applied sciences

MDPI

Article

Case Study: Students' Code-Tracing Skills and Calibration of Questions for Computer Adaptive Tests

Robert Pinter [1,*], Sanja Maravić Čisar [1], Attila Kovari [2], Lenke Major [3], Petar Čisar [4] and Jozsef Katona [5]

1 Department of Informatics, Subotica Tech—College of Applied Science, 24000 Subotica, Serbia; sanjam@vts.su.ac.rs
2 Department of Natural Sciences and Environmental Protection, Institute of Engineering, University of Dunaujvaros, 2400 Dunaújváros, Hungary; kovari@uniduna.hu
3 University of Novi Sad Hungarian Language Teacher Training Faculty, 24000 Subotica, Serbia; major.lenke@magister.uns.ac.rs
4 Department of Informatics, University of Criminal Investigation and Police Studies, 11080 Belgrade-Zemun, Serbia; petar.cisar@kpu.edu.rs
5 CogInfoCom based LearnAbility Reseacrh Team, University of Dunaujvaros, 2400 Dunaújváros, Hungary; katonaj@uniduna.hu
* Correspondence: pinter.robert@vts.su.ac.rs

Received: 31 August 2020; Accepted: 8 October 2020; Published: 11 October 2020

Featured Application: Authors are encouraged to provide a concise description of the specific application or a potential application of the work. This section is not mandatory.

Abstract: Computer adaptive testing (CAT) enables an individualization of tests and better accuracy of knowledge level determination. In CAT, all test participants receive a uniquely tailored set of questions. The number and the difficulty of the next question depend on whether the respondent's previous answer was correct or incorrect. In order for CAT to work properly, it needs questions with suitably defined levels of difficulty. In this work, the authors compare the results of questions' difficulty determination given by experts (teachers) and students. Bachelor students of informatics in their first, second, and third year of studies at Subotica Tech—College of Applied Sciences had to answer 44 programming questions in a test and estimate the difficulty for each of those questions. Analyzing the correct answers shows that the basic programming knowledge, taught in the first year of study, evolves very slowly among senior students. The comparison of estimations on questions difficulty highlights that the senior students have a better understanding of basic programming tasks; thus, their estimation of difficulty approximates to that given by the experts.

Keywords: computer adaptive testing; code tracing; basic programming skills

1. Introduction

Modern technologies offer numerous possibilities for improving knowledge assessment and the process of education. In higher education, testing is one of the most commonly used methods of measuring student knowledge. The main goal of testing is to determine the level of students' knowledge in one or more areas of the course material on which the test is based. In a class held in an online environment, self-testing via computer-based tests (CBT) provides feedback that shows the students how well they are progressing in the acquisition of knowledge or skills.

The most common form of computer tests are linear fixed-length tests. All students are given the same test; in fact, it "imitates" traditional pen-and-paper tests, representing their digital version. This type of test does not take into account the abilities of each individual student. While doing such a one-size-fits-all test, respondents may feel discouraged if the questions are too difficult, or, on the other hand, they may lose interest if the tasks are too easy for their level of knowledge. The solution to this problem may be the application of computer adaptive tests, which have the ability to change the level of question difficulty on the basis of the respondents' abilities similar to in an oral exam.

A computer adaptive test offers the ability to create a test based on the respondent's actual abilities. Questions are selected from a database, and the individual capabilities of the candidates are taken into account during the test. By applying the Item Response Theory (IRT) in CAT, different versions of the test are possible, e.g., if the respondent answers the question correctly, the following question is selected from a group of questions that are one level higher than the previous question. On the other hand, if the respondent's answer is incorrect, the next question is selected from a group of questions that are one level easier than the previous questions the respondent failed to answer correctly. Many different possibilities (algorithms) for selecting the next question can be applied. With this adaptive selection, the respondents with less knowledge will be given easier questions, while those better prepared for the test will receive a set of more difficult questions. Therefore, the end result of the test for each respondent also depends on the level of question difficulty that they answered correctly. Thus, individual students may have the same percentage of correct answers, but those who answered the more difficult questions will be given a higher grade [1].

There are several advantages of computer adaptive tests worth considering, when compared to traditional linear tests, such as

- individualization of tests,
- promoting a positive attitude toward testing,
- determining the level of knowledge with greater accuracy.

To develop a system with adaptive tests requires a database with many questions. Every question needs calibration, i.e., objective values indicating the question's difficulty. With the difficulty values determined properly, adaptive algorithms can then select the "right" next question in the test. One way of calibration is when the difficulty of the question is determined by the expert(s). Another way of calibration relies on a given population, e.g., a set of students. The large number of answers from participants for whom the adaptive tests will be created can provide information about the question's difficulty [2].

The primary goal of this study is to compare results of how experts (teachers) determined the difficulty of the questions with the values that were determined by the students. The authors aim to use the results and the experience acquired in this research in the further development of CS1 courses (introductory programming courses) and CAT systems. The secondary goal is to analyze what happens with the basic programming knowledge students are taught in their first year of study throughout the course of their education. In the case of students of informatics, does such knowledge develop, or will it be mainly forgotten in later years of study?

Writing a computer program is a difficult cognitive skill to master and it is also difficult to measure it [3,4]. Despite the best efforts of teachers, many students are still challenged by programming. In order to write a simple program, they need to possess a basic knowledge of variables, input/output of data, control structures, and other areas. Even given all necessary theoretical knowledge, they face a problem when having to apply their knowledge as a whole and actually write a programming code [5].

The authors used a code-tracing type of task and questions to assess students' programming skills. It has been confirmed that students cannot learn to write code without having previously learned to read/trace code as a precursory skill [6–8]. It must also be pointed out that being aware of how code functions is not the same as being able to use it in problem solutions [9].

The characteristics of code tracing can be summarized as follows:

- Tracing refers to following the flow and data progression of a given program.
- In many instances, tracing represents a single user's journey through a program or program snippet.
- Its purpose is not reactive, but instead, it focuses on optimization.

- By tracing, developers can identify bottlenecks and focus on improving performance.
- When a problem does occur, tracing allows the user to see how it came to be: which function, duration of a function, which parameters passed, and how deep into the function the user could delve.

2. Related Works

Problems that arise in novices when learning programming is a field that has been the subject of numerous studies, as have the ways of adopting those new concepts [10]. Xie et al. [11] proposed a theory that identified four distinct skills that novices learned incrementally. These skills were tracing, writing syntax, comprehending templates, and writing code with templates. They assumed that the explicit instruction of these skills decreased cognitive demand. The authors conducted an exploratory mixed-methods study and compared students' exercise completion rates, error rates, ability to explain code, and engagement when learning to program. They compared the learning material that reflected this theory to more traditional material that did not distinguish between the skills. The findings of their study were as follows: teaching skills incrementally resulted in an improved completion rate on practice exercises and decreased error rate and improved understanding on the post-test.

The report of a 2001 ITiCSE (Innovation and Technology in Computer Science Education) working group [12] assessed the programming ability of 216 post-CS1 students from eight tertiary institutions in various countries. The "McCracken group" used a common set of programming problems. The majority of students performed much worse than their teachers had expected. The average score was 22.89 out of 110 points. While such a report by an author at a single institution might be dismissed as a consequence of poor teaching at that particular institution, dismissing a multinational study is not done so lightly. Given the scale and the multinational nature of the collaboration, these results were widely viewed as significant and compelling. The McCracken study did not isolate the causes of the problem. A popular explanation for the students' poor performance was that they lacked the ability to problem-solve. In fact, students lack the ability to take a problem description, decompose it into sub-problems, implement the necessary steps, and then reassemble the pieces to create a complete solution. Based on the McCracken group research, an ITiCSE 2004 working group (the "Leeds Group") tested students from seven countries in two ways. First, students were tested on their ability to predict the outcome of executing a short piece of code. Next, the students were given the desired function of a short piece of near-complete code and tested on their ability to select the correct completion of the code out of a small set of possibilities. An alternative explanation is that many students have a weak grasp of the basic programming principles and were missing the ability to systematically carry out routine programming tasks, such as tracing through code [13].

This working group established that many students lacked knowledge and skills that are a precursor to problem-solving. These missing elements were more associated with the students' ability to read code than to write it. Many showed weakness in systematically analyzing a short piece of code. The working group did not argue that all students who manifested weakness in problem-solving were doing so due to reading-related factors. They accepted that a student who scored high on the type of tests used in this study, but was unable to write a novel code of similar complexity, was most likely suffering from a weakness in problem-solving. The working group merely stated that any research project aiming to study problem-solving skills in novice programmers had to include a mechanism to screen for subjects weak in the precursor, code reading-related skills.

