Developing a Test for Assessing Incoming Students' Cognitive Competences

https://doi.org/10.3991/ijep.v7i4.7433

Veronika Thurner, Daniela Zehetmeier, Sabine Hammer, Axel Böttcher(✉)
Munich University of Applied Sciences, Munich, Germany
ab@cs.hm.edu

Abstract—Every year, a few weeks after new freshmen students started on their course of studies in Computer Science, many lecturers complain that some of their students are ill-equipped with those competences that are necessary to study successfully. Hence, these students struggle. This situation can be observed universally, at different universities and in various countries. On the other hand, at the same time there usually are some students around, which meet study requirements rather easily. To effectively deal with this heterogeneity, lecturers need to gain a quick overview of initial competences in their student cohorts. This paper describes the development of a test that assesses first-year students' initial cognitive competences as well as basic knowledge in maths and computer usage. On this basis, lecturers can adapt their lectures to address the students' current level, in order to quickly develop those skills that are still missing. We have been using this test for four years, for a pilot study as well as three regular runs. So far, we have collected test results of over 750 students. First insights into the results confirm our assumption that important competences are lacking in many freshmen students. As a consequence, we adapt our teaching in introductory courses, to enable students to close the gap and quickly meet study requirements.

Keywords—cognitive competence, assessment, skill level, study requirements

1 Motivation

All over the world, university lecturers report that some students meet study requirements rather easily, while others struggle. Although the academic community has been dealing with this situation for years, the key factors that make some students successful while others struggle are not yet sufficiently well understood. Therefore, we attempt to analyse freshmen students' initial competences and relate them to study requirements.

Our observations from past teaching experience suggest that more and more first-semester students are ill equipped with those base competences that are necessary to study successfully. More precisely, many of them have significant deficits in those cognitive competences that are crucial for studying Computer Science successfully, such as abstract and logical thinking. Based on these observations, we built the hy-

pothesis that some students have competences that others still need to develop – and that these essential competences are not focused on sufficiently during first semester classes.

Usually, lecturers make a great effort to provide study conditions that allow all students in their cohort to have a fair chance to complete their studies successfully. To achieve this, lecturers need to know their students' level of initial competences as soon as possible, i.e. *before* the first season of regular exams. On this basis, they can adapt their teaching to meet students' requirements.

In order to facilitate this, we developed and applied a program for the student entry phase, which comprises two well-designed tests that assess a set of competences that are essential for the study process. In this work, we focus on assessing those *cognitive* competences that are especially relevant for studying Computer Science or related topics successfully.

The test designed in this paper is integrated into a framework of various activities, focussing on four main aspects: facilitating transition from school to university, establishing a network among the students, providing insights into students' initial competences, as well as confronting our students with typical core tasks of computer scientists to gain an impression of typical tasks in professional life.

2 Goals and Benefits

At the beginning of their studies, most students possess only little knowledge in the domain of their selected course of studies, i.e. Computer Science (CS) in our example. Therefore, our test does not address CS related content. Rather, we focus on those cognitive base competences that are required to develop CS related skills later on. As a first step in designing the test for incoming students, these competences have to be selected in a systematic way. Applying the approach described in [1], we identified the following cognitive competences as being highly relevant: systematic thinking, logical thinking, thinking in an abstract way, thinking concretely, analytical thinking, and thinking holistically (cf. Table 2)

To systematically evaluate these competences, we employ a set of test items that attempt to address each single competence on its own, i.e. as isolated as possible. For example, if we want to assess the competence of abstract thinking, we try to keep the textual definition of the test item as simple as possible, to ensure that any deficiencies in reading abilities do not interfere with the competence in focus. To identify suitable tasks for testing these competences, we thoroughly analyse existing material, by rating each task according to the cognitive competences we focus on.

The same test can be applied to incoming students, and again at a later point of the study process. Thus, it is possible to assess the change of (and hopefully increase in) competences throughout the study process. As well, by comparing test results from different cohorts that did (or did not) run through any interventions that were applied to foster certain competences, we can evaluate the effectiveness of these activities.

All data collected using our set of tests can be enriched by information that describe the students' academic success or failure, like exam participation and the aver-

age grade students finally receive. On this basis, data analysis can be employed to identify competences as success factors – or indicators for possible struggle, if the respective competence is not sufficiently developed.

The results of our tests are beneficial for both lecturers and students. By gaining insights into their students' current skill level, lecturers can compare these to their implicit expectations, thus creating awareness for their students' needs. Having identified which competences are the key to success, lecturers can focus their teaching on developing these skills in their students. For example, lecturers can create and integrate interventions into their lectures, which explicitly address and improve these competences. Over time, this leads to a portfolio of well-established teaching methods that address typical students' needs.

