of the digital world are satisfied with the non-existence of end-user computing. However, it has been found that this approach(es) leads to erroneous end-user computing [1]-[11], institutionalized bricolage [6]. The consequences resulting from incorrect figures cause serious financial losses and also mean that human and computer resources are used up in vain [1] [2] [8] [9] [10].

Research found that those involved in end-user computing—central corporate IT groups, general corporate management, information systems researchers [1] [2] [9] [10], education policy-makers, teachers, publishers, and end-users [3] [4] [5] [11]—seem to be blind to effective teaching methods, to error handling, recognition, and correction, to how the brain works, how it can be utilized effectively in digital environments. Even those methods are not adopted which have proved effective in real world problem solving in other sciences and in programming. Kadijevich’s research in spreadsheets found that the widespread misconception is that “(1) there are no interesting problems related to spreadsheets, and (2) teaching spreadsheets is not necessary because spreadsheet learning occurs naturally through practice.” [7]. Conferences on computer education also claim that end-user text management (word processing) is “not really relevant to computing education”, “not focused on computing education”. We claim that these findings and statements are alarming and we provide further proofs that neither education nor end-user programs and their developers support end-user problem solving and the development of the end-users’ computational thinking [12], and these approaches lead to erroneous end-user computing.

1.2. Time for Changes

It is mutually agreed upon that end-users are those participants of the digital world who do not carry out any “serious” computer related activities, which re-

mains the privilege of the trained professional of CS/Informatics (Computer Sciences/Informatics).

As it is mentioned above, end-users are usually untrained or self-trained, and they seem to be invisible to digital professionals. On the other hand, they can be self-confident to the extremes, especially those who are trained in Informatics and find themselves in end-user roles. (The sample of **Figure 1** clearly presents how an overconfident end-user evaluates his knowledge. The list of IT Skills in general holds the major end-user subjects, but even their classification shows lack of knowledge.)

To clarify end-user activities and roles in the digital world, we launched a project focusing on computer problem solving (CPS). Our project is different in nature from previous studies, where only one aspect of end-user computing is thoroughly examined—spreadsheet, text, presentation, etc. management. We claim that CPS is an umbrella project, since end-user activities are considered in symbiosis, where knowledge transfer plays a crucial role in developing computational thinking.

In the present paper, we argue that (1) end-user computing should not differ in nature from any other problem solving, (2) effective teaching methods and approaches can be adapted from other sciences (3) novel methods can and have to be invented and introduced, (4) what teachers do matters, and (5) what “some” teachers, the expert teachers, do matters [13] [14].

1.3. Sampling Process

Considering that we launched a pilot project, our first concern was the data collection and the analyses of the available sources in the subject, in different languages and countries. The sampling process was planned in advance, but during

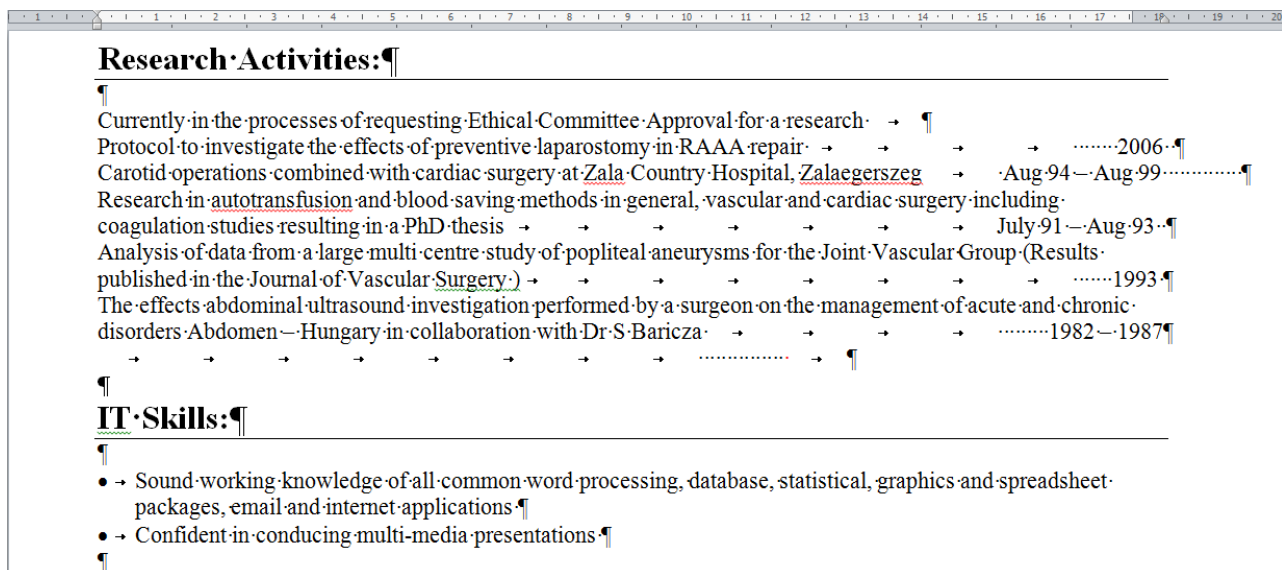


Figure 1. The end-user’s lack of knowledge in word processing is clearly presented in Section Research Activities, while his overconfidence is revealed in the comparison of his product and his knowledge listed in Section IT Skills. It is not clear what he meant by “internet applications”, “email”, why “multi-media presentations” is listed separately, and what meant by “all” in his first sentence.

the work, without any previously published guides, we had to reframe our methods and strategies. (Reasons for this: Lack of financial support, lack of availability, language barriers, the diversity of countries, lack of acceptance of the subject, etc. However, we would like to express our gratitude for all the volunteers for helping us.)

We planned to collect teaching, learning materials developed both for institutionalized and self-trained educational purposes, user guides, helps for supporting daily end-user activities, and sources of widely accepted approaches to end-user computing and teaching. At the time of the publication, we hold 85 and 23 printed materials in our native and foreign languages, respectively. Beyond the printed materials we heavily rely on sources available on the Internet and in the built-in helps. The number of digital sources increases more rapidly than of the printed sources for at least three reasons: (1) their easy access, (2) the online versions of the printed materials, (3) and the changing proportion of the printed and digital sources. Beyond the official supporting sources, we have a huge private collection of end-user documents, the results of end-user activities. This collection consist of several hundreds of documents downloaded from the Internet, sent in emails, collected in our previous projects, and collected by supporting researchers and students.

After collecting, analyzing, and evaluating the content of the sources, the results were compared to the results of recent studies in education, thinking modes, and problem solving. This comparison clearly revealed the discrepancies which led us to end-user bricolage, ineffective end-user computing.

In the present paper, we provide samples of the collected sources (Section 5) and the reason why most of the teaching and supporting materials do not match the requirements of effective end-user problem solving. Those materials which are found supporting are mentioned in Section 7, but their number is infinitely small, compared to the number of the analyzed materials. Here, however, we have to mention that our collection is only a thin slice of the existing materials, and consequently, we are open to any further sources which we do not have access.

2. Problem Solving Approaches

To talk about different computer problem solving approaches we all need a typology which consists of all the possible approaches and their definitions. At the beginning of our project, we were faced with the lack of such typology; back then, there was no available complete typology which consisted of all end-user problem solving approaches. Consequently, our major concern was at that time to reveal what other sciences and computer programming had already recognized. We found, by comparing the contents of several typologies, that there is a considerable overlapping, however (1) the terminology used in the typologies and (2) the non-recognition of end-user computing would explain that these similarities are not obvious and that the typologies do not cover all end-user problem solving approaches. The results of our analysis, comparison, and exten-

sion are summarized in Section 2.1.

In the following phase of our project, we proposed the extended typology of end-user problem solving approaches, where the mathability levels of the named approaches were set up. Such ranking of the mathability levels allows us to a quantitative measuring system of the recognizable methods (Section 2.2).

2.1. Typologies of Problem Solving

Our typology of computer problem solving approaches (published in 2014 in Hungarian) [15] identifies five clearly distinguishable approaches in two hypernym classes—deep approach (DA) and surface approach (SA) classes (Figure 2, Figure 3).

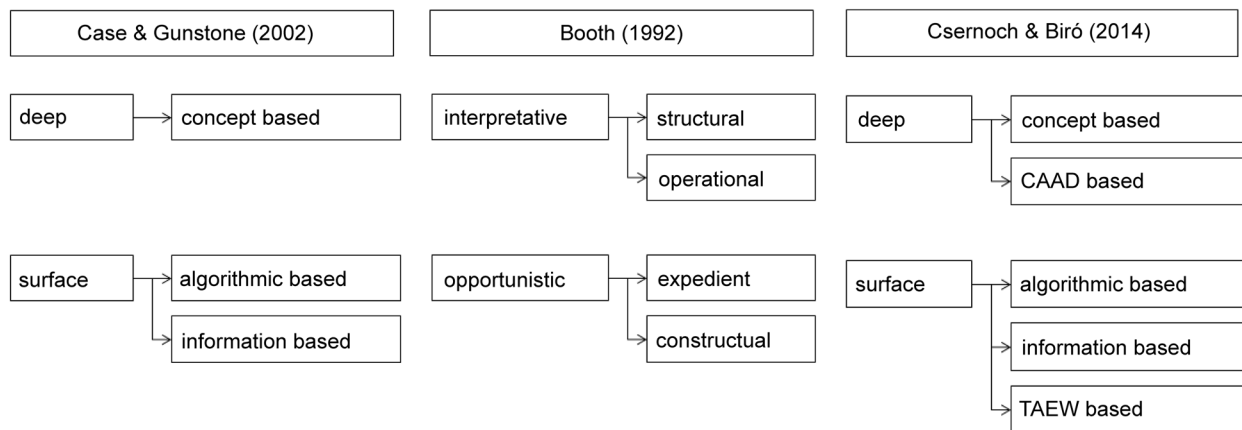


Figure 2. Two typologies of problem solving approaches in sciences (Case & Gunstone) and computer programming (Booth) (left and middle, respectively), which partially cover the end-user problem solving approaches, and our extended typology presented in 2014 (Csernoch & Biró) (right).