Kopec et al. [14] analyzed programmers' examination errors but focused on intermediate programmers (i.e., with some programming experience and understanding of basic programming concepts). They concluded that educators had to pay careful attention to their problem description and presentation, since novices were easily confused by nested loops and recursion, which differentiated between intermediate programmers' and novice programmers' errors.

Code-tracing problems (e.g., debugging a program, identifying the output of a program) are attractive in that they can be solved in shorter stints of time, using formats such as multiple choice that are easier to grade [13]. In this context, Computer Science researchers were keen to see whether there was a correlation between students' performance on code-tracing problems and their ability to write code. Lopez et al. [6] studied the responses of novices in an examination and found strong support for a relation between their skills for code tracing and code writing, and between explaining code and writing code. They showed that there were strong correlations between code tracing and code writing.

The hypothesis of Kumar's [7] study was that tracing code would lead to an improvement in code-writing skills. The results of this study indicated that code-tracing activities helped students learn to write both syntactic and semantic components of code. However, the extend of the effect was found to be small to medium. Therefore, code-tracing exercises can be used as a supplement rather than a substitute for code-writing exercises.

Moreover, some other research areas present alternative technologies such as virtual reality (VR) [15,16], alternative approaches for teaching programming [17], brain–computer interface (BCI) systems [18,19], and serious games [20], which could support learning efficiency and help complement learning difficulties as well.

3. Research

The authors of this paper conducted a study among all three student cohorts of bachelor Informatics students at Subotica Tech—College of Applied Sciences. The research included 182 student participants who were taking an introductory course called 'Algorithms and data structures'. The course is composed of lectures, practices, and laboratory practices. The course runs in the spring semester over 15 weeks in the first year of study. It consists of one 90 min lecture per week as well as a 45 min practice and another 90 min lab practice.

The aim of the course is to introduce the students to the basic concepts of algorithms and data structures in C/C++ programming language. The course covers the basic principles of programming, variables, control statements (*if, while, do-while, break* and *continue* statements), arrays, functions, pointers, and some basic algorithms (sort and search).

Based on historical data, the grade average in this given course is 7.02 (5 is the minimum grade and means fail, and the maximum grade is 10), while the pass rate is about 50%.

During their studies, informatics students learn new developing methods and developing environments as well as new programming languages. The novel techniques help them solve more complex information and communications technology (ICT) problems. Although the problems are more complex, and the developing environment and the programming language are new to them, when solving a particular problem, students frequently rely on those basic algorithms and data structures that they learned in their first year of study, e.g., iterations, arrays, or conditional statements. The programming languages that students use in their second and third years are so-called C-like programming languages, so the syntax and logic of the language are highly similar to those covered in the course 'Algorithms and data structures' during the first year of study.

As stated earlier, the aim of this research is to analyze what happens to those basic programming skills acquired during the first year. The following questions arise:

1. Do computer science students forget the basic algorithms and elements of language, or because of using them actively, do they understand them even better?
2. What is the impact of the constantly developing programming skills when estimating the difficulty of basic tasks?

The authors formulated three hypotheses:

Hypotheses 1 (H1): *Students in later years of study have a better understanding of basic algorithms and data structures.*

Hypotheses 2 (H2): *Subjective difficulty estimation of one task at later years of study will be lower.*

Hypotheses 3 (H3): *The difference between the students' and teachers' subjective estimation decreases with the years of study*.

4. Data Collection and Analysis

The primary data collected were the students' answers in a test. A total of 182 students contributed data to this part of the study, but only 117 of those gave answers to all 44 questions.

The students in the second and third year of a study were tested in January 2020. The test was conducted as a paper-and-pencil test with limited time. During the test, students were not allowed to use computers, smart phones, or any other type of help in answering the questions. The test consisted of multiple-choice questions, so the students had to mark one or more of the given answers or write the output of the code snippet. The students were given a time limit of 90 min for completing the test.

The participants in Year 1 were given the same test in May 2020, but they had to complete it in an online environment due to the COVID-19 pandemic. Students took the test in the Moodle system. In all aspects, including questions themselves, their order, and the time limit, the online test was identical to the paper-and-pencil test.

Altogether, 182 records were collected. With the relatively short time limit and the large number of questions, even for students with an average programming language knowledge and tracing skills, completing this test correctly was not an easy task. The underlying reason for compiling a challenging test was to highlight the differences in individual skills. As a result of such a test design, there were students who did not answer all the questions either in the tracing part or the difficulty estimation of questions part.

4.1. Test Design

The test was composed of 44 tasks or questions. Out of these, 24 were multiple-choice questions with four possible answers, while 22 questions required the students to write down the output of the program or program snippet. The test questions were categorized based on two criteria: the subject matter of the question and the subjective estimation of the task's difficulty. Students filling in the test essentially had to do two things for each question: first, to solve the question as required by the task, then, to give their subjective estimate regarding the difficulty of the given question. In short, the authors asked them to do the task and then to indicate how easy or difficult (or medium) they found the task.

4.2. The Categories of the Questions.

Apart from skills tracing, the authors were also keen to explore students' programming knowledge; therefore, the test contained six types of tasks (i.e., six categories). The task categories were the following:

1. **Statements**. Through these questions, the authors aimed to explore the students' knowledge of various basic concepts of the C programming language. For example, students had to calculate an arithmetic problem with different operators, determine the correct names of variables, find syntax errors, etc. The "statement" category contained 12 questions.
2. **Conditional statements**. This category consisted of questions that used the *if*, *if-else*, *if-else-if*: (ternary operator), and switch structures. The conditions that the students had to examine/conclude ranged from simple to complex. Altogether, there were six conditional statements in this category.
3. **Iterations.** With the help of these iteration questions, the authors studied how the students would solve questions with *for*, *while,* and *do-while* cycles. The variation in task difficulty was achieved by adjusting the conditions of the cycle: they ranged from simple to complex. There were ten iteration-type questions in this category.

4. **Structures**. This category of questions focused on the use of arrays and two-dimensional arrays with integers and characters, although there were no tasks of data sorting. This category contained six questions.
5. **Functions**. Questions in this category were related to the knowledge of function arguments (call by value, call by reference) and the use of local, static, and global variables. There was a total of six function-related questions in the category.
6. **Recursion**. Questions in this category were consciously created as "difficult", so that they could reveal whether senior students still knew the principles of recursion functioning or would use other methods. There were four recursion questions in the category, three of which were basic, while the fourth question had double recursion.

The following list contains a few examples of categories, with the boxes for their level of difficulty that had to be ticked:

- Conditional statements

```
What is the output of the following code? Hard □ Medium □ Easy □
void main(){
int i = 0;
if((13 + 1)/2) {
if(13 > 5)
i+=5;
else
i++;
} else
i+=2;
cout<<i;
}
```

- Iteration statements

```
What is the output of the following code? Hard □ Medium □ Easy □

main (){
int i = 0;
int s = 9;
for (; i < s; i++)
s -= i;
     cout<<s;
}
```

- Data structures

```
What is the output of the following code? Hard □ Medium □ Easy □

void main(){
int j, i;
int ar[] = {1, 2, 3, 4, 5, 6, 11};
for(j = 5, i = 0 ; i < j ; j--,i++)
ar[i] = ar[j];
     i = ar[i];
     cout<<i;
}
```

- Functions

```
What is the output of the following code?  Hard □ Medium □ Easy □

void swap(int&, int&);
int main(void) {
int a = 10, b=20;
         swap (a, b);
         cout<<a<<"\t"<<b<<"\n";
         return 0;
}
void swap(int& x, int& y) {
     x+=2;
     y+=3;
}
    a.  14, 24
    b.  11, 21
    c.  10, 20
    d.  Nothing, there is an error in the code
    e.  12, 23
```

4.3. Estimation of the Question Difficulty

Students' subjective estimation of question difficulty is very useful feedback. Based on the students' answers, teachers gain information about which part of the study material the students understand the least. Furthermore, the teachers can rethink and revise their own estimation of the questions' difficulty. For example, a given question thought to be 'easy' by the teacher may be perceived as an unsolvable problem by the students themselves. This feedback can be used by teachers when preparing questions for the exam in order to only include questions with approximately the same difficulty. As mentioned above, a subjective estimation of question difficulty can also serve in computerized adaptive testing that adapts to the examinee's level of ability. This is advantageous from the examinee's perspective, as the difficulty of the exam seems to tailor itself to their level of ability. For instance, if an examinee performs well on an item of intermediate difficulty, they will then be presented with more difficult questions. Or, if they performed poorly, they will be presented with a simpler question.