Students, too, profit from the results of our tests. For one thing, the list of those competences identified as being relevant for successfully studying computer science helps students to realize what is expected of them. For another thing, the results of our test and a corresponding self-assessment [2] help students to become aware of their own initial competence profile and to identify improvement potential early on in their study process. Thus, students can take effective measures to sharpen their profiles by enrolling in appropriate interventions that especially address their individual needs.

3 Related Work

Thurner and Böttcher conducted a regional survey among lecturers, analysing their expectations of incoming students' initial competences and comparing them to those competences these students possess in reality [3]. This survey focusses on soft skills and cognitive skills (rather than professional skills), as they are the foundation for the learning process. As a next step, Thurner et al. [1] identified a set of competences that are relevant, but often missing in first-year students of CS or related topics.

3.1 Task Collections

In order to identify suitable items for the test, we analysed a number of different sources, e.g. workbooks and textbooks (e.g. [4], [5], [6], [7]), certification tests, school contests (such as [8]) and publications. As well, we considered professional tests from psychology as another useful source. The Goldstein-Scheerer [9] test addresses abstract and concrete thinking, while the Halstead Category Test [10] focusses on abstract thinking. The emphasis of our search was laid on tasks that do not require any specific CS professional knowledge.

One valuable source is the so-called "Informatik-Biber"[1] [8], which is an annual contest (online test) for pupils between the ages of 8 and 19. The "Informatik-Biber" is the German release of the "Bebras International Contest on Informatics and Computer Literacy", which originated in Lithuania 2003. After one year of creating tasks and preparing technology to implement the test, the first contest started in Lithuania

[1] Supported by: German Informatics Society, Fraunhofer Information and Communication Technology Group, Max-Planck Institute for Informatics and the federal ministry of education and research.

in October 2004 [11]. The goal of this contest is to draw the pupils' interest on informatics, by setting exciting tasks that require no previous CS knowledge. Tasks focus on the "understanding of the principles, ideas and concepts that are involved in informatics systems" [11]. At the same time, the seemingly omnipresent fear of coming into contact with informatics is reduced. Thus, pupils are motivated to get engaged in informatics.

Dagienė and Futschek [11] also provide several criteria for successfully developing suitable Bebras tasks. Furthermore, they sketch an approach for transferring complex concepts and processes that require Computer Science knowledge into tasks that pupils can easily understand, but which still represent these concepts and processes.

The "Informatik-Biber" started in 2007. All tests as well as their solutions are available online[2]. Each task of the tests is characterised by three kinds of meta-information: form, level of difficulty, and a description why this task is related to informatics.

3.2 Relating Tasks to Competences

"Tasks can be classified by very different properties, such as the difficulty, the purpose, the type or form (e.g. open or multiple choice questions), or the context, to mention only some features" [12]. Many papers deal with the classification of assignments, using different aspects for their classification. Whereas Tharp [13] concentrates on programming goals, Hansen [14] combines the students' assessed sophistication in the topic with the underlying study material to classify an assignment. A classification according to context of the programming assignment is described by Layman et al. [15]. Wilson [16] focuses on which type of assignment is preferred by which gender, where game, application for an applied field or an application for mathematics are distinguished as available types.

Table 1. Categories of questions from the Bebras contest [11].

Category		Description
INF	Information comprehension	Representation (symbolic, numerical, visual) Coding, encryption
ALG	Algorithmic thinking	Including programming aspects
STRUC	Structures, patterns and arrangements	Combinatorics Discrete structures (graphs, etc.)
PUZ	Puzzles	Logical puzzles Games (mastermind, minesweeper, etc.)
USE	Using computer systems	E.g. search engines, email, spread sheets, etc. General principles, but no specific systems
SOC	ICT and Society	Social, ethical, cultural, international, legal issues

Members of the Bebras Organising Committee proposed six topics for the Bebras contest in September 2007. These categories improve the initially suggested task

[2] http://informatik-biber.de/archiv

types of [17] to the effect that tasks within a category are of equal importance, and that the names are easy to understand [11]. The categorization developed is shown in Table 1. Each task can belong to one or more categories.

However, there are no categorizations for well-known techniques for analysing the cognitive competences necessary to solve a given problem. In order to analyse which competences are involved in solving a task, we employed methods such as think aloud activity [18] and decoding the discipline [19]. These methods aim at making expert thinking visible, and thus accessible for students. They can be adapted to make any thinking process evident, for different purposes.

4 Knowledge Test Design

As each student cohort differs with respect to their initial competences, the current state of competences must be evaluated for each of them. To achieve this, we develop a questionnaire-based test tool set for assessing the relevant competences. Our student cohorts tend to be rather large. Therefore, the test design ensures that the test can be processed automatically.

