Design and First Results of a Psychometric Test for Measuring Basic Programming Abilities

Andreas Mühling  
TUM School of Education  
Technische Universität München  
Arcisstr. 21  
80333 München, Germany  
andreas.muehling@tum.de

Alexander Ruf  
TUM School of Education  
Technische Universität München  
Arcisstr. 21  
80333 München, Germany  
rufa@in.tum.de

Peter Hubwieser  
TUM School of Education  
Technische Universität München  
Arcisstr. 21  
80333 München, Germany  
peter.hubwieser@tum.de

ABSTRACT

We present the design of a test for measuring students’ abilities concerning the application of control structures. Validated test instruments are a valuable tool for the evaluation of teaching both in a research setting as well as in a classroom setting. Our test is based on item-response-theory, in particular the Rasch model, and comprises a set of items all following the same format and using a simple, artificial programming language. We field-tested and modified the instrument in four iterations using only small samples and special statistical methods instead of the large samples usually required for IRT models. After the fourth iteration, the test has now reached a usable state. Based on the results, we were able to identify two misconceptions that are occurring very frequently in our test population - students of grade 7 to 10 in secondary schools.

CCS Concepts

• Social and professional topics → Student assessment; 
• K-12 education;

Keywords

Algorithmic control structures; assessment; computer science education; computing concepts; latent-trait; novice programmer; program flow; psychometric test

1. INTRODUCTION

Research in Computer Science Education is still a rather young scientific endeavor. Fincher & Petre identified several areas of interest for future work in CSEd research [8], among them the areas of “students understanding” and “assessment”. Also, “[t]he study of cognitive aspects of learning to program or computer system comprehension has been and should be given attention” [12].

Particularly for beginning programmers, there is much literature about the success or inherent difficulties of educational approaches or teaching interventions [16]. “[T]here are obvious difficulties in empirically evaluating such courses” [12]. Often, the effect of such a design is determined based on pre- and post-test of the abilities or knowledge of the students. As long as the same test is administered on both occasions and as long as a retention effect of the test itself can be disregarded, such a design may very well yield valid results. However, it would be much more favorable to have a validated psychometric test (instrument) that is publicly available and can be used “off the shelf” for measuring the effects of an educational intervention. Such instruments exist in other areas already, for example for measuring intelligence or motivation (e.g. [19]). As Buffum et al. state: “developing assessments of student learning is an urgent area of need for the relatively young computer science education community as it advances toward the ranks of more mature disciplines such as physics that have established standardized assessments over time” [4]. The need for more data-driven research has also been noted in the CSEd community before [24].

Classroom teachers, as well as researchers, can greatly benefit from a set of validated instruments. Teachers may use them as a way to generate feedback [28] that is independent of exams that are often designed by the teachers themselves and therefore subject to personal influences that affect the results in unpredictable ways. Researchers can use the instruments to evaluate the effectiveness of interventions, teaching approaches, or even curricula.

At the moment, we are still lacking a tried and true set of such instruments designed specifically for CSEd. We were faced with this very situation when conducting a study comparing the effect of two different software tools for beginning programmers [23]: Based on the idea of using existing questions of the Bebras contest [6] to form our pre- and post-test, we set out to design, validate and field-test an instrument for testing students’ abilities concerning the basic programming concepts of sequence, conditional statement, and loops. This is inline with a trend in educational research towards validated instruments that are measuring only a very specific aspect of a subject area [7].

The contribution of this paper is twofold: First, it presents the first steps of the design and validation of an instrument for measuring students’ abilities concerning the application of control structures. The instrument is now in a preliminary...
state, however it is usable and has already yielded some interesting results. Second, we present our design approach and the methods used in order to encourage the replication for other areas of interest by the CSEd community.

2. RELATED WORK

There are many different studies who are employing some form of test to evaluate their design, e.g. [3].

Sudol & Studer present a comprehensive evaluation of several items from an assessment as an example of the type of information that Item Response Theory (IRT) can provide [24].

