Validity and Reliability of Student Models for Problem-Solving Activities

Tomáš Effenberger
tomas.effenberger@mail.muni.cz
Masaryk University
Brno, Czech Republic

Radek Pelánek
xpelanek@fi.muni.cz
Masaryk University
Brno, Czech Republic

ABSTRACT

Student models are typically evaluated through predicting the correctness of the next answer. This approach is insufficient in the problem-solving context, especially for student models that use performance data beyond binary correctness. We propose more comprehensive methods for validating student models and illustrate them in the context of introductory programming. We demonstrate the insufficiency of the next answer correctness prediction task, as it is neither able to reveal low validity of student models that use just binary correctness, nor does it show increased validity of models that use other performance data. The key message is that the prevalent usage of the next answer correctness for validating student models and binary correctness as the only input to the models is not always warranted and limits the progress in learning analytics.

CCS CONCEPTS

• **Human-centered computing** → **User models**; • **Applied computing** → **Interactive learning environments**.

KEYWORDS

student modeling; skills; difficulties; validity; reliability; performance measures; problem solving; introductory programming

ACM Reference Format:
Tomáš Effenberger and Radek Pelánek. 2021. Validity and Reliability of Student Models for Problem-Solving Activities. In *LAK21: 11th International Learning Analytics and Knowledge Conference (LAK21), April 12–16, 2021, Irvine, CA, USA.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3448139.3448140

1 INTRODUCTION

Student modeling is a key step in the development of adaptive learning environments [28]. Based on data about student performance, a student model provides an estimate of a student state. This estimate is used to drive adaptive behavior, such as challenging students with appropriately difficult problems. Student models are often evaluated using predictive accuracy, using metrics like RMSE and AUC for the task of predicting the next answer correctness [27]. This approach to evaluation is, however, quite narrow [15]. We care about more general properties of the models: their validity and reliability. Predictive accuracy is just a proxy for these general properties.

The simplified approach to evaluation may be reasonable for models that utilize data from simple exercises like multiple-choice questions. In the case of more complex problem-solving activities, its limitations become apparent. For illustration, consider the following scenario. Imagine you are developing an online learning system to teach introductory programming. The system contains several exercises, such as turtle graphics and Python programming. Each exercise contains dozens of problems (e.g., draw a triangle, compute factorial), and you want to order them from the easiest to the most difficult. The system also features a teacher mode that should support ordering students by their programming skill so that teachers can instantly see who is struggling. Finally, you want to show predicted problem-solving times to provide students with a personalized challenge to overcome themselves [31].

In the case of programming exercises, many aspects of student performance are potentially informative about student state: not just the correctness of answers, but also response time, the number of edits, or the length of code. Which of them are useful and which only make models more complicated? How should we compare models which utilize different aspects of performance? If we want to use predictive accuracy for evaluation, which aspect of performance should we predict? When we choose a single aspect of performance, we implicitly favor models that focus on this performance aspect. When we consider the various uses of student models (outlined in the previous paragraph), it is clear that there is not a single objective criterion that can serve as a measure of model quality.

Drawing inspiration from psychometrics [6, 8], we propose methods for assessing the validity and reliability of student models. We demonstrate these methods in a case study of estimating problems' difficulties and students' skills in programming exercises. To explore the generalizability of the presented results, we use six types of programming exercises (e.g., turtle graphics realized in block-based programming or textual programming in Python). Our case study illustrates the insufficiency of the binary correctness prediction task, as it is neither able to reveal the low validity of student models that use just binary correctness, nor does it show increased validity of models that use other performance data.

We compare binary correctness with non-binary performance measures primarily to illustrate the methods for assessing validity and reliability. As a secondary contribution, the results of the comparison confirm that the prevalent usage of the binary correctness as the only input to the student models is not warranted in the problem-solving context. Our case study provides evidence of

LAK21, April 12–16, 2021, Irvine, CA, USA

ACM ISBN 978-1-4503-8935-8/21/04…$15.00
https://doi.org/10.1145/3448139.3448140

non-binary performance measures' utility since they increase the validity and reliability of skill and difficulty estimates.

2 RELATED WORK

We first describe the general concepts of validity and reliability as they are used in psychometrics and then review the current approaches for evaluating student models.

2.1 Validity and Reliability

In psychometrics, the validity of a measurement tool is defined as the degree to which the measurements can be interpreted as representing the underlying construct with respect to the intended purpose of the measurement[1] [6, 8]. In learning analytics, construct might be a programming skill, measurement tool a student model, and the purpose to order students in an overview for a teacher.

2.1.1 Validity through Utility. Unlike most of the properties predicted by machine learning models, skills and other psychological constructs are not directly observable. Measurement tools, therefore, cannot be validated simply by comparing the measurements to the ground truth labels; there is no ground truth. Instead of seeking for measurements that are "true," we should strive for measurements that are useful [4]. This means that we cannot establish the validity of a measurement tool without considering its intended uses [23]. One measurement tool can have multiple uses and be valid for some of them, while invalid for others [19]. Therefore, to validate a measure, we should ask about the practical consequences of the measurements [6, 19, 23].

For example, to validate a student model that estimates skills, we need to know how the measured skills affect decision making. For ordering the students in an overview for a teacher, specific values are irrelevant, while placing all struggling students at the top of the list is crucial. For deciding on mastery [32], the ordering of students is irrelevant, while the specific values for students close to the mastery threshold are crucial.

2.1.2 Sources of Validity Evidence. Providing strong evidence of the utility of a measure is often infeasible in an early stage of research, as the consequences of the decisions based on the measurements take a long time to manifest. As an alternative, we can provide evidence for the validity of the intended interpretations of the measurements, using multiple sources such as the internal structure and content of the measurement tool, thought processes of students (through think-aloud), and relationships to related variables, which is denoted as *criterion validity* [6, 8].

Two special cases of criterion validity are *convergent* and *predictive*. The convergent validity argument is based on high correlations between measurements of related constructs. Convergent validity was used to reduce the number of items in a questionnaire for measuring intrinsic motivation in ASSISTments, ensuring that the shorter questionnaire still measures the same construct as the original long one [24]. The predictive validity argument is based on the ability of the measurements to predict related variables measured in the future, e.g., a score on the final exam. Predictive validity is the most common (and usually the only) evidence of validity provided in student modeling research.

As an example of providing multiple sources of evidence for validity, let us look at validating a language-independent CS1 assessment [33]. First, the authors created an assessment specification, which was reviewed by experts to ensure that all relevant CS1 concepts are covered (content validity). The test was then administered to students of four introductory courses, and both performance data and think-aloud interviews were collected. The assessment's internal structure was evaluated using Item Response Theory to ensure that all problems are appropriately difficult, discriminate between students, and have a low guessing rate. Students' mental processes captured in think-aloud interviews were examined to see whether they used the intended programming concepts to solve the problems. The authors also provided a predictive validity argument by comparing their assessment results to the final CS1 exam scores. The original paper did not show evidence of the assessment's utility, but it was since then used to help to answer several research questions [26, 34].