After identifying competences that are relevant for studying computer science or related topics successfully (Thurner, Böttcher, & Kämper, 2014), the project team started to develop different assessments to evaluate students' competences in 2013. In October 2013, the tool set of test instruments (cf. Fig. 1) contained three tests: a questionnaire focussing on personal information, a self-assessment focussing non-technical skills, and a knowledge test. The self-assessment is described in detail in [3].

The first version of the knowledge test was designed in winter term 2013/14. By now, the test is in its fourth iteration. Over time, the knowledge test was refined, but has been rather stable since winter term 2014/15.

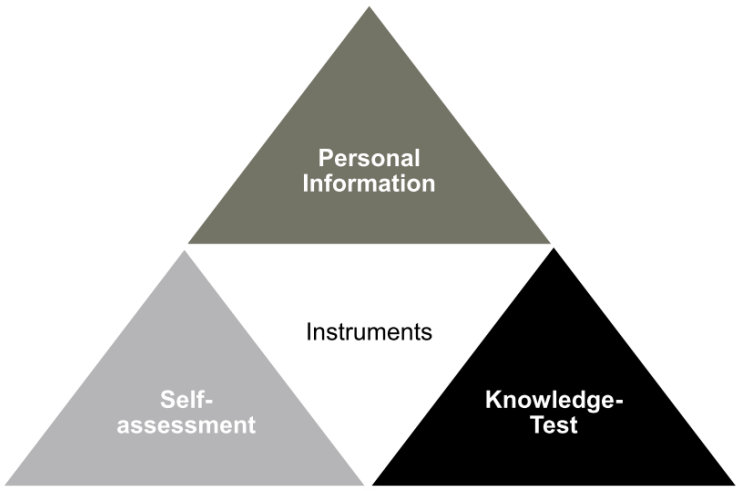

Fig. 1. Tool set of test instruments.

4.1 Designing the Prototype and Pilot Study

In winter term 2013/14, data collection started with a prototype of the knowledge test. It consisted of six parts: German reading comprehension, English reading comprehension, basic math, computer knowledge (theory and practice) and CS-specific competences. The part with CS-specific competences was based on a selection of cognitive competences identified in [1] as being crucial for studying Computer Science or related topics successfully. [2] provides short definitions of these competences, to help students to reflect on their respective skills. Table 2 offers a selection of competences and their description. The initial test was paper-based and transcribed into electronic data manually.

Table 2. Selection of relevant competences [1] and their descriptions [2].

systematic I consequently work off given instructions and apply well known methods.
logical thinking I develop my thoughts step by step, regarding cause and effect.
thinking in an abstract way My perception of the world is a theoretical one. I look for general laws and principles.
thinking concretely I take in the world by experiences that I make myself, gather impressions and think in examples.
analytical thinking When I look at things, I search for individual elements, and like to get to the bottom of things.
thinking holistically I look at things in their entirety and like to maintain an overall view.

German reading comprehension was adapted from [20]. The task is composed of a text about "learning", consisting of 563 words. We selected ten single choice questions out of 20, which students have to answer. In five of these ten questions, more than 80% of the students gave the correct answer. The poorest result was the last question, which only 62% of the students got right. However, this is still a very good result. Further data analysis indicated that these questions have no significant correlation to the competences in question.

Text and questions for the English reading comprehension were adopted from the University of Victoria [21]. The text contained 333 words about "interest groups" and "lobbies", and provided ten associated questions. We adapted some of these questions and invented some new ones, resulting in a total of 12 questions. Of these questions, five were answered correctly by over 90% of the students, as opposed to 42% of the students that answered the "hardest" question correctly. All in all, for 10 out of 12 questions over 55% of the students gave the correct answer. Here again, data analysis did not indicate any reasonable correlation to the students' performance on cognitive tasks.

Tasks for the test part focussing on basic math were provided from several lecturers of our math department. These tasks focus on basic mathematical principles that students should already know from school. Thus, we got a portfolio that contains questions on fractions, functions, statistics, stochastic and problems specified in natu-

ral language. In addition, two questions were added from the "ZEIT-Mathetest" [22]. All in all, 10 of the 13 questions in the test require free text as answer. Based on those student answers that are given most often in the pilot test, answer possibilities for the next run can be derived, thus moving from free text to multiple choice and automatic grading.

As the test aims at providing insight into skills of first-year students of computer science, the section on computer knowledge seemed to be rather important. Here, we selected a number of multiple choice questions from INCOBI-R [23]. Six questions asked for theoretical knowledge, whereas another two focused on practical skills.

To address CS-specific cognitive competences such as analytical and logical thinking, we initially included six questions. Two of them were so-called Bongard-problems (Bongard, 1970), which can be found in (Süddeutsche Zeitung). Furthermore, three tasks on logic were added from (Süddeutsche Zeitung) as well. The last task graphically represents a system of equations (Loyd & Gardner, 2003) (see Fig. 2) that has to be solved, using a combination of different cognitive competences.