As described in her dissertation [25] and her paper [26], Tew developed the Foundational CS1 (FCS1) Assessment Instrument, claiming that this is “the first assessment instrument for introductory computer science concepts that is applicable across a variety of current pedagogies and programming languages”. She applied well-approved methods from educational and psychological test development and adapted these to fit the context of CS education. Additionally she conducted a large scale empirical study to demonstrate that pseudo-code was an appropriate mechanism for achieving programming language independence and validated her instrument.

Buffum et al. developed a seven-step approach to designing, iteratively refining, and validating knowledge assessment instruments designed to measure the efficacy of teaching interventions [4]. Additionally, they explain how this process can be applied in practice by presenting a three-year project to implement a game-based learning environment for middle school computer science. The paper can be regarded as a practical guide for adapting widely accepted psychometric practices to the development and validation of computer science knowledge assessments to support research.

Winters & Payne describe how an IRT (computer-)based assessment can be used for instant feedback in CS classes ([28] and [29]).

The design of a test based on item response theory is closely related to competencies, because typically these are measured in a very similar way. Currently, our research is not based on a specific competency-model. Instead, we are concerned with designing an instrument for a very specific part of computer science/programming. However, there are others who are investigating this area in a “top-down” fashion, e.g. the MoKoM project [15].

3. ITEM-RESPONSE-THEORY

The Rasch model [22] has been successfully applied in educational research for decades (cf. [9]). It is based on item-response-theory. While classical test theory assumes that a test measures some directly observable trait - in the presence of measurement errors and noise - IRT specifically assumes that the trait is by itself unsolvable (latent) but the answers to a set of items can be taken as indicators of this trait. The latent trait is then described by some form of (statistical) model. IRT approaches have been well received especially in educational measurements [7]. One advantage of the IRT approach is that it is possible to explicitly test the model fit based on the results. For the Rasch model, the probability of a person solving a (dichotomous) item x correctly is dependent on the person’s ability β and the item’s difficulty α, which are both measured as a single numeric value on the same scale (usually ranging from -4 to 4). The probability function is (cf. [2]):

\[
P(x = 1 | \alpha, \beta) = \frac{\exp(\alpha + \beta)}{1 + \exp(\alpha + \beta)}
\]

(1)

If a person’s ability equals the item’s difficulty, the probability of this person correctly solving this item is 0.5. The more the ability exceeds the difficulty, the more this probability approaches 1 (or 0 if the difficulty exceeds the ability). The typical “characteristic curves” of this dependency can be seen in the experimental section - e.g. in Fig. 5.1. The Rasch model is based on certain, rather strict, premises that must be met by the data for the model to be a good and valid fit or descriptor/predictor. These preconditions are (cf. [15]):

1. **Local stochastic independence**: The probability of solving an item is independent of the probability of solving any other item given the latent-trait.

2. **Specific objectivity**: Each pair of items with fixed difficulties can be used to compare two persons and each pair of persons with fixed abilities can be used to compare two items. In other words, subgroups of items or persons are not showing characteristics different from the whole set. In particular, this precondition prevents differential item functioning.

3. **Unidimensionality**: All items of the test are measuring the same, single latent construct (trait).

One of the nice properties of the Rasch model is its simplicity. In particular, the number of correctly solved items of a person is sufficient information for this person’s ability and the total number of persons that solved an item correctly is sufficient information for this item’s difficulty (cf. [2]).

To make sure that the Rasch preconditions are met, there are several test that can be applied in posterior to check whether or not the Rasch model is a good fit. Well known tests are Andersen’s LR test [1], the Wald test [27], or the Martin-Löf [11] test. All of these test, however, require a large sample to be collected for testing purposes [13]. Pearson’s χ² test, as described in [2], for example, requires about 5 · 2^n samples for n items to be applied safely.

For smaller samples, there are several non-parametric, “quasi-exact” model test described in [13] and [21]. The tests are based on a bootstrapping methodology. Instead of using a large enough result matrix directly, a large number of matrices with the same rank sums as the (too small) result matrix derived from the test are generated. Then, the test statistic evaluate how well the actual matrix fits the model in light of all the generated ones - as all matrices with equal rank sums are equally probable under the Rasch model. Among the tests that are possible, are the following four that are used in this paper (cf. [13]):

\( T_1 \): Tests whether or not the inter-item covariance is too high (violation of stochastic independence).

