Towards Competency Based Testing and Feedback

Competency Definition and Measurement in the Field of Algorithms & Data Structures

Peter Hubwieser, Marc Berges
School of Education
Technical University of Munich
München, Germany
peter.hubwieser@tum.de
berges@tum.de

Michael Striewe, Michael Goedicke
The Ruhr Institute for Software Technology
University of Duisburg-Essen,
Essen, Germany
michael.striewe@paluno.uni-due.de
michael.goedicke@paluno.uni-due.de

Abstract—At least since the first PISA surveys, the intentions of education are increasingly expressed in terms of competencies instead of learning goals. In consequence, learning outcomes should be measured in terms of these intended competencies as well. This holds for large-scale investigations like PISA as well as for the very small scale that is represented by the examinations that a single teaching person performs with his/her students. Yet, to assure validity, competency measurement requires proper models for competency structure and levels, which are based on empirical research. The goal of this project is to identify a set of relevant competencies in the domain of object-oriented programming (OOP), to develop a suitable competency model to represent their structure and to construct proper corresponding test instruments. This paper describes a study on the outcomes of several assignments that represents one step of this process with focus on the subdomain of data structures. The data were collected during an introductory lecture on programming in the winter term 2015/16. We analyzed the student solutions of three mini-projects that deal with the implementation of abstract data, applying a specific e-assessment system to score the students’ solutions. To find suitable item sets that could be used for the definition and measurement of certain competencies, different methodologies were applied that are based on classical test theory as well as on contemporary Item Response Analysis. Up to now, several suitable item sets for the competencies were identified, which are required to implement and operate 1- and 2- dim arrays, linked lists, and binary trees.

Keywords—competency models; Algorithms & data structures; introductory programming; Rasch Model; Item Response Theory

I. INTRODUCTION

Normally, teaching persons try to get information on the learning outcomes of her/his students by giving certain assignments to them. As nowadays teaching goals are defined in terms of competencies usually, the results of these assignments should reflect the competency level reached by the students. The same requirements have to be met by large-scale investigations on student learning outcomes, which require automatic assessment additionally. Unfortunately, competency measurement requires certain knowledge of the structure of the targeted abilities. Additionally, the outcomes of these assignments should depend predominantly on these competencies. As soon as the tested target group reaches larger sizes, the evaluation of the outcomes might produce considerable workloads. In consequence, automatic evaluation becomes desirable or even essential, particularly for large-scale investigations like the PISA studies (“Programme for International Student Assessment”) by the OECD that survey hundreds of thousands of participants in many countries.

Our project aims to develop models for the structure of programming competencies as well as methodologies to investigate the fitting of assignments to assist CS teachers and lecturers in monitoring their teaching outcomes. Among other research activities, the authors are evaluating assignments and outcomes of automatic testing systems for several reasons. First, considerable numbers of solutions of the assignments are required, usually some hundreds. Second, programming lectures are quite often attended by several hundreds of students. Third, our long-term goal is to provide the prerequisites for large-scale investigations in programming to enable a PISA survey on programming competencies one day.

The most important prerequisite is a comprehensive competency structure model for programming. For this purpose, a set of relevant competencies in this field has to be defined by identifying each of them with a set of suitable tasks. To test these competencies, the corresponding test instruments have to meet certain quality criteria, which are based on the Item Response Theory. Based on a prior proposal for the competency structure model [1], we have analyzed the student responses on programming assignments that were given during a large introductory programming lecture at a German university. The submitted programs were evaluated by the automatic assessment tool JACK [2]. In fact, we have found several combinations of items among these assignments that are likely to test certain competencies that fit in the proposed model, in particular in its first substantial dimension of data structures like arrays, lists, or trees. In this paper, we explain the theoretical background that is necessary to understand the applied methodology, which is described afterward. Following the presentation of the lecture, the assignments and the automatic assessment too, we present several results that might show how to construct items to test competencies in the field of Data Structures.

II. THEORETICAL BACKGROUND AND RELATED WORK

A. Competency Definition

In this paper, the authors refer to the well-known definition of competence by Weinert [3], who defined competencies as “the cognitive abilities and skills possessed by or able to be

© IEEE, 2017. This is the author's version of the work. It is posted here by permission of IEEE for your personal use. Not for redistribution. The definitive version was published in EDUCON (2017).

https://doi.org/10.1109/EDUCON.2017.7942896
learned by individuals that enable them to solve particular problems, as well as the motivational, volitional and social readiness and capacity to use the solutions successfully and responsibly in variable situations.”