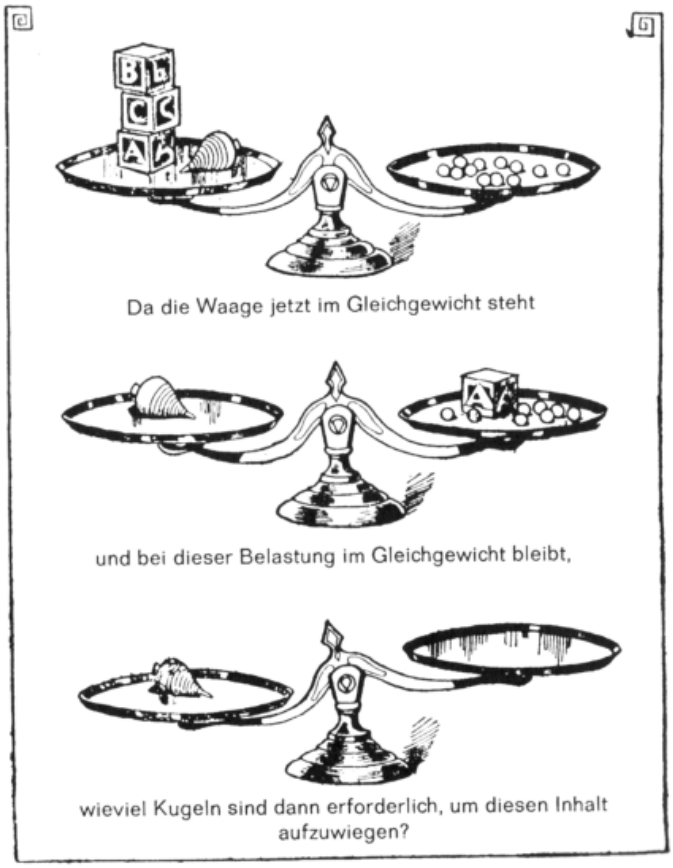

Fig. 2. Example question based on [26]. Weight is balanced in the upper two situations. How many balls are required to balance it in the lower image as well?

4.2 Improvements Towards the Final Version

After conducting our pilot study, we moved to a test that is still paper based, but can be evaluated automatically after scanning the students' test sheets.

In the pilot test, the vast majority of students performed really well in German and English reading comprehension with texts from a non-IT context. As a consequence, it is not possible to identify significant differences in skills in this area. Therefore, we omitted this part in the final version of the test. Analogously, other tasks with more than 80% correct answers were removed from the test as well. As a consequence, the two test parts focussing on theory and practice of computer-related knowledge were thinned out and summarised into one.

For the transition from a text-based test with free-text answers towards an electronic test with multiple-choice questions, we analysed students' answers from the pilot study and clustered them. For example, in the test items on basic math, we grouped the different solutions for a single task according to their underlying solution process or typical misconceptions. This resulted in about five typical answers for each task. For example, the initial version of the questionnaire contained the task of reducing the fraction 294/210. Clustering of the solutions led to five groups of solutions: correct ones (7/5, 1 2/5), minimalists (1 84/210), complete misconceptions (94/10) and calculation errors (everything else).

The most complex part in the improvement of the knowledge test were the tasks for assessing relevant cognitive competences. The simple logic questions provided in the pilot test were solved rather well by most students. Nevertheless, these students had difficulties in applying logic when dealing with their lab session assignments later on. To gain a more detailed insight into students' initial skills in this area, it was necessary to improve and expand this section of the test.

4.3 Enhancing Test Items for Cognitive Competences

To improve our tasks that focus on cognitive competences, we started by scanning several workbooks for proper tasks. However, all tasks that we found required specific knowledge of concepts or principles in Computer Science. Hence, they are not appropriate for testing first-year students that have not yet learned how to program. Instead, tasks should address the Computer Science mind-set in a non-CS context, thus not requiring any prerequisites. As a consequence, other sources that represent the Computer Science mind-set in a non-CS context need to be found.

Another possible source are commercial tests on selected cognitive competences, which are used by psychologists or in assessment centres. However, many of these tests focus only on a single competence, testing it thoroughly in different variants and from different angles. As we need to address not just one, but seven cognitive competences, this kind of test does not meet our requirements for test efficiency.

