Evaluation of Source Code with Item Response Theory

Marc Berges
TUM School of Education
Technische Universität München
Arcisstr. 21, 80333 München, Germany
berges@tum.de

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

ABSTRACT

The analysis of source code produced by novice programmers could provide interesting insights into their learning progress, particularly in introductory programming courses. Yet, as the programming ability of a person is assumed to be quite complex, it is not likely that it would be observable directly in its total. Instead, we regard those abilities as latent psychometric constructs and apply the methodology of item response theory (IRT) to assess their manifestations. In preparatory work, we had identified a list of items that represent the central concepts of object-oriented programming. In this paper we propose a methodology that allows the evaluation of coding abilities by analyzing the application of those concepts. We demonstrate this methodology by exemplarily analyzing source code that was produced during programming projects. The results provide interesting information about the difficulty of the concepts' application and the distribution of the respective coding abilities among the students.

Categories and Subject Descriptors

K.3.2 [**Computer and Information Science Education**]: Computer science education

Keywords

programming novices; code analysis; item-response theory

1. INTRODUCTION

A few years ago, we have introduced a preliminary programming course for the freshmen of computer science at our university [9]. These courses offered a broad research field for the investigation of source code that was written by students with well-known levels of programming experience in a closely controlled setting.

Although the direct evaluation of source code is difficult, several methodologies for assessing programming abilities from code have been presented, e.g. qualitative analysis of students' solutions [13], measures of code quality [8] or investigations about misconceptions [16].

Yet, due to its complexity, we do not assume that any programming ability could be measured in a direct way. Nevertheless, we could regard certain attributes of the source code as manifestations of latent psychometric constructs according to the principles of item response theory. More detailed, we could treat the application of certain structural elements like loops, conditional statements or inheritance operators as positive responses on certain items (e.g. "existence of loops"). In consequence, the probability of such positive responses in dependence on the item difficulty and the estimated person abilities could be described by certain psychometric models, e.g. the Rasch Model.

In the research project of this paper, we have applied the item response theory to evaluate coding abilities of freshmen on the basis of items gathered from their source code.

2. RELATED WORK

The analyzing and scoring of object-oriented code is a central topic in educational research ever since object-oriented programming has been taught in introductory courses in computer science.

For example, Börstler et al. proposed in [5] three categories for the evaluation of object-oriented example programs according to certain criteria like content, style or modeling. Sanders and Thomas introduced in [16] a check-list for scoring object-oriented programs by investigating concepts and misconceptions in object-oriented programming. In contrast, an automatic approach is conducted in [17], based on a framework for static code analysis of students' programs. For this, Truong et al. summarized common poor programming practices and common logical errors from literature. Additionally, they conducted a survey among teaching staff and students. The framework is working on a XML basis and enables the students to get feedback on their programs and rate them automatically.

Currently, there are several tools for educational purposes that assess code in an automatic way. Many work online like the one introduced in [18]. In [10] the scoring of the International Olympiad in Informatics is investigated. Kemkes et al. score the code with 1 if it runs successfully on a given set of input data. Otherwise it is scored with 0. In addition, they compared this methodology to other scoring schemes with the help of item response theory. Finally, they state that the simple scoring is the most applicable if an automatic scoring is needed.

One solution for the problem of a simple assessment system could be to investigate the syntax elements of a programming task with an open solution. A recent example is the research presented in [12]. The authors investigated differences in the correct solutions of students. For this purpose, they define a taxonomy that distinguishes the code according to structure, syntax and presentation. Structure means different control flow in the code, syntax means differences in the code with the same control flow structure. Finally, presentation means variation in the identifier names or number of white-spaces for example.

Some additional tools and algorithms for student assessment in computer science courses are presented in [19]. Winters and Payne conducted an item-response analysis on students' score data that was gathered during the semester. Additionally, existing tools were evaluated and the advantages of such a methodology for the process of identifying suitable items were pointed out.