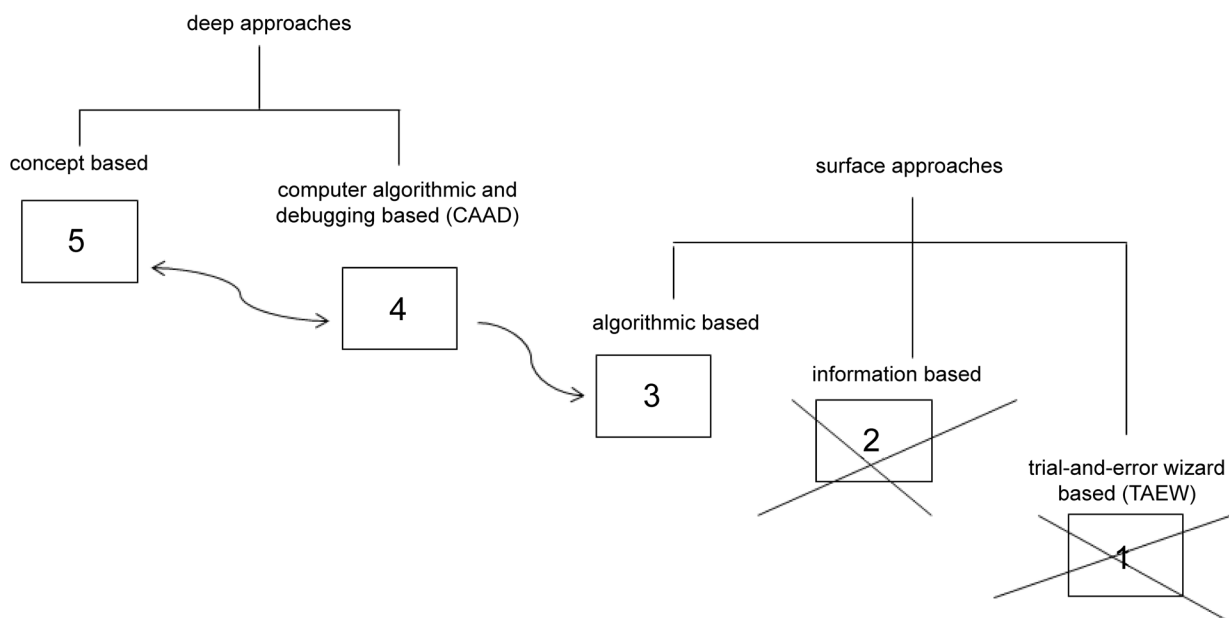


Figure 3. The mathability of computer problem solving approaches, introduced in 2015 by Biró & Csernoch, allow us to a quantitative measuring system for developing and evaluating teaching and guiding materials in end-user computing.

Some methods of these approaches are adapted from other sciences, while others are recognizable mainly in computer environments, computer related problem solving, and in data and information management.

The concept based approach (DA class), the algorithmic and the information based approaches (SA class), are well defined in Case & Gunstone' typology for problem solving in sciences [16] (Figure 2). All three approaches are identified, with similar contents in Booth's typology [17] for functional programming a decade earlier (using different terminology, Figure 2), remained almost unnoticed. In her typology, Booth named one more DA hyponym category, which focuses on building algorithms for programming problems (operational). The similar contents of the categories made it clear for us that these typologies can be merged, expanded, and tuned to fit the requirements of the digital era [15] (Figure 2).

For emphasizing that building algorithms (Booth—operational) and applying algorithms (Case & Gunstone—algorithmic based) requires different thinking modes, slow and fast, respectively [1] [25], in the DA hypernym class, we introduced the “Computer Algorithmic And Debugging” expression (CAAD) [15] [17], instead of Booth's operational. Our expression also includes the debugging process, which is essential in the testing and evaluating process of the computer generated outputs.

Since our concern was to identify all the end-user computing approaches, we had to expand the already existing typologies and added the Trial-And-Error Wizard based (TAEW) SA methods [15]. The trial-and-error activities are different in nature from the others, since they are related to problem solving, but real problem solving is hardly carried out; instead, a surface navigation over-arches the whole process, where any output, if reached, is accepted. This approach is not unknown in methodology, but not considered as problem solving, which would explain not being included in any of the previous typologies.

However, TAEW is so widely accepted and practiced in computer related activities that we cannot ignore it anymore. The question is why TAEW and information based approaches are so popular in the digital world, while they seem to be the detours of computer problem solving. For the presence of the TAEW based approach one of the best explanations can be adapted from Polya [18]. He claimed that

“We know, of course, that it is hard to have a good idea if we have little knowledge of the subject, and impossible to have it if we have no knowledge. Good ideas are based on past experience and formerly acquired knowledge.” ([18], p. 9)

Considering the information based approach, Polya stated that

“Mere remembering is not enough for a good idea, but we cannot have any good idea without recollecting some pertinent facts; materials alone are not enough for constructing a house but we cannot construct a house without collecting the necessary materials.” ([18], p. 9)

Rewording Polya's note in the digital world, we find that computers alone are not enough for solving problems but we cannot solve problems without "collecting" the necessary hardware and software tools. However, we have to note here that it is difficult to find the border line between the information and the TAEW based approaches, since both focuses on the tools, without considering the problem; "I can use it, that's enough". The difference between them is rather on the proportion of learned materials. The ECDL exams, for example, require the candidates to follow a long list of familiar instructions, without giving any thought to the problems. These participants of the digital world mainly carry out information based activities during the ECDL exams. However, in other situations, this knowledge is hardly useful, so TAEW based methods are applied outside of the exams. We also found that most of those end-users who never had the opportunity to participate in any formal digital education primarily carry out TAEW based bricolage and they cause serious financial losses [8]; we have reached a level which Polya considered as the worst.

"The worst may happen if the student embarks upon computations or constructions without having understood the problem." ([18], p. 6)

2.2. The Mathability of Computer Problem Solving Approaches

The results of one further research let us complete our typology, namely research in mathability [19] [20] [21] [22]. According to the authors, they apply the concept of mathability to the usage of computer tools.

"This usage has basically two forms:

- (1) In some cases we use existing functions and methods provided by a system, and we apply these tools to solve the problems.
- (2) Another possibility is, if we, based on existing means of the system, develop new programs and functions for solving new problems."

In our typology-building process, we recognized that the two approaches to the usage of computer tools matches the two hypernym classes of problem solving approaches. The first usage covers the surface approach (SA) methods, while the second usage the deep approach (DA) methods. This finding of ours, led us to further consequences: based on our completed computer problem solving typology [15] and the concept of mathability [19] [20] [21] [22], in 2015, we defined the mathability of software tools [19] [20] [21] [22].

The adaptation of the mathability of software tools to computer problem solving approaches allows us to create a quantitative measuring system [23] [24]. In this measuring system, the concept based approach is associated with the highest mathability level, Level 5, while the TAEW based approach with the lowest level, Level 1. The DA CAAD, and the SA algorithmic and information based methods are at Level 4, Level 3, and Level 2, respectively, presented in **Figure 3**.

With the association of the typology of end-user problem solving approaches

and the mathability of software tools, we invented the mathability of end-user problem solving approaches, which provides us guide lines in recognizing the mathability of problem solving approaches, in teaching methods, in tasks presented in classes, in teaching materials, in exams, in developing curricula, frameworks, etc. With this measuring tool we would be able to provide comparable quantitative values associated to teaching and guiding materials.

Due to the facts that our end-user problem solving typology is an extension of previously published typologies in sciences and computer programming and that problem solving in general in the digital era is mostly computer based, our typology would be accepted in other sciences and programming also. The introduction of our mathability associated typology into other subjects is our further concern.

The arrows in the typology of computer problem solving (**Figure 3**) indicate the direction of communication between the different mathability levels. In real world problem solving there is continuous communication between Levels 5 and 4, until the problem is understood, the relations between the input data and the required output(s) are recognized, and the plan—the algorithm—is completed.

If the problem is too complicated we can simplify it, until a stage is reached which leads to discussable or acceptable results. This second option, namely the acceptance of partial results, is a well-known phenomenon in problem solving. Polya suggested that if the problem seems too difficult

“...we must look around for some other appropriate point of contact, and explore the various aspect of our problem; we have to vary, to transform, to modify the problem. Could you restate the problem?” ([18], p. 10)

What Polya suggested, Kahneman [25] proved, and stated that

“This is the essence of the intuitive heuristics: when faced with a difficult question, we often answer an easier one instead, usually without noticing the substitution. The spontaneous search for an intuitive solution sometimes fails—neither an expert solution nor a heuristic answer comes to mind. In such cases we often find ourselves switching to a slower, more deliberate and effortful form of thinking.” ([25], p. 12)

In general, we can conclude that in our typology we can recognize four approaches which require slow thinking: the DA methods for planning, substituting, and discussing—concept and CAAD based approaches—and the two lowest mathability approaches for the aimless usage of the tools—information and TAEW based. There is only one computer problem solving class which utilizes fast thinking; this is the algorithmic based approach (SA class). The proportion of slow and fast thinking approaches is 4:1. Being aware of this number, there is no wonder that end-user computing is erroneous.

2.3. Increasing the Proportion of Level 3 Activities

Activities at Level 4 have multiple purposes. Beyond building algorithms and

discussing and debugging the outputs, the construction, association, and accommodation of schemata [26]—schemata construction for short—also take place at this level. Schemata construction plays a crucial role in the learning process. Back in 1971 Skemp referred to this process as intelligent learning [26]. In novel research—e.g. in Cognitive Load Theory [27]—the multilevel hierarchy of schemata is considered the measurement of proficiency.