In the end, we focused on well-known task types for selected competences. For example, we chose Bongard-problems [24] to assess abstract thinking. Furthermore, we identified the "Informatik Biber" as a really valuable source, as the tasks are well designed by experts and match many of our requirements, such as:

- Focus on competences relevant in Computer Science
- Appropriate level of difficulty
- No Computer Science prerequisites
- Represents Computer Science mind-set

The collections of tasks in the "Informatik Biber" or from the Bebras are a suitable basis for identifying and designing new tasks for testing cognitive competences. More precisely, we initially selected 39 tasks from the "Informatik Biber", taking into account the respective target group, level of difficulty and form (multiple choice or possibility to transfer to multiple choice, to allow for automated evaluation).

For a systematic selection of suitable tasks from this initial set, we have to keep in mind which kind of insight we want to gain from the tests. As stated before, we want to assess our students' skill level in selected cognitive competences. Therefore, we have to identify which competences are involved in solving the different tasks, as well as the level of expertise required in each competence, respectively. Furthermore, we want to test the students' ability to employ several cognitive competences at the same time and in an integrated way.

Classifying a task. Firstly, we analyse and classify the collected tasks from different perspectives. Classifications from literature that we analysed (cf. section 3) focus on learning Computer Science related knowledge in combination with specific programming tasks or at least the programming context, and thus do not suit our purpose. Therefore, we define our own classification schema. In a first step, we specify the IT Concept the task focuses on, as well as its level of difficulty (see Table 3). Note that the difficulty is specified with respect to high school students as intended target group; i.e. professionals might find tasks classified as "tough" easier as a student.

Table 3. IT concepts represented in each task and level of difficulty [8].

Name (Year)	IT Concept	Level of difficulty
Coding Pictures (2010)	coding, data compression, data representation	easy
Back Side (2013)	implications, logic	tough
Flow Diagram (2013)	flow chart	medium (tough for grades 9 - 10)
Magical Machine (2013)	Petri net	tough
Minimal Duration of Studies (2011)	network diagram, path finding	tough
File Salad (2010)	search pattern, regular expressions	medium
RAID (2013)	RAID technology	tough
Bebrocarina (2012)	coding, data representation	easy
Loophole in the Dark (2011)	pledge-algorithm	medium

Secondly, for each task we analysed which cognitive competences are essential for solving it, and on which level each of these competences is required *(easy – medium – tough*). For this analysis, we use *think aloud* [18] as well as *decoding* [19] methods, to make the process of solving the task visible. Then, all steps of the solution process are mapped to the cognitive competences specified in Table 2.

In the following, as an example we describe this analysis process for a selected task (from [8, pp. 7, year 2010], see Fig. 3; original description in German). The picture on the left side of the Figure is encoded by a program. For each row of the picture, the resulting code is depicted on the right side. However, the code in the third row is lost. Which code is the correct one for the third row? Available alternatives are aobobicio, bodiao (correct), bocibo, oociaio and no answer.

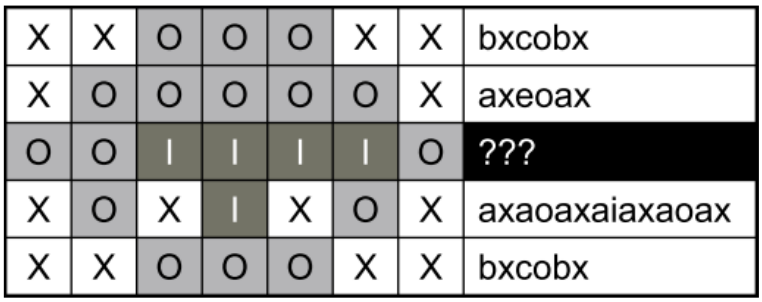

Fig. 3. Example for a task from the "Informatik Biber" [8, pp. 7, year 2010]

Several test persons were observed in their attempts to solve this task. **Error! Reference source not found.** lists the steps of an attempt that lead to the correct result, whereas the approach documented in Table 5 was unsuccessful.

Table 4. Steps of a test person's think aloud process that lead to the correct result.

	Step
	Picture and code contain the same characters (x, o, i)
	All other characters like a and b in the code must carry some information
	Both X in the picture have the same color and might be a unit
	Try find correlation between _ _XX and code bxcobx as well as _ _ XX and code bxcobx
	Sequence XX twice in the picture and bx twice in the code
	From XX follows bx
	Test hypothesis with last row → works
	Both bx as well as XX can be 'removed' for further work
	Remains OOO and co for the first line
	b → 2, c → 3, a → 1
	Character in front of x, o, i represents the amount of signs X, O, I in the code until it changes and it matches the number to the character in the alphabet on the position equal to the amount
	Code consists of tuple (amount of same character, the character itself)
	Test hypothesis with all examples → works
	Apply rule to third and missing row
	Check if result is a possible answer

Table 5. Steps of a test person's think aloud process that lead to a false result.