2.1.3 Reliability. In psychometrics, the reliability of a measurement tool is defined as the consistency of the measurements [6, 8]. For example, a reliable test of intelligence should result in the same score if taken repeatedly by the same person. There are many types of reliability since there are many variables that we may want our measurements to be independent of. We may desire consistency of the measurements through time (test-retest reliability) [8], between raters (inter-rater reliability) [16], between multiple variants of the measurement tool (alternate forms reliability) [8], or between subsets of items in the measurement tool (internal consistency) [6].

For each of these general types of reliability, there are several specific ways to quantify them. For example, the inter-rater reliability can be quantified by percent agreement or by Cohen's Kappa, which accounts for the agreement by chance [22]. Yet other metrics are needed in case of multiple raters or non-nominal scales [16]. As a second example, consider the internal consistency between items. One way to quantify the internal consistency is a split-half method, i.e., computing correlation between measurements obtained using two disjoint subsets of items [21]. If all items are supposed to measure the same construct, Cronbach's alpha is often used to quantify the agreement between all individual items by a single number [7].

Consistency is only needed over those variables that we want our measure to be independent of [19]. For example, as the students in online learning systems are learning, we do not expect skill measurements of a single student to be stable through time.

2.1.4 Validity vs. Reliability. Reliability can be considered as one type of validity evidence [6]. High reliability is necessary but not sufficient for validity [6, 19]. For instance, using problem IDs as difficulty estimates is completely reliable — the IDs do not change between repeated measurements — but clearly invalid. The interaction between validity and reliability is, however, more subtle. For example, by increasing the coverage of programming concepts, we might increase the validity of a test at the cost of reducing its reliability [17]. The desirable trade-off depends on the use of the measurement: for high-stake tests, reliability is crucial; for frequent summative assessment, it is not.

[1] Measurements are also denoted as scores, and measurement tool as a measure, scale, instrument, inventory, questionnaire, or test.

2.2 Evaluation of Student Models

Measuring the validity and reliability of psychological constructs is difficult, but it is even more so in the context of online learning systems. Students are learning, so we only observe a single attempt for a given student before her skill changes — a rather extreme data sparsity! Furthermore, each student solves only a subset of available problems, and neither the subset nor the order of the problems is random, resulting in various data collection biases [29].

There is no standard approach to assess the reliability of adaptive tests [9], let alone the reliability of student models that account for learning. In contrast, there is a standard approach to assess the validity of student models: through the performance on the next answer correctness prediction task [27]. This approach, however, has been criticized as potentially misleading because increasing predictive accuracy does not always increase the model's utility [15].

We can demonstrate the utility of a student model using a randomized controlled trial, but there is still a need for methods to validate student models in isolation using just collected performance data. Such *layered evaluation* [5, 25] has three advantages compared to a randomized controlled trial: (1) it disentangles the effects of various components in the learning system, allowing us to diagnose specific weak points, (2) it allows us to evaluate many student models early, quickly, and without negatively impacting students, and (3) by using data from multiple exercises and learning systems (and even simulated data), we can explore the behavior of the model in many situations, ensuring its reusability.

When evaluating a student model in isolation, we should provide multiple sources of validity evidence [18, 25] — this is analogical to the common advice in psychometrics. In addition to the predictive performance, previous studies assessed the plausibility and consistency of fitted model parameters [18] and expected instructional effectiveness (effort needed to learn a new concept) when coupled with a specific instructional policy [15].

2.3 Student Models with Performance Measures

We explore one specific context in which the next answer correctness prediction task is insufficient: comparison of student models for problem-solving activities that use other performance data beyond correctness. If each problem takes a few minutes to solve, binary correctness provides too little information for meaningful adaptation. Without richer input, the state-of-the-art student models might already be close to their ceiling performance [3]. The importance of the input data is a consistent result; predictive accuracy of various student models was improved by incorporating response time [14, 20], the number of incorrect attempts [35, 37], or hints usage [35, 36].

Previous work proposed an *answer classification interface* between the data collection layer and applications to facilitate the reusability and comparability of extended student models [30]. The detailed observations about a student's attempt are compressed by a *performance measure* into just one of a few performance classes (e.g., "correct, weak"). Student models then use only these discrete performance measurements, as illustrated in Fig. 1. Our case study supports this proposal by showing that the use of such performance classes can increase the validity and reliability of skill and difficulty estimates.

3 SETTING

Our case study is an extension of the scenario described in the Introduction, which poses the following three challenges: (1) to order problems by their difficulties, (2) to order students by their skills, and (3) to predict various aspects of future performance such as problem-solving time. Our aim is not to find the best model to solve these challenges; instead, we aim to clarify how to evaluate models in such scenarios properly.

Therefore, we intentionally chose the simplest student models that allow us to illustrate the evaluation issues clearly. The chosen student models are based on a simple use of mean performance and deviations from the mean performance. The only aspect in which the compared models differ from each other is the performance measure that prepares input to them (e.g., binary correctness measure feeds the model with 0s and 1s). To be useful for the posed challenges, each model: (1) estimates a one-dimensional difficulty for each problem, (2) tracks a one-dimensional skill for each student after each attempt, and (3) can predict future performance for a student-problem pair. Presented evaluation methods are applicable to any model that can provide these estimates and predictions.

Before we proceed to the details of the models and data, let us clarify the used terminology. An *exercise* is a collection of problems with a shared theme and user interface; an example of an exercise is turtle graphics with a block-based programming interface. A *problem* is a specific instance of an exercise, e.g., drawing a square using turtle graphics. An *attempt* is a time-series of student-problem interactions like edits and code executions without a long break. If the student has eventually solved the problem, we call the attempt *correct*; if not, we call it *incorrect*. Fig. 1 illustrates the overall context and used terminology.

3.1 Baseline Approaches

Let us first think about some simple approaches to solve the three challenges. Problems could be ordered by their success rate. Students could be ordered by their success rate as well, but to account for the difficulty of the attempted problems, we might instead estimate the skill as the mean deviation between the observed performance (0 or 1) and the success rate of the problem. As for the time-prediction task, a reasonable non-personalized baseline is a per-problem average of times, or preferably an average of *log-transformed* times since problem-solving times are usually log-normally distributed [31]. To make the predictions personalized, we might slightly adjust the per-problem average in proportion to the estimated skill; we describe one way to do this in section 3.3.