The evaluation of task difficulty was conducted on two levels, by the participant teachers and students. The teachers involved in this study gave their subjective estimation both of the difficulty (how challenging a given task was) and complexity (how complicated the question was, how many steps it involved) of each question, resulting in two separate marks for every question. The underlying reason was to increase the objectivity and accuracy of difficulty estimation. This step was followed by a detailed professional discussion of the given marks, which were then transformed into the difficulty values of "easy", "medium" and "hard". While the teachers performed this double estimation, primarily to make the overall estimation more objective, such double estimation was not required from the students, because the authors believed it would prove too distracting for students, given that they had to pay close attention to actually solving the question and then evaluating its difficulty. In the test itself, the students were able to give a subjective estimation of the question difficulty by selecting one of the three offered options: "easy", "medium" and "hard", as seen above, next to the sample questions.

4.4. Participants

As stated before, the total number was 182 bachelor students of informatics. Table 1. shows the structure of the participating students by year of study.

Table 1. The number of students per year of study.

	n	%
First year	78	42.9
Second year	58	31.9
Third year	46	25.3
Total	182	100.0

Two professors from Subotica Tech, with more than two decades of experience in teaching computer science courses, have designed the test and conducted the data acquisition.

4.5. Comparison of the Performance in Each Year of Study

Comparing the test results for each year of study, the students from the first year gave the largest number of correct answers, averaging 25.73 out of 44. The students in Year 2 had an average of 21.21 points, while the weakest result came from the third-year students, who only achieved a point average of 17.39 (see Table 2.).

Table 2. Average of points.

Year of Study	Average Point	Standard Deviation (SD)
1st	25.73	6.1
2nd	21.21	7.2
3rd	17.39	6.0

The authors also examined the results achieved by the students on a question-by-question basis. The percentage of correct answers for each question is illustrated in Figure 1. The questioned revealed for which items the students' correct answer reached a level of 51% (the lower limit of the points to "just pass"). Those questions where the value did not surpass the 51% limit are marked (e.g., Q2, Q6, Q11, etc.)

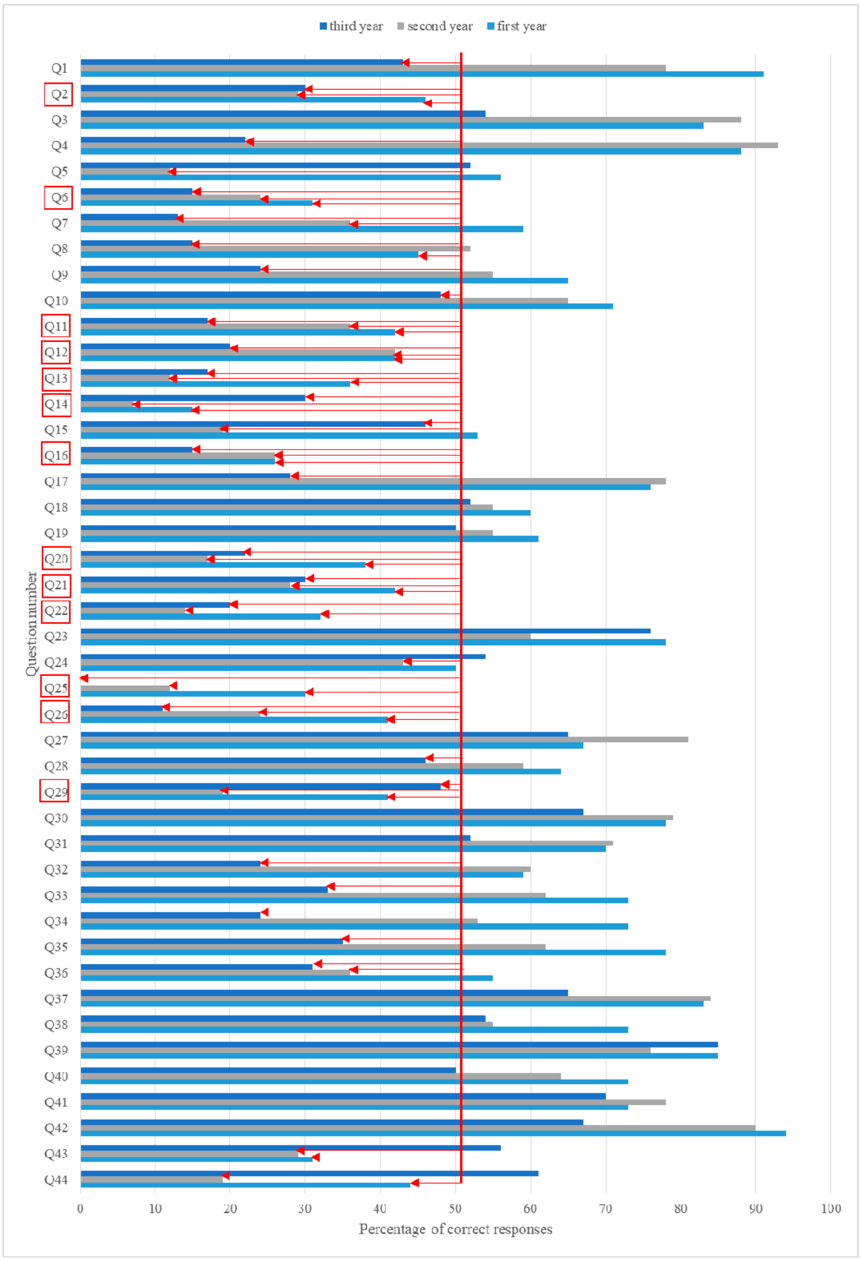

Figure 1. Percentage of correct answers for each question.

The first step of data analysis was the comparison of performance for each year of study separately. The one-way ANOVA test was used in order to determine the differences in grade results. Based on the Tukey B test, which followed the ANOVA, the following relationship can be recorded (Table 3):

Table 3. Results comparison by ANOVA.

	First Year		Second Year		Third Year		One-Way ANOVA	
	Mean	***SD***	**Mean**	***SD***	**Mean**	***SD***	**F**	***p***
Points	25.73	6.1	21.21	7.2	17.39	6.0	25.2	0.001

[first grade] > [second grade] > [third grade] F = 25.2 p = 0.001.

This was followed by a two-sample t-test so that the authors could examine the ratio of deviations in the light of the above results (Tables 4–6).

Table 4. Comparing results for the first and the second-year students.

	Results		Two-Sample *t*-test	
	Mean	***SD***	***t***	***p***
First year	25.73	6.1	3.97	0.001
Second year	21.21	7.2		

Table 5. Comparing results for the first and the third-year students.

	Results		Two-Sample *t*-test	
	Mean	***SD***	***t***	***p***
First year	25.73	6.1	7.44	0.001
Third year	17.39	6.0		

Table 6. Comparing results for the second and the third-year students.

	Results		Two-Sample *t*-test	
	Mean	***SD***	***t***	***p***
Second year	21.21	7.2	2.92	0.004
Third year	17.39	6.0		

The results highlighted that students in Year 1 achieved significantly better results than those in Year 2 and Year 3. The results of those in Year 2 and Year 3 also show a statistical difference in favor of the second year. Studies in Year 3 performed considerably poorer than the students in Years 1 and 2.

Therefore, these results do not support the H1 hypothesis: the students in Year 1 performed significantly better than the second- and third-year students.

4.6. Analysis Based on the Subjective Difficulty Estimation of the Test Questions

4.6.1. Comparing Students' Estimations

The following step in examining the results was to compare how students evaluated the difficulty of each test questions.

Given that the test contained a large number of questions, the analysis was not conducted on the whole set of questions (full scale). Five questions were selected for which the students had given the highest percentage of correct answers, and another five questions were selected for which students had given the lowest percentage of correct answers.

The responses were ranked on a three-point Likert scale, where the easy questions were rated as 1, medium questions were rated as 2, and the hard questions were rated as 3. Although the Likert scale is strictly a ranking scale, it is generally accepted that in order to take advantage of the possibilities offered by complex statistical procedures, it is treated as an interval scale. According to Selltiz et al. [21], the scores can be added and averaged.

The following analysis presents the degree of estimated difficulty for the top five correctly answered questions and top five questions with the lowest correct-answer rate, for all students from all years of study. The results show estimations on the full scale.

As already indicated, the students could rate every single question as easy, medium, or hard.