3. BACKGROUND

In classical test theory (CTT), the construct of interest (e.g. student abilities) is considered to be measured directly by item scores, yet this might be error-prone. This straight-forward approach is not suitable for measuring such complex constructs as programming abilities. In contrast, the item response theory (IRT) regards the constructs of interest as latent psychometric constructs that cannot be measured directly. Nevertheless, the probability of correct answers depends from those constructs in a certain way: $P(X_{ik} = 1|\theta_i, \beta_k) = f(\theta_i, \beta_k)$, where θ_i is the parameter of person i, representing the manifestation of the psychometric construct, β_k the parameter of item k, representing its difficulty, and $f(\theta_i, \beta_k)$ a function that is determined by the psychometric model (e.g. the Rasch Model (RM), see below)

The latent construct might be uni- or multi-dimensional. Depending on the theoretically based assumptions of the structure of the psychometric construct, different mathematical models can be applied according to IRT.

A logistic model with one parameter (1pl) was introduced by Rasch [15]. The basic idea of the Rasch model is that the probability of solving an item is determined by the difference of a person's ability in the latent dimension θ and the difficulty of the item itself (β).

There are three model restrictions related to the Rasch model: the items have to be locally stochastically independent, fulfill specific objectivity, and measure one latent construct [1, pp. 20].

Basically, there are two approaches in the item analysis. First, a set of homogeneous items has to be found that is conducted to one latent psychometric construct. Second, the itemset has to be fit to a proper item response test or model, respectively. For these purposes, there are several tests.

The nonparametric tests for the Rasch model that are based on a Markov Chain Monte-Carlo algorithm are a suitable test framework for small sample sizes. Basically, the lack of data is reduced by simulating data matrices that have equal row and column sums than the estimated dataset. For that purpose two columns of the original matrix are randomly chosen. Afterwards, the rows with different values are randomly changed. This procedure leads to a new matrix with equal margins. As the algorithm needs the matrices to be independent of the initial matrix and to occur with the same probability, not all simulated matrices can be used for the calculation [11, pp. 4]. For the nonparametric tests a valid Rasch model is assumed. in a valid Rasch model the row and column sums are a sufficient statistic and because of that matrices with different values but equal row and column sums can be compared.

The general idea behind the nonparametric tests is the comparison of all item pairs [14]. A violation in homogeneity can be assumed if there are more unequal item pairs in the observed matrix than in the simulated ones. So, the test compares the observed test statistic with simulated test statistics [11]. For the characterization of a latent construct, a set of homogeneous items is necessary.

The local stochastic independence has to be proven for validating the Rasch model. On the one hand, the items are not allowed to be too similar. With regard to two items, this can be expressed by the number of equal response patterns. In contrast to the assumption of homogeneity, violation of the local stochastic independence is expressed by too many equal response patterns.

On the other hand, violation of the local stochastic independence can be a result of a learning effect within the items; participants answer on one item is dependent of the solution of another one. For this reason, only those patterns that are both answered correctly are summed. The local stochastic independence is important to find items for a proper test framework. For the definition of the latent dimension, dependent items can be useful as well.

If the ratio of the items' number and the number of participants is suitable, parametric tests can be applied. Generally, all these tests are based on the assumption that the model is valid in every sub population that is grouped by chance. So, if there is no significant difference in the parameters if the investigated population is separated into two groups the model is assumed to be valid. Especially, if there are other measures like previous knowledge or gender aspects, these criteria should be used for finding different groups [1, p. 63].

The first test is the Likelihood Ratio Test (LRT). For the test, the likelihood is calculated for the estimated parameters of each group and for the complete population as well. Afterwards, the likelihoods are compared. If the estimated parameters fit the model in the subgroups as well as in the complete population, the model is assumed to be valid. Otherwise it would be better to estimate the parameters for each single group and calculate separate models [7, pp. 86].

Another test for the model fitting that is based on the idea of separating the population into two groups is the Wald-test. In contrast to the LRT, the standard Wald-test is comparing the items directly. So, the standard Wald-test can identify items that violate the model assumptions [7, pp. 89].

4. COURSE DESIGN

In autumn 2008 we developed a course at our university for the freshmen studying Computer Science (CS) [9]. It takes place just before the first semester. All the students starting their studies are invited during their enrollment process. The participation is voluntarily. The necessity of the course results from the German lecture system at universities. During the semester there are mainly lectures with only very little time for practical experiences. Nevertheless, it is officially communicated that it would be possible to study CS without any prior programming knowledge, which implies that students without such prior knowledge should be accommodated somehow, too. Therefore we have developed and installed specific programming courses for this purpose that take place before the first semester.