Such baselines would not work particularly well. First, due to the amount of non-serious attempts, which are unevenly distributed, the success rate is considerably biased [10]. Second, as most attempts are eventually correct, the skill estimate is based on much less than one noisy bit of information per each attempt, and there are typically not so many attempts per student in problem-solving exercises (each attempt can take up to few minutes). As we will see in section 4.2, that is not enough information to reliably differentiate between students. Third, the ability to solve a problem is

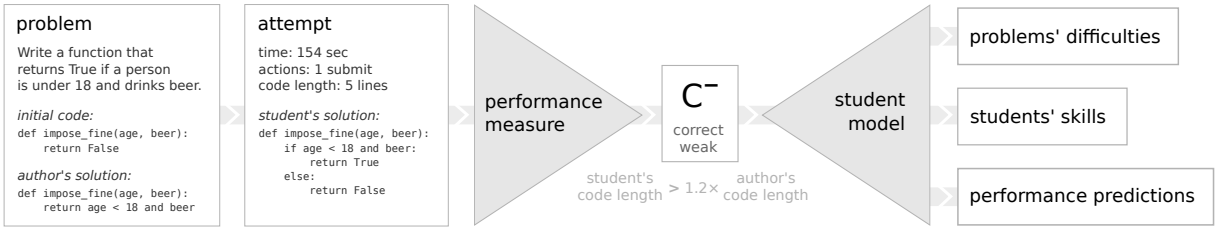

Figure 1: Example of a problem from Python exercise, a student's attempt, and a performance measurement based on final code length. The performance measurement ("correct, weak") is the only information about the attempt passed to the student model.

only one aspect of the programming skill; we also care about the students' speed and the quality of their code. Note that all these issues are primarily caused by poor input data, not by the admittedly trivial student model; more complex student models are thus unlikely to miraculously help. To alleviate these issues, we need to use additional performance data.

3.2 Performance Measures

We compare eight performance measures. Our goal is an illustration of diverse options, not a systematic treatment of performance measures. In addition to the standard binary correctness, we define one continuous measure based on solving time, three discrete measures based on solving time, two discrete measures based on code length, and one combined measure that uses both time and code. The discrete measures adhere to the *answer classification interface* proposed in [30], using the following five classes: correct excellent C^+, correct normal C^0, correct weak C^-, incorrect normal I^0, and incorrect non-serious I^-.

The measures and their thresholds are described in Table 1. We do not focus on the development of performance measures. Issues such as the choice of performance criteria and suitable thresholds are addressed in [12, 30]. Since we aim to study how to compare student models, not how to optimize them, we set the thresholds manually using just a few meta-parameters per measure.

The continuous and time-Q measures are relative with respect to the problem difficulty, as they base their measurements on within-problem median or other quantiles. This makes the measurements informative about the relative skill of a student compared to others, but less about the absolute difficulty of the problem — except for the information about the proportion of incorrect attempts. Time-H is a hierarchical measure that combines median times within problem, level (a group of problems of similar difficulty), and exercise, making the measurements more informative about the difficulties. Time-L exploits the length of the author's solution. It estimates "good time" as the solution length divided by the average speed (the number of lines per second) computed per exercise. In problems with a block-based programming interface, we count only the blocks that correspond to new lines. In problems with scaffolding (initial code), we count the scaffolded lines as only 0.5.

Code-L and code-D base their measurement on the length of the student's solution, compared to the author's solution. Their measurements differ only in the problems with scaffolding; code-D counts just the added lines, not the initial code. The combined measure provides an example of a conjunctive condition on multiple criteria: it requires both a short time *and* a short code to achieve good or excellent performance. The thresholds are the same as in time-L and code-L measures.

3.3 Student Models

We extend the baseline approaches based on mean performance (section 3.1) with the performance measures (section 3.2). The use of arbitrary performance measurements instead of just binary correctness is the only change to the baseline student model; the rest is the same: we average performance measurements to estimate difficulties, we average the deviations between performances and difficulties to estimate skills, and we adjust per-problem average baseline by an amount proportional to the skill to estimate future performance.

To compute the difficulty — or rather easiness — as the mean performance on a given problem, we need to assign numerical values to the performance classes. We assign the values uniformly $(I^0, C^-, C^0, C^+) = (0, 0.5, 1, 1.5)$. We do not assign any value to non-serious attempts; instead, we filter them out to reduce noise in the estimates. Denoting the set of students who seriously attempted problem p as A_p and the observed performance of a student s on a problem p as y_{ps}, the estimated difficulty is:

$$d_p = \frac{1}{|A_p|} \sum_{s \in A_p} y_{ps}$$

The skills are tracked online, i.e., the model updates the estimated skill after each observed attempt. The skill is estimated as the mean deviation between the measured performances and difficulties of the attempted problems, regularized by 5 pseudo-attempts with 0 deviations.[2] For example, after solving two problems with difficulties 1.1 and 1.2, both with excellent performance, the estimated skill would be $((1.5 - 1.1) + (1.5 - 1.2))/(2 + 5) = 0.1$. Denoting the set of seriously attempted problems of a given student by A_s the estimated skill is:

$$\theta_s = \frac{1}{|A_s| + n_{pseudo}} \sum_{p \in A_s} (y_{ps} - d_p)$$

To predict future performance (e.g., log-time) of a student s on a problem p, we start with a per-problem average of the predicted performance aspect $\overline{y}_p$ and adjust it by an amount directly proportional

[2]This can be seen as a simplified version of a Bayesian model with a normal prior on student skill centered at zero.

Table 1: Performance measures and their thresholds. All thresholds are computed per problem. Variables g, h, l, and d are also computed per problem, and they refer to a *good time*, *hierarchical good time*, *length*, and *delta*, respectively. The Q, H, L, and D in the names stand for *quantile*, *hierarchical*, *length*, and *delta*, respectively.

Measure	$C^{\pm}$ thresholds (excellent and weak attempts)	I^{-} threshold (non-serious)
binary	no thresholds; all correct attempts are classified as C^0	—
continuous	no thresholds; performance = $\max(1 - \text{time}/(2 \times \text{median}), 0)$	—
time-Q	0.3 and 0.7 quantiles within the problem	0.05 quantile (correct attempts)
time-H	$0.5h$ and $2h$, where h = exp(mean(log-median-times within the problem, level, and exercise))	$h/3$ (min 15s, max 60s)
time-L	$0.5g$ and $2g$, where g = (solution length)/(median speed)	$g/3$ (min 15s, max 60s)
code-L	l and $1.2l$, where l = length of the author's solution	$0.2l$
code-D	d and $1.2d$, where $d = l$ − scaffolding length	$0.2d$
combined	$0.5g$ and $2g$ for time, l and $1.2l$ for code length	$g/3$ and $0.2l$

to z-normalized skill z_s:

$$\widehat{y}_{ps} = \overline{y}_p + k \cdot z_s$$

The proportionality constant k controls the sensitivity of the predictions to the skill, i.e., how much is the predicted performance higher for the student with skill one standard deviation above average. This constant is shared for all problems in the exercise but differs between exercises and between prediction tasks. We choose the optimal constant k_{ey} for exercise e and prediction target y to minimize the sum of squares between the observed and predicted performance with L2 penalty (using regularization strength $\alpha = 1$) over all attempts A_e in the exercise:

$$k_{ey} = \arg\min_k \left\{ \sum_{p,s \in A_e} (y_{ps} - \overline{y}_p - k \cdot z_s)^2 + \alpha \cdot k^2 \right\}$$

This sum can be minimized using regularized linear least squares regression with a single feature (the z-normalized skill z_s), without intercept, and with targets being the baseline residuals $y_{ps} - \overline{y}_p$.