These findings are in complete accordance with Kahneman, when he states that System 2 (thinking slow or Attention Thinking Mode, ATM) is the only one that can follow rules, compare objects on several attributes, and make deliberate choices between options, and a reliable System 1 (thinking fast or Automatic Thinking Mode, AUM) requires knowledge built up in learning and practice [25].

“As you become skilled in a task, its demand for energy diminishes. ... the pattern of activity associated with an action changes as skill increases, with fewer brain regions involved.” ([25], p. 35). “...repetition induces cognitive ease and a comforting feeling of familiarity.” ([25], p. 66)

Applying this rule, we can transfer knowledge from the highest mathability levels to Level 3, where System 1 can work. For reducing the errors in spreadsheet management a reliable System 1 should be used, suggested Panko in his paper, entitled “The Cognitive Science of Spreadsheet Errors: Why Thinking is Bad” [1].

Consequently, the algorithmic based approach is different in nature from the information and the TAEW based approaches in the SA class. The algorithmic based level is where System 1 operates. At this level the stored schemata are activated, and decisions are made quickly based on these schemata. On the other hand, the information and the TAEW based levels heavily rely on System 2, which Panko claimed as erroneous approaches to spreadsheet problem solving. Consequently, we can claim that mathability Levels 2 and 1 are pseudo problem solving approaches. At these levels there is no real problem solving, the goal is to perform satisfactorily in tests or exams and/or to achieve any kind of output. The information based approach [16] or habit learning [26] focuses on the details of the tools, without recognizing or understanding the problems. The TAEW based approach is pure surface navigation [15]; aimlessly clicking and browsing until some form of output is produced. In these low mathability approaches the problem is not determined, plans are not built, and consequently the discussion of the outputs is not part of the process; in general, leading to erroneous end-user computing, documents, conclusions, and consequences. System 2 is heavily used, instead of the fast System 1, however, when System 1 responds there is no background knowledge to make reliable decisions ([25], pp. 237-239).

Kahneman explains that for a reliable System 1 we need intuition and real experts who can trust their intuition. (The definition of intuition is from Herbert Simon and cited in Kahneman ([25], p. 237):

“The situation has provided a cue; this cue has given the expert access to information stored in memory, and the information provides the answer. Intuition is nothing more and nothing less than recognition.”

However, it is difficult to define who the real experts are. Even Klein and Kahneman took time to find the sources of their misunderstanding.

“He [Klein] was more willing to trust experts who claim an intuition because, as he told me, true experts know the limits of their knowledge. I argued that there are many pseudo-experts who have no idea that they do not know what they are doing (the illusion of validity), and that as a general proposition subjective confidence is commonly too high and often uninformative. ([25], p. 237), [28]

The illusion of validity is not unknown and has been researched extensively. Some of the earliest works is the paper of Kruger and Dunning [28], who cited Confucius claiming that

“Real knowledge is to know the extent of one’s ignorance.”

Further considering the subject of true experts Kahneman explains that acquiring a skill is fundamental.

“If subjective confidence is not to be trusted, how can we evaluate the probable validity of an intuitive judgment? When do judgments reflect true expertise? When do they display an illusion of validity? The answer comes from the two basic conditions for acquiring a skill: (1) an environment that is sufficiently regular to be predictable (2) an opportunity to learn these regularities through prolonged practice. When both these conditions are satisfied, intuitions are likely to be skilled.” ([25], p. 240)

3. Hypotheses

Considering the different approaches to end-user computing and the education of end-users, our hypotheses are the following:

[H1] End-user teaching materials do not support high mathability end-user computer problem solving.

[H2] Teachers unconditionally accept low mathability end-user computing teaching materials.

[H3] Computer Sciences/Informatics Education research does not consider end-user computing and problem solving as important as programming.

[H4] End-user-program developers and end-user-programs do not support understanding and effective computer problem solving.

In our hypotheses the generally accepted views, the main tracks, the most widely accepted, and institutionalized approaches are in the focus. However, it is also expressed in the present paper that there are teachers, teaching materials, approaches which realized the dangers and consequences of the institutionalized bricolage, and try to introduce their findings.

4. The Meaning System Model Applied to End-User Computing

The question we need to address is what it is that leads to low mathability end-user activities. To find an explanation we have analyzed both the direct human participants in the teaching-learning process—teachers and education policy makers—and the indirect human participants—ICT (Information Communication Technology), IT (Information Technology), CS (Computer Sciences) curricula, teaching materials, and software supports.

It is wildly accepted that end-user computing is boring [35], comprising low level routine tasks [36], nothing more than computer skills [32]; consequently, it is not suitable for developing students' or end-users' computational thinking [12], and as such it is restricted to serious programming. These opinions are in accordance with the software companies' "user-friendly" slogans, and rest on the assumption that low mathability approaches are sufficient and effective in end-user computing.

It is obvious that low mathability approaches are not in accordance with those opinions and approaches which claim that there are various tools beyond declared programming languages which would serve the development of algorithmic skills and the computational thinking of students [37]-[42], and their problem solving skills.

The meaning system model of Chen *et al.* [43] (Figure 4) explains the connection between teachers' belief in the nature of science, their goals, and the results achieved by students in the teaching-learning process.

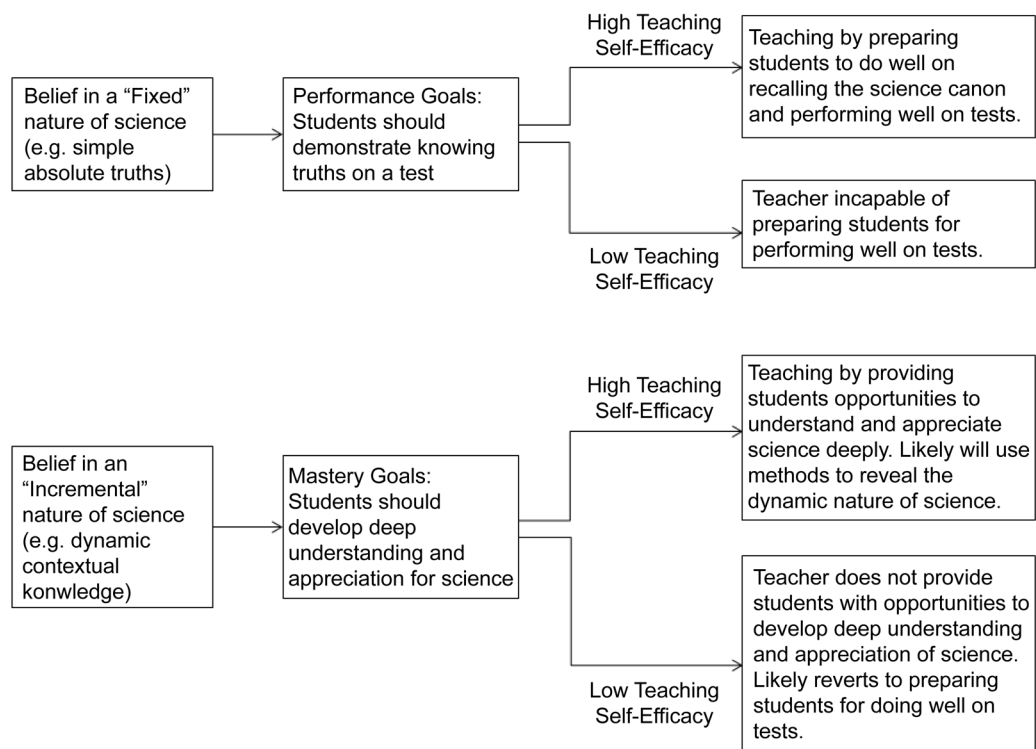


Figure 4. The meaning system model [43].

End-user activities are mostly taught with a belief in the fixed nature of science—if it is considered science at all. In this framework the various software tools are separated from each other, and emphasis is placed on toolbars, menus, and the programming languages as something unique to each application and their different versions. This approach is well presented by the teaching materials which we have analyzed (Figures 5-17) and also by the Spreadsheet Competency Framework, launched in 2016 [44].

5. Low Mathability End-User Activities and Training

5.1. Low Mathability Tasks

In end-user computing the steps of problem solving are reduced to one: “carry out” something, use the computer, the software, the Internet. In end-user terminology this is usage. Most of the end-users are convinced that this is what matters. However, in practice it is nothing more than surface navigation [15], *i.e.* bricolage [3]. As it was mentioned in the previous sections, on the mathability scale these approaches are at the lowest levels [23]. In educational environments we can distinguish two different approaches to software usage (software usage ≠ problem solving): (1) apply whatever is available, regardless of whether the content requires it or not (Figure 5, Figure 9, Figure 17), if there is any content (Figure 6, Figure 10, Figure 11, Figure 12), (2) follow the instructions step-by-step regardless of their correctness or of their functionality (computer cooking for short) (Figure 7, Figure 8, Figure 10), and the combination of the two (Figure 6). In the following we present samples of low mathability tasks, where the focus is on the software tools.

In end-user text management, Level 2 activities are present when end-users are forced to learn how to apply various font and paragraph formattings to pieces of texts—they learn the tool—, but they do not plan or design the document

▪ **Star Wars Family Tree**

Anakin and Padme are married.	Luke and Leia are their children.	Anakin and Padme are the parents of Luke and Leia.	Luke and Leia are siblings.
Leia falls in love with Han Solo.	They get married and a child is born, Ben.	Leia is Ben's mother and Han Solo is Ben's father.	Luke is Ben's uncle. Luke's nephew is Ben.
Anakin is Ben's grandfather and Padme is Ben's grandmother.	<pre> graph TD P[Padme] --- M1(()) A[Anakin] --- M1 M1 --- L[Luke] M1 --- M2(()) M2 --- M3(()) M3 --- H[Han] M3 --- B[Ben] </pre>		

Figure 5. An example of low mathability problems, focusing on the software tools and features. Star Wars Family Tree. The sample is part of a lesson plan, written by a pre-service teacher of Informatics.