	Step
	Picture has 7 columns whereas the code has 6 and more characters
	Match of one sign in the picture to a character in the code is not possible
	Row 1 and 5 are identical
	Difference between code in row 1 and 2, 2 has no b
	Conclusion XX equals b
	Correlation discovered that 2X → b
	What does the o in the code mean?
	What does the O in the picture stand for?
	Correlation XX → b, X → a
	Solution cannot start with b or a as the picture shows no X
	Elimination process on the answer possibilities leaves only one choice oociaio (wrong answer)

As a next step, we analysed the documented approaches and identified which of the cognitive competences under consideration are used, and on which skill level.

The task does not provide any detailed instructions that should be followed. Thus, it does not require a systematic way of working, as defined in [2]. Different correlations between picture and code, such as x in the code representing an X in the picture, are revealed by logical thinking, as they represent cause and effect. Furthermore, participants need to identify subsets in the code and matching subsets in the picture, e.g. XX → bx. Hence, analytical thinking is necessary to solve the task. The interrelation of code and picture can be described by a general law that can be derived from the given setting, which requires abstract thinking. As it is not necessary to have an overall view to solve this task, holistic thinking is not required.

In summary, we identified that logical thinking, abstract thinking and analytic thinking are required to solve this tasks. All these cognitive competences are needed on level *easy*.

Participants who came up with a wrong solution did not see the correlation between the picture and the code. Rather, they were talking of circle and o instead of calling both symbols an "o". Hence, they had a mistake in their logical thinking process. Moreover, the participants did not identify the matching subsets. Thus, their analytical thinking was not sufficient in this process. Furthermore, they did not develop a correct general law. If they had tried to apply the law on other rows, they would have recognised the mistake. Instead, they just executed a procedure of exclusion on the answer possibilities with an unproven hypothesis.

Summing up, test persons that were not able to solve this task correctly did not show any of the required competences. Note that the steps in the solution process seem to be intertwined, as we assume that getting one step correctly might lead to a different process, and thus to another solution as well.

Selecting Appropriate Tasks for the Test. In a similar way, we analysed and classified all the tasks in our initial set, and documented which cognitive competences are required to solve them. On this basis, we selected suitable tasks to include them

into our final test. In our selection process, we applied the following criteria for choosing tasks:

- Only small number of competences necessary to solve the task (4 or less)
- Test each competence at least twice
- Test different levels of one competence
- Allow proof of several assumptions, like logical thinking combined with systematic is easier than logical thinking on its own
- Provide a German as well as an English task for the same set of competences, to evaluate English reading ability in correlation with cognitive tasks

On the other hand, we explicitly excluded tasks that showed at least one of the following characteristics:

- Difficult to decide which competences are involved
- Has at least two distinct solution processes (often differing between experts and novices) using different competences, as we cannot know from the result which process was performed to solve the task; thus we cannot deduce any information on the competences involved
- Eliminate tasks with "fear factors", such as unknown and big formula
- Programming knowledge or similar professional skills are required for solving the task

From our initial set of 39 tasks, 11 made it into the final test. Table 6 documents the cognitive competences that are necessary to solve them, as well as the required competence level.

Table 6. Relating tasks and cognitive competences required for solving them

Name (Year)	systematic	logically thinking	abstract thinking	thinking concretely	analytic	thinking holistically
Coding Pictures (2010)	-	easy	easy	-	easy	-
Back Side (2013)	-	tough	-	-	-	-
Flow Diagram (2013)	-	-	-	easy	-	-
Magical Machine (2013)	tough	medium	-	easy	-	easy
Minimal Duration of Studies (2011)	easy	-	-	-	-	easy
File Salad (2010)	-	easy	-	medium	-	-
RAID (2013)	easy	tough	-	easy	-	easy
Bebrocarina (2012)	easy	easy	-	medium	-	-
Loophole in the Dark (2011)	medium	easy	-	medium	-	-
Bongard 1	medium	-	medium	-	-	-
Bongard 2	medium	-	medium	-	-	-

5 First Results

Each student that takes the test receives a personal result sheet. It shows a radar chart that compares the lecturers' expectations (*Model Student*) with the self-assessed level of competences. Furthermore, it depicts a table stating the points a student achieved in the knowledge test, as opposed to the desired and the maximum number of points. Finally, students find a detailed evaluation of the tasks that test their cognitive competences. Each task is listed with its required competences, and the information whether the student answered it correctly or not (see Fig. 4.).

Student Report

Knowledge Test – Overview

	Achieved	Target	Maximum
Computer Knowledge	6	4	8
Cognitive Competences	9	6	11
Basic Math	10	9	13
Total	26	19	34

Explanation
- Achieved: Points you achieved in the test.
- Target: Points lecturers would expect from first-semester students.
- Maximum: Points you can achieve at most.