In terms of the average of the responses for students from all three years of study, the top five correctly answered questions were Questions 3, 30, 37, 39, and 42. For these specific questions, most of the estimations fell into the easy category, which is in accordance with the number of correct answers to those questions (Figure 2).

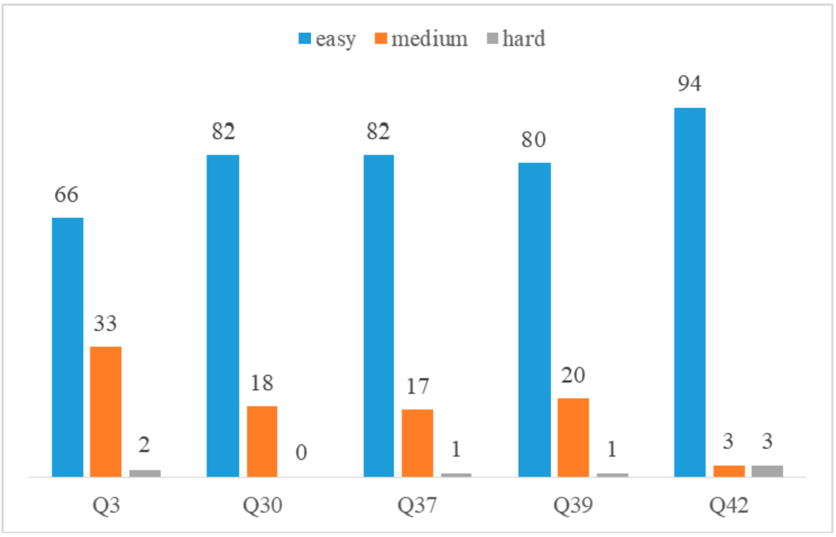

Figure 2. Estimation of the top five highest score questions (percentage of correct answers).

The descriptive results were analyzed with a one-way ANOVA test. Three out of the top five highest score questions (Q30, Q39, Q42) show a significant difference between students' estimation (Table 7). Based on the results, we can state that the students from the third year estimated the questions as significantly harder than the Year 1 and Year 2 students. The students from Year 2 estimated question Q39 as harder than the Year 1 and Year 3 students. The results for each question are less significant than the data for the whole scale, and this result partially supports the H2 hypothesis.

Table 7. The top five highest score questions.

	Year 1		**Year 2**		**Year 3**		**One-Way ANOVA**	
	Avg.	***Sd.***	**Avg.**	***Sd.***	**Avg.**	***Sd.***	**F**	***p***
Q3	1.31	0.5	1.36	0.4	1.48	0.5	1.52	0.02
Q30	1.12	0.3	1.17	0.3	1.30	0.4	3.18	0.04
Q37	1.22	0.4	1.24	0.4	1.00	0.0	2.92	0.05
Q39	1.12	0.3	1.33	0.4	1.20	0.4	4.49	0.01
Q42	1.00	0.00	1.10	0.2	1.24	0.6	6.1	0.003

The five worst-performing questions were Questions 6, 13, 14, 20, and 25 on the average of the responses for students from all three years of study. For these questions, most of the estimations fell into the medium or hard category (Figure 3). Based on the results, a review of question Q6 is definitely recommended, as the majority of students rated it as an easy question; however, they received one of the lowest scores (the lowest rate of correct answers) for this question.

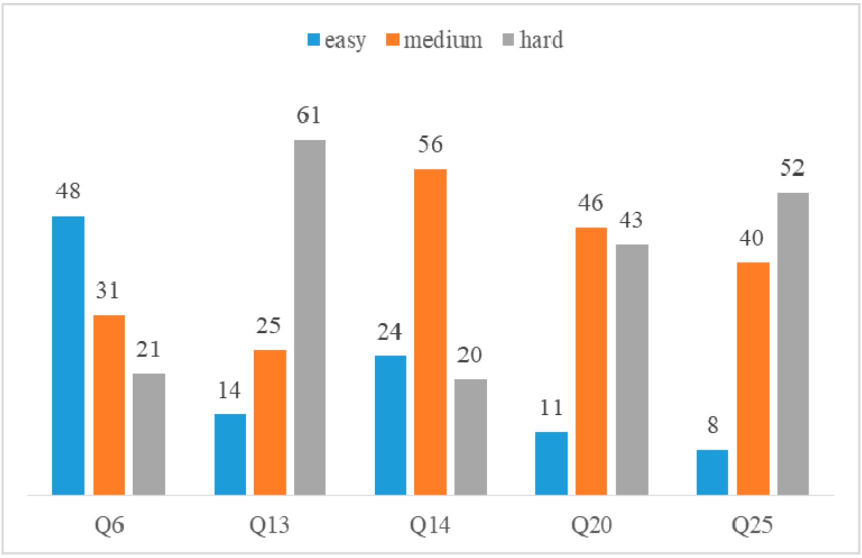

Figure 3. Estimation of the top five lowest-score questions (percentage of correct answers).

Four from the top five lowest-score questions (Q6, Q14, Q20, Q25) show a significant difference between students' estimation (Table 8). The Q6 question was estimated as significantly harder by the Year 3 students than the Year 1 and Year 2 students. In the case of Q14, opposite estimation was done: this question was "harder" for the Year 1 students. The result for Q14 supports the H2 hypothesis. The estimations for Q20 and Q25 questions differ from the previous three.

Table 8. The top five lowest-score questions.

	Year 1		**Year 2**		**Year 3**		**One-Way ANOVA**	
	Avg.	***Sd.***	**Avg.**	***Sd.***	**Avg.**	***Sd.***	**F**	***p***
Q6	1.47	0.6	1.60	0.6	2.37	0.7	23.2	0.001
Q13	2.48	0.7	2.44	0.7	2.48	0.6	0.02	0.9
Q14	2.12	0.5	1.96	0.6	1.70	0.8	5.93	0.003
Q20	2.21	0.6	2.62	0.5	2.11	0.6	6.86	0.001
Q25	2.22	0.6	2.63	0.4	2.59	0.7	7.23	0.001

These results do not support hypothesis H2 and show only that the students' estimation was close to the real difficulty of the questions: when a question was estimated as hard, then they achieved a lower score.

This section presents the results of the analysis answering the following questions:

- Is the lowest-score question estimated on the same level for all students?
- Is the highest-score question estimated on the same level for all students?
- What score did students achieve on those questions?

Score in this case refers to the rate of correct answers.

It was found that question Q25 received minimum correct answers (low score), and question Q42 was given the highest correct answers rate (high score) from students from all three years of study. The question with the highest score (Q42) was also rated as the easiest question by students of Year 1 and Year 2. Without exception, all students from Year 1 estimated its difficulty as easy, only 12% of students from Year 2, and 10% from Year 3 estimated the question's difficulty as medium difficult (Figure 4).

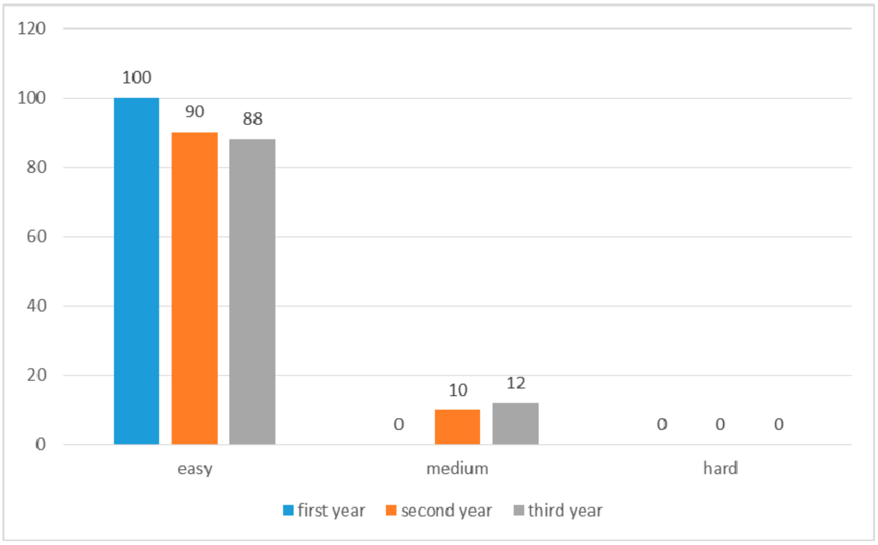

Figure 4. Estimation of difficulty for the highest-score (Q42) question.