\( T_{1n} \): Tests whether or not the inter-item covariance is too low (violation of unidimensionality).

\( T_{10} \): Tests whether or not subgroups of the responses show different traits than the whole set (violation of specific objectivity).
If this is the case or not can be the same, code-tracing skills also affect code-writing skills. Also, as reported in [14], even though they are clearly not task which makes the design of a test instrument worthwhile. Abstractions available in Nassi-Shneiderman diagrams [30]. Abstraction for procedural programming mostly means the ability to create and use composite types and subroutines - which is also an explicit operation in the Nassi-Schneiderman diagrams. We left out this aspect and concentrated on students’ ability to execute a given program based on the above control structures, which leads to the following set of programming concepts/operations that our test should cover:

- Sequencing: doing one step after another.
- Selection (if-then-else): doing either one thing or another.
- Repetition (Iterative loops or recursion).
- Language constructs that support abstraction: wrapping up a [...]"

This covers all the typical control structures found in procedural programming and closely follows the set of operations available in Nassi-Shneiderman diagrams [30]. Abstraction for procedural programming mostly means the ability to create and use composite types and subroutines - which is also an explicit operation in the Nassi-Schneiderman diagrams. We left out this aspect and concentrated on students’ ability to execute a given program based on the above control structures, which leads to the following set of programming concepts/operations that our test should cover:

- sequence of operations,
- conditional statement with and without alternative,
- loop with fixed number of iterations,
- loop with exit condition (conditional loop),
- and the nesting of these structures to create more complex programs.

We are restricting ourselves to the application of these structures, i.e. the correct execution of a given program. In the CS-specific taxonomy of learning objectives of Fuller et al. [10], this corresponds to the Apply/Understand cell of the Producing/Interpreting dimension. The reasoning for this restriction is that 1) this increases the likelihood of our test measuring a single, latent trait, 2) we want to start with a clearly defined, small aspect of the whole domain of programming or computer science, and 3) the experimental results support our assumption that for our designated target population reaching this cell is already a non-trivial task which makes the design of a test instrument worthwhile. Also, as reported in [14], even though they are clearly not the same, code-tracing skills also affect code-writing skills.

If we are striving to use a regular Rasch model, the abilities that we are trying to measure must form a homogeneous psychological (latent) trait. If this is the case or not can be determined with hindsight as soon as data has been collected - in our case the results have been positive so far. Additionally, our a priori reasoning for why we expect this to be the case is twofold:

First, unidimensionality of a latent trait in educational settings, according to [9], means that “each single problem can be solved by applying a limited number of rules (operations).” This is the case here, since executing a program is by its very definition the application of a limited number of operations - the control structures.

Second, in a different study the conceptual knowledge underlying our proposed latent-trait has been investigated with concept maps. A detailed description of the research setting and experimental results are given in [17] and [18]. The concept maps of a large group of beginning CS students have been analyzed regarding the predominant structural similarities and differences in their knowledge configuration. This has been done by - among other methods - analyzing the (graph theoretic) communities of a graph derived from aggregating the set of concept maps. It shows that the basic concepts of programming that we are concerned with here are all ending up in the same community. While conceptual knowledge as measured by concept maps is only a weak indicator of procedural knowledge, this at least indicates that these concepts are closely enough related to be “chunked” and to form the knowledge basis for a specific ability that forms a latent psychological trait.

We can check our assumption with a statistical analysis of the first field-test, by checking whether or not the tests for unidimensionality are pointing to problem spots.