Knowledge Test – Details

Task (Year)	Result	systematic	logically thinking	abstract thinking	thinking concretely	analytic	thinking holistically
Coding Pictures (2010)	X	-	easy	easy	-	easy	-
Back Side (2013)	-	-	tough	-	-	-	-
Flow Diagram (2013)	X	-	-	-	easy	-	-
Magical Machine (2013)	-	tough	medium	-	easy	-	easy
Minimal Duration of Studies (2011)	-	easy	-	-	-	-	easy
File Salad (2010)	X	-	easy	-	medium	-	-
RAID (2013)	X	easy	tough	-	easy	-	easy
Bebrocarina (2012)	X	easy	easy	-	medium	-	-
Loophole in the Dark (2011)	X	medium	easy	-	medium	-	-
Bongard 1	X	medium	-	medium	-	-	-
Bongard 2	X	medium	-	medium	-	-	-

Explanation X: correct; -: wrong; difficulty of task: easy, medium, tough

Fig. 4. Example evaluation for a student.

In order to be able to respond to their students' needs, every lecturer receives an overall evaluation for his/her study group. This evaluation contains the following information:

- Distribution according to university entrance diploma
- Distribution according to whether students completed an apprenticeship or not
- Overall radar chart with the median result of the self-assessment [2]
- Distribution of points in the knowledge test
- Distribution of points in specific parts of the knowledge test
- Percentage of students who passed / failed the threshold of the knowledge test

To gain an overview of the distribution of answers, we looked at the percentage of students per answer possibility. There are some tasks that seem to be easy, as more than 75% of the students gave the correct answer. However, there are also tasks that

just 25% got right. It is also interesting that on average, 29% of the students stated that they have no clue how to solve the Code task which we described in detail in section 0, Fig. 3. A detailed analysis of the Code task is depicted in Table 7.

Table 7. Results of the code task.

aobobicio	bodiao	bocibo	oociaio	n.A.	Year
5.6%	29.1%	7.7%	18.9%	34.0%	2014/15
7.2%	31.7%	8.6%	20.8%	26.2%	2015/16
8.0%	45.2%	2.7%	14.1%	26.2%	2016/17
6.9%	***35.4%***	***6.2%***	***17.8%***	***29.1%***	***Average***

These results, compared with the information gained by the decoding processes (cf. Table 6), suggest that there are some competences that students possess right from the beginning, such as thinking concretely. For example, the task *Flow Diagram* solely requires concrete thinking, and over 75% of the students got this correctly. This was also the highest score in the test part that focusses on cognitive competences. In contrast to this, other competences such as logical and abstract thinking seem to be missing, as less students have tasks correct that require these competences.

6 Conclusion and Future Work

Our first results support the hypothesis from Thurner et al. [3] that specific cognitive competences are not sufficiently developed in many first-year students in Computer Science or related topics. Systematically selected tasks help to diagnose the students' initial competence profile.

The feedback of the students on the test is positive, as they get an impression of the professional mind-set required in Computer Science, right from the beginning of their studies. Furthermore, they find out about their strengths and deficits early in their study process, especially in maths, without having to fail an exam beforehand. This offers the opportunity to work on their deficits before students are left behind.

With the developed test, lecturers get an early feedback about the strengths and deficits of their cohort, too. Thus, they can take these into account during their lectures. We distinguish between competences that are not vital for the topics of the introductory classes, and competences that are essential for their successful completion. An example from our software development class (CS1) is, that we phased out programming tasks that require elementary algebra, as students often struggle with it, like operations on rationales, and introduced more tangible examples.

A first glance on the data suggests that there exist competences which influence each other. This should be investigated in a next step. Furthermore, we want to analyse whether educational qualification, gender or other factors influence the development of specific competences or not.

The insights we gain from this analysis will influence our teaching in the introductory courses for maths and software development. New teaching units can be designed, focussing on competences that have to be developed.

7 References

[1] V. Thurner, A. C. Böttcher and A. Kämper, "Identifying Base Competencies as Prerequisites for Software Engineering Education," in Global Engineering Education Conference (EDUCON), 2014 IEEE, 2014.

[2] D. Zehetmeier, M. Kuhrmann, A. Böttcher, K. Schlierkamp and V. Thurner, "Self-Assessment of Freshmen Students' Base Competencies," in Global Engineering Education Conference (EDUCON), 2014 IEEE, 2014. https://doi.org/10.1109/EDUCON.2014.6826130

[3] V. Thurner and A. Böttcher, "Expectations and deficiencies in soft skills," in Global Engineering Education Conference (EDUCON), 2012 IEEE, 2012. https://doi.org/10.1109/EDUCON.2012.6201197

[4] E. Jung, Java 8 - Das Übungsbuch, MITP-Verlags GmbH & Co. KG, 2014.

[5] K. Sierra and B. Bates, Head first java, " O'Reilly Media, Inc.", 2005.