This simple model could be improved in many ways. We could make the model more flexible by learning non-uniformly spaced values of the performance classes or fitting proportionality constant k for each problem instead of exercise. We could model knowledge components or aggregate performance in a more sophisticated way than just by averaging (e.g., to account for learning). But we stop here and focus on the essential question: How to assess the validity and reliability of such a model? Is the model with a performance measure any "better" than the baseline?

3.4 Data

We use data from six diverse introductory programming exercises from two online learning systems, which allows us to discuss the generalizability of the results. Table 2 provides a summary of the exercises. Each exercise contains 40–100 problems divided into 5–12 levels. After filtering active students (those with at least 10 attempts), an average student attempts 18–32 problems, depending on the exercise. All problems require at most 20 lines of code (or blocks). Most of the attempts are eventually correct (0.87–0.97 depending on the exercise). Per-exercise median times range from 36 seconds (Arrows) to nearly 3 minutes (Python).

Table 2: Exercises and data used in the case study after filtering active students (with at least 10 attempts).

Exercise	Problems	Students	Attempts
Arrows	94	10,300	322,000
Spaceship	85	6,000	119,000
Shapes	54	1,900	35,000
Blockly Turtle	77	6,100	126,000
Python Turtle	46	1,100	23,000
Python	73	800	20,000

Each exercise targets a different age group, from the primary school (Arrows) to the university (Python). The exercises differ in the programming interface, scaffolding, and adaptive behavior. In the Arrows exercise, students put directions directly into the grid world; in the Spaceship, Shapes, and Blockly Turtle, they use block-based programming interface with snapping code blocks [2]; and in the last two exercises, they program in Python. Some problems include scaffolding — an initial program to complete (as in Fig. 1). The Python and Python Turtle are scaffolded heavily, while the Arrows and Spaceship do not use any scaffolding at all. The Spaceship features an adaptive algorithm for problem recommendation [11], while the other exercises recommend the problems in a fixed order (the ordering is occasionally updated based on the collected performance data). In all exercises, the students do not have to follow the problem recommendation.

Due to the limited space, we sometimes include plots with only a subset of these six exercises. In such cases, we comment on the generalizability of the results across the other exercises and provide complete results with all exercises as supplementary materials at github.com/adaptive-learning/lak2021.

4 RELIABILITY

In this section, we present methods to assess the reliability of student models. These methods are applicable to any model that produces a one-dimensional estimate of difficulty for each problem and a one-dimensional estimate of skill after each attempt.

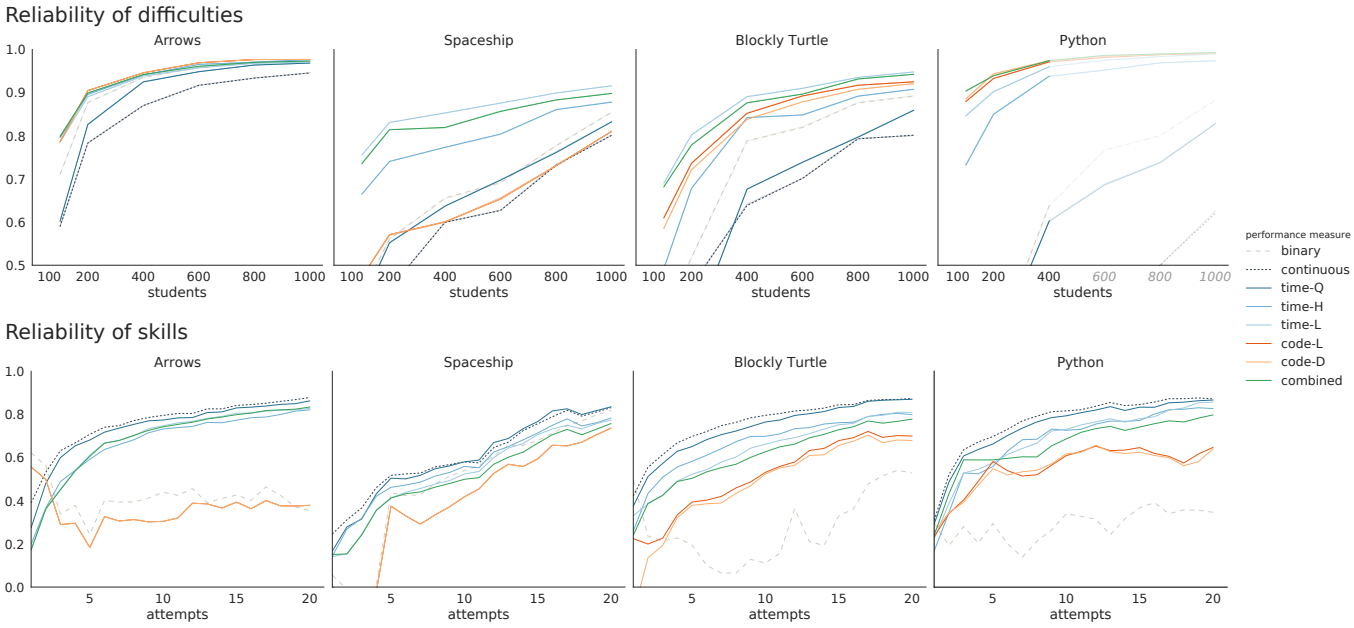

Figure 2: Split-half reliability of difficulties (top) and odd-even reliability of skills (bottom) in 4 exercises. See supplementary materials for other exercises. In the Python exercise, there are not enough students to create two disjoint sets of more than 400 students, so the reliability is, in this case, approximated by drawing students with replacement.

4.1 Split-half Reliability of Difficulties

Our goal is to see how much data we need to collect to obtain a consistent ordering of problems. For this purpose, we adapt the split-half method used in psychometrics [21]. A similar approach was used previously to compare the stability of difficulty indicators such as success rate and the number of executions in a single programming exercise [10].