Our basic design methodology follows the paradigm of evidence centered design (ECD). “The basic principle of ECD is that the items should be carefully designed to provide as efficiently and completely as possible the necessary evidence required to satisfy the assessment purpose” [7]. To this end, we were trying to (re-)design our test based on our initial goal as well as preliminary results, but also in such a way that we are still covering all of the constructs and that the test itself is well-designed concerning the number of items and their sequence - based on the personal experience as a teacher. Our basic design cycle for the instrument is as follows, similar to [4]:

1. Create or modify items to cover all of the basic operations of program flow.
2. Collect data from a relatively small set of students.
3. Analyze the data, in particular the model-fit of the Rasch model.
4. Identify items that are posing problems and should be modified or removed in the next iteration.

The target audience - for the time being - are students of Bavarian secondary schools. Later on, we hope to have a test that is also applicable in a bigger variety of contexts. Concerning the item format, we were looking for one that is:

- inherently dichotomous,
- easily understandable for our designated target group, and
- making it easy to modify items or create new ones.
The format that we settled on is based on a grid world in which a figure can be moved by a set of simple programming constructs. Each item contains a situation, i.e. a grid and a placement of the figure in the grid, and a short program code. The answer for each item is the placement of the figure after the given program has run. This consists of the coordinates of the cell and the orientation of the figure itself. Figure 4 shows one of the items as an example. The setup is reminiscent of “Karel the Robot” [20]. This is on purpose, since in Bavarian secondary schools Karel the robot is often used as a tool for the introduction to programming. Clearly, we want to have versions of the instrument in the future that are not relying on this particular context, too. We used our own version of the syntax for the program code to make the test self-contained, based on the findings of [25]. The semantic of each syntactic element is described briefly before the first appearance. We expect that this will not pose problems to students who have worked with a similar kind of “programming language” before. Many elements are similar or identical to Karel the robot.

The first two tests are each resulting in item pairs that violate the tests, the latter two test results in a “global” p-value that tests the null hypothesis that the Rasch model is a valid fit for the data. It is noteworthy, that removing items from the test can alter the test statistics of all other item pairs as well since the row sums of the test matrix changes, which also affects the expected number of co-occurrences. Therefore, removing all items that are posing problems in each of the tests is not always necessary. It is often possible to only remove a subset of these items and still arrive at a set that passes all tests.

The set of items that we used throughout the four iterations and their evolution over time can be seen in Table 2. The name of an item indicates the content:

<table>
<thead>
<tr>
<th>Iteration</th>
<th>School</th>
<th>Participants</th>
<th>Grades</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>A</td>
<td>32</td>
<td>10th</td>
</tr>
<tr>
<td>2</td>
<td>A</td>
<td>119</td>
<td>8th and 9th</td>
</tr>
<tr>
<td>3</td>
<td>A</td>
<td>103</td>
<td>7th</td>
</tr>
<tr>
<td>4</td>
<td>B</td>
<td>91</td>
<td>9th</td>
</tr>
</tbody>
</table>

For each iteration the following statistical tests were conducted, as described above:

**Inc** Test for increased inter-item correlation. If the observed correlation is too high, the assumption of local stochastic independence may be violated.

**Dec** Test for decreased inter-item correlation. If the observed correlation is too low, the assumption of unidimensionality is most likely violated.

**Inv** Test for subgroup invariance. Based on the assumption of specific objectivity, the estimated parameters of the Rasch model must not change significantly when using only a subgroup of the responses. This test checks this assumption.

**Obs** Deviations between observed and expected response patterns. If the observed response pattern deviate too much from the ones expected under the assumptions of a valid Rasch model, the model is not a good fit for the data.

The set of items that we used throughout the four iterations and their evolution over time can be seen in Table 2. The name of an item indicates the content:

**SO** Sequence of simple statements

**CSx** Conditional statement (CS1 with and CS2 without alternative, and CS3 both with- and without alternative)

**LFx** Loop with fixed number of iterations

**LCx** Loop with exit condition

**NCx** Nesting of control structures (loop in loop for NC1 and conditional statement in loop for NC2)

Each item receives a single dichotomous score. Sub-items like LF1a and LF1b indicate, that the items were “belonging together” from a content perspective, i.e. were presented on the same page of the test or were based upon each other. However, over the course of our field-tests all such interactions between items have been removed. Also, the response
format for each items has been (nearly) always the same: the students give the position and direction of the robot after the given code has been executed on a given scenario. The item is considered correct if and only if both position and direction are correct. The only exceptions are items LC1b and LC2c: Here, the students had to give a written explanation for a phenomenon. However, both items were removed after the first iteration. Item CS3 consists of two different scenarios and two different responses and is scored correctly if and only if both parts are answered correctly. While some items remained constants throughout all cycles, others were removed due to problematic test results and some were modified. The next subsections give an overview over the results and subsequent decisions of each iteration.