In the first line, set the font type to Courier New, the font size to 14 pt, and the font style to bold. ¶
 In the second line, set the font type to Arial, the font size to 16 pt, and the font style to italic. ¶
In the third line, set the font type to Comic Sans MS, the font size to 18 pt, the font style to underline, and the font color to red. ¶
In the fourth line, set the font type to Times New Roman CE, the font size to 20 pt, the font color to blue, and the font style to double underline. ¶

Figure 6. A text without any specific content, created only for the manipulation of software tools (sample text of a lesson plan written by an in-service teacher of Informatics).

according to its content; they do not know whether applying these tools leads to a typographically correct document or not. In **Figure 5** a Star Wars Family Tree was created, however the over- and misuse of tools (table, alignments, colors, font types, font styles) leads to the loss of the content.

The task in **Figure 6** is borrowed from a lesson plan of an in-service teacher of Informatics. However, this piece is not unique, since several similar ones can be found on the Internet. The greatest problem with this type of tasks is that they have no content, and without content we cannot associate any format, consequently, the task is completely meaningless.

According to the lesson plan, the teacher gave instructions which the students had to follow: (1) the teacher wrote the text on the board, (2) the students had to type it, (3) and then the lines and pseudo lines had to be formatted according to the “computer cooking” instructions of the teacher and/or the text, without giving any thought of typography and content.

Unfortunately, one of the greatest institutionalized computer cooking is the ECDL exam system, where long lists of demands have to be performed in pre-practiced tasks (**Figure 7**, **Figure 8**). Beyond licensing low mathability end-user activities, one further serious consequence of this approach is that both non-expert teachers and students are misled;

“...they identify computer science with a computer driving license. They think that studying computer science is not a challenge, and that anybody can learn it. Computer science is not considered a scientific discipline but a collection of computer skills.” [30].

In **Figure 7** and **Figure 8** an ECDL Excel and PowerPoint sample is presented, respectively. Without further details, the samples clearly demonstrate that the primary purpose of these tasks is forcing the students to follow instructions and giving marks for pure software usage. In PowerPoint Task 8 even the picture is distorted by changing both the height and the width. We also have to call attention to the inattentive terminology usage: spreadsheet application is opened (**Figure 7**), while presentation application is started (**Figure 8**).

1. Open the spreadsheet application and open the file called **improvements.xlsx** from your candidate drive. Save the file as **costings.xlsx** to your candidate drive. [1 Mark]
2. On the **projection** worksheet, zoom the display to **100%**. [1 Mark]
3. Widen **column A** so that the content of the column is fully visible. [1 Mark]
4. Enter the number **2,000** into **cell C7**. [1 Mark]
5. Enter a function in **cell B11** to calculate the sum of the **cell range B5:B10**. [1 Mark]
6. Copy the sum function in **cell B11** to the **cell range C11:F11**. [1 Mark]

Figure 7. An ECDL Excel example of computer cooking [29].

1. Start the presentation application and open the file called **sample answerfile6.1.pptx** from your candidate drive.
8. On slide 5 titled **Italy** resize the "Italian Flag" image so that it is 2.36 cm high and 3.81 cm wide and save. [1 Mark]
9. On slide 5 enter the following bullet point text as below: [1 Mark]
 - **Store opening in Milan**
 - **New fashion ranges**
 - **20% discounts**

Figure 8. An ECDL PowerPoint example of computer cooking [29].

In **Figure 9** the picture is positioned behind the text. The author used the tool, but did not pay attention that with this setting of the picture the text would become unreadable.

Further examples of Level 1 activities in text management are when Space, Tabulator, and Enter characters are typed until end-users think that the arrangement of the text is "not too bad"; it would look acceptable or quite well in the printed form (**Figure 1**, **Figure 9**, **Figure 15**). In spreadsheet management, the argument list of a function is filled in until there is no syntactic error in the formula. In presentations, animation sequences are created, loaded with annoying entrances and exits and/or with redundancies. Software independent features are the meaningless and misleading cell coloring, alignments, and diagrams [31], as well as animations, transitions, and mismatching pictures and figures [32].

One further consequence of not using real contents is that we are losing an effective motivating tool. Contemporary students want to acquire knowledge which is directly adaptable to the real world around them. They do not necessarily like purely educational purpose software, in spite of its good quality. Students want immediate links to everyday life [33] [34] and end-user problem solving would be a link between programming and document management.

5.2. Low Mathability Teaching Materials

If teachers believe in the incremental nature of science, this changes their teaching goals, and developing a deep understanding and appreciation of science in the students becomes the focus of the teaching. However, even with this teacher-belief the teaching-learning process can go astray if passing exams is the main



Figure 9. The sample presents (1) an example of using a tool (picture positioned behind the text, making it unreadable) without checking its effect and (2) an example of the irrational use of Space and Enter characters from item 10 on the list.

goal in this process, rather than the use of methods to reveal the dynamic nature of science.

In computer problem solving the dynamic nature of CS/Informatics can be revealed through knowledge transfer, which is primarily based on the high-mathematicity approaches to problem solving.

Real world problem solving, presented through authentic contents, is designed to increase motivation and give opportunities for practice, and ultimately to enable the building of concepts. Students who are interested in the contents are more likely to be interested in the problems based on them. For the development of concepts practice, problem solving, and repeated activities are needed. Similar to other sciences,

“A concept requires for its formation a number of experiences which have something in common. ... Concepts of a higher order in a hierarchy than those which a person already has cannot be communicated by definition.”
[26].

In the following end-user activities, we have found, in the comparison of teaching materials which aim is to support office document management, repeated contents general ICT knowledge, over tens or hundreds of pages: how to start the program, what are in the window, in the toolbars, in the menus, what the new features are, how to color, how to create a border, how to open a file, how to save a file, how to save a file in older formats, how to make a selection, and how to copy and move, etc. There is no reference to the fact that these are

basic ICT skills, which are commonly found in all software tools, and as such, clearly presents the teachers', publishers' and software companies' fixed belief in the nature of science, that the different applications should be taught, handled, and used in seclusion. The connections and the common features of these similar programs are not discovered, the knowledge transfer between them is ruled out, the revelation of knowing from other situations is never reached, and these programs are not only tools, but the only and ultimate purpose of the teaching and guiding process. Consequently, they do not fulfill the requirements of textbooks [45]. According to Chen *et al.* [43], this approach results, in the best case, in students passing the required exam.

The other feature of low mathability teaching materials is that there are no problems presented at all, no real world problem solving, and no design; only surface navigation and computer cooking. We have also found that if tasks are presented, they are mostly meaningless—with no content at all, or at most a couple of lines of fictitious content. These types of pseudo-contents are extremely boring (Figure 6, Figure 10, Figure 11). The second set of tasks of Figure 10 is even more spoon-feeding than the first one, since here the functions are named. Students do not have to think and do not have to make the least mental effort; there is no problem to solve, the only requirement is to follow meaningless orders. This teaching approach is in complete accordance with the requirements of the ECDL and the school leaving exams of our country. The students pass these exams to everyone satisfaction. However, due to the low

1. Type numbers in cells A2:A5 then calculate the sum of them with the SUM function in cell A6.
 2. Type numbers in cells B2:B5 then write out the smallest of them with the MIN function in cell B6.
 3. Type numbers in cells C2:C5 then write out the largest of them with the MAX function in cell C6.
 4. Type numbers in cells D2:D5 then write out the average of them with the AVERAGE function in cell D6.
 5. Type numbers and texts in cells E2:E5 then write out with a suitable function in cell E6 that how many numbers are among them.
1. Type addition in cell A1. Type two numbers in cells A2 and A3 then calculate the sum of the two numbers in cell A4.
 2. Type subtraction in cell B1. Type two numbers in cells B2 and B3 then calculate the difference of the two numbers in cell B4.
 3. Type multiplication in cell C1. Type two numbers in cells C2 and C3 then calculate the product of the two numbers in cell C4.
 4. Type division in cell D1. Type two numbers in cells D2 and D3 then calculate the quotient of the two numbers in cell D4.

Figure 10. Boring, meaningless, computer cooking tasks from an official Informatics course book for grades 9 - 10 [47].

	A	B	C	D
1		Signe	Exemple	Résultat
2	Addition	+	=45+78	123
3	Soustraction	-	=854-584	270
4	Multiplication	*	=12*43	516
5	Division	/	=9394/854	11
6	Puissances	^	=12^3	1728

	A	B	C	D
1		Signe	Exemple	Résultat
2	Addition	+		=A8+A9
3	Soustraction	-		
4	Multiplication	*		
5	Division	/		
6	Puissances	^		
7				
8		987		
9		123		
10				

Figure 11. Boring, meaningless tasks from a French online tutorial [48].

mathability teaching approaches, their knowledge and problem solving methods stay at low mathability levels, which we found in the TAaAS project [56] by testing first year students of Informatics in tertiary education.

In **Figure 10** and **Figure 11** we present the same low mathability tasks in printed and online forms. Recently, most of the teaching materials published online, proposing that that form matches the requirements of the digital children, the way they think, the way they learn. However, we claim that it does not matter in which form low mathability materials are presented, since they would not help the understanding and the appreciation of the science.

The sample of **Figure 12**, copied from another tutorial, tries to explain formulas on an empty table. In this example not even pseudo-data is typed, not even computer cooking is required, making it completely meaningless and useless.

It is also remarkable that these meaningless tasks are country and language independent; similar “tasks” can be found in different teaching materials, regardless of language and location. The spreadsheet examples of **Figures 10-12** are from teaching and tutorial materials of three different places of the world, in three languages, and they are in complete accordance with the ECDL exam samples of **Figure 7**, **Figure 8**.