For a given exercise and the number of students N, we draw two disjoint sets of N students. For each of these groups, we estimate difficulties using the mean performance student model described in section 3.3. This includes computing thresholds of performance measures described in Table 1 using the attempts of the selected students. For example, for the time-Q measure and each problem, we compute the 0.05, 0.3, and 0.7 quantiles of solving times from correct attempts for this problem. These thresholds are used to classify the attempts into the five performance classes, which are fed to the mean performance model. The reliability is then computed as the Spearman correlation coefficient between the two lists of difficulties, which is equivalent to the Pearson correlation coefficient between the induced problem orderings. We repeat the whole computation 10 times (resampling the students each time) and report the mean correlation.

For larger values of N, we did not have enough students in some exercises to create two groups of N students, so we approximated the reliability by drawing students with replacement. We explored the impact of this choice on the values of N for which we could afford to create two disjoint sets. The difference in reliability due to sampling with replacement was small and not optimistically biased (the mean difference between the measurements with and without replacement was -0.0001).

The results vary quite a bit between exercises (Fig. 2). For example, binary and code-based measures behave differently in different exercises, from being ones of the most reliable to ones of the least reliable. The speed of convergence also differ; in particular, Spaceship needs many more students than other exercises to achieve consistent problem ordering. Some trends do generalize across exercises. Time-L and combined measures are robustly reliable once we have a few hundreds of students, and relative performance measures lead to the lowest reliability.

4.2 Odd-Even Reliability of Skills

In contrast to difficulties, student skills change after each attempt. Thus, we do not want to measure how consistent is the skill of a student over time, but rather *at a given time*. To do so, we adapt the split-half method to the situation with temporal patterns, making the two halves as comparable to each other as possible. Specifically, we split all previous attempts of the student (at a given time) into two groups: odd and even. Then we use the student model to compute skills using separately only odd/even attempts. For a given number of attempts N, we compute the Spearman correlation coefficient between the skills computed from N odd and N even attempts for all students.

The results are strikingly different from the reliability of difficulties (Fig. 2). Relative measures (continuous and time-Q) lead to the most reliable skill estimates. This suggests that students' relative performance (who is faster or slower) is a relatively stable attribute. Code-based measures are less reliable than time-based measures in all six exercises, especially in Arrows and Spaceship, where the code length variability is low. The combined measure is also less

reliable than time-based measures, but only slightly. The binary measure provides unreliable skill estimates in all exercises.

There are several limitations of the presented method. Taking only every other attempt creates possibly unrealistic students, which might be especially problematic for models that assume learning. By computing correlation only between skills estimated after the same number of attempts, it fails to capture whether the student model can reliably distinguish between students with a different level of activity. Even if we manage to overcome these limitations, the odd-even reliability of skill estimates needs to be interpreted with caution: it is just a measure of the estimates' consistency, not of their validity. A model can provide highly reliable estimates of skills that are not valid at all, e.g., because it measures some stable characteristics of the student rather than the ability to solve programming problems (a trivial example is the length of the student's name).

5 VALIDITY

The validity of a student model depends on its uses. Although we might not anticipate all future uses of a model, we should always have some of them in mind when validating a model and explicitly declare them. In our case study, we assume that the estimated skills and difficulties are used to order students and problems and to display predicted performance.

We present methods that require only collected performance data. This allows us to validate many student models early, but it does not allow us to provide definitive evidence of their utility. Our methods provide weaker validity evidence through convergent and predictive validity arguments described in section 2.1.2.

5.1 Convergent Validity

Correlation analysis is a simple yet powerful diagnostic tool. Even without any external measurements of difficulties and skills (such as final exam scores), we can still examine correlations between skill/difficulty estimates by various student models. If two models behave similarly across exercises, we do not need to continue to consider both of them. On the other hand, pairs of models with low correlation are worth exploring.

5.1.1 Difficulty Estimates Correlations. In most exercises, the difficulty estimates induce similar problem ordering (measured by Spearman correlation) for all performance measures (Fig. 3). The two code-based measures have a correlation close to 1 across all exercises. The time-based measures have high correlations as well, and the relative performance measures (time-Q and continuous) also highly correlate with the binary performance measure. However, there is a prominent exception, which provides a warning about premature generalization claims: in Python exercise, the continuous measure has a low correlation with all other measures, and the time-L measure is more similar to the code-based measures than to other time-based measures.

Let us illustrate a follow-up exploration informed by the correlation analysis. We zoom to the low correlations in the Python exercise. Fig. 4 shows the difficulty of problems according to all performance measures. There is a clear group of outlying problems with unexpectedly high difficulties according to some of the measures. None of these problems has a suspicious success rate, but students write long programs because they miss the "return value of logic expression" idiom (see example in Fig. 1). This issue is detected by code-based measures and also — since the students need more time than suggested by the short author's solution — by time-L and combined measures. This is an example of utility evidence because these models give us an actionable insight; we might add an explanation or impose limits to force students to use the intended programming idiom.

5.1.2 Skill Estimates Correlations. Fig. 3 also provides correlations for skill estimates. All four time-based measures lead to similar orderings of students, and the same holds for the two code-based measures. The correlation between these two groups is weak in some exercises, especially in Python. The combined measure is a rather robust compromise between the time-based and code-based measures, having a high or moderate correlation with all other measures across all examined exercises. On the other hand, the low correlations between binary and time-based and code-based measures suggest that the binary measure is insufficient to capture these other facets of programming skill.

5.2 Predictive Validity

Predictive validity is based on the ability of the models to predict future performance. In addition to the next attempt correctness, we use three other prediction tasks. These tasks reflect a potential use of a model to flag problems that are predicted as too difficult for the student and to display a personalized challenge on the solving time and the number of lines in the solution.

5.2.1 Prediction Tasks. In each task, the goal is to predict some aspect of performance on the next attempt for a given student and a problem.

correctness The goal is to predict whether the problem will be eventually solved. This is the most common prediction task for student model evaluation.

too difficult The problem is considered too difficult if the student either fails to solve it or solves it with weak performance. The performance is assessed using conjunctive criteria on the solving time and code length as specified by the combined performance measure. Only serious attempts are used for evaluation; i.e., attempts classified as non-serious according to the combined performance measure are filtered out.

solving time The time is capped at 15 minutes to avoid the negative impact of outlying solving times and log-transformed to obtain a less skewed distribution. Only correct attempts are used for evaluation.

code length We use relative length compared to the author's solution, clipped between 1 and 2. Only correct attempts are used for evaluation. In introductory programming, the code length is a proxy for the solution's quality since a longer code is typically associated with an unapplied programming pattern (e.g., iteration, function abstraction). This prediction task is specific to introductory programming, but other proxies for the solution's quality are applicable in other domains.