5.1 First Iteration
The original test contained 10 items, which were presented in the following order: SO1 - LF1a - LF1b - NC1 - LC1a - LC1b - LC2a - LC2b - LC2c - CS1

The statistical tests gave the following results:

Inc The two item-pairs LC1b-LC2a and LC2b-LC2c are showing too high a correlation.

Dec The correlation is too low for the pairs SO-NC1 and NC1-LC2a.

Inv With a p-value of 0.22, the null hypothesis of model fit holds.

Obs The p-value of this test is 0.16, again not small enough to discard the null hypothesis of model fit.

The high correlations are partly explainable: LC2a used a “while (x)...end” type of loop and LC2b used a “do...while (x)” type of loop. LC2c asked for the difference between the two loops. Obviously, only those students who understood the difference between the loops were able to correctly execute the corresponding code in their mind. However, it is not obvious why LC1b and LC2a are showing a high correlation but not LC1a and LC2a - which are very similar.

There are several subsets of items that do pass all tests, though. Item LC2c has to be removed in all of these cases, so this is clearly an items that poses problems. Additionally, it is enough to remove item LC2a in order to pass all tests. For this combination, we can estimate the Rasch model parameters. Fig. 5.1 shows the resulting item difficulties and their characteristic curves. As we can see, overall the test is rather easy - as most of the ICC curves are placed in the negative part of the x-axis. Also, the two easiest items (SO1 and LF1a) have the same difficulty and the hardest item is LC2b. This makes sense, as the “do...while(x)” construct was new to the students.

5.2 Second Iteration
The second version of the test was again based on 10 items. Using the results of the first iteration, we were trying to eliminate the inter-item dependencies. To this end, several LCx items were removed and the LFx items were modified. Additionally, two new items were introduced in order to get the overall difficulty of items more evenly spread out. The order of the items was: SO1 - LF1a - LF1b - NC1 - LC1a - CS2 - NC2 - LC2a - LC2b - CS1

The statistical tests are giving the following results:

Obs The p-value is 0.12, not low enough to reject the Rasch Model.

Inc The item pairs SO-LC1a, LF1a-CS2, LF2b-NC1, and NC2-CS1 show a higher correlation than expected.

Dec The item pairs LC1a-CS2 and CS2-LC2b are showing a lower correlation than expected.

The overall p-value for model fit is 0.004, therefore, the hypothesis of model-fit must be rejected.

On the first glance, there are more items that are posing problems in this test than in the first iteration. However, removing the items LC1a and CS2 is enough to have the reduced set of items pass all tests. Fig. 5.2 shows how the model parameters were estimated in this case.

We also noticed for some items that there were typical errors that the students made. For example, for item CS2, many students gave a wrong answer that would have been right if the conditional statement had been a loop with exit condition. Since the item that directly precedes CS2 is exactly such a loop and the two constructs look rather similar, syntactically, we suspected that the problems with item CS2 were due to careless errors of the students.

5.3 Third Iteration
Based on the prior experience, we were trying to consolidate our test so that each different control structure still appears, but overall there are as few items as possible. This was done in order to decrease the amount of data that is eventually necessary to validate the test in the standard fashion. Therefore, only one of the LCx items was kept. We decided to keep item CS2 of the second iteration but move the preceding item to a different location. We did remove item LF1a which has caused problem together with item CS2 and has also been somewhat redundant for the
Table 2: The single iterations of our test and how the items evolved over time. A * denotes that the item was rephrased or otherwise slightly modified from the original version. A ** denotes that the modified version has been modified again.