All students were asked at the registration to self-assess their prior programming experience in one of three levels: (1) "I have no experience at all", (2) "I have already written programs", (3) "I have already written object-oriented programs".

Based on this information we tried to compose the groups – 6 to 15 participants each – as homogeneously as possible. The demands of the programs the students should realize differed according to their respective level of programming experience. The students of the first level were asked to program a "Mastermind" game. The groups of level 2 should realize a tool for managing results from a sports tournament (e.g. a football league). The groups of the 3rd level should program a version of the dice game "Yahzee".

The course took two and a half days. All participants worked on their own (instead of in teams), because we wanted to investigate the individual learning outcome. However, the students were actively encouraged to talk to each other. The material was presented in the form of worksheets that contained all the required information. Additionally, each group was coached by an experienced student as a peer tutor. The students were encouraged to approach the tasks in a self-directed learning process. So, the tutors were responsible for helping the students to understand the worksheets and tools, but they were advised not to give any assistance (or instruction) on programming itself. We suggested that the students use BlueJ due to the reasons mentioned by Bergin [4] for their first steps in programming. Towards the end of the course, they had the choice to switch over to Eclipse.

5. DATA COLLECTION

Based on a concept extraction from the materials of the underlying course, a list of 21 concepts was formed according to the method described in [3]: *access modifier (AM), array (AR), assignment (AG), association (AC), attribute (AT), class (CL), conditional statement (CS), constructor (CO), data encapsulation (DE), datatype (DT), inheritance (IN), initialization (IS), instance (IT), loop statement (LO), method (ME), object (OB), object orientation (OO), operator (OP), overloading (OV), parameter (PA),* and *state (ST)*. For the final concept list, four of them are eliminated because of different reasons. OO is eliminated because it is provided by design of Java. CL and DT are excluded because it cannot be distinguished between "implementation by the students" and "implementation forced by the IDE". Finally, IT is the same as OB and because of that only OB is included into the list.

For the calculation of a model in the item-response theory, a set of items is needed. In [3] the concepts mentioned above, were split up into observable items in the code. These items cover all observable aspects that are related to a specific concept. The list below presents these items, formulated as questions which can be 1-rated if the code answers the item with "yes", or 0-rated otherwise. The abbreviations in front of each item points to the underlying concept. The items included in the final model are underlined.

IN1	Is there inheritance from existing classes?
IN2	Is the code using a manually created inheritance hierarchy?
ME1	Is there a method call in the code?
ME2	Is there a method declaration?
ME3	Is there a return value in a method?
AG1	Is an assignment used in the code?
CO1	Is there a declaration of a new constructor?
CO2	Is there a call of a constructor?
ST1	Is it possible to save the state of an object?
ST2	Is it possible to change the state of an object?
ST3	Is it possible to use the state of an object?
AC1	Is there an association between classes in the code?
AC2	Is there any use of associations between classes?
DE1	Is the visibility of the attributes other than public or default?
OP1	Is the assignment operator used?
OP2	Are there any logical operators used in the code?
OP3	Are there any other operators used, apart from the assignment or logical operators?
AR1	Are there arrays with pre-initialization declared in the code?
AR2	Are there arrays without pre-initialization declared in the code?
AR3	Is there any access of the elements of an array in the code?
AR4	Is there an initialization with *new*?
AR5	Are methods of the class `Arrays` used in the code?
IS1	Is there an explicit initialization of the attributes?
PA1	Is there a method call with parameters in the code?
PA2	Are there any method declarations with parameters used?
PA3	Are the parameters of a method declarations used in the method body?
AT1	Are there attributes declared in the code?
AT2	Are attributes of other classes accessed?
AT3	Are attributes of class accessed within this class?
CS1	Is there an IF-statement without ELSE?
CS2	Is there an IF-statement with ELSE?
CS3	Is there a SWITCH-statement?
OB1	Is there a declaration of any object?
OB2	Is a declared object used in the code?
OB3	Is there a reference to its own object using *this*?
OV1	Is there a declaration of an overloaded method?
OV2	Is there an overloaded method used in the code?
LO1	Is there a use of loops?
AM1	Are the access modifiers *public*, *private* or *protected* used with attributes or methods?

