Learner Modeling for Integration Skills

Yun Huang
Intelligent Systems Program, University of Pittsburgh
Pittsburgh, PA, USA
yuh43@pitt.edu

Julio Guerra-Hollstein
Instituto de Informática & School of Information Sciences,
Universidad Austral de Chile & University of Pittsburgh
Valdivia, Chile
jguerra@inf.uach.cl

Jordan Barria-Pineda
School of Information Sciences, University of Pittsburgh
Pittsburgh, PA, USA
jab464@pitt.edu

Peter Brusilovsky
School of Information Sciences & Intelligent Systems
Program, University of Pittsburgh
Pittsburgh, PA, USA
peterb@pitt.edu

ABSTRACT

Complex skill mastery requires not only acquiring individual basic component skills, but also practicing integrating such basic skills. However, traditional approaches to knowledge modeling, such as Bayesian knowledge tracing, only trace knowledge of each decomposed basic component skill. This risks early assertion of mastery or ineffective remediation failing to address skill integration. We introduce a novel integration-level approach to model learners' knowledge and provide fine-grained diagnosis: a Bayesian network based on a new kind of knowledge graph with progressive integration skills. We assess the value of such a model from multifaceted aspects: performance prediction, parameter plausibility, expected instructional effectiveness, and real-world recommendation helpfulness. Our experiments based on a Java programming tutor show that proposed model significantly improves two popular multiple-skill knowledge tracing models on all these four aspects.

KEYWORDS

learner modeling; knowledge tracing; programming patterns; skill integration; Bayesian network

1 INTRODUCTION

Complex skill mastery requires not only the acquisition of individual basic component skills, but also practice in integrating such component skills with one another [1, 25]. Despite this recognized need, learner models in modern intelligent tutoring systems [2, 34, 43] have predominantly focused on teaching and assessing individual basic component skills (following a prerequisite-outcome ordering), yet haven't explicitly or systematically monitored the level of knowledge that is present in integrating basic component skills. For example, the most popular learner modeling approach, Bayesian knowledge tracing (BKT) [10], is based on decomposing domain knowledge into individual basic component skills, and assuming that each basic component skill takes full responsibility for overall problem-solving performance. Even some more advanced learner models [9, 14, 26, 47] that address the responsibility assignment in a more sophisticated probabilistic method still decompose domain knowledge into individual basic component skills, and continue to ignore any possible integration or interactions among them. Defining and assessing skills without explicit and systematic specification of integration skills makes knowledge engineering and modeling easier (since the integration or interaction space can be much bigger), but it risks an early assertion of mastery by merely observing student success in basic component skill practices.

Recent research on algebra has provided empirical evidence to demonstrate that there is additional knowledge related to specific skill combinations; in other words, the knowledge about a set of skills is greater than the "sum" of the knowledge of individual skills [17], some skills must be integrated with other skills to produce behavior [25]. For example, students were found to be significantly worse at translating two-step algebra story problems into expressions (e.g., 800-40x) than they were at translating two closely matched one-step problems (with answers 800-y and 40x) [17]. This indicates that, at least in some domains, it is necessary to pay specific attention to skill integration in modeling student knowledge. Computer programming is arguably one of these domains. Research on computer science education and pedagogy has long argued that knowledge of a programming language can't be reduced to a sum of knowledge about different programming constructs, since there are many stable combinations or patterns (also known as schemas or plans) that must be taught and practiced [13, 41].

While the existence of integrating skills have been acknowledged in both cognitive science and teaching practices, there's almost no existing work that explores modeling integration skills in learner models. This involves changes in both parts of a learner model: its underlying *skill model* (specifying the skills in the domain, skills required by each problem and relationships among skills) and its *modeling approach* given a skill model (specifying the use of either a flat or a hierarchical structure in a Bayesian network). In this work, we introduce a novel integration-level approach to model learners' knowledge and provide fine-grained diagnosis: a Bayesian network based on a new kind of knowledge graph with progressive integration skills. We also introduce a novel multifaceted evaluation

UMAP'17, July 9–12, 2017, Bratislava, Slovakia
© 2017 ACM. 978-1-4503-4635-1/17/07…$15.00
DOI: http://dx.doi.org/10.1145/3079628.3079677

framework that includes assessments of performance prediction, parameter plausibility, expected instructional effectiveness, and real-world recommendation helpfulness. Following this framework, we perform an extensive evaluation and demonstrate that our model significantly improves upon two popular multiple-skill learner models in all four aspects. The remaining part of the paper starts with an extensive review, and then gives an introduction of the new modeling approach, the multifaceted evaluation framework, the results, and ends with the conclusion.