Even more, some authors identify the competency directly with a set of tasks, e.g. [4] p. 15 (translated by the authors): “A competency consists of certain sets of tasks that can be performed by those who have this competency”. In the light of these statements, we aim to define certain competencies by a set of tasks each.

B. Competency Models

Klieme et al. [5] describe three types of competency models:

- Competency Structure Models, usually structured by dimensions (e.g. competency areas or competency characteristics) describing the cognitive dispositions that learning individuals need to solve tasks and problems in a specific content or requirement area,
- Competency Level Models, giving information about the levels or profiles of the described competencies, and
- Competency Development Models aiming to describe, how competencies will develop over time.

Usually, level or development models have to be based on structure models.

The German MoKoM-project aimed to develop a competency model for computer science based on empirical results [6]. The work had started with a theory-driven model that was enriched with empirical data. In addition to the objective of developing competencies, the MoKoM-project aims to develop appropriate test instruments. In contrary to our project, the scope of MoKoM was very broad, covering the four dimensions:

- System application,
- System comprehension,
- System development, and
- Dealing with system complexity.

Recently a first Competency Structure Model for OOP was proposed [1]. It is based on an extensive literature study on competency models of different subject areas and results of a survey on declarative knowledge (see [7]). Further, it was validated by several surveys among researchers, teachers, and students. It comprises the following dimensions:

1 OOP knowledge and skills
   1.1 Data structure (graph, tree, array)
   1.2 Class & object structure (object, attribute, association)
   1.3 Algorithmic structure (loops, conditional statement)
   1.4 Notional machine (data, working memory, processor, statement, program, automaton)

2 Mastering representation (language defined by syntax & semantics)
3 Cognitive Process
   3.1 Problem solving stage (understanding the problem, determine how to solve the problem, translating the problem into a computer language program, testing and debugging the program)
3.2 Cognitive Process Type (Interpreting, Producing)

C. Test Quality Criteria

To be regarded as “good”, any test (independent from its scale) has to meet at least three classical criteria: Objectivity, Reliability, and Validity. Yet, as already detected by Adams as a result of his extended literature review nearly a century ago [8], there are a lot of divergent, partly contradictory definitions of these criteria. Additionally, the criteria are partly interdependent. Adams concluded: “Validity is a quality, not a quantity. When a test measures a function, simple or complex, as completely as possible, it is a valid measure of that function regardless of whether it measures with high or low accuracy. [...] Reliability, also, is a quality, not a quantity. It is associated fundamentally with absence of systematic errors. When systematic errors appear, the retest usually measures a function different from that measured by the test. [...] When test and retest measure the same function twice, then the test is reliable. Objectivity, again, is a quality, not a quantity. Objectivity exists only when all errors of measurement are random. With the advent of correlated errors, subjectivity appears.”

Nowadays, the mostly referred aspect of validity is construct validity, see [9]: “Construct validation is involved whenever a test is to be interpreted as a measure of some attribute or quality, which is not “operationally defined.” The problem faced by the investigator is, “What constructs account for variance in test performance?” This corresponds to the basic assumption of Item Response Theory (see section E below). Additionally, he warns “Construct validity cannot generally be expressed in the form of a single simple coefficient”.

D. Measuring Competencies

According to Klieme et al. [5], “Competencies cannot be reflected by or assessed in terms of a single, isolated performance. Rather, the range of situations in which a specific competence takes effect always spans a certain spectrum of performance. Narrow assessments cannot meet the requirements of competency models. [...] Competence must be assessed by an array of tasks and tests that do more than simply tap factual knowledge”.

Obviously, it has to assured that the solution of this “array of tasks” requires predominantly certain levels of the competency that should be measured. This requirement closely resembles the criterion of Internal Consistency of classical test, which can be calculated by Cronbach’s Alpha Coefficient [10]. Alpha will be negative whenever there is greater within-subject variability than between-subject variability. The common rule of thumb for Internal Consistency is “excellent” for $\alpha \geq 0.9$, “good” for $\geq 0.8$ and acceptable for $\alpha \geq 0.7$. 
E. Item Response Theory