As for the question with the lowest score (Q25), the students' estimations were quite different. The majority of students in Year 1 found the question moderately difficult (medium), while the majority of students in Years 2 and 3 found its difficulty hard (Figure 5). The same trend can be observed with the points achieved on these two questions: the Year 1 students received the highest amount of points, while the Year 2 and Year 3 students received the lowest points. At the same time, the Year 2 and Year 3 students also felt that it was more difficult for them to solve the task, which supports hypothesis H2.

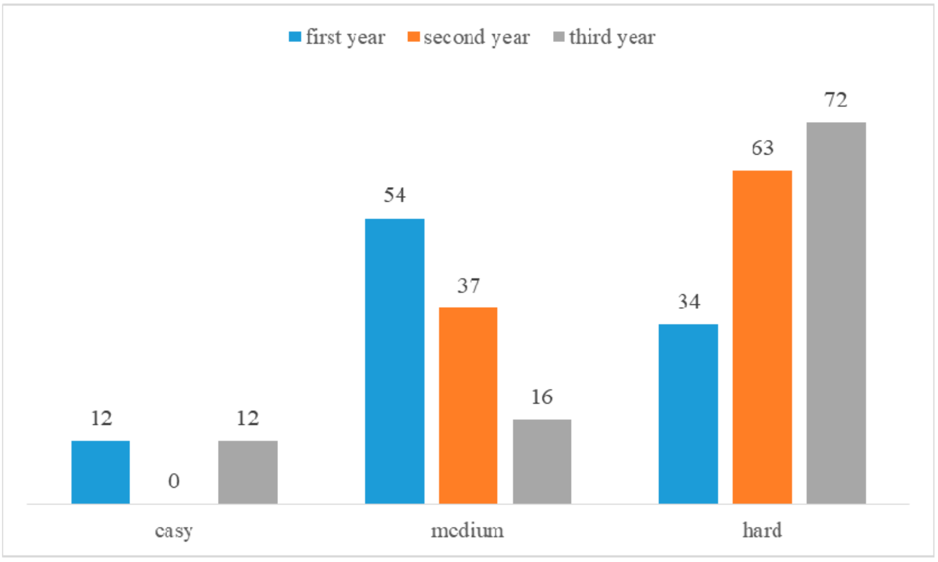

Figure 5. Estimation of difficulty for the lowest-score (Q25) question.

When estimating the degree of difficulty of the full scale (the whole test), the majority of students identified it as being of medium difficulty (Table 9).

Table 9. Estimating the degree of difficulty for the whole test.

	n	%
easy	30	16
medium	147	81
hard	5	3
Total	**182**	**100**

The differences in students' estimation for each year of study are illustrated in Figure 6.

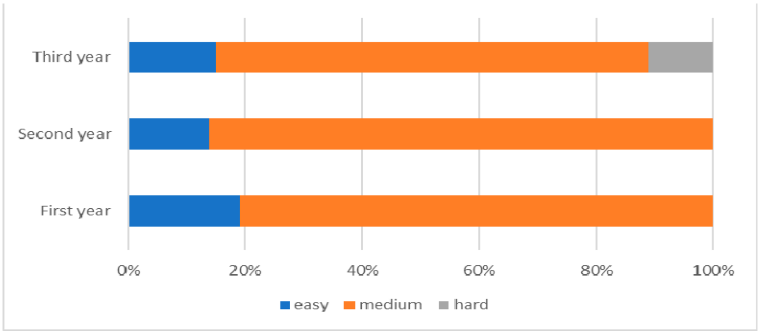

Figure 6. Estimating the difficulty of questions by year of study.

The results were tested with the chi square probe (Table 10).

Table 10. Correlations between year of study and estimated difficulty of the questions.

	Easy	**Medium**	**Hard**	**Total**
Year 1	12	66	0	78
Year 2	5	53	0	58
Year 3	7	36	3	46
Total	24	155	3	182

The data are consistent with the values shown in the descriptive statistics: the majority of students from all three years of the study estimated the test with medium difficulty value. The results show a correlation between the two variables (performance and task difficulty estimation) (χ^2 = 10.73 p = 0.03). When students from a higher year of the study estimated a question as medium or hard, then, they achieved a lower score there. This result supports hypothesis H2.

Based on year of study, the authors separately examined the five best- and worst-performing questions and the estimation associated with them. For these questions, they also compared the estimated value of the questions given by students with those defined by the instructors.

In addition to Questions 42 and 25 already analyzed above, several questions received similar estimations from the students of all three years of study. Question 37 received a high score from the students in their first and second year, and Q39 received a high score from the students in their second and third year. Students in Year 1 and Year 2 achieved low scores with Q14. Questions 5 and 16 were those where students in Year 1 and Year 3 gave fewer correct answers.

In the estimation of the difficulty of the five highest-scoring questions, the opinions voiced by students in Year 1 and the instructor are very similar (Figure 7). Estimations were also compared with a nonparametric statistical analysis, where the result of the Mann–Whitney test also supported the agreement of the estimations.

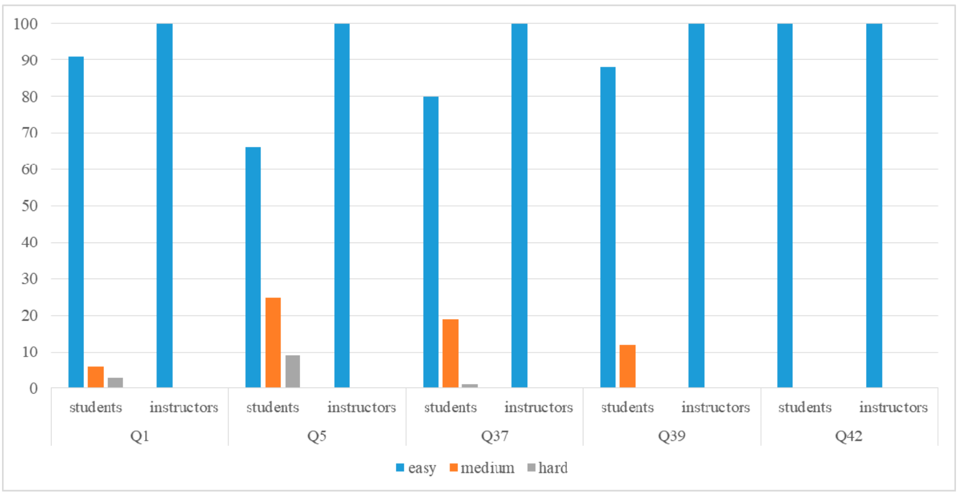

Figure 7. Comparing estimation difficulty of the five highest-scoring questions defined by the students in Year 1 and the instructors.

For the lowest-scoring questions, the estimations of the students and the instructor were no longer so consistent (Figure 8). For Questions 6 and 16, the majority of students and the instructor were of the same opinion. Question 25 was marked as medium by the students and hard by the instructor. Question 43 was rated as hard by the students as opposed to medium by the instructor. Statistically, these differences are not significant. However, for Question 14, where students estimated the question as medium and the instructor estimated the question as easy, there was a significant difference in estimation based on the Mann–Whitney test (Z = −1.97 p = 0.04). This result supports hypothesis H3: it indicates that the Year 1students gave a different estimation of difficulty than the instructor.

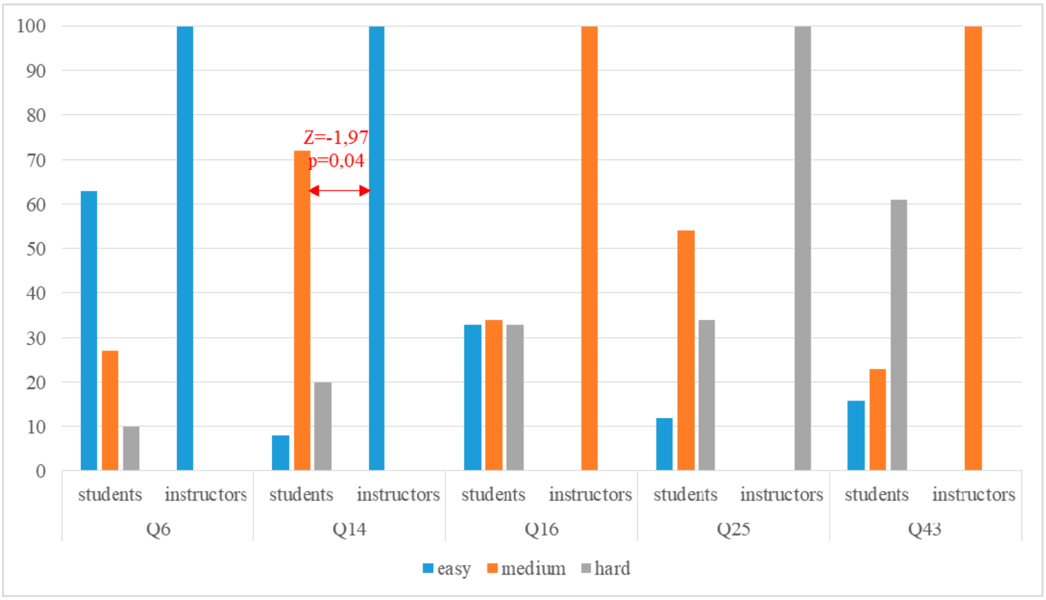

Figure 8. Comparing estimation difficulty of the five lowest-scoring questions defined by the students in Year 1 and the instructors.