[6] P. Ackermann and L. Leowald, Schrödinger programmiert Java: das etwas andere Fachbuch, Galileo Press, 2014.

[7] S. Reges and M. Stepp, Building Java Programs, Pearson, 2014.

[8] BWINF, Gesellschaft für Informatik e.V., "Die "Bundesweiten Informatikwettbewerbe"".

[9] K. Goldstein and M. Scheerer, "Abstract and Concrete Behavior -- An Experimental Study With Special Tests.," Psychological Monographs, vol. 53, no. 2, 1941. https://doi.org/10.1037/h0093487

[10] W. Halstead and G. Settlage, "Grouping behavior of normal persons and persons with lesions of the brain," Archives of Neurology and Psychiatry, vol. 49, pp. 489--506. https://doi.org/10.1001/archneurpsyc.1943.02290160011001

[11] V. Dagienė and G. Futschek, "Bebras International Contest on Informatics and Computer Literacy: Criteria for Good Tasks," in Proceedings of the 3rd International Conference on Informatics in Secondary Schools - Evolution and Perspectives: Informatics Education - Supporting Computational Thinking, Berlin, 2008. https://doi.org/10.1007/978-3-540-69924-8_2

[12] A. Ruf, M. Berges and P. Hubwieser, "Classification of Programming Tasks According to Required Skills and Knowledge Representation," in International Conference on Informatics in Schools: Situation, Evolution, and Perspectives, 2015. https://doi.org/10.1007/978-3-319-25396-1_6

[13] A. L. Tharp, "Getting more oomph from programming exercises," in ACM SIGCSE Bulletin, 1981. https://doi.org/10.1145/953049.800968

[14] S. A. Hansen, "Analyzing programming projects," ACM SIGCSE Bulletin, vol. 41, pp. 377-381, 2009. https://doi.org/10.1145/1539024.1508999

[15] L. Layman, L. Williams and K. Slaten, "Note to self: make assignments meaningful," in ACM SIGCSE Bulletin, 2007. https://doi.org/10.1145/1227504.1227466

[16] B. C. Wilson, "Gender differences in types of assignments preferred: Implications for computer science instruction," Journal of Educational Computing Research, vol. 34, pp. 245-255, 2006. https://doi.org/10.2190/7FLU-VKJL-86RM-5RQG

[17] M. Opmanis, V. Dagienė and A. Truu., "Task Types at "Beaver" Contests," in Proceedings of the 2nd Int. Conference Informatics in Secondary Schools: Evolution and Perspectives, 2006.

[18] R. D. Pea, "Language-independent conceptual 'bugs' in novice programming," Journal of Educational Computing Research, vol. 2, pp. 25-36, 1986. https://doi.org/10.2190/689T-1R2A-X4W4-29J2

[19] D. Pace and J. K. Middendorf, Decoding the disciplines: Helping students learn disciplinary ways of thinking, Jossey-Bass, 2004.
[20] "Lernen: Lust oder Last," 2016.
[21] "Pulp Friction: Critical Reading Exercise".
[22] "Der große ZEIT-Mathetest," 2013.
[23] T. Richter, J. Naumann and H. Horz, "Eine revidierte Fassung des Inventars zur Computerbildung (INCOBI-R)," Zeitschrift für pädagogische Psychologie, 2010. https://doi.org/10.1024/1010-0652/a000002
[24] M. Bongard, Pattern recognition, J. K. Hawkins, Ed., New, York: Spartan Books, 1970.
[25] Süddeutsche Zeitung, "IQ Tester -- Der kostenlose IQ-Test online mit Sofortergebnis".
[26] S. Loyd and M. Gardner, Mathematische Rätsel und Spiele: Denksportaufgaben für kluge Köpfe. 283 Aufgaben und Lösungen, DuMont Buchverlag GmbH, 2003.

8 Authors

Axel Böttcher and **Veronika Thurner** are professors at Munich University of Applied Sciences, Department of Computer Science and Mathematics, teaching Software Engineering and related topics. Their current area of research is technical didactics and computer science education.

Sabine Hammer is a research assistant and didactic expert at Munich University of Applied Sciences, Department of Computer Science and Mathematics. She studied mathematics and physics for teaching profession, wrote her doctoral thesis in the area of mathematics education and has varied experience in the field of empirical education research.

Daniela Zehetmeier is a research assistant at Munich University of Applied Sciences, Department of Computer Science and Mathematics. She studied Computer Science and is currently working on her doctoral thesis in the field of teaching abstraction in the context of software development. Before starting her academic career, she gained work experience in several software projects.

This article is a revised version of a paper presented at the EDUCON2017 conference held in Athens, Greece, 25-28 April 2017. Article submitted 15 July 2017. Published as resubmitted by the authors 03 September 2017.