Regarding the evaluation of surveys on competencies, the Item Response Theory (IRT) is the current state of the art. While in Classical Test Theory, the psychometric construct of interest (in this case a certain competency) is considered to be measured directly by item scores (apart from a certain fuzziness), the IRT considers this construct as latent and not directly measurable. Instead, the probability of correct answers on a certain item depends on the competency in a certain way:

\[ P(X_{ik} = 1 | \theta, \beta_k) = f(\theta_i, \beta_k) \]  

(1)

Where \( \theta \) is the Ability Parameter of person \( i \), representing his/her level of competency, \( \beta \) the Difficulty Parameter of Item \( k \), and \( f(\theta_i, \beta_k) \) a certain function that is determined by the psychometric model (e.g. the Rasch Model, see below) that is assumed to fit the observations. In most cases, these parameters have to be estimated by effortful numerical calculations. Depending on the structure of the psychometric constructs that are to be measured, several different models may be applied, e.g. unidimensional models that cover only one single competency or, alternatively, multidimensional models. One of the simplest and most widely used Models is the basic unidimensional Rasch Model (RM) with one parameter:

\[ P(X_{ik} = 1 | \theta, \beta_k) = \frac{\exp(\theta - \beta_k)}{1 + \exp(\theta - \beta_k)} \]  

(2)

Provided that this model is applicable, some very convenient simplifications can be made. For example, the sum of the scores of all individual items is a sufficient statistics, which means that the (estimated) person parameter depends only on the total number of correct answers of this person. It does not matter, which items the person had responded to correctly.

The graph of this function looks as displayed in Figure 1 for four different values of \( \beta \) ("Ability"), taken from a set of items of one of the investigated assignments (MP2, see section III.C). These graphs are called Item Characteristic Curves (ICCs).

Figure 2 displays the ICCs of an item set (from MP2 again) that vary in difficulty (horizontal position) \( \beta \) as well as in discrimination (slope) \( \delta \). As the ICC of item A1c demonstrates, the variation of slope can cause intersections of ICCs. This would mean that the difficulty order of the regarded items depends on the person parameter, which would violate the requirement of specific objectivity (see below). On the other hand, low variation of slopes and missing intersections of a certain set of items in the BM can be regarded as a good indicator that the RM is applicable on this set.

![Fig. 1. Exemplary Item Characteristic Curves of the Rasch Model](image1)

![Fig. 2. Exemplary Item Characteristic Curves of the 2-parameter Birnbaum model](image2)

The application of the RM and the BM requires three preconditions that have to be met:

1) Homogeneity of items: All items are measuring the same psychometric construct. In this case, this set of items is called homogenous in this paper.

2) Local stochastic independence: the underlying psychometric construct is the only coupling factor between items.

3) Specific objectivity: for all samples from the population, the item parameters are independent of the specific person sample; the same holds for all samples of items and person parameters.

In the case that there is more than one psychometric construct to be measured, multidimensional models have to be applied. Unfortunately, it is not possible to explain these in this limited space here. For details, see the original paper of Rost & Carstensen [12].
F. Factor analysis of dichotomous data

To investigate the homogeneity of a set of items, classical explorative factor analysis is applied traditionally. Yet, as the score scale is dichotomous in this case, this is not applicable. The reasons are explained in detail in [13]. Thus, we chose the methodology of latent trait analysis (LTA) as presented in Chapter 8 of [13]. In the following section, this methodology will be outlined.

According to this methodology, it is assumed that the responses of the students to a given set of items can be described by a certain psychometric model, for example by the Rasch Model (see section II.E). Under this assumption, one can estimate all person and item parameters based on the scoring matrix of the responses. Using the estimated values of the parameters, by calculating the probability $P$ in equation (1) of section II.E, the expected number of occurrences $E(r)$ of all possible response patterns $r$ (e.g. 01101 in the case of 5 items) can be calculated. For $p$ dichotomous items, there are $2^p$ response patterns (i.e. combinations of 0s and 1s with the length $p$). For each pattern $r$, its expected frequency $E(r)$ is compared to the actually observed pattern frequency $O(r)$. For the differences, the log-likelihood test statistic $G^2$ and the common $X^2$ statistic are calculated that both describe the differences of the expected and the measured values. Both statistics are approximately $\chi^2$ distributed. Thus, we can estimate the goodness of fit of the applied Rasch Model by this way. The precondition for this calculation is a sufficient number of datasets, which assures that the frequency of each pattern has an expectation value of at least 5. This process is explained in detail in [13].