The same two analyses were conducted for the estimation of students in Year 2. In the case of the questions with the highest scores, four questions (Q3, Q4, Q37, Q42) were estimated to be on the same level by both students and instructors (Figure 9). In the case of Question 27, the instructor rated the question as hard, while most of the students rated it as medium. The difference is only shown in the descriptive statistics; the Mann–Whitney test does not show a statistically significant difference. This result supports the H3 hypothesis. The questions with lowest scores showing in Figure 10.

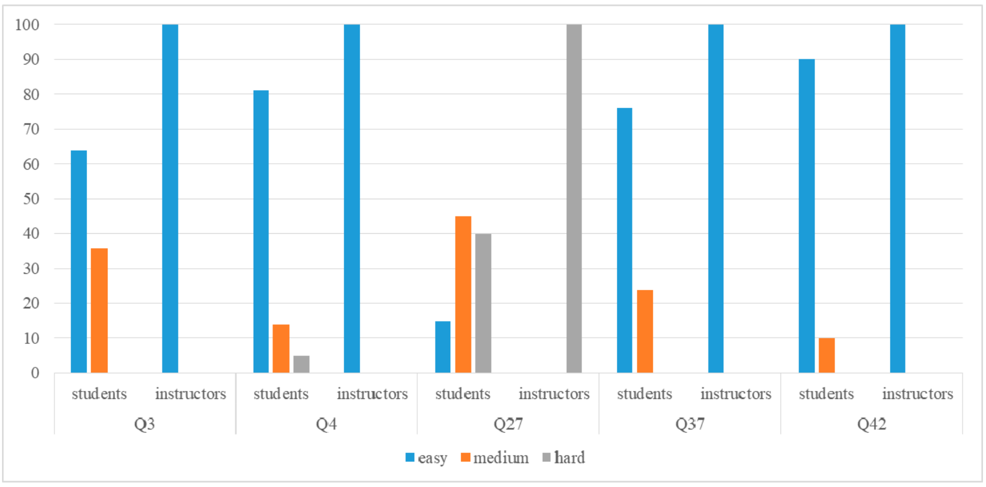

Figure 9. Comparing estimation difficulty of the five highest-scoring questions defined by the students in Year 2 and the instructors.

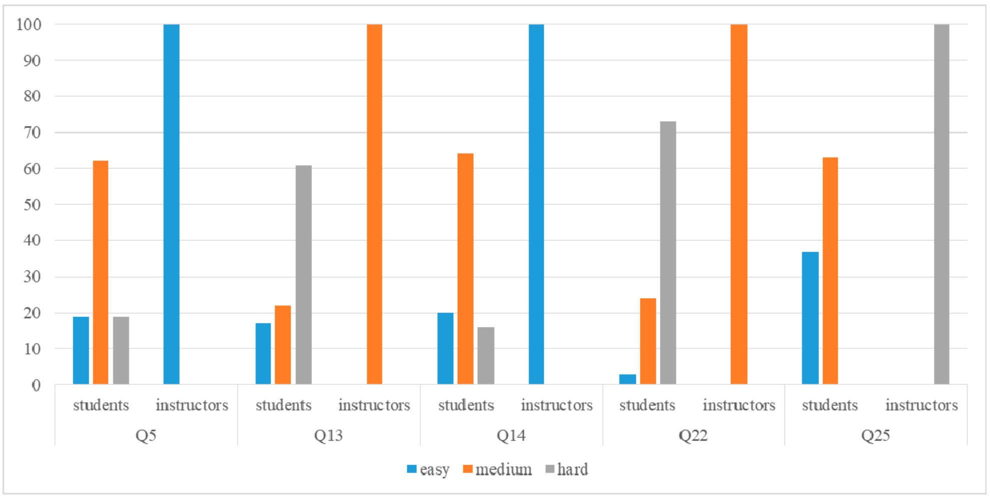

Figure 10. Comparing estimation difficulty of the five lowest-scoring questions defined by the students in Year 2 and the instructors.

With the students in Year 3, regarding the questions that received the most points, there was a difference in the estimation for the students and the teachers for Question 23 (Figure 11). The students rated the question as easy, whereas the teacher saw it as medium. The other questions' estimation did not show much diversion: the majority of students as well as the teachers found the questions easy.

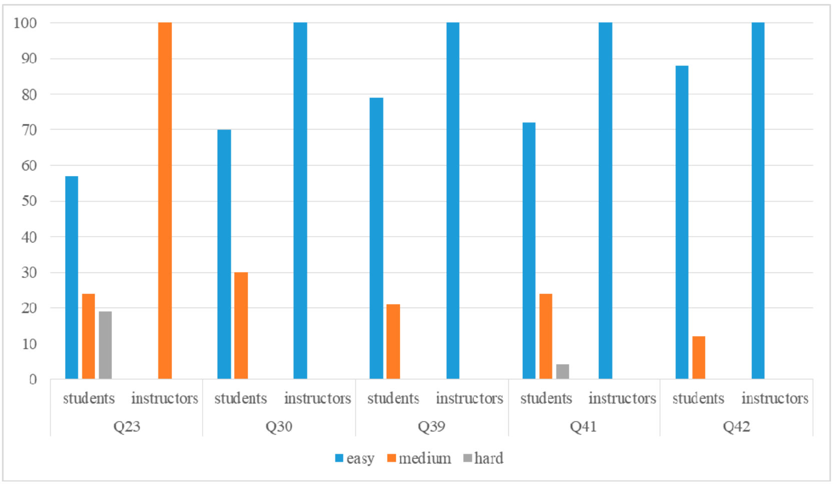

Figure 11. Comparing estimation difficulty of the five highest-scoring questions defined by the students in Year 3 and the instructors.

In the case of the questions with the lowest score (Figure 12), the estimations of the students and the instructor were the same for Questions 25 and 26, as both parties saw their difficulty as hard. Question 6 was considered hard by the students, as opposed to easy by the teacher. Question 7 was considered easy by the majority of students and of medium difficulty by the teacher. Question 16 was estimated as hard by the students and of medium difficulty by the teacher. In all cases, the Mann–Whitney test performed with these results showed no relevant difference between the estimation of the students from the third year and the teacher. This result also supports hypothesis H3 for the students of Year 3.

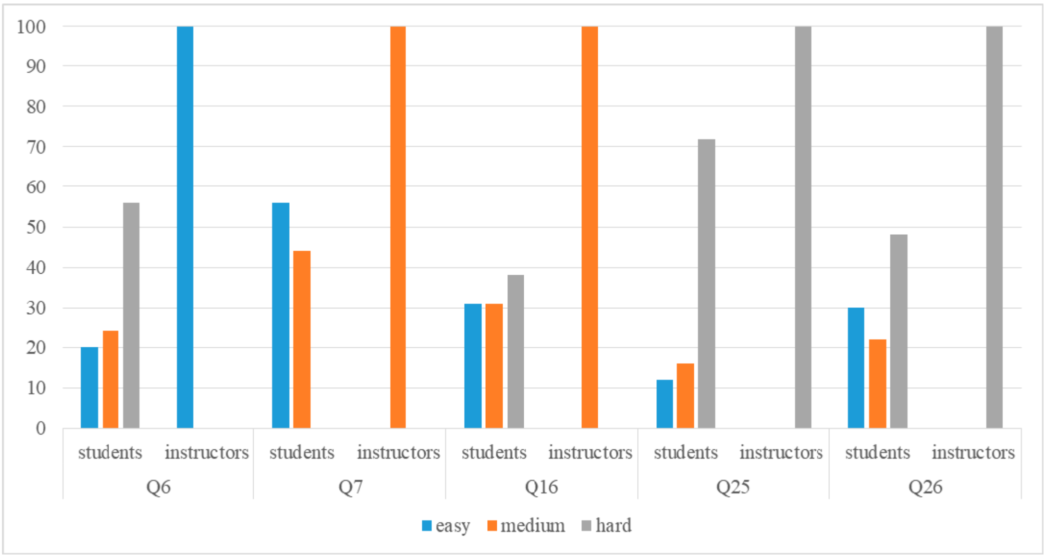

Figure 12. Comparing estimation difficulty of the five lowest-scoring questions defined by the students in Year 3 and the instructors.