During the investigation, the students were asked to implement a small project on the basis of an assignment that did not include explicit questions on programming. Nevertheless, the resulting programming code contains the responses on these questions. This is why we can assume the code-items to be assignments posed to the participants.

In total 321 datasets, gathered from 2008 to 2011, are included in this research project. Each dataset consists of the personal data and a vector with the responses on all code items.

6. RESULTS

The main goal of this research project was to develop and evaluate a methodology for assessing personal coding abilities from source code. For this purpose, we had to find a model that would describe the measured outcomes in a suitable way. Additionally, we had to validate the model by evaluating its outcomes.

6.1 Model Validation

The first step towards a valid item set for a Rasch model is to identify items that violate the preconditions of the Rasch model. First, all items that are related to the same structural element as others are eliminated. For example, both items `ST1` and `AT1` need a variable declaration in the code to be 1-rated. More precisely, this affects the items `AG1, ST1, ST2, ST3`, and `OB1`.

As all the tests that rely on dividing the population or item set into two parts need differing items in both parts of the population, the trivial items are removed in advance. An item is said to be trivial if it is either 1-rated for almost all or almost none of the participants. For that reason a limitation level of 0.01 is defined for this study. In particular, this affects only the item `OP1`, which deals with the use of an assignment operator.

Due to the large number of items in comparison to the number of participants at the beginning, the calculation of an exact test on local stochastic independence and homogeneity is not possible. Because of that, the non-parametric tests are applied. First, the item set is reduced to those items that are homogeneous.

Starting with the 33 items after the exclusion process, all items that violate the homogeneity criterion are eliminated. In the first run the items `DE1, OP2, IS1, AR5, CS2, CO1, AR4, CS3, OV1, AR1, PA2, PA3, ME2`, and `IN1` violate the homogeneity criterion. The selection criteria for which items are eliminated is the frequency of the dependent items. Thus, `DE1` has the most dependencies, while `IN1` has the least. In general, all pairs of dependent items are listed and the items are removed from that list one by one until the list is empty; then, no dependencies are left. Afterwards, a new set of simulated matrices is calculated based on the new item set. The second run results in elimination of `AR3, AR2, LO1, OB3, ME1, AC2, AT1, AT3, ME3, PA1`, and `AC1`. Once again, the items are ordered by their number of dependencies. After a third and fourth elimination run, the items `AM1` and `CS1` are removed from the item set. In the end the remaining items are homogeneous. Now, as the item set has been reduced to six items, the exact tests can be applied for justifying the nonparametric tests. The Martin-Löf test is conducted on the resulting item set. Here, the p-value is 0.66. Thus, the items are assumed to be homogeneous and locally stochastically independent. After that, the two test statistics presented in [2] are calculated. For the general goodness-of-fit test statistic G^2, a value of 62.1 is the result. Additionally, the χ^2 test statistic X^2 results in a value of 139.4. Both are not significant for 13 degrees of freedom to a level of 0.05 in the χ^2-distribution. Again, the H_0-hypothesis is rejected and the items are assumed to be homogeneous.

Following the test on homogeneity, the items that violate local stochastic independence in the way of being too similar in their results have to be found. Actually, only the item `OB2` is dependent on another item and is, therefore, eliminated from the result set. Last, the learning aspect of the local stochastic independence is tested. Here, no item violates the presumption.

The resulting item set with the items `IN2, CO2, OP3, AT2`, and `OV2` is valid with regard to the presumption of the Rasch model.

After validating the presumptions of the Rasch model, fitting of the data and a valid model are calculated for the given data. The likelihood-ratio test has a p-value of 0.5 for the first pre-knowledge splitting criterion (pre-knowledge level 1 vs. pre-knowledge levels 2&3). The test is not significant and, because of that, the model is assumed to be valid. A look at the Wald test for the items also shows no significant model violations ($p<0.05$). For both tests we have to assume a vector that splits the population into two parts. For this reason students' self-assessments of the previous knowledge (pk) concerning programming is chosen as a separator and, additionally, gender is chosen to find differences. Concerning the students' previous knowledge, two levels are put together and compared with a third level to get two groups. No model violations can be found for all three combinations. Gender as a separation criterion is not applicable as only 19% of the participants were female. As mentioned in [11, p. 99], the two groups should be almost of equal size.