5.2.2 Within-Fold RMSE Normalization. To obtain a stable estimate of models' performance and its variability, we use 10-fold student-level cross-validation [29], reporting mean RMSE and standard

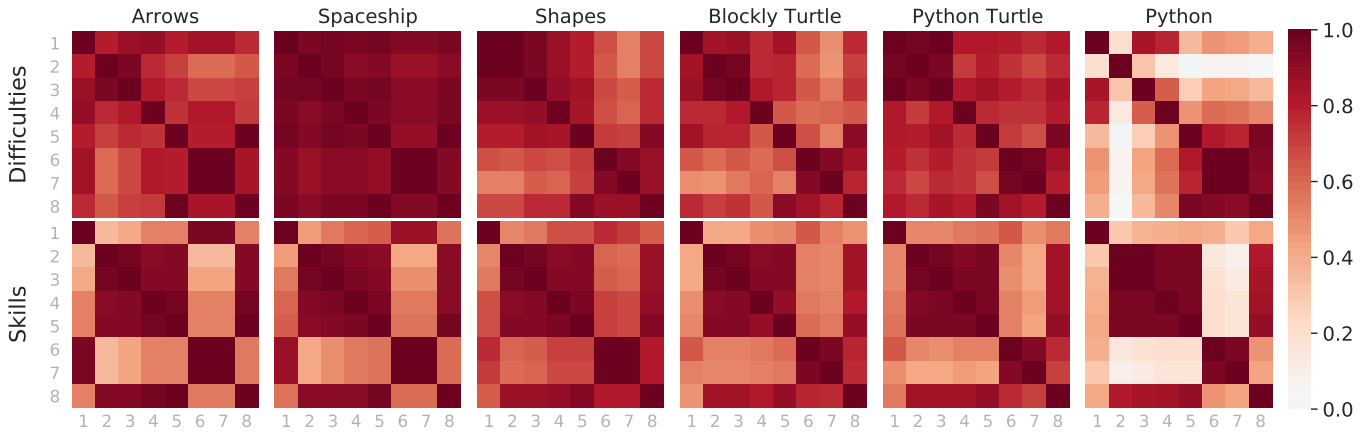

Figure 3: Correlations between difficulties and skills measured by 8 performance measures described in section 3.2, encoded as follows: 1 – binary, 2 – continuous, 3 – time-Q, 4 – time-H, 5 – time-L, 6 – code-L, 7 – code-D, 8 – combined. See supplementary materials for a more detailed version of this plot with labels and correlation values.

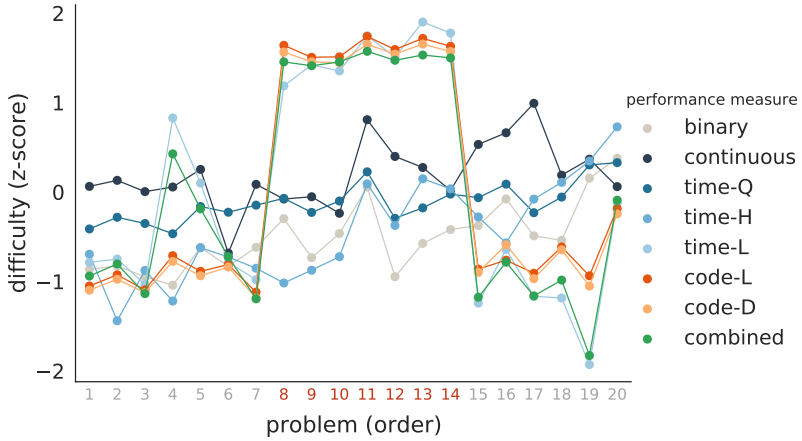

Figure 4: The difficulty of the first 20 items in the Python exercise. The difficulty is z-normalized to allow comparison between different performance measures. The highlighted items 8–14, which are disproportionately difficult, belong to the same level.

deviation across folds. As noted in a recent study on the impact of methodological choices [13], the variability in RMSE might be dominated by the differences between the data in individual folds (which we are not interested in) instead of the differences between models (which we are interested in). To reduce the impact of between-folds variability, we normalize RMSE within each fold, as suggested in the study [13]. Within each fold, we calculate RMSE relative to the per-problem average baseline:

$$rRMSE = \frac{RMSE(\widehat{y}_{sp})}{RMSE(\overline{y}_p)} = \frac{\sqrt{\frac{1}{n}\sum_{sp}(y_{sp} - \widehat{y}_{sp})^2}}{\sqrt{\frac{1}{n}\sum_{sp}(y_{sp} - \overline{y}_p)^2}}$$

We include both absolute (unnormalized) and relative (normalized) RMSE in supplementary materials. Relative RMSE is less noisy, making it easier to see which models perform consistently better than others. Relative RMSE is also easier to compare across exercises and prediction tasks, as it can be interpreted as relative performance compared to the baseline, with the value of 1 always corresponding to the baseline performance. Relative RMSE is closely related to the commonly used coefficient of determination R^2, which can be similarly interpreted as relative improvement of a *full* model over a *reduced* model [1]:

$$R^2 = 1 - \frac{SS_{full}}{SS_{reduced}} = 1 - \frac{\sum_{sp}(y_{sp} - \widehat{y}_{sp})^2}{\sum_{sp}(y_{sp} - \overline{y})^2}$$

Relative RMSE differs from R^2 by not subtracting the relative error from 1 (the lower the *rRMSE*, the better the model), monotonically transforming the sums of squares ($RMSE = \sqrt{SS/n}$), and normalizing the error by a stronger baseline (the per-problem average $\overline{y}_p$ instead of the global average $\overline{y}$).

It is still worth taking a look at the absolute RMSE as well. For example, in the code length prediction task for the Arrows and Spaceship exercises, all models achieve RMSE close to 0 due to low variability in the length of possible solutions. The low RMSE explains why all models perform similarly; there is simply no room

for improvement over the baseline. The evaluation of predictive validity should explore other choices that could impact the results and their interpretation, e.g., the choice of metric and its averaging across students or problems [13].

5.2.3 Results. Fig. 5 shows the results for Blockly Turtle. Although the specific values differ, the trends are similar across all exercises, except for code-based measures and code length prediction task in Arrows and Spaceship. As already discussed, these two exercises have low variability in the length of the submitted solutions.

Not surprisingly, to predict response time, it is best to use time-based measures, to predict code length, it is best to use code-based measures, and to predict too difficult problems, it is best to use performance measure whose criteria for weak performance matches the criteria for detecting too difficult problems. Even the binary correctness is best predicted by the binary performance measure; using additional performance data seems even to harm the performance. For each measure, we can construct a prediction task in which it is the best — this is exactly what makes a fair evaluation difficult.

Since using any single metric would be misleading, we recommend to look at multiple prediction tasks, which represent proxies for various potential uses of a model. In our case, the combined measure seems to be a robust compromise, improving RMSE compared to the binary performance measure for all prediction tasks except for the correctness prediction.

6 DISCUSSION

The aim of this paper is to contribute to the methodology of student model evaluation. We explored methods for assessing the validity and reliability of student models for problem-solving activities. As a side product, our case study demonstrates that binary correctness is insufficient in the problem-solving context and that using other aspects can increase validity and reliability of the student models.