5.3. Low Mathability Help Materials

Help features and materials are the specialties of the digital world, representing both a blessing and curse. These materials are mainly written by programmers untrained in psychology and education, using language understandable to professionals in Computer Sciences and programming.

In the following spreadsheet examples (**Figure 13**, **Figure 14**), concepts are used which end-users have not mastered; consequently they will not understand the functions described in the wizards.

In the definition of both the IF() (**Figure 13**) and the MATCH() functions (**Figure 14**) concepts are mentioned which end-users usually do not have (e.g.

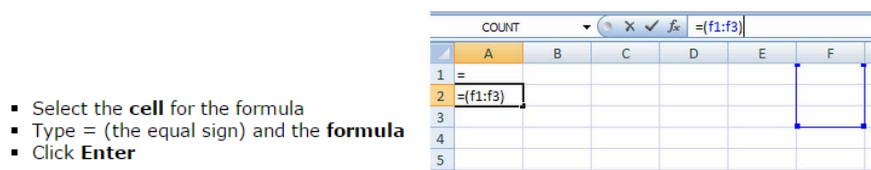


Figure 12. Formula created on an empty table [46].

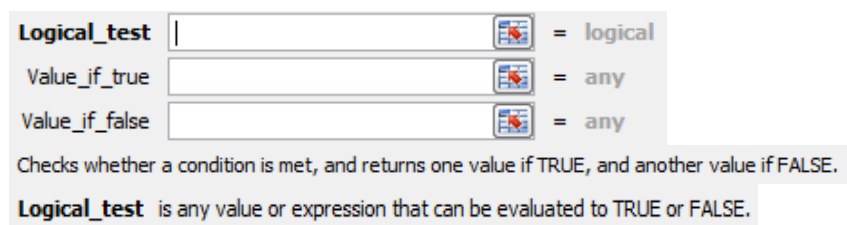


Figure 13. The definition of the IF() function in MS Excel.

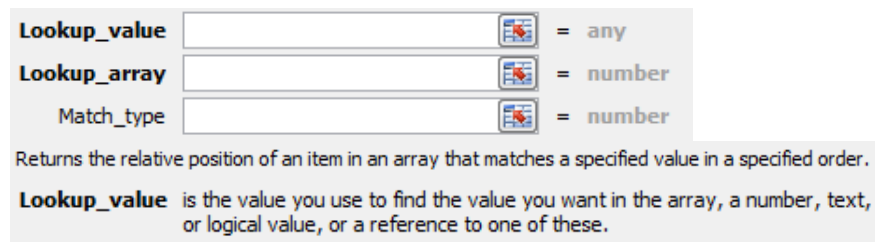


Figure 14. The definition of the MATCH() function in MS Excel.

“argument”, in IF(): “logical test”, “condition”, “any value or expression”, “evaluated”, in MATCH(): “array”, “relative position”, “item”, “specified value”, “specified order”, “logical value”, “reference”).

It also frequently occurs that these definitions are not correct, and furthermore, the published teaching materials repeat the errors, and teachers also accept them without any more thought.

Presented in **Figure 14**, in the definition of the MATCH() function, “lookup_array” is mentioned as the second argument; however, any array cannot be entered, only a vector, which is a one-dimensional array. In the analysis of the teaching materials, we only found two books which correct this error [49] [50]; all the others go along with the help. A further problem with the help features is that they use meaningless, empty, misleading, longish descriptions and information which is irrelevant when making decisions, and in some cases, incorrect and ambiguous expressions, sentences, and samples.

We also found in the helps that the examples focus on the details of the language. In the sample of **Figure 17** a composite function is presented just for the sake of showing that the language allows one to do so. There is no problem presented, just the code and the output.

5.4. Spreadsheet Competency Framework

The Spreadsheet Competency Framework was launched in 2016 at the EuSpRIG Conference. It also consists of the framework specification, where spreadsheet skills are listed [44]. The skills are organized in eleven groups and four levels of proficiency is defined.

Levels of proficiency: (1) basic user, (2) general user, (3) creator, and (4) developer.

Groups of skills: (1) Design and best practice, (2) Reviewing and team working, (3) Basic skills, (4) Efficiency of use, (5) Formulas, (6) Formatting, (7) Charting, (8) Protection and errors, (9) Data analysis, (10) Macros and automation, (11) Development and problem solving.

Since the analysis of the framework is beyond the scope of the present paper, only three of the features are mentioned here. The Basic skills consist of only three items, Access and save files, Read and enter data, and Set up printing. Two and a half of these are not spreadsheet skills, but general ICT skills, while reading data is a skill, which would be essential for problem solving. However, problem solving is only the skills of creators and developers. Even such framework

does not require any problem solving from end-users.

Creating formulas are also essential in problem solving. In this framework, the focus is on the functions and formulas, not on the complexity of the problems. A list of groups of functions are presented, without mentioning that the formulas and functions can be used at any level, but the problems should match the level of the students. In this framework, we are again faced with the problems mentioned and listed in connection with the teaching materials and teaching approaches to problem solving: the focus is on the tools, not on the problem solving and not on the effective problem solving approaches.

5.5. What Teachers Do Matters

It is obvious that the low mathability methods, wildly accepted, supported, and influenced by teachers, software companies, teaching and help materials, examination forms, etc., are clearly recognizable in the erroneous document handling, in the lack of knowledge regarding transfers between applications, and in computer problem solving in general. It is clear that low mathability computer solving approaches have a negative impact on students. It is not widely accepted that end-users are not the low quality participants of the digital world. They are just not professional programmers, on one hand, on the other hand, they seem to be misjudged, mistreated, and misguided.

Several of low mathability samples are displayed and discussed in the paper. However, a sample of one of the victim's work is presented in **Figure 15** (left).

The example is selected from a lesson plan of a pre-service teacher of Mathematics in the final year of her tertiary studies. The non-printable characters of the document clearly reveal the bricolage, which is obviously a TAEW based approach. However, the footnote added to the formula clearly reveals that she is not aware of the lack of her knowledge: "At the end of the lines the '=' sign cannot be entered because of the Word program." She even blames the software not allowing the entering of the "=" at the end of the lines, instead of creating the equations with an Equation Editor (**Figure 15**, right). (In the second line she created the "=" by applying double underline font style on two Space characters.)

She is a victim of low mathability teaching approaches in various senses: (1) her knowledge in computer problem solving is so low that she cannot use computers effectively, (2) as an in-service teacher, she would not be able to use computer tools for real problem solving and to motivate her students, (3) and

$$\frac{d}{3} + \frac{c}{4} - \frac{2d}{6} - \frac{3c}{12} + \frac{d}{12} - \frac{d}{6} - \frac{c}{6} =$$

$$= \frac{4d}{12} - \frac{4d}{12} + \frac{d}{12} - \frac{2d}{12} + \frac{3c}{12} - \frac{3c}{12} - \frac{2c}{12}$$

Figure 15. A sample from a lesson plan of an overconfident pre-service teacher of Mathematics (left).

finally, based on her note added to her formula, she is not aware of her lack of knowledge.

An example of misleading Microsoft messages is presented in **Figure 16**. Our testing of students of Informatics in the TAAAS project (Testing Algorithmic and Application Skills) [56] proved that even students whose major is Informatics are convinced (99%) that if we changed the extension of a file it cannot be used any more. End-users do not realize that by changing the filename, the content does not change. (With data files only the program assigned to them is changed, but the files can be opened in any previously initiated suitable programs. While the .exe files are just not recognized as executable.) However, the extensions can be renamed to the original, while the same warning message is displayed.

How different teachers can handle this warning message?

Microsoft and experienced teachers want to protect end-users and students. Both know that if there is no extension, there is no problem. Consequently, Microsoft decided a couple of versions ago that it is better to hide the extensions, and most of the teachers accept this setting. This is a low mathability approach.

However, there are teachers who display the extension intentionally and teach its role and importance. They also provide various examples of file conversions and cases when the extension is changed intentionally. (e.g. a .csv extension is changed to .txt for a more convenient opening in a European Excel or the .exe extension is deleted to attach the file in an email.) With this high mathability approach students would understand the different file types, the connection between them, their conversions, and that extension can be changed without doing any harm to the content of the file. In Chen's meaning system model these are the teachers with belief in the incremental nature of science and high teaching self-efficacy, the experts.

Another type of Microsoft low mathability example is presented in **Figure 17**, where the power of embedded IF() functions is demonstrated, the focus is on the tool. The task is to calculate the grades based on marks stored in A2:A4.

However, grading should not be solved with embedded IF() functions, because Excel has solutions which are more suitable for this type of problem (e.g. the INDEX(MATCH()) composite function) [50] [51] [52].

Another two low mathability aspects of the MS solution are that in the help three four-level composite functions are presented for the three inputs, which contain A, B, C, D, and F and 89, 79, 69, and 59 as constants.

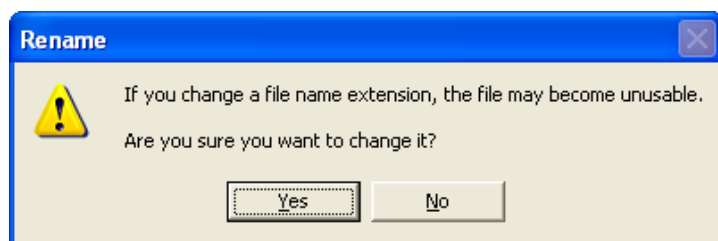


Figure 16. Misleading warning message from Microsoft.