Concerning the gender aspect, a closer look at the different items with separated participant groups show the violations in the model fit. For that reason, all items are split by gender. Afterwards for each group the items are tested if they are trivial. Due to the small group size of the female participants, a ratio of 0.01 is too small as it is less than one person. Because of that, the limit is set to 0.02, which means that at least one person has to have a different answer than the others. The problem occurs with the item `OP3`, where there is less than 0.02 different answers for the female group. Additionally, the item `IN2` has a value of only 0.03 for the female participants. Nevertheless, this is not critical for the test. All other items have a distribution between the answers of 0.3 and 0.7 (`CO2` and `OV2`) and 0.5 for both groups (`AT2`).

As described above, the Rasch model is a one-parametric test model where the items only differ in their level of difficulty. To show that another model with more estimated parameters does not fit better, a two-parametric test model is calculated and both models are compared. The comparison coefficients AIC and BIC [6] to get two groups have almost the same values. The two-parametric model provides no advantage by introducing an additional parameter.

6.2 Model Interpretation

After fitting the dataset to a valid Rasch model, the next paragraphs present the results of the model. In Figure 1 the item characteristic curves for all items that are included in the model are shown. According to the definition of the Rasch model, they only differ in their level of difficulty. This is expressed in the figure by a shift on the x-axis, which shows the latent parameter on a scale of -10 to 10. All curves are parallel and only differ in the value of the latent parameter at the probability of 50% for rating a code item with "yes" (1). The probability that an individual with a specific value of the latent parameter has solved a specific item is drawn on the y-axis.

By definition, the item parameters sum up to 0. The items `OP3` and `IN2` have values of -5.4 and 5.0, respectively. The item that is closest to the average of 0 for the investigated population is `AT2` (0.84). The use of attributes of other classes, either direct or by using a method, indicate participants with an average ability in coding, concerning the investigated items. Interestingly, all items except `AT2` and `OV2` have the same distance between each other. In general, Figure 1 presents a ranking of the items. The simplest item is `OP3`, which represents the use of arithmetic operators. The underlying concept is simple to code and all projects

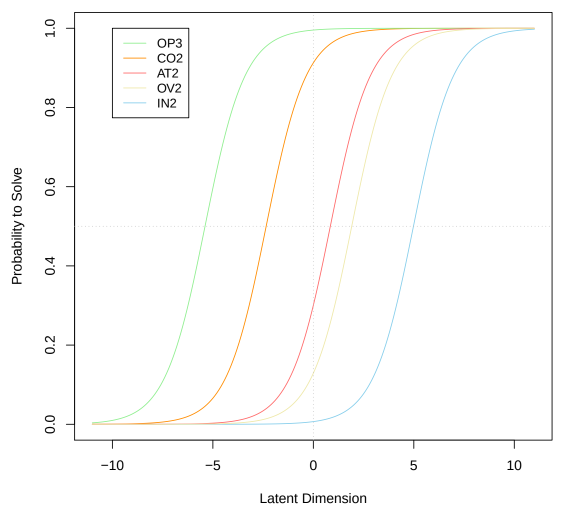

Figure 1: Item characteristic curves (ICC) of all items included in the Rasch model

need calculations. As a result, the position within the items is not surprising. The next concept in the ranking is `CO2`, which indicates the use of a constructor or an initialization of an object. Again, the underlying concept is easy, but the basic object-oriented notions have to be implemented as well. Next, the items `AT2` and `OV2` indicate the use of interrelations between classes. As mentioned above, the first one represents the use of foreign methods and attributes. The second one represents the use of overloaded methods. Regarding the last item `IN2` (use of an own class hierarchy), these two items represent more advanced concepts of object orientation. Thus, the item set contains representatives of simple coding concepts that can be related to the procedural paradigm, as well as representatives of advanced object-oriented notions.

In general, if the Rasch model is valid, the marginals of the underlying dataset are a sufficient statistic. Because of that, each possible person score is related to a person parameter. For mathematical reasons the parameters for the margins 0 and 5 cannot be estimated, but have to be interpolated. The mean value of the person parameters is 0.11; the median is -0.92.