6.1 Evaluation of Student Models

The next answer correctness prediction task is insufficient to validate student models in the problem-solving context. In the presented case study, if we evaluated the models using just the next answer correctness, we would conclude that the model with binary correctness input is the best option and that non-binary performance measures not only do not help, they are even harmful (Fig. 5). This interpretation would be misleading; a more comprehensive evaluation shows that non-binary performance measures do increase the validity of the student model.

We conjecture that the misleading evaluation stems from two properties of the binary correctness in our context: class imbalance (few attempts are incorrect) and noise (incorrect attempts are frequently caused by other factors than low skill). Accounting for the imbalance in the evaluation (e.g., using average precision metric) and reducing the noise (e.g., by filtering non-serious attempts detected by a previously validated performance measure) could improve the next answer correctness prediction task and make it actually informative.

But even then, there would still be a more fundamental limitation of the next answer correctness. To assess validity, we need to consider the uses of the model. In the problem-solving context, we care about other aspects of performance than just the ability to produce correct solutions; the students' speed and the quality of their solutions are often important as well. This would not be an issue if the ability to produce correct solutions would highly correlate with the other performance aspects; however, our exploration of the convergence validity (Fig. 3) suggests that, in some cases, the correlation is weak.

We do not offer an easy solution to this fundamental limitation of the next answer correctness; we do not believe there is an easy solution. Instead, we advocate a more comprehensive evaluation of student models since any single evaluation metric can be misleading, especially if the model is used for more than one purpose. For example, we have seen that a performance measure that leads to reliable skill estimates (time-Q) can lead to unreliable difficulty estimates, and vice versa.

Drawing inspiration from psychometrics, we proposed several methods for assessing validity and reliability of student models: split-half reliability of difficulty estimates, odd-even reliability of skill estimates, convergent validity comparing estimates of multiple student models, and predictive validity using multiple prediction tasks. All these methods use only collected performance data. However, there are other potential sources of validity evidence: thought processes of the students observed through think-aloud, an inspection of the estimated skills by teachers, external skill estimates such as exam results and post-test performance, and utility of an improved student model in a randomized control trial. These other sources of evidence require substantially more effort, so an important question is when we need them and when a comprehensive evaluation of the collected performance data (as done in this paper) is sufficient.

6.2 Utility of Performance Measures

The same reasons that make binary correctness an insufficient prediction task also make it an insufficient input to the student models for problem-solving activities. Due to the class imbalance and noise, we need many attempts for the estimates of skills and difficulties to become reliable. This is a concern especially for skill estimates since the number of attempts per student is low (Fig. 2).

Reliability aside, the validity of the skill and difficulty estimates computed from just binary correctness is also questionable. If we care about other performance aspects, we obtain substantially different skill estimates (Fig. 3). Difficulty estimates are more consistent, but in a few cases, important differences do exist. For example, we have seen how binary correctness hides a systematic issue in code quality that deserves an intervention (Fig. 4). Finally, skills based on binary correctness were not much useful for predicting too difficult problems, solving times, or code length (Fig. 5).

The reliability and validity of student models can be increased using performance measures based on criteria relevant for a given exercise. The best performance measure depends on the specific exercise and use, but it would be convenient to have as few measures as possible. A straightforward combined measure (conjunction on all relevant performance criteria) seems to be a robust compromise.

The generalizability of the presented observations about performance measures requires further work. For introductory programming, we believe that our experiments provide good coverage of

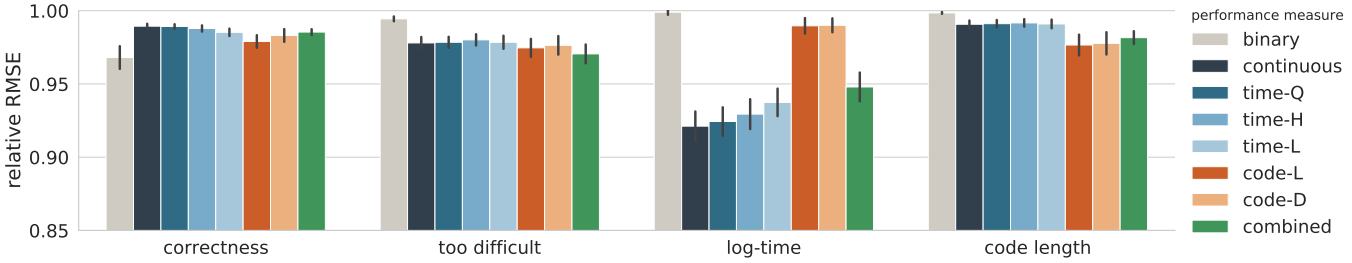

Figure 5: Predictive accuracy of performance measures in Blockly Turtle exercise on four prediction tasks. The height of the bars and the length of the vertical black lines show the mean and standard deviation of the relative RMSE across 10 folds. The lower the relative RMSE, the better the model. Results for the other exercises are included in supplementary materials.

the domain — we used data from six types of exercises covering a wide range of introductory programming activities. As a next step, it would be useful to explore other domains to understand better when are non-binary performance measures beneficial. We believe that in many contexts, advancing methods for designing and evaluating performance measures have much more potential than the usage of more complex student models (e.g., Bayesian models or models based on neural networks).

7 CONCLUSION

The next answer correctness is an overused prediction task, which is insufficient to validate student models. The solution is a more comprehensive evaluation, e.g., using multiple prediction tasks, comparing estimates of multiple student models, and exploring the reliability of the estimates. Binary correctness of answers is an overused performance measure, which is insufficient to construct valid and reliable student models for problem-solving activities. A solution is to employ discrete performance measures based on performance criteria relevant for the given exercise.

SUPPLEMENTARY MATERIALS

Results for all exercises are available as supplementary materials at github.com/adaptive-learning/lak2021.