4.6.2. Relationship between Student and Teacher Estimations

Apart from the students, all questions used in this research were also evaluated by the teachers. The teachers estimated the average difficulty of the questions' full scale as "medium". The results for the full scale are derived from the average score of the answers to each question. The difference between the teachers' and the students' evaluation was examined in a one-sample t-test, where the teachers' evaluation was viewed as an external standard average (Table 11).

Table 11. Comparing teacher and student estimation of test difficulty.

	Average of Marks by Students	*SD*	**Average of Marks by Teachers**	**One-Sample *t*-test**	
				t	***p***
First year	1.85	0.3	2	−3.74	0.001
Second year	1.91	0.2		−2.31	0.02
Third year	1.91	0.5		−1.27	0.2

Based on the obtained results, students in their first and second year estimated the questions as significantly easier than the medium difficulty level specified by the teachers. The third-year students' estimation was the same as that of the teachers.

The results support the H3 hypothesis, namely, there is no difference between the students' and the teachers' estimation in the third year of study on question difficulty. However, the discrepancy in the estimation of students in their first and second year suggests that the students perceived the tasks as easier than the teachers did.

5. Discussion

Three hypotheses were formulated in this research. The first was to prove that students were perfecting their knowledge and skills during their studies. The assumption was that if a given material was taught in the first year and subsequently used in the later years of education, that knowledge or skill would be improved. As a result, the students in the later years of study would solve the basic programming tasks more easily than their peers in the first year. The hypothesis was also supported by the assumption that during computer science education, students were also using the process of debugging to identify and remove errors from application in many other courses. Code tracing is one of the main techniques in debugging. To summarize these assumptions, the authors find that in the later years of study, students have more experience in understanding how the whole program, or just a segment of the code, works.

- The test results of this hypothesis show a reverse tendency. Possible reasons for hypothesis H1 not being confirmed may be as follows. First, programming requires many skills. Some of those programming skills are taught, while others are not. Those skills that are developed during the years of study tend to be independent from each other. For instance, improving algorithmic thinking or making different abstraction layers, which are the basis for developing a quality solution, does not depend on, e.g., code-writing, debugging, or code-tracing skills. This supports the view that the profession of a software developer is a complex one, and it is possible to define important skills for specific areas. Some research works confirm this [10,22].
- Students generally do not, or only to a very slight extent, associate knowledge from one course (subject) with the knowledge from another course with similar topics. This is possibly caused by the structure of the curriculum, which fails to sufficiently emphasize the connection between topics of different courses. There is another possible explanation: many students believe that a successfully passed exams means they are somehow "finished with it" and there will be no continuation. They erroneously assume that each new course, even with similar topics, is something completely new, for which they would not need to use previously learned topics. The fact that first-year students obtained better results can be explained by them not yet having completed the course 'Algorithms and data structures' and thus had not yet taken the final exam. For them, the first date to take the exam was three weeks away, counting from the date of doing the test for this research. Cognitive flexibility theory focuses on this behavior. The theory is largely concerned with the transfer of knowledge and skills beyond their initial learning situation. The theory asserts that effective learning is context-dependent, as argued by Spiro and Jehng. [4].

- The reason for poorer performance in later years of study may also lie in the fact that students with a greater amount of experience more often use libraries and advanced objects when solving problems. Functions from those libraries or objects' methods with one instruction solve a (complex) task, such as inserting a new element at the beginning of an array, doing binary search in array, sorting an array, etc. This may be the underlying reason for why senior students make less use of the basic elements of programming language such as statements, conditionals, and iterations to create a solution. Conversely, this also means that some of their vital software developing skills, such as code tracing, fail to develop and evolve.
- In their second and third years of the study, students frequently use ready-made solutions that can be found online. By querying the expression "how do you break string to words in C?", the Google search engine offers more than 170 million hits. One of those hits is sure to contain a suitable solution for the given situation. Students apply these ready-made solutions without too much thinking, analyzing or, in fact, understanding the actual code.

The second and third hypotheses are based on the assumption that during their studies, students will enhance their programming skills, because they have to solve an increasing number of programming tasks and develop projects with growing complexity. More hours spent solving programming tasks should ideally contribute to the creation of better abstraction layers, improved task comprehension, and finding the optimal solution. The authors' assumption was that due to the lack of experience in first-year students, their perception of task difficulty would mostly be medium or hard. The assumption was that the difference between subjective evaluation of the task difficulty given by students and teachers would be smaller, since students are improving their skills in solving programming tasks.

Most of the first-year students lack experience in code writing, algorithmic thinking, and debugging, thus making it more challenging for them to objectively evaluate the difficulty of test questions.

Since they do not have enough experience solving programming tasks, they do not have any basis of comparison for a given task's difficulty. This is precisely why they perceived the tasks as more difficult than they actually were and also more difficult than how the second- and third-year students evaluated them. Through participating in various projects and exercises, senior students gain experience in solving tasks and knowledge of how to solve similar problems, so that they can faster recognize the elements needed for creating the solution, thus enabling them to more objectively evaluate the difficulty of the task at hand. The results of the conducted research confirm this assumption, and by this, the second hypothesis is also confirmed.

The same trend can be observed regarding hypothesis H3: the results of estimating the task difficulty given by the third-year students approximate the evaluation given by the teachers. Since the teachers defined the level of every task by taking into account both task complexity and difficulty, it can be stated that objective estimation skills can be acquired with more and more years of education. This is why senior students were able to objectively evaluate the complexity and difficulty of the task.

The results of comparing the answers on questions on a full scale can be used in the so-called computerized adaptive testing (CAT). These tests adapt to the given student's level of ability. CAT selects questions so as to maximize the exam accuracy based on what is known about the student from previous questions [23]. For the student, this means that he or she will always be given a "tailored" test suited to their level of ability. For instance, if the student does well on a medium question (i.e., medium in difficulty), they will then receive a more difficult next question. However, a wrong answer to a given question means the student performed poorly and will be given simpler questions. For tailoring the optimal item (subsequent question) in CAT, the system uses IRT and requires a database in which all the levels of difficulty for all questions (item) are precisely determined (calibrated). Figure 13 shows the probability of a correct response depending on the skill level. The left-hand curve indicates an easy item, the middle-dotted line refers to a medium item, while the broken line on the right side signifies an item whose difficulty level is hard. The horizontal axis shows skill levels between −4 and 4.

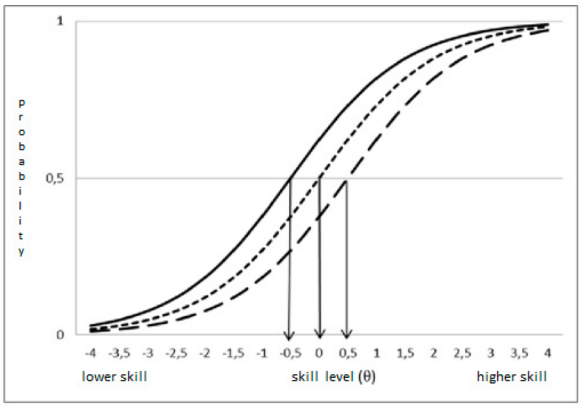

Figure 13. Probability for the correct answer.

The results of this research can help instructors in creating items for such a database. The following passage offers some ways to apply the results of the research to CAT.

Based on the score (correct answers) results (Figure 1), groups of difficulties can be formed. For example, questions for which less than 20% of the students supplied a correct answer would be classified as hard. Questions with scores between 21% and 40% would be medium, while the rest would be categorized as easy. This is a categorization of the items and not the calibration. However, this categorization can be used to build different variations of tests with approximately the same difficulty by selecting tasks from low, medium, or hard categories. The here-presented results can also be used in the process of item calibration for CAT.