Actually, there is a medium correlation (0.42) between the self-assessment of the students' previous knowledge and their person parameters (p-value $\ll 0.01$). Regarding gender of the participants, females (-0.13) have a lower – but not significant – average person parameter than male students (0.26). On the other hand, the self-assessment has a significant difference (p-value $\ll 0.01$) in the person parameters. The students with previous knowledge have a mean value of 1.06, while those without any previous knowledge have a mean value of -0.93.

In addition to students' previous knowledge and their gender, lines of code are another measurement that can be conducted on the source code. In particular, the projects differed a lot in their complexity. There were projects with only a few lines of code (min. 6 LOC) and some with more than one thousand lines of code (max. 1330 LOC) containing a GUI and other features . The mean value of project size regarding the lines of code is 212.7 LOC. For all participants, the median is 129 LOC, while the first quartile is 73 LOC and the third is 275 LOC. Furthermore, the projects developed by those with previous knowledge have significantly (p-value $\ll$ 0.01) more lines of code. The mean value for those with pre-knowledge is 253.2 LOC versus 160.7 LOC for those without pre-knowledge. Regarding the person parameters of the Rasch model, there is no correlation (0.07) to the lines of code.

7. DISCUSSION

The resulting model contains only five items. Except `OP3` which deals with arithmetic operators, the items are related to the object-oriented paradigm. As shown in Figure 1, the most simple one is the use of operators (`OP3`) followed by calling a constructor (`CO2`) and accessing attributes of other classes. The difficulty of this item is close to the use of overloaded methods (`OV2`). The most difficult item according to the Rasch model is the use of manually created inheritance hierarchies (`IN2`).

Generally, there are more homogeneous item sets than the presented one. The order of removing dependent items is important for the resulting item set. Here, the number of dependencies built the criterion. This results in a broad item set. In further research other criteria could be taken into account.

Concerning the evaluation of programmers, the person parameters of the model are of interest. As mentioned above, the participants had to assess their own previous knowledge. This self-assessment correlates with the results of the model. Furthermore, there is a significant difference in the use of the items between those with previous knowledge to those without any previous knowledge.

In addition to the code items we calculated the lines of code (LOC). Again, there is a significant difference between the previous knowledge levels. But, the person parameters of the model have no correlation to the lines of code. Concerning the lines of code, the projects produced by the participants without any previous knowledge are more similar than those produced by the other participants. Nevertheless, even in the group without any previous knowledge, there are projects with more than 500 LOC.

8. FUTURE WORK

Validating whether the items really measure the programming ability as the latent dimension is difficult to proof. In fact, the items only cover a part of programming ability as some concepts that are not mentioned in the material for the preprojects are missing. Additionally, the facet of problem solving that is a large part of the programming ability cannot be assessed by a simple structural analysis.

Although the model is fitting the data, there still remain some problems. Obviously, there are different kinds of difficulty concerning the code items. On the one hand, there are concepts that are difficult in a common understanding and there are concepts that force the programmer to recognize its use for a better programming code. So, this distinction implies two different kind of programming abilities and because of that more than one dimension. The investigation of the relationship of these dimensions is content of future work.

Another problem we figured out, is the dependence of some items on the programming assignments. Especially, some force specific concepts like *inheritance* while others do not. So, to create a generally valid model for programming assessment the assignments have to be chosen in a proper way so that no concept is privileged.

During the model fit tests we found several items that did not fit the model and because of that were excluded from the model. In a future research it would be important to find a way either to include the concepts by identifying other code items or by extending the investigated population.

9. CONCLUSION

We have shown in this paper that the investigation of source code with the help of item response theory could provide interesting outcomes. Yet, it seems difficult to find appropriate assignments that show the persons' programming abilities. The most common way is to provide small coding tasks that assess a specific part of that ability. Nevertheless, if the application and especially the combination of different concepts should be assessed this methodology does not work. Although, our method still has to be improved, the a posteriori identification of programming concepts in a bigger assignment might allow figuring out the complete facets of programming. Additionally, small tasks can be assessed as well at the time a general assessment tool based on item-response theory is conducted.

10. REFERENCES

[1] R. J. d. Ayala. *The theory and practice of item response theory.* Methodology in the social sciences. Guilford Press, New York, 2009.