Formula	Description	Result
=IF(A2>89,"A",IF(A2>79,"B", IF(A2>69,"C",IF(A2>59,"D","F"))))	Assigns a letter grade to the score in cell A2	F
=IF(A3>89,"A",IF(A3>79,"B", IF(A3>69,"C",IF(A3>59,"D","F"))))	Assigns a letter grade to the score in cell A3	A
=IF(A4>89,"A",IF(A4>79,"B", IF(A4>69,"C",IF(A4>59,"D","F"))))	Assigns a letter grade to the score in cell A4	C

Figure 17. A help example for only the sake of language details. This problem can be solved a lot faster, safer, and flexible solution with the INDEX(MATCH()) composite function.

For calculating the grades a more expert solution can be provided with the following settings:

- Instead of the four-level IF(IF(IF(IF()))), the INDEX(MATCH()) two-level composite functions is created.
- Instead of the constants, a side table is set up (e.g. D1:E5) which holds the grade limits and the alphabetical grades, arranged in vectors (D1:D5 and E1:E5, respectively).
- Instead of the three functions for the three inputs, an array formula can be used [50] [51] [52], [53] [54] [55] (with the array formula the copying and the absolute and relative references can be avoided which are the major sources of erroneous documents [1]).
- The solution is {=INDEX(E1:E5, MATCH(A2:A4, D1:D5))} formula.

One can argue that the HLOOKUP() and VLOOKUP() functions would be faster. However, we claim that there are so many restrictions on these two functions that from the high mathability problem solving and the meaning system model aspects, they do not worth mentioning. They are another low mathability solution.

While Microsoft's is a low mathability, ours is a high mathability solution. Non-expert teachers can argue that the {INDEX(MATCH())} array formula is difficult, and students, end-users would not understand it.

However, expert teachers can develop students' understanding of algorithms and building and applying schemata which support their flexible problem solving. The {INDEX(MATCH())} solution presents integrated knowledge, the application of previously learned algorithms, flexible and high-efficacy solution. These are the features which experts teachers have and are able to develop with their students.

6. Expert Teachers

6.1. Attributes of Expert Teachers in Computer Problem Solving

In the following, mainly citations of different sources are presented to prove that there are effective, high mathability teaching methods which can be adapted and applied in end-user computing. As it was mentioned in the previous chapters, we are in lack of approaches which modify the proportion of fast and slow thinking, banish low mathability interface navigation, and focus on high mathability problem solving and schema construction, to build up knowledge.

In the Meaning System Model [43] Chen *et al.* claimed that "Teaching by providing students opportunities to understand and appreciate science deeply,

[which] likely use [s] methods to reveal the dynamic nature of science” is the approach to effective teaching, and this has been proved by Hattie’ research, in his biggest ever research project on teaching strategies [13] [14].

In the present paper we focus on teaching approaches and teaching materials for end-user computing and problem solving. Within this framework we have selected from Hattie’s 16 attributes of expert teachers those which are relevant in this context [13] [14].

Expert Teachers [13] [14]	Computer Problem Solving
“Experts and experienced teachers do not differ in the amount of knowledge [they possess] ... Experts possess knowledge that is more integrated, in that they combine new subject matter content knowledge with prior knowledge; [they] can relate current lesson content to other subjects in the curriculum.”	The different subjects of Informatics and Computer Sciences are thought of and taught in exclusion. End-user computing is taught without real contents and problems, relying heavily on concepts which novices and end-users usually do not have.
“Expert teachers teach and are prepared for a greater store of algorithms that students might use when solving a particular problem.”	This characteristic of expert teachers plays a crucial role in guiding students when they develop intuitive expertise. Intuitive expertise is the knowledge which System 1 needs for making reliable fast decisions [25]. Expert teachers are able to teach how knowledge at mathability Level 4 can be transferred to Level 3.
“Both expert and experienced teachers perform better than novices ... because their cognitive skills become automatic with extensive practice” (Chase & Simon, 1973; Chi <i>et al.</i> , 1981 in [13]), [25]. “The difference, rather, is that experts develop automaticity so as to free working memory to deal with other more complex characteristics of the situation, whereas experienced non-experts do not optimise the opportunities gained from automaticity.”	Expert teachers are much effective in activating System 1 and System 2 in the right proportion and within the right time frame, which has great importance in effective problem solving, in lightening the load of working memory, and in our case, in reducing the number of erroneous and demanding documents.
“The expert teacher more often than the experienced teacher seeks further information, whereas experienced teachers focus more on directly available data...”	Typing data, presenting meaningless and/or low mathability formulas, l’art-pour-l’art formattings, etc., and the acceptance of low mathability materials are widely accepted.
“Experts are more adept at anticipating problems and then improvising. They tend to spend a greater proportion of their solution time trying to understand the problem to be solved as opposed to trying out different solutions.”	In this aspect, the difference between experts and the others in computer problem solving is that they prefer high mathability and TAEW based approaches, respectively.
“They [experts] are better able to filter relevant from irrelevant information, and are able to monitor, understand, and interpret events in more detail and with more insight than experienced teachers.”	This is the aspect mentioned in connection with the definition and description of functions, where irrelevant information is focused on.
“Expert teachers aim for more than achievement goals. They also aim to motivate their students to master rather than perform, they enhance students’ self-concept and self-efficacy about learning, they set appropriate challenging tasks, and they aim for both surface and deep outcomes”,	This statement is in complete accordance with the Meaning System Model [43].
“Expert teachers are more likely to set challenging rather than “do your best” goals, they set challenging and not merely time consuming activities, they invite students to engage rather than copy.”	This high mathability approach was hardly found in the analyzed materials; they primarily expected copying and surface navigation from students.

6.2. Utilizing Fast and Slow Thinking Effectively

Finally, we return to the computer problem solving approaches, to the two hypernym categories. Hattie found in his research that

“We can make a distinction between surface and deep learning. Surface learning is more about the content (knowing the ideas, and doing what is needed to gain a passing grade), and deep learning more about understanding (relating and extending ideas, and an intention to understand and impose meaning). The claim is that experts are more successful at both types of learning, whereas both experienced and expert teachers are similar in terms of surface learning.”

Experts are more aware of when there is a need to activate System 2 than experienced but non-expert teachers. To be able to find the right proportion of System 1 and System 2 requires being an expert in our major, which is pedagogical content knowledge: Hattie argues that

“...content knowledge is necessary for both experienced and expert teachers, and is thus not a key distinguishing feature... We are not underestimating the importance of content knowledge—it must be present—but it is more pedagogical content knowledge that is important: that is, the way knowledge is used in teaching situations.”

Being aware of the characteristics of the deep and surface approach problem solving methods and the attributes of expert teachers, we go one step further and claim that the problem is not with “overuse of ATM thinking”, since this is needed in real world problem solving to build strategic procedures—implemented in System 2 (ATM thinking)—, but using ATM thinking when AUM thinking would do better. As we have seen, expert teachers are those who can teach students how to build schemata and when they can be recalled, and how to find the right proportion of deep and surface learning.

One further reason to find the right proportion of thinking fast and thinking slow is explained by Polya [18] and is in complete accordance with Kahneman’s [25] and Hattie’s [13] [14] findings: “If you cannot solve a problem, then there is an easier problem you can solve ...” Expert teachers are those who have the ability, based on their attributes, to teach this.

6.3. Sunk-Cost Fallacy

Why is there a shortage of expert teachers in Informatics? Why can teachers not see that being experienced is not enough; they have to be expert? Kahneman’s sunk-cost fallacy is one explanation [25]:

“The decision to invest additional resources in a losing account, when better investments are available”.

Teachers have invested a lot in being experienced, most of them are self-taught—similar to spreadsheet developers and users [7], which was found in the

TAaAS project regarding teachers of Informatics [56]—, and it is not easy to accept that they have to leave behind everything and change their teaching approach to a completely new and more demanding one(s). If we can call attention to the problem and make teachers and education policy makers listen, we can avoid the mistake which companies make:

“All too often a company afflicted by sunk costs drives into the blizzard, throwing good money after bad rather than accepting the humiliation of closing the account of a costly failure.”

We practice in end-user computing, in general, and research focusing on spreadsheet errors proves that in end-user computing pseudo-experts outnumber the true experts [1] [2] [9] [10]. The environment is not sufficiently regular, since software companies boast new features, instead of regularities, which is accepted unconditionally by teachers, education policy makers, curricula developers, etc., however, it is not accepted that the regularities have to be learnt through practice. Schema construction is not accepted/used in end-user computer trainings, even though it is necessary for reliable fast thinking and consequently error-free problem solving.

7. Examples of High Mathability Materials

Some attempts to change end-users' computational thinking, their approach to problem solving have been made. However, it seems that without institutionalized support—both from education policy and education research—they will remain isolated and forgotten.

It was declared around three decades ago that functional programming is more effective for developing students' algorithm skills than imperative languages, due to the simplicity of the languages [17]. This programming approach focuses on problem solving instead of the language details. The idea has recently emerged again, entitled as Functional Model, and been accepted in some of the states of Germany, based on Hubwieser's research [57]. His finding, however, clearly shows the isolation of researchers working in the field. In 2004 he claimed his model as novelty [57], not realizing that it had been around since 1992 [17]. Similar research is being conducted in Hungary, where Sprego—Spreadsheet Lego—has been introduced [53]. The Sprego language is similar to Logo, but in a spreadsheet environment. It serves both as a functional language and office application. However, at present it is completely isolated for various reasons: (1) the Hungarian language, (2) teachers' reluctance to switch from low to high mathability approaches, (3) the fact that it takes several years to prove its effectiveness 100%. Both Hubwieser [57] and Csernoch [50] [51] [52] focus on real world problems with high mathability approaches. Sprego is similar to Hubwieser's approach in that it builds the functional models for the problems, but goes one step further and does coding in its simplified language to introduce programming concepts to novice users.

A problem solving approach is also present in spreadsheet management in the

“Succeeding in Business with Microsoft Excel” series [49]. However, the authors use a made-up company and made-up examples, mix spreadsheets with ICT, and do heavy computer cooking, not able to distinguish problems from language details.