G. Automated Assessment

Automated assessment is a common feature of introductory courses at universities to cope with the high numbers of participants. For the subject of programming exercises, a wide range of e-assessment systems exists that detect errors by executing test cases and analyzing source code [14]. In general, automated systems are performing similarly well compared to human graders regarding scoring and the generation of feedback [15]. Fine-grained feedback and grading schemes have been proven to be effective in this context as well [16].

III. DATA COLLECTION AND ANALYSIS

For the development of competencies and test instruments, the scope has to be kept quite narrow in order to get useful results. Thus, for the beginning, the focus is set on the subdimension 1.1 of the structural proposal of [1], see section II.B. This subdimension was chosen because it seemed the most promising to find assignments that would comprise homogeneous items. Indeed, suitable data were found among the responses of the assignments of an introductory programming course at the University of Duisburg-Essen. These responses had been collected and evaluated by an automated grading system JACK [2].

A. Course Design and Assignments

The data of this study was collected during the introductory “Programming” lecture in the winter term 2015/16. The lecture was designed for students with major in Applied CS or Information Systems and CS teacher students. The programming language was Java. The course did not require any prior knowledge in programming. It was structured according to the “objects first” approach and covers the following topics in the given order: (1) Classes, fields, and methods; (2) Primitive data types and expressions; (3) Control structures; (4) Simple data structures; (5) Inheritance. The course format consists of 4 hours of lecture and 2 hours of exercise per week. Both lecture and exercise are presented to the whole class in the lecture hall. Additional tutoring sessions for smaller groups are offered for voluntary participation. The class started with approx. 700 participants. About 2/3 of them refrained from taking the final exam and dropped out rather early. Unfortunately, this was neither particularly characteristic for that course nor for this term. Instead, similar drop-out rates of students had to be observed in prior terms for that lecture as well as in introductory lectures in other subjects at the same university.

To be admitted to the final exam, students had to pass at least three out of six tests during the semester. To prepare for the tests, exercises in the form of so-called “mini-projects” (MPs) with the following scope were presented to the students:

- MP1: implementation of mathematical formula, using methods and primitive data types,
- MP2: implementation of 1- and 2-dimensional Arrays
- MP3: implementation of Linked Lists
- MP4: implementation of Binary Trees,
- MP5: implementation of Inheritance,
- MP6: implementation of a Data Base System, using predefined frameworks.

In all cases, the mini-project was composed of several tasks dealing with different aspects of the respective topic. For all mini-projects, the students had to use one or several predefined classes, which they should adopt and complete according to the definition of the respective task.

Due to their apparent correspondence with the assumed competency levels on sub-dimension 1.1 of the proposed model (see section II.B), we decided to investigate the outcomes of MP2, MP3, and MP4, which deal with the implementation of abstract data types. To give the reader an impression of the nature of the assignments, the tasks are summarized in the following paragraphs. In the parentheses, the identifiers of the subtasks are indicated, which are regarded as the items of the intended IRT-analysis.

MP2 deals with 1- and 2-dimensional Arrays and is accordingly subdivided into two separate parts. In both parts, a class implementation is provided which defines a field with a 1- or 2-dimensional array of type integer, but no initialization for that field. In the first part of the project the students are asked to:

1a) add a constructor to the first class that provides a proper initialization of the existing field with a 1-dimensional array of size 0,
1b) add a method that extends the existing array by one and adds a value provided as a method parameter to that new position,
(1c) add a method that takes a 1-dimensional array as a parameter and appends all of its values to the existing array,

(1d) add a method that returns the minimum value in the array, and

(1e) add a method that returns a new array containing all values from the existing one that are above a threshold provided as a method parameter. In the second part of the project, students are asked to

(2a) add a constructor to the second class that initializes the existing field with a 2-dimensional array provided as a constructor parameter,

(2b) add a method that takes a single integer value as a parameter and returns the result of a scalar multiplication of that value and the existing array,

(2c) add a method that takes a 2-dimensional array of type integer as a parameter and returns the sum of this array and the existing one,

(2d) add a method that returns the column vector for one of the columns in the existing array,

(2e) add a method that takes a 2-dimensional array of type integer as parameter and returns true if this array is identical to the existing one, and

(2f) add a method that transposes the existing array and returns the result.