[2] D. J. Bartholomew. *Analysis of multivariate social science data.* Chapman & Hall/CRC statistics in the social and behavioral sciences series. CRC Press, Boca Raton, 2nd edition, 2008.

[3] M. Berges, A. Mühling, and P. Hubwieser. The Gap Between Knowledge and Ability. In *Proceedings of the 12th Koli Calling International Conference on Computing Education Research - Koli Calling '12*, pages 126–134, New York, 2012. ACM Press.

[4] J. Bergin, K. Bruce, and M. Kölling. Objects-early tools: a demonstration. In *Proceedings of the 36th SIGCSE technical symposium on Computer science education*, pages 390–391, New York, 2005. ACM.

[5] J. Börstler, Henrik B. Christensen, Jens Bennedsen, M. Nordström, Lena Kallin Westin, J. E. Moström, and Michael E. Caspersen. Evaluating OO example programs for CS1. In *Proceedings of the 13th annual conference on Innovation and technology in computer science education*, pages 47–52, New York, 2008. ACM Press.

[6] K. P. Burnham and D. R. Anderson. *Model selection and multimodel inference: A practical information-theoretic approach.* Springer, New York, 2nd edition, 2002.

[7] G. H. Fischer and I. W. Molenaar. *Rasch Models: Foundations, recent developments, and applications.* Springer, New York, 1995.

[8] B. Hanks, C. McDowell, D. Draper, and M. Krnjajic. Program quality with pair programming in CS1. In *Proceedings of the 9th annual SIGCSE conference on Innovation and technology in computer science education*, volume 36, pages 176–180, New York, 2004. ACM Press.

[9] P. Hubwieser and M. Berges. Minimally invasive programming courses: learning OOP with(out) instruction. In *Proceedings of the 42nd ACM technical symposium on Computer science education*, pages 87–92, New York, 2011. ACM Press.

[10] G. Kemkes, T. Vasiga, and G. Cormack. Objective Scoring for Computing Competition Tasks. In R. Mittermeir, editor, *Informatics Education – The Bridge between Using and Understanding Computers*, volume 4226 of *Lecture Notes in Computer Science*, pages 230–241, Berlin, 2006. Springer.

[11] I. Koller and R. Hatzinger. Nonparametric tests for the Rasch model: explanation, development, and application of quasi-exact tests for small samples. *InterStat*, 11:1–16, 2013.

[12] A. Luxton-Reilly, P. Denny, D. Kirk, E. Tempero, and S.-Y. Yu. On the differences between correct student solutions. In *Proceedings of the 18th ACM conference on Innovation and technology in computer science education*, pages 177–182, New York, USA, 2013. ACM Press.

[13] M. McCracken, V. Almstrum, D. Diaz, M. Guzdial, D. Hagan, Y. B.-D. Kolikant, C. Laxer, L. Thomas, I. Utting, and T. Wilusz. A multi-national, multi-institutional study of assessment of programming skills of first-year CS students. In *Working group reports from ITiCSE on Innovation and technology in computer science education*, Working Group Reports, pages 125–180, New York, 2001. ACM Press.

[14] I. Ponocny. Nonparametric goodness-of-fit tests for the rasch model. *Psychometrika*, 66(3):437–460, 2001.

[15] G. Rasch. *Probabilistic models for some intelligence and attainment tests.* University of Chicago Press, Chicago, 1980.

[16] K. Sanders and L. Thomas. Checklists for grading object-oriented CS1 programs: concepts and misconceptions. In *Proceedings of the 12th annual SIGCSE conference on Innovation and technology in computer science education*, pages 166–170, New York, 2007. ACM Press.

[17] N. Truong, P. Roe, and P. Bancroft. Static analysis of students' Java programs. In *Proceedings of the 6th conference on Australasian computing education*, pages 317–325, Darlinghurst, 2004. Australian Computer Society, Inc.

[18] A. Vihavainen, T. Vikberg, M. Luukkainen, and M. Pärtel. Scaffolding students' learning using test my code. In *Proceedings of the 18th ACM conference on Innovation and technology in computer science education*, pages 117–122, New York, USA, 2013. ACM Press.

[19] T. Winters and T. Payne. What Do Students Know?: An Outcomes-based Assessment System. In *Proceedings of the first international workshop on Computing education research*, pages 165–172, New York, 2005. ACM Press.