<table>
<thead>
<tr>
<th>Iteration</th>
<th>SO1</th>
<th>CS1</th>
<th>CS2</th>
<th>CS3</th>
<th>LF1a</th>
<th>LF1b</th>
<th>LC1a</th>
<th>LC1b</th>
<th>LC2a</th>
<th>LC2b</th>
<th>LC2c</th>
<th>LC3</th>
<th>NC1</th>
<th>NC2</th>
<th>Sum</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>X</td>
<td></td>
<td>X</td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td>10</td>
</tr>
<tr>
<td>2</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td>*</td>
<td>*</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td>X</td>
<td></td>
<td>10</td>
</tr>
<tr>
<td>3</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td>*</td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td>*</td>
<td></td>
<td>7</td>
</tr>
<tr>
<td>4</td>
<td>X</td>
<td></td>
<td>*</td>
<td></td>
<td>X</td>
<td>*</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td>**</td>
<td>8</td>
</tr>
</tbody>
</table>

The mean difficulty over all seven items is -0.7 - which corresponds to the visual interpretation of the items “leaning to the left side” of the x-axis a bit. When comparing the normal Rasch model to a two parameter latent-trait model, we find the AIC value to favor the simpler Rasch model slightly. Using a two parameter model, we can visually check whether the fixed slopes for the ICCs that the Rasch model uses - a consequence of specific objectivity - are a good fit. The slope is the second parameter in the two parameter model, i.e. it is estimated for each item separately based on the data. It shows that item LC1a has a very shallow slope which leads to crossings between the curves - a situation which is generally to be avoided as it means that the difficulty of an item is dependent on the ability of a person taking the test - a violation of specific objectivity. However, this is the only item that shows a clear deviation in slope. Our suspicion is that this deviation results from the specific code of item LC1a. The code lends itself towards a direct “guessing” of the final position of the figure, which often turns out to be wrong (loop was stopped one iteration to soon). In this case, the item would not measure the students ability to execute a loop correctly but instead their ability to intuitively visualize the result. In general, too shallow slopes are an indication that the item is not measuring one specific construct (cf. [24]).

5.4 Fourth Iteration

For the final, fourth iteration, we kept almost everything the same, since the third iteration proved to be (almost) a well designed test already. The only problem spots that remained are the CS2 item which is frequently solved incorrectly due to a careless error and the very shallow slope of item LC1a. Therefore, we introduced a new item CS3. Also, we decided to change our syntax such that the conditional statement no longer starts with the same letter as the loop: Previously, the loop started with “wiederhole” and the conditional statement started with “wenn”. We changed “wenn” to “falls” (which is also the German word used in the Scratch syntax). This change also affects item NC2. Additionally, we introduced a new item for the conditional loop in order to see whether or not this is performing better than LC1a, which was also kept in the test for comparison. The new item LC3 is exactly like item LC1a but with a different, more complex code that - hopefully - forces the students to actually trace the code in their minds instead of incorrectly “seeing” the result. The order of items was: SO1 - LF1b - NC1 - LC1a - LC3 - CS3 - NC2 - CS1

In this case, the evaluation showed that only item CS1 posed a problem for this particular test because it was too easy. When removing it and keeping item LC3 instead of LC1a - which was the purpose of testing both, all tests pass.
Also, CS1 is redundant in the sense that CS3 tests the same control structure. This means that the fourth iteration finally yields a set of items that encompass all control structures and pass all tests - with high p-values of 0.63 and 0.93 for the Inv and Obs tests.

Again, the model with only one parameter is favored in comparison to a two-parameter model by the AIC and BIC values. The ICC curves - as shown in Fig. 5(a) show that each item has a distinct difficulty - the overall test is still a bit too easy, though. The slopes of the ICC curves in the 2 parameter model, as shown in Fig. 5(b) are - basically - non-intersecting, only for very low abilities where guessing may play a large role in our small sample. Clearly, SO1 is a very easy item that also separates very strictly. Such an item is sometimes called a “Guttman item” (cf. [24]).