MP3 illustrates the concept of Linked Lists by the metaphor of trains, which are considered as a linked list of coaches. Several operations have to be performed on this list. A complete implementation of a class representing the coaches (the list elements) is provided to the students along with the project description. In particular, the class defines some properties of the coaches such as its capacity or the number of passengers on board. It also provides methods to manipulate these properties. All tasks in this MP have to be solved by implementing additional methods in another class representing the train (= the list). In particular, tasks are to implement methods that

(1a) count the number of coaches in the train,
(1b) count the number of passengers on the train, (c) compute the total capacity of the train,
(1d) append a new coach to the train,
(1e) let some passengers board the train following some simple strategy on how they spread over the coaches,
(1f) uncouple coaches after a given index,
(1g) insert a coach at a given index, (h) invert the ordering of coaches,
(1i) remove a particular coach from the train, and
(1j) return the coach at a particular index.

MP4 covers Binary Trees by depicting the hierarchy of noble and normal persons in a kingdom. The root of the tree is the king with male persons as left successors and female persons as right successors. Some simple rules are provided in the project description on how persons are added as left or right successors below the king. A complete class definition is provided for a class representing the persons (tree nodes) and some of their properties such as gender and rank. The tasks of the MP ask to add methods in another class representing the kingdom (the tree). In particular, students are asked to implement

(1) the constructor for the kingdom to set up a proper tree with a king as root,
(2) a method to add persons to the tree following the rules given in the project description,
(3) a method that transforms the tree into an array following some criteria for the ordering of elements,
(4) a method for searching an element in the tree,
(5) a method for removing an element in the tree, and
(6) a method inverting the order of tree elements, thus making the king the least important person in the kingdom.

B. The E-assessment system

The applied e-assessment-system AA was used to score the students’ solutions and to provide automated feedback. It makes use of several techniques for grading and feedback generation by the abilities to

- execute the standard Java compiler and pass all compiler messages to the student in order to grade and comment on syntactical correctness of a solution,
- perform teacher-defined rule-based checks on the syntax graph of a solution and provide arbitrary feedback on unwanted or missing code structures in order to grade and comment on non-functional properties of the solution,
- execute teacher-defined test cases and provide both arbitrary feedback on test results and generic feedback on Java runtime exceptions and infinite loops in order to grade and comment on functional properties of the solution.

Teachers can define exercise-specific grading schemas with different weights for rules and test cases.

C. Data preparation

For the analysis, the data had been extracted from the e-assessment system’s database according to the following procedure: In the first step, a list of students was created for each of the three Mini Projects, referring to all students who made at least one submission to this project. Submissions that did not change the predefined classes were ignored. The resulting lists contained submissions of 548 individual students for MP2, 433 individual students for MP3 and 340 individual students for MP4. In the second step, a single list was created containing the intersection of the three lists from the first step. This list contained submissions of the 321 students that tried all three Mini Projects. In the third step, all submissions from these 321 students to all three Mini Projects were analyzed with respect to successful and failed test cases. For that purpose, each test case was mapped to one particular task within the exercise. A task was considered to be solved successfully if all test cases associated with this task were executed successfully. In turn, a
task was considered not to be solved successfully, if any of its test cases failed or could not be executed because of compiler errors. For each exercise, a matrix has been created where each line represents one submission, and each column represents one task. The matrix contains a 1 where a submission solved a task and 0 in all other cases. Additionally, the total scores were extracted that were reached by each submission according to the scoring strategy of the e-assessment system. Table I displays some statistics of the data.

| TABLE I. BASIC STATISTIC DATA OF THE MINI-PROJECTS |
|-----------------|--------|--------|--------|
|                 | MP2    | MP3    | MP4    |
| Nr of participants | 321    | 321    | 321    |
| Nr of submissions  | 1726   | 3214   | 1532   |
| Max submissions per participant | 55     | 124    | 61     |
| Mean Submissions per participant | 5.38   | 10.01  | 4.77   |
| Mean (max) Scores over all students | 94.27  | 90.13  | 77.60  |

As many students had submitted several (up to 124) different solutions to the e-assessment system, we had to decide between two choices. First, we could regard each submission as a single “virtual” participant with his/her own person parameter. This would increase the number of cases dramatically (up to more than 1700). Second, we could restrict the analysis to one submission per student. In this case, we would have to decide which submission of each student we should pick (e.g. the first, the last or a random one). We decided to follow both paths by evaluating the results based on three sets of submissions:

- the total set of all submissions (ALL),
- a first random selection of one submission per student (SINGLE1),
- a second random selection of one submission per student (SINGLE2).