By combining the score and the estimated difficulty, it is possible to pinpoint, for example, which are the easiest or most difficult questions in the set. Figure 2 presents those questions that were rated as easy; thus, they had the most correct answers, whereas Figure 3 contains those questions that were difficult for all students. Furthermore, instructors can identify the problematic questions in the set, such as Q6, which was rated as easy, but the majority of the students failed to answer it correctly. In this study, the analysis was limited to the five best- and five worst-performing questions. Extending the analysis to the whole set of questions may produce a subset of questions that can be used in CAT—and, due to very different scores and estimations, those that cannot be used in CAT.

The results, as summarized in Figures 7–12, show the level of difference between the estimation of difficulty given by the students and those given by the instructors. This comparison can help determine the real level of difficulty for a given question, because:

- It excludes an estimation that is based only on one teacher opinion or several of them.
- It excludes the possibility of defining the rate based on estimations given by students from a single year of study: the level of knowledge and skills may vary from one cohort to the next. Using questions with estimations from a so-called academically speaking "strong" cohort, i.e., students with good results, will give faulty results in the not-so-stellar, academically "weaker" cohort.

Significant differences between the estimation of difficulty given by students from different years of study and the teachers may indicate specific questions where the probability of giving the correct answer also depends on certain other factors. There may be situations when giving the correct answer depends on knowledge in some other topic, but then such items should be excluded from the database of questions.

As for future plans, the next phase of this particular research will include a more detailed analysis of the results. Comparing the results by the categories (sub-scale) can offer a better understanding of the efficiency of the implemented teaching methodology. The results of this comparison can aid teachers in pinpointing the specific parts of the material in the CS1 course that need to be taught or explained in a different way.

Author Contributions: Conceptualization, A.K.; methodology, R.P. and S.M.Č.; software, R.P. and S.M.Č.; validation, L.M., R.P. and S.M.Č.; formal analysis, A.K. and J.K.; investigation, R.P.; resources, S.M.Č.; data curation, S.M.Č.; writing—R.P. and A.K.; writing—review and editing, S.M,Č and J.K.; visualization, L.M.; supervision, P.Č.; project administration, A.K.; funding acquisition, A.K. All authors have read and agreed to the published version of the manuscript.

Funding: The project is sponsored by EFOP-3.6.2-16-2017-00018 "Produce together with the nature" project.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Reckase, M.D.; Ju, U.; Kim, S. How Adaptive Is an Adaptive Test: Are All Adaptive Tests Adaptive? *J. Comput. Adapt. Test.* **2019**, *7*, 1–14. [CrossRef]
2. Luecht, R.M.; Nungester, R.J. Some Practical Examples of Computer-Adaptive Sequential Testing. *J. Educ. Meas.* **2005**, *35*, 229–249. [CrossRef]
3. Lister, R.; Clear, T.; Simon, B.; Bouvier, D.J.; Carter, P.; Eckerdal, A.; Jacková, J.; Lopez, M.; McCartney, R.; Robbins, P.; et al. Naturally occurring data as research instrument: Analyzing examination responses to study the novice programmer. *ACM SIGCSE Bull.* **2010**, *41*, 156–173. [CrossRef]
4. Spiro, R.J.; Jehng, J.C. Cognitive flexibility and hypertext: Theory and technology for the nonlinear and multidimensional traversal of complex subject matter. In *Cognition, Education, and Multimedia: Exploring Ideas in High Technology*; Nix, D., Spiro, R.J., Eds.; Lawrence Erlbaum Associates, Inc.: Mahwah, NJ, USA, 1990; pp. 163–205.
5. Maravić Čisar, S.; Radosav, D.; Pinter, R.; Čisar, P. Effectiveness of Program Visualization in Learning Java: A Case Study with Jeliot 3. *Int. J. Comput. Commun. Control* **2011**, *6*, 668–680. [CrossRef]
6. Lopez, M.; Whalley, J.; Robbins, P.; Lister, R. Relationships between reading, tracing and writing skills in introductory programming. In Proceedings of the Fourth International Workshop on Computing Education Research (ICER '08), Sydney, Australia, 6–7 September 2008; pp. 101–112.
7. Kumar, A. Solving Code-tracing Problems and its Effect on Code-writing Skills Pertaining to Program Semantics. In Proceedings of the 2015 ACM Conference on Innovation and Technology in Computer Science Education (ITiCSE'15), Vilnius, Lithuania, 4–8 July 2015; pp. 314–319.
8. Cunningham, K.; Blanchard, S.; Ericson, B.; Guzdial, M. Using tracing and sketching to solve programming problems: Replicating and extending an analysis of what students draw. In Proceedings of the 2017 ACM Conference on International Computing Education Research, Tacoma, WA, USA, 18–20 August 2017; pp. 164–172.
9. Nelson, G.L.; Xie, B.; Ko, A.J. Comprehension First: Evaluating a Novel Pedagogy and Tutoring System for Program Tracing in CS1. In Proceedings of the 2017 ACM Conference on International Computing Education Research, Tacoma, WA, USA, 18–20 August 2017; pp. 2–11.
10. Sheard, J.; Souza, D.D.; Klemperer, P.; Porter, L.; Zingaro, D. Benchmarking Introductory Programming Exams: Some Preliminary Results. In Proceedings of the 2016 ACM Conference on International Computing Education Research, Melbourne, Australia, 8–12 September 2016; pp. 103–111.
11. Xie, B.; Loksa, D.; Nelson, G.L.; Davidson, M.J.; Dong, D.; Kwik, H.; Tan, A.H.; Hwa, L.; Li, M.; Ko, A.J. A theory of instruction for introductory programming skills. *Comput. Sci. Educ.* **2019**, *29*, 205–253. [CrossRef]
12. McCracken, M.; Almstrum, V.; Diaz, D.; Guzdial, M.; Hagen, D.; Kolikant, Y.; Laxer, C.; Thomas, L.; Utting, I.; Wilusz, T. A Multi-National, Multi-Institutional Study of Assessment of Programming Skills of First-year CS Students. *ACM SIGCSE Bull.* **2001**, *33*, 125–180. [CrossRef]
13. Lister, R.; Adams, E.S.; Fitzgerald, S.; Fone, W.; Hamer, J.; Lindholm, M.; McCartney, R.; Mostrom, J.E.; Sanders, K.; Seppala, O.; et al. A Multi-National Study of Reading and Tracing Skills in Novice Programmers. *ACM SIGCSE Bull.* **2004**, *36*, 119–150. [CrossRef]
14. Kopec, D.; Yarmish, G.; Cheung, P. A description and study of intermediate student programmer errors'. *ACM SIGCSE Bull.* **2007**, *39*, 146–156. [CrossRef]
15. Lampert, B.; Pongracz, A.; Sipos, J.; Vehrer, A.; Horvath, I. MaxWhere VR-learning improves effectiveness over clasiccal tools of e-learning. *Acta Polytech. Hung.* **2018**, *15*, 125–147.
16. Budai, T.; Kuczmann, M. Towards a modern, integrated virtual laboratory system. *Acta Polytech. Hung.* **2018**, *15*, 191–204.

17. Csapó, G. Placing event-action-based visual programming in the process of computer science education. *Acta Polytech. Hung.* **2019**, *16*, 35–57.
18. Katona, J.; Kovari, A. Examining the learning efficiency by a brain-computer interface system. *Acta Polytech. Hung.* **2018**, *15*, 251–280.
19. Katona, J.; Kovari, A. The evaluation of bci and pebl-based attention tests. *Acta Polytech. Hung.* **2018**, *15*, 225–249.
20. Sik-Lanyi, C.; Shirmohammmadi, S.; Guzsvinecz, T.; Abersek, B.; Szucs, V.; Van Isacker, K.; Lazarov, A.; Grudeva, P.; Boru, B. How to develop serious games for social and cognitive competence of children with learning difficulties. In Proceedings of the 2017 8th IEEE International Conference on Cognitive Infocommunications (CogInfoCom), Debrecen, Hungary, 11–14 September 2017; pp. 321–326.
21. Selltiz, C.; Jahoda, M.; Deutsch, M.; Cook, S.W.; Chein, I.; Proshansky, H.M. *Research Methods in Social Relations*; Methuen & Co. LTD.: London, UK, 1964.
22. Cooper, S.; Wang, K.; Israni, M.; Sorby, S. Spatial skills training in introductory computing. In Proceedings of the Eleventh Annual International Conference on International Computing Education Research, Omaha, NE, USA, 9–13 August 2015; pp. 13–20.
23. Maravić Čisar, S.; Pinter, R.; Čisar, P. Evaluation of knowledge in Object Oriented Programming course with computer adaptive tests. *Comput. Educ.* **2016**, *92–93*, 142–160. [CrossRef]