Also in relation to spreadsheets we must mention, as a good example, the attempts in France. They teach spreadsheets as part of Mathematics, and as such focus on the fundamental formulas and functions, and this knowledge is tested in the “Brevet des colleges” [58]. This approach is present in many spreadsheet teaching materials, but even French education is not free of low mathability teaching materials. It must also be mentioned that some of the French maths course books offer math-oriented real world problems, but most of the on-line materials are meaningless, empty “examples”, without tables and without real problems to solve, focusing on the language details. In spite of this effort, several papers have been published lamenting the low spreadsheet knowledge of French students [59] [60]. Our own studies within the framework of the TAAS project [56] revealed that most French students of Informatics do not even understand the concept of spreadsheet formula. Research beyond spreadsheet errors focuses on spreadsheet design and programming in spreadsheet [40] [41] [50]-[55] but just as with other sources, they are hardly known [7] [62] [63].

Ben-Ari focused on text management [3] [6] and found that most of the text-based documents are bricolage. Similar results have been published since 1997, but have remained unnoticed [4] [5] [61]. In presentation and webpage management the situation is similar; it seems that only a very few listen.

The list of good practices is obviously not complete, just like the low mathability examples in the previous section, but this cannot be the goal of the present paper. In general, the overwhelming proportion of low mathability tools and teaching approaches in end-user computing is clearly present.

8. Conclusions

In this paper we focus on the training and problem solving approaches of the non-professional participants of the digital world, end-users for short. We report the results of our findings based on the collection and analyses of the available end-user teaching, guiding, self-learning, testing materials, along with the different teaching approaches in different languages and countries. We also summarize our base and mathability-extended typology of the computer problem solving approaches (published in 2014 and in 2015, respectively), which is a quantitative measuring system for evaluating educational contents.

Our analyses of end-user materials led us to the following conclusions:

- End-user educational sources primarily support low mathability slow thinking activities, where surface navigation and usage of software tools are in the focus [H1].
- There are hardly any traces of high mathability slow thinking activities, where real world problem solving has to be carried out [H1], [H4].
- Intuition based reliable fast thinking activities are scarcely supported and

carried out, due to the lack of true expertise [H2], [H3], [H4].

- End-user computing lacks expert teachers who have the appropriate attitude to maximize the impact on learning. The majority of the CS/Informatics teachers are novice and/or experienced, who do not possess the necessary pedagogical content knowledge. As such they themselves do not know and consequently they are not able to teach their students when and how to apply fast and slow thinking effectively. These teachers focus on tools, on surface navigation, and accept low mathability and frequently erroneous teaching and guiding materials [H2], [H3].
- Sunk-cost fallacy further deepens the problem. The experienced participants of end-user training and support invested so much that they do not want to or are not able to give up their achievements, but stick with them [H2].

We also found during our analytical process that researches in CS/Informatics education have already recognized some of the problems in end-user computing. However, they decided that end-user computing should be banished from CS/Informatics, and only pure programming has to be taught, that is the only way for developing the students' computational thinking [H3].

However, we claim that end-user computing in the digital world is not less important than traditional CS/Informatics studies. We found that the results of recent researches in educational studies would provide guidelines for fundamental changes in end-user education, training, and support.

- With expert teachers end-user computing can be taught as effectively and efficiently as programming and other majors in CS/Informatics.
- The path to professionalism in CS/Informatics and to professional problem solving in other sciences starts at end-user computing.
- End-users outnumber professionals, so they cannot be ignored and/or banished. Well trained end-users would be those participants of the digital world who have great impact on developments.

We have a large number of experienced teachers (and also a large number of unexperienced and novice teachers), but that is not enough. The enormous number of erroneous documents, the wasting of human and computer resources, the low mathability teaching materials focusing on technical and language details, ill-titled and misleading coursebooks and many other teaching materials, as well as the fact that teachers unconditionally accept these circumstances, all outline the necessity of change. The effect of experienced teachers and software developers on end-user computing is obvious: what they do matters. However, we are faced with a very unfortunate situation in that they have a negative effect on end-user problem solving.

We are in great need of expert teachers to develop students' computational thinking, the skills required for the digital era, effective end-user computing and computer problem solving. We must emphasize that "it is excellence in teachers that make[s] the greatest differences, not just teachers" [14]. We do not claim that our teachers of CS, IT, ICT, Informatics, or whatever we choose to call it, are poor teachers; we only argue that for various reasons they are non-excellent,

and it is high time for change.

We need teachers who are ready to give up their “experienced” status and are open to novel high mathability approaches. Avoiding the sunk-cost fallacy [25] might be a humiliation for some, but a great opportunity for the next generation. Expert teachers are needed who can provide students with opportunities to understand and appreciate science deeply, and who will likely use the methods required to reveal the dynamic nature of science [43].

References

- [1] Panko, R. (2013) The Cognitive Science of Spreadsheet Errors: Why Thinking Is Bad. *Proceedings of the 46th Hawaii International Conference on System Sciences*, 7-10 January 2013, Maui, 4013-4022. <https://doi.org/10.1109/hicss.2013.513>
- [2] EuSpRIG Horror Stories. <http://eusprig.org/horror-stories.htm>
- [3] Ben-Ari, M. and Yeshno, T. (2006) Conceptual Models of Software Artifacts. *Interacting with Computers*, **18**, 1336-1350. <https://doi.org/10.1016/j.intcom.2006.03.005>
- [4] Csernoch, M. (2009) Teaching Word Processing—The Theory Behind. *Teaching Mathematics and Computer Science*, **7**, 119-137. <https://doi.org/10.5485/TMCS.2009.0212>
- [5] Csernoch, M. (2010) Teaching Word Processing—The Practice. *Teaching Mathematics and Computer Science*, **8**, 247-262. <https://doi.org/10.5485/TMCS.2010.0252>
- [6] Ben-Ari, M. (1999) Bricolage Forever! PPIG 1999. 11th Annual Workshop, 5-7 January 1999, Computer-Based Learning Unit, University of Leeds, Leeds, UK. <http://www.ppig.org/papers/11th-benari.pdf>
- [7] Kadijevich, D. (2013) Learning about Spreadsheet. In: Kadijevich, D., Angeli, C. and Schulte, C., Eds., *Improving Computer Science Education*, Routledge, New York, 19-33.
- [8] Van Deursen, A. and Van Dijk, J. (2012) Ctrl Alt Delete. Lost Productivity Due to IT Problems and Inadequate Computer Skills in the Workplace. Universiteit Twente, Enschede.
- [9] Panko, R. (2015) What We Don't Know about Spreadsheet Errors Today: The Facts, Why We Don't Believe Them, and What We Need to Do. arXiv:1602.02601 [cs]. <http://arxiv.org/abs/1602.02601>
- [10] Panko, R. and Port, D. (2013) End User Computing: The Dark Matter (and Dark Energy) of Corporate It. *Journal of Organizational and End User Computing*, **25**, 1-19. <https://doi.org/10.4018/joeuc.2013070101>
- [11] Csernoch, M. and Biró, P. (2016) Teaching Methods Are Erroneous: Approaches Which Lead to Erroneous End-User Computing. *EuSpRIG2016*. London. <http://www.eusprig.org/mcsernoch-2016.pdf>
- [12] Wing, J.M. (2006) Computational Thinking. *Communications of the ACM*, **49**, 33-35. <https://doi.org/10.1145/1118178.1118215>
- [13] Hattie, J. (2003) Teachers Make a Difference: What Is the Research Evidence? Distinguishing Expert Teachers from Novice and Experienced Teachers. *Australian Council for Educational Research (ACER) Annual Conference on: Building Teacher Quality*. https://www.det.nsw.edu.au/proflearn/docs/pdf/qt_hattie.pdf
- [14] Hattie, J. (2012) Visible Learning for Teachers: Maximizing Impact on Learning.

Routledge, New York and London.

- [15] Csernoch, M. and Biró, P. (2015) Computer Problem Solving. In Hungarian: *Számítógépes problémamegoldás, TMT, Tudományos és Műszaki Tájékoztató*, **62**, 86-94.
- [16] Case, J.M. and Gunstone, R.F. (2002) Metacognitive Development as a Shift in Approach to Learning: An In-Depth Study. *Studies in Higher Education*, **27**, 459-470. <https://doi.org/10.1080/0307507022000011561>
- [17] Booth, S. (1992) Learning to Program: A Phenomenographic Perspective. Göteborg Studies in Educational Sciences 89. Acta Universitatis Gothoburgensis, Gothenburg.
- [18] Polya, G. (1957) How to Solve It. *A New Aspect of Mathematical Method*. 2nd Edition, Princeton University Press, Princeton.
- [19] Baranyi, P., Csapo, A. and Varlaki, P. (2014) An Overview of Research Trends in CogInfoCom. *IEEE 18th International Conference on Intelligent Engineering Systems INES2014*, 181-186. <https://doi.org/10.1109/INES.2014.6909365>
- [20] Baranyi, P. and Csapó, A. (2012) Definition and Synergies of Cognitive Infocommunications. *Acta Polytechnica Hungarica*, **9**, 67-83.
- [21] Baranyi, P. and Gilanyi, A. (2013) Mathability: Emulating and Enhancing Human Mathematical Capabilities. 2013 *IEEE 4th International Conference on Cognitive Infocommunications (CogInfoCom)*, 555-558.
- [22] Borus, Gy. and Gilanyi, A. (2013) Solving Systems of Linear Functional Equations with Computer. 2013 *IEEE 4th International Conference on Cognitive Infocommunications (CogInfoCom)*, 559-562. <https://doi.org/10.1109/coginfocom.2013.6719310>
- [23] Biró, P. and Csernoch, M. (2015) The Mathability of Computer Problem Solving Approaches. 2015 *6th IEEE International Conference on Cognitive Infocommunications (CogInfoCom)*, 111-114.
- [24] Biró, P. and Csernoch, M. (2015) The Mathability of Spreadsheet Tools. 2015 *6th IEEE International Conference on Cognitive Infocommunications (CogInfoCom)*, 105-110. <https://doi.org/10.1109/coginfocom.2015.7390573>
- [25] Kahneman, D. (2011) Thinking, Fast and Slow. Farrar, Straus; Giroux, New York.
- [26] Skemp, R. (1971) The Psychology of Learning Mathematics. Lawrence Erlbaum Associates, New Jersey.
- [27] Merriënboer, J.J.G. van and Sweller, J. (2005) Cognitive Load Theory and Complex Learning: Recent Developments and Future Directions. *Educational Psychology Review*, **17**, 147-177. <https://doi.org/10.1007/s10648-005-3951-0>
- [28] Kruger, J. and 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**, 1121-34. <https://doi.org/10.1037/0022-3514.77.6.1121>
- [29] ECDL Sample Tests (Spreadsheets, Presentation). http://ecdcl.org/media/spreadsheets_sampleparttest_ms2010.zip
http://ecdcl.org/media/presentation_sampleparttest_ms2010.zip
- [30] Hromkovič, J. (2009) Algorithmic Adventures. Springer Berlin Heidelberg. Berlin, Heidelberg. <http://link.springer.com/10.1007/978-3-540-85986-4>
<https://doi.org/10.1007/978-3-540-85986-4>
- [31] Mireault, P. (2016) Characteristics of Spreadsheets Developed with the SSMI Methodology. *EuSprIG2016*. <http://www.eusprig.org/pmireault-2016.pdf>
- [32] Reynolds, G. (2011) Presentation Zen: Simple ideas on Presentation Design and De-