Yet, it turned out that the first selection (ALL) was not suitable in several regards. For example, it did not produce any homogenous item combination in MP2 and MP3. Therefore, only the random selections were used for most analysis.

IV. DATA ANALYSIS AND RESULTS

A. Item Difficulty

Obviously, the most interesting information for teachers is the frequency of correct solutions. Table II shows the ranking from easy to hard for each mini-project for the selection SINGLE2.

| TABLE II. ITEM DIFFICULTIES (PORTION OF CORRECT SOLUTIONS) |
|-----------------|--------|--------|--------|
| MP2 Item        | Diffi-  | MP3 Item | Diffi-  | MP4 Item | Dif-  |
|                 | culty   |         | culty   |         | culty  |
| A1a             | 0.8972  | A2d      | 0.6978  | A1       | 0.6729 |
| A1d             | 0.8442  | A2a      | 0.6916  | A2       | 0.5327 |

B. Internal Consistency

Despite the importance of IRT, the classical methods for analyzing Internal Consistency seemed advisable to control out intended IRT results. In particular, for each mini-project Cronbach’s Alpha Coefficient of the total item sets based on SINGLE2 was calculated, which turned out to be excellent in all cases (see Table III).

| TABLE III. CRONBACH’S ALPHA |
|-----------------|--------|--------|--------|
| Project         | MP2    | MP3    | MP4    |
| Nr Items        | 11     | 10     | 6      |
| Alpha           | 0.94   | 0.96   | 0.94   |

Additionally, the Point Biserial Correlation between each item and the total score (including and excluding this item) was calculated. Again, the correlations showed high values from high (in 2 cases between 0.6 and 0.7, all others > 0.7) in all cases.

From a classical point of view, these results suggest that all three items sets show a high internal consistency and thus are likely to measure a common psychometric construct for each set. Yet, unfortunately, the requirements of the IRT turned out to be much harder to meet.

C. Latent Trait Analysis

In this case, the number of cases (students) of 321 (see section III.A) restricts the size of item sets to be considered by LTA to a maximum of 6. Unfortunately, LTA is confirmatory in nature and therefore requires an a priori defined set of items to be tested. We solved this problem by calculating both statistics $G^2$ and $X^2$ for all possible combinations of $p = 3, 4, 5,$ and 6 out of the total set of items for each Mini Project. Following this, we selected all combinations of $p$ items where $G^2$ or $X^2$ did not exceed the $\chi^2$ limits for the respecting degrees of freedom. As we did not want to lose combinations which were nearly homogenous, we allowed a tolerance of 10% on the $\chi^2$-limits. All computing was performed in GNU R, using the package ltm [17].

Indeed, it turned out that a lot of combinations of subtasks met the requirements of the LTA.
TABLE IV. NUMBERS OF HOMOGENEOUS ITEM SETS.

<table>
<thead>
<tr>
<th></th>
<th>ALL</th>
<th>SINGLE1</th>
<th>SINGLE2</th>
</tr>
</thead>
<tbody>
<tr>
<td>MP2  (11 Items)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4-Combs</td>
<td>0</td>
<td>26</td>
<td>48</td>
</tr>
<tr>
<td>5-Combs</td>
<td>0</td>
<td>1</td>
<td>15</td>
</tr>
<tr>
<td>6-Combs</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>MP3  (10 Items)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4-Combs</td>
<td>0</td>
<td>58</td>
<td>50</td>
</tr>
<tr>
<td>5-Combs</td>
<td>0</td>
<td>19</td>
<td>11</td>
</tr>
<tr>
<td>6-Combs</td>
<td>0</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>MP4  (6 Items)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4-Combs</td>
<td>1</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>5-Combs</td>
<td>0</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>6-Combs</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

 Interestingly, the two random selections SINGLE1 and SINGLE2 produced slightly different results regarding the potentially homogenous subsets, while the ALL selection showed a rather bad picture of homogeneity. The following analysis is restricted to the selection SINGLE2, which produced the best results.

As a consequence of the high number of potentially homogenous item subsets, nearly every item belongs to more than one of these combinations. This arises the question, which items might be combined at the end. To visualize the psychometric structure of the items over all combinations, we aggregated a graph of all items of each mini-project. An edge between two items is drawn if these two items belong to the same homogenous item set. Hence, the edges show which pairs of items might measure the same construct.