5.5 Discussion

The test in its fourth iteration presents a ranking of difficulty of the different aspects of control structures. Unsurprisingly, the sequence of simple statements is the easiest item. However, the rest of the ranking is not as obvious. A loop with a fixed number of iterations is easier than anything with conditions. A simple conditional statement is easier than a loop with exit condition. And, perhaps most surprisingly, the nesting of control structures is also easier than a loop with exit condition - note that the items NC1 and NC2 both use a loop with a fixed number of iterations as the outer structure. By taking a closer look at the actual responses, we were able to identify two common misconceptions that may have impact on the observed difficulty of the items.

Loop with exit condition Many students seem to think that the loop stops as soon as the exit condition is false - even if this happens in the middle of the loop. In this case, the students simply skip the rest of the loop-code. About 30% of the students showed this misconception in the last iteration.

Conditional statement A conditional statement without an enclosing loop proved to be difficult for the students. This is something that they most probably haven’t seen in their lessons, as a single conditional statement in such a scenario is rather uninteresting. What happens is that the students then seem to be implicitly assuming that the conditional statement is actually a repetition that they are executing until the condition is false.

Even though our final interest lies on a standardized instrument that can be applied for a summative assessment and - more importantly - the comparison of groups, a closer inspection of the answers that the students are giving is also a powerful way of identifying misconceptions. In a second step, items can be constructed in ways to explicitly test for the occurrence of these misconceptions. This is in line with current trends in educational research: “There is rapidly emerging a powerful need, and demand, for tests designed to formatively assess an appropriately chosen moderate number of relatively fine-grained chunks of knowledge in major subject or important cognitively defined areas. By ‘formative’ is meant that the results of the assessment are used to directly support teaching and learning” [7].

When taking items SO1, CS1, LC1a, and NC1 - which have been present in every iteration, we are able to run the conventional Rasch model tests as the number of participants is - in total - enough for this subset of items. Both the Wald test and the LR test show, that item SO1 is too easy to be included in the test. The other three items are, however, valid under the assumptions of the Rasch model. This backs the validity of our design and is a promising result for the yet to be done larger field-tests for the final validation.

6. CONCLUSION AND FUTURE WORK

We presented the design of an instrument for measuring students’ abilities in executing control structures. The test has been designed in an iterative fashion and is based on item-response-theory. At the moment, we have strong indications that - after four iterations - the set of seven items in their current format works as a psychometric test\(^1\) and that the Rasch model is a valid descriptor for the latent trait. However, the final validation will have to be done in a larger field test that allows us to apply the typical statistical tests on the data.

Future work focuses on two aspects: Finishing the work on this particular instrument as well as determining future directions for different tests as well.

Concerning this instrument, the items must be refined, more difficult items must be included and the set of items itself must grow in order to provide a more elaborate set of choices. Also, the test itself should be made more flexible. This means, among other aspects, designing and testing an English version of the instrument, so that it can be made available to a broader community of CSEd researchers. Also, we want to try out different contexts of the items in order to

\(^1\)If you are interested in the instrument, for the time being please contact the authors directly. In the future we are planning to make it publicly available in a suitable form.
make them less reminiscent of the Karel the Robot world. This can either mean generating a more abstract version of the test, which is not based upon any of the currently used tools for learning programming, or it can mean creating several versions of the instrument with each one oriented towards a particular type of these tools, like a Karel version and a Scratch version (in which, e.g., the code segments could be color-coded similar to the “puzzle-pieces” of Scratch). Also, there are several design choices that we made which may have to be examined in more detail. For example, it may be worthwhile to create a multiple-choice version of the instrument or to present it electronically instead of the current pen-and-paper format. All of these steps will require additional tests and validations however, so the iterations should be kept minimal. We hope that the CSEd community will be able to help us out with acquiring the necessary student data.

Concerning possible future directions for new tests, we clearly need to expand toward more elaborate cognitive processes than “just” application of control structures. The higher order cognitive processes - evaluating and creating - may therefore prove to be an interesting working ground for the future. Also, CS obviously is more than programming and programming is more than procedural programming with control structures. Object-oriented programming may, for example, provide a valuable context for future work.

7. REFERENCES