- livery. New Riders, Berkeley.
- [33] Message, R. (2013) Programming for Humans: A New Paradigm for Domain-Specific Languages. Technical Report. University of Cambridge. Computer Laboratory. <https://www.cl.cam.ac.uk/techreports/UCAM-CL-TR-843.pdf>
- [34] Stöckli, T. (2011) Lebenslernen: Ein zukunftsfähiges Paradigma des Lernens als Antwort auf die Bedürfnisse heutiger Jugendlicher. Von der Fakultät I—Geisteswissenschaften (Institut für Kunstwissenschaft und Historische Urbanistik) der Technischen Universität Berlin zur Erlangung des akademischen Grades Dr. phil. genehmigte Dissertation. https://depositonce.tu-berlin.de/bitstream/11303/3165/1/Dokument_43.pdf
- [35] Gove, M. (2012) Michael Gove speech at the BETT Show 2012. Published 13 January 2012. Digital Literacy Campaign. <http://www.theguardian.com/education/2012/jan/11/digital-literacy-michael-gove-speech>
- [36] Bell, T. and Newton, H. (2013) Unplugging Computer Science. In: Kadjevich, D.M., Angeli, C. and Schulte, C., Eds., *Improving Computer Science Education*, Routledge, New York and London, 66-81.
- [37] Soloway E. (1993) Should We Teach Students to Program? *Communications of the ACM*, **36**, 21-24. <https://doi.org/10.1145/163430.164061>
- [38] Mayer, R.E. (1981) The Psychology of How Novices Learn Computer Programming. *ACM Computing Surveys*, **13**, 121-141. <https://doi.org/10.1145/356835.356841>
- [39] Kirschner, P.A., Sweller, J. and Clark, R.E. (2006) Why Minimal Guidance During Instruction Does Not Work: An Analysis of the Failure of Constructivist Discovery, Problem-Based, Experiential, and Inquiry-Based Teaching. *Educational Psychologist*, **41**, 75-86. https://doi.org/10.1207/s15326985ep4102_1
- [40] Wakeling, D. (2007) Spreadsheet Functional Programming. *Journal of Functional Programming*, **17**, 131-143. <https://doi.org/10.1017/S0956796806006186>
- [41] Sestoft, P. (2011) Spreadsheet Technology. Version 0.12 of 2012-01-31. IT University Technical Report ITU-TR-2011-142. IT University of Copenhagen, December 2011.
- [42] Ben-Ari, M. (2011) Non-Myths about Programming. *Communications of the ACM*, **54**, 35. <https://doi.org/10.1145/1965724.1965738>
- [43] Chen, J.A., Morris, D.B. and Mansour, N. (2015) Science Teachers' Beliefs. Perceptions of Efficacy and the Nature of Scientific Knowledge and Knowing. In: Fives, H. and Gill, M.G., Eds., *International Handbook of Research on Teachers' Beliefs*. Routledge, New York and London, 370-386.
- [44] Spreadsheet Competency Framework (2016) A Structure for Classifying Spreadsheet Ability in Finance Professionals. ICAEW. <http://www.icaew.com/-/media/corporate/files/technical/information-technology/it-faculty/spreadsheet-competency-framework.ashx>
- [45] Freiermuth, K., Hromkovic, J. and Steffen, B. (2008) Creating and Testing Textbooks for Secondary Schools. *Proceedings of the 3rd International Conference on Informatics in Secondary Schools—Evolution and Perspectives. Informatics Education—Supporting Computational Thinking*, Springer-Verlag, Berlin, Heidelberg, 216-228.
- [46] Florida Gulf Coast University 2013. Excel 2013 Tutorial. <http://www.fgcu.edu/Support/excel2013.html>
- [47] Dancsó, T. and Korom, P. (2013) Informatics for Grades 9-10. In Hungarian: In-

- formatika 9-10. A gimnáziumok számára. Nemzedékek Tudása Tankönyvkiadó (Oktatókutatató és Fejlesztő Intézet), Budapest.
- [48] Cours Excel: Formules de calculs et fonctions.
http://www.excel-pratique.com/fr/cours/excel_formules_calculs_fonctions.php
- [49] Gross, D., Akaiwa, F. and Nordquist, K. (2014) Succeeding in Business with Microsoft Excel 2013: A Problem-Solving Approach. Cengage Learning, Delhi.
- [50] Csernoch, M. (2014) Programming with Spreadsheet Functions: Sprego. In Hungarian, Programozás táblázatkezelő függvényekkel—Sprego. Műszaki Könyvkiadó, Budapest.
- [51] Csernoch, M. and Biró, P. (2015) Sprego Programming. *Spreadsheets in Education (e/SiE)*, 8, Article 4. <http://epublications.bond.edu.au/ejsie/vol8/iss1/4>
- [52] Csernoch, M. and Biró, P. (2015) Sprego Programming. LAP Lambert Academic Publishing, Saarbrücken.
- [53] Guidelines and Examples of Array Formulas (2010).
<https://support.office.com/en-us/article/Guidelines-and-examples-of-array-formulas-7D94A64E-3FF3-4686-9372-ECFD5CAA57C7>
- [54] Walkenbach, J. (2003) Excel2003 Formulas. John Wiley & Sons, Foster City.
- [55] Walkenbach, J. (2010) Excel 2010 Bible. Wiley Pub., Indianapolis, Indiana, United States. <https://ebooks-it.org/0470474874-ebook.htm>
- [56] Csernoch, M., Biró, P., Máth, J. and Abari, K. (2015) Testing Algorithmic Skills in Traditional and Non-Traditional Programming Environments. *Informatics in Education*, 14, 175-197. <https://doi.org/10.15388/infedu.2015.11>
- [57] Hubwieser, P. (2004) Functional Modelling in Secondary Schools Using Spreadsheets. *Education and Information Technologies*, 9, 175-183.
<https://doi.org/10.1023/B:EAIT.0000027929.91773.ab>
- [58] Mathématiques - Brevet (série générale).
<http://www.bankexam.fr/etablissement/8-Brevet/632-Mathematiques>
- [59] Tort, F. (2010) Teaching Spreadsheets: Curriculum Design Principles. arXiv:1009.2787 [cs]. <http://arxiv.org/abs/1009.2787>
- [60] Tort, F., Blondel, F.M. and Bruillard, É. (2008) Spreadsheet Knowledge and Skills of French Secondary School Students. In Mittermeir, R.T. and Syslo, M.M., Eds., *Informatics Education—Supporting Computational Thinking*, Springer, Berlin and Heidelberg, 305-316.
http://link.springer.com/chapter/10.1007/978-3-540-69924-8_28
https://doi.org/10.1007/978-3-540-69924-8_28
- [61] Csernoch, M. (1997) Methodological Questions of Teaching Word Processing. *3rd International Conference on Applied Informatics*, Eger-Noszvaj, 375-382.
- [62] Kadijevich, D. (2009) Simple Spreadsheet Modeling by First-Year Business Undergraduate Students: Difficulties in the Transition from Real World Problem Statement to Mathematical Model. In: Blomhøj, M. and Carreira, S., Eds., *Mathematical Applications and Modeling in the Teaching and Learning of Mathematics. Proceedings the 11th International Congress on Mathematical Education*, Monterrey, Mexico, 241-248.
- [63] Angeli, C. (2013) Teaching Spreadsheets: A TPCCK Perspective. In: Kadijevich, D.M., Angeli, C. and Schulte, C., Eds., *Improving Computer Science Education*. Routledge, New York and London, 132-146.

Submit or recommend next manuscript to SCIRP and we will provide best service for you:

Accepting pre-submission inquiries through Email, Facebook, LinkedIn, Twitter, etc.

A wide selection of journals (inclusive of 9 subjects, more than 200 journals)

Providing 24-hour high-quality service

User-friendly online submission system

Fair and swift peer-review system

Efficient typesetting and proofreading procedure

Display of the result of downloads and visits, as well as the number of cited articles

Maximum dissemination of your research work

Submit your manuscript at: <http://papersubmission.scirp.org/>

Or contact jsea@scirp.org