The production of these graphs was performed separately for each combination length $k = 4, 5, 6$. It turned out that for $k=4$, all items of each mini-project were connected, indicating that all these items belonged at least to one homogenous item combination of length 4. The same was true for $k=5$ in MP2. Thus, we give the graph of MP2 for $k=6$, which provided the most selective information (see Figure 3).

Apparently, the Items 2a, 2c, 2d, and 2e are measuring slightly different constructs compared to the rest, which all belong to at least one combination of length 6.

In MP3, only item 2c does not belong to any combination of length 5. Yet, $k=6$ gives the most interesting information, see Figure 4.

In MP4, the maximum length of homogeneous combinations was $k=5$. As all items belong to at least one combination of this length, the respective graph does not provide useful information.

In summary, the results of LTA suggest that all items of all mini-projects seem to belong to at least one homogenous combinations of items, which would allow the application of the Rasch Model according to LTA. This means that these item sets are likely to measure the same competency regarding homogeneity. Yet, this does not mean that they would represent “really good” test item sets, as the following analysis of the model fit demonstrates.

Additionally, several non-parametric tests for the Rasch model fit were applied that can be used to proof homogeneity if the number of cases is too low for LTA. The lack of data is overcome by simulating data matrices that have the same row and column sums as the estimated data sets by applying a Markov Chain Monte-Carlo algorithm [18]. The non-parametric
tests and the underlying test statistic are explained in [19]. The procedure to gather homogeneous item sets is described in detail in [20]. The algorithm starts with the complete item set and reduces the items regarding the number of homogeneity violations. We used these tests to double-check the results of the latent trait analysis. In the end, the non-parametric probabilistic calculations resulted in the same item sets, approving the results of LTA.

D. Model Fit Analysis

To inspect the fit of the Rasch Model, we first applied the 2-parameter Model to all combinations, aiming to detect the variation of item discriminators $\delta$ and possible intersections of ICCs. Unfortunately, it turned out that most of the combinations of maximum lengths showed substantial variations in item discrimination, causing cross-overs as displayed in Figure 5 for one of the 6-combinations in MP3.

Yet, several shorter combinations seemed to have acceptable variations in discrimination parameters $\delta$ (see Figure 6).

The “best” item combination regarding its length, the variation of the discrimination parameter and the Model Fit was one of the two combinations of 5 items of MP4 (see Figure 7), called MP4-C5 in the following text. The only flaw of MP4-C5 was that three items A4, A5 and A6 have nearly the same item difficulty, causing a coincidence of their ICCs by this way. Yet, it would be easy to correct this flaw by slightly tuning the difficulty up in two of these cases.

A further analysis of the Model Fit by comparing the AIC and BIC Parameters of the Models with 1 and 2 parameter (for explanation and details see [17]) shows that in this case indeed the Rasch Model is the best fitting model. Apart from the flaw mentioned above, MP4-C5 represents an item set that enables us to define and measure a competency called “Implementing Binary Trees”.

As explained in section II.E, the application of the Rasch Model to a set of data requires the estimation of the item and person parameters of equation (1) in II.E. Table V displays the item parameters of MP4-C5, resulting in the same difficulty ranking as produced by the classical item difficulty (portion of correct solutions)

<table>
<thead>
<tr>
<th>Parameter</th>
<th>value</th>
<th>std.err</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dffclt.A1</td>
<td>-0.5194</td>
<td>0.2208</td>
</tr>
<tr>
<td>Dffclt.A2</td>
<td>0.0046</td>
<td>0.0158</td>
</tr>
<tr>
<td>Dffclt.A4</td>
<td>0.5849</td>
<td>0.1182</td>
</tr>
<tr>
<td>Dffclt.A5</td>
<td>0.6240</td>
<td>0.0694</td>
</tr>
<tr>
<td>Dffclt.A6</td>
<td>0.6166</td>
<td>0.0786</td>
</tr>
<tr>
<td>Dscrmn</td>
<td>18.4405</td>
<td>23.0895</td>
</tr>
</tbody>
</table>