REFERENCES

[1] Richard Anderson-Sprecher. 1994. Model comparisons and R2. *The American Statistician* 48, 2 (1994), 113–117.
[2] David Bau, Jeff Gray, Caitlin Kelleher, Josh Sheldon, and Franklyn Turbak. 2017. Learnable Programming: Blocks and Beyond. *Commun. ACM* 60, 6 (2017), 72–80.
[3] Joseph E Beck and Xiaolu Xiong. 2013. Limits to accuracy: How well can we do at student modeling. In *Proceedings of Educational Data Mining*.
[4] Yoav Bergner. 2017. Measurement and its uses in learning analytics. *Handbook of learning analytics* 35, 2 (2017).
[5] Peter Brusilovsky, Charalampos Karagiannidis, and Demetrios Sampson. 2004. Layered evaluation of adaptive learning systems. *International Journal of Continuing Engineering Education and Life Long Learning* 14, 4-5 (2004), 402–421.
[6] David A Cook and Thomas J Beckman. 2006. Current concepts in validity and reliability for psychometric instruments: theory and application. *The American journal of medicine* 119, 2 (2006), 166–e7.
[7] Lee J Cronbach. 1951. Coefficient alpha and the internal structure of tests. *Psychometrika* 16, 3 (1951), 297–334.
[8] Holli A DeVon, Michelle E Block, Patricia Moyle-Wright, Diane M Ernst, Susan J Hayden, Deborah J Lazzara, Suzanne M Savoy, and Elizabeth Kostas-Polston. 2007. A psychometric toolbox for testing validity and reliability. *Journal of Nursing Scholarship* 39, 2 (2007), 155–164.
[9] Christopher Doble, Jeffrey Matayoshi, Eric Cosyn, Hasan Uzun, and Arash Karami. 2019. A data-based simulation study of reliability for an adaptive assessment based on knowledge space theory. *International Journal of Artificial Intelligence in Education* 29, 2 (2019), 258–282.
[10] Tomáš Effenberger, Jaroslav Čechák, and Radek Pelánek. 2019. Measuring Difficulty of Introductory Programming Tasks. In *Proceedings of Learning at Scale*. ACM.
[11] Tomáš Effenberger and Radek Pelánek. 2018. Towards making block-based programming activities adaptive. In *Proceedings of Learning at Scale*. ACM, 13.
[12] Tomáš Effenberger and Radek Pelánek. 2019. Measuring Students' Performance on Programming Tasks. In *Proceedings of Learning at Scale*. ACM.
[13] Tomáš Effenberger and Radek Pelánek. 2020. Impact of Methodological Choices on the Evaluation of Student Models. In *Proceedings of Artificial Intelligence in Education*. Springer, 153–164.
[14] Luke Glenn Eglington and Philip I Pavlik. 2019. Predictiveness of Prior Failures is Improved by Incorporating Trial Duration. *Journal of Educational Data Mining* 11, 2 (2019), 1–19.
[15] José González-Brenes and Yun Huang. 2015. Your model is predictive - but is it useful? theoretical and empirical considerations of a new paradigm for adaptive tutoring evaluation. In *Proceedings of Educational Data Mining*.
[16] Kilem L Gwet. 2014. *Handbook of inter-rater reliability: The definitive guide to measuring the extent of agreement among raters*. Advanced Analytics, LLC.
[17] Wynne Harlen. 2005. Trusting teachers' judgement: Research evidence of the reliability and validity of teachers' assessment used for summative purposes. *Research papers in education* 20, 3 (2005), 245–270.
[18] Yun Huang, José P González-Brenes, Rohit Kumar, and Peter Brusilovsky. 2015. A Framework for Multifaceted Evaluation of Student Models. In *Proceedings of Educational Data Mining*.
[19] Michael T Kane. 2013. Validating the interpretations and uses of test scores. *Journal of Educational Measurement* 50, 1 (2013), 1–73.
[20] S Klinkenberg, M Straatemeier, and HLJ Van der Maas. 2011. Computer adaptive practice of Maths ability using a new item response model for on the fly ability and difficulty estimation. *Computers & Education* 57, 2 (2011), 1813–1824.
[21] G Frederic Kuder and Marion W Richardson. 1937. The theory of the estimation of test reliability. *Psychometrika* 2, 3 (1937), 151–160.
[22] Mary L McHugh. 2012. Interrater reliability: the kappa statistic. *Biochemia medica: Biochemia medica* 22, 3 (2012), 276–282.
[23] Samuel Messick. 1993. Foundations of validity: Meaning and consequences in psychological assessment. *ETS Research Report Series* 1993, 2 (1993), i–18.
[24] Korinn S Ostrow and Neil T Heffernan. 2018. Testing the Validity and Reliability of Intrinsic Motivation Inventory Subscales Within ASSISTments. In *Proceedings of Artificial Intelligence in Education*. Springer, 381–394.
[25] Alexandros Paramythis, Stephan Weibelzahl, and Judith Masthoff. 2010. Layered evaluation of interactive adaptive systems: framework and formative methods. *User Modeling and User-Adapted Interaction* 20, 5 (2010), 383–453.
[26] Miranda C Parker, Mark Guzdial, and Shelly Engleman. 2016. Replication, validation, and use of a language independent CS1 knowledge assessment. In *Proceedings of International Computing Education Research*. 93–101.
[27] Radek Pelánek. 2015. Metrics for Evaluation of Student Models. *Journal of Educational Data Mining* 7, 2 (2015), 1–19.
[28] Radek Pelánek. 2017. Bayesian knowledge tracing, logistic models, and beyond: an overview of learner modeling techniques. *User Modeling and User-Adapted Interaction* 27, 3-5 (2017), 313–350.
[29] Radek Pelánek. 2018. The details matter: methodological nuances in the evaluation of student models. *User Modeling and User-Adapted Interaction* 28 (2018), 207–235. Issue 3. https://doi.org/10.1007/s11257-018-9204-y
[30] Radek Pelánek and Tomáš Effenberger. 2020. Beyond Binary Correctness: Classification of Students' Answers in Learning Systems. *User Modeling and User-Adapted Interaction* 27, 1 (2020), 89–118.

[31] Radek Pelánek and Petr Jarušek. 2015. Student Modeling Based on Problem Solving Times. *International Journal of Artificial Intelligence in Education* (2015), 1–27.
[32] Radek Pelánek and Jiří Řihák. 2018. Analysis and design of mastery learning criteria. *New Review of Hypermedia and Multimedia* 24, 3 (2018), 133–159.
[33] Allison Elliott Tew and Mark Guzdial. 2011. The FCS1: a language independent assessment of CS1 knowledge. In *Proceedings of ACM Technical Symposium on Computer Science Education*. 111–116.
[34] Ian Utting, Allison Elliott Tew, Mike McCracken, Lynda Thomas, Dennis Bouvier, Roger Frye, James Paterson, Michael Caspersen, Yifat Ben-David Kolikant, Juha Sorva, et al. 2013. A fresh look at novice programmers' performance and their teachers' expectations. In *Proceedings of Innovation and Technology in Computer Science Education*. 15–32.
[35] Eric G Van Inwegen, Seth A Adjei, Yan Wang, and Neil T Heffernan. 2015. Using Partial Credit and Response History to Model User Knowledge. In *Proceedings of Educational Data Mining*.
[36] Yutao Wang and Neil Heffernan. 2013. Extending knowledge tracing to allow partial credit: Using continuous versus binary nodes. In *Proceedings of Artificial Intelligence in Education*. Springer, 181–188.
[37] Yutao Wang, Neil T Heffernan, and Joseph E Beck. 2010. Representing Student Performance with Partial Credit. In *Proceedings of Educational Data Mining*. 335–336.