V. PERSONAL ABILITIES

Of course, practicing lecturers will be mostly interested in the learning outcomes of their students. According IRT, these are indicated by the individual person parameters $\Theta$. As mentioned in section II.E, the total scores (number of correctly solved items) represent sufficient statistics in the case that the Rasch model is applicable. Thus, there are only $k+1$ values of $\Theta$ for a set of $k$ items in this case. Table VI shows the person parameters $\Theta$ of MP4-C5 that corresponds with all possible response patterns. Additionally, the observed and the expected (according to the estimated parameters) frequencies of the response patterns are given. A look at the table demonstrates that there are only six different values of $\Theta$, corresponding to the number of five items.
TABLE VI. PERSON PARAMETERS OF RESPONSE PATTERNS IN MP4-C5

<table>
<thead>
<tr>
<th>A1</th>
<th>A2</th>
<th>A3</th>
<th>A4</th>
<th>A5</th>
<th>A6</th>
<th>Obs. Frequ.</th>
<th>Exp. Frequ.</th>
<th>Θ</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>105</td>
<td>113.609</td>
<td>-0.695</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>45</td>
<td>48.724</td>
<td>-0.219</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>39</td>
<td>42.228</td>
<td>0.230</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>3</td>
<td>2.146</td>
<td>0.568</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>2</td>
<td>1</td>
<td>1.871</td>
<td>0.568</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>6</td>
<td>5.808</td>
<td>0.644</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>6</td>
<td>3.853</td>
<td>0.568</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>11</td>
<td>11.958</td>
<td>0.644</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>10</td>
<td>10.426</td>
<td>0.644</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>94</td>
<td>80.372</td>
<td>0.837</td>
<td></td>
</tr>
</tbody>
</table>

Regarding the distribution of the students over these values of Θ, it was found that this is far from being normal. Instead, the low and high values dominate. Apparently, many students were able either to solve all items or not solve any of them.

Figure 8. Distribution of Person Parameters Θ in MP4-C5

This result is supported by the outcomes of a cluster analysis of the students based on the dichotomous response matrix of MP4-C5. The clusters were calculated applying the R function hclust (of package stats), which allows to use several different clustering algorithms. We found that the algorithm complete combined with a Manhattan distance metrics produced the best results in this case. According to the structure of the respective dendrogram, four seemed to be the most suitable number of clusters. Table VII shows the number of students in each cluster.

Figure 9 displays the average success rates (portion of correct solutions) of each cluster over all items of MP4-C5. For example, about 40% of the students in cluster 4 were able to solve item 3. Apparently, the students of cluster 1 were the least successful. About 25% of them solved item 1, yet none was able to solve any further item. In contrast, cluster 2 comprises the best students, having a success rate of more than 90% over all items. The 20 students of cluster 3 were similarly successful, except in item 5, where they failed altogether. Fortunately, 60% of them did not give up but solved item 6. Cluster 4, finally, describes the students that started very good in item 1 and 2. But then, only 50% of them solved item 3. Afterwards, they failed altogether.

It can also be taken from Figure 9 that item 1 is the easiest and item 5 the most difficult, in perfect accordance with the classical and IRT difficulties.

TABLE VII. NUMBER OF STUDENTS IN THE CLUSTERS

<table>
<thead>
<tr>
<th>Cluster</th>
<th>Nr Students</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>150</td>
</tr>
<tr>
<td>2</td>
<td>112</td>
</tr>
<tr>
<td>3</td>
<td>20</td>
</tr>
<tr>
<td>4</td>
<td>39</td>
</tr>
</tbody>
</table>

VI. CONCLUSION AND FURTHER WORK

In summary, we have applied classical and IRT methods to analyze the homogeneity and the suitability of all possible combinations of items of the three mini-projects. As the results demonstrate, these items can be used to define three competencies that could be identified with the abilities to “implement and operate

• one- and two-dimensional arrays,
• linked lists, or
• binary trees”.

In addition, we have found several item sets that might form good starting points to construct suitable test instruments for these competencies. The best set we have found was MP4-C5, which only has to be fine-tuned for this purpose.
Once we have designed these items, a second pilot will take place in the same introductory lecture. Provided that it succeeds, we would have constructed an online measurement instrument for the searched competencies that allows survey in PISA-like scales.

By the practicing lecturer, the resulting item sets could be taken as blueprints for his/her examinations. Keeping close to these items would assure that the examinations measure predominantly the intended competencies.

In longer terms, we aim to vary the items on the remaining dimensions of the proposed competency model, e.g. on the representation dimension by using other programming languages. By this way, we hope to validate the model step by step over all its dimensions.

REFERENCES