2 RELATED WORK

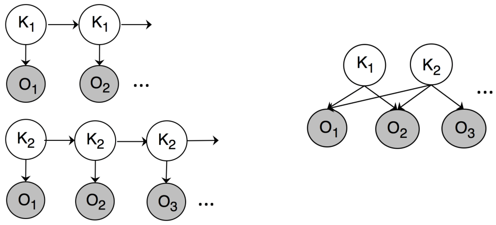
2.1 Knowledge Tracing for Multiple Skills

One major functionality of learner modeling is to maintain explicit knowledge estimations of domain skills over time. This process is called *knowledge tracing* [10, 11]. Such explicit knowledge estimations are critical for understanding learners' cognitive states and providing targeted remediation. In this work, we focus on learner models with explicit knowledge estimations on fine-grained skills with levels at which remediation can directly operate.

While many models have been constructed for single-skill knowledge tracing, multiple-skill knowledge tracing has long been a challenge. Each observed assessment unit (i.e., an observation, a step, an item, or a problem) involving multiple skills poses substantial challenges in assigning responsibility (credit or blame) to each individual skill for practical performance. Currently, there are two main streams of work that address this problem.

The first stream of work [14, 16, 26, 47] converts the many-to-many skill to item mapping into a one-to-many (or one-to-one) mapping during the training process, where a classic *Bayesian knowledge tracing* [10] paradigm can be applied[1]. Such models separately train each individual skill using a hidden Markov model which assumes that each skill is fully responsible for performance by duplicating the observations for each of the required skills (Figure 1a). This oversimplifies the responsibility assignment issue, but reduces modeling complexity. The parameters are the probability of initially knowing the skill (*init*), the probability of transferring from an unlearned to a learned state (*learn*), the probability of accidentally failing a known item (*slip*), and the probability of correctly answering an item by chance (*guess*). Variants on these models differ in how they conduct prediction and updating during the predicting phrase. One variant we consider in this paper is called *weakest knowledge tracing (WKT)*, which has been shown to have the best predictive performance on several datasets, as compared with other variants [14, 16, 47]. It takes the minimum of the predicted probability of success among involved skills as the final prediction. This model only updates the knowledge of the weakest skill when observing an incorrect response, and updates all skills otherwise. This serves as a low baseline for our later experiments.

The second stream of work [9, 31, 32] maintains the many-to-many skill-to-item mapping in both the training and predicting phrases (Figure 1b). Each individual skill is assigned responsibility according to the conditional probability table and the Bayesian rule. Here, we focus on the models that assume a conjunctive relationship among skills (i.e., success in an item requires knowing all required skills) and that use noisy-AND gates for modeling the conjunctive relations. Noisy-AND gates were commonly used in many prior studies [6, 9, 44], due to their linear rather than exponential complexity in inference. We call such a model that uses item level noisy-AND gates with a flat structure among skills *conjunctive knowledge modeling (CKM)* and use it as a high baseline in this paper. These models closely relate to the popular psychometric model DINA [23], but they ultimately conduct dynamic knowledge estimation rather than static ability estimation. Each noisy-AND gate uses a *slip* parameter to capture the the probability of accidentally failing a known item, and a *guess* parameter to capture the probability of correctly answering an item by chance. In this avenue of work, some use a hierarchical structure among skills, yet focus on either the prerequisite relations among intrinsically different skills [6, 9, 24], or granularity relationships (including competency-based networks) [8, 9, 32, 33, 35], where higher level nodes denote more abstract, more general, aggregated skills at which level remediation doesn't directly operate. They are substantially different from the integration relationship that we model and the level of remediation that we target here. Also, most work doesn't model transition probabilities across time steps, due to the complexity imposed by the skill model in an arbitrary practice order.



(a) WKT

(b) CKM

Figure 1: Main knowledge tracing models for multiple skills. O nodes represent binary observed student performance and K nodes represent binary latent skill knowledge levels.

2.2 Learner Model Evaluation

Evaluation methodology has been considered an important research topic in the field of user-adaptive systems. In the early years, the cumulative value of personalization was assessed in a user study by comparing performance achieved with a personalized system against performance achieved in a similar system that had personalization disabled [7]. A similar approach was used to compare two versions of personalization. More recently, it has been recognized that personalization is a result of several stacked processes, user or learner modeling. and the proponents of *layered evaluation* argued that holistic empirical evaluation should be complemented by approaches that independently assess each layer [5, 37].

Nowadays, a separate data-driven assessment of learner modeling has become popular in the field of adaptive educational systems, with predictive performance evaluation on held-out datasets [10] emerging as the gold standard. However, several researchers have recently expressed concerns about using prediction performance as the *only* approach. It has been shown that a highly predictive model can be useless for adaptive tutoring [15], and can have low parameter plausibility, as shown by our previous framework *Polygon* [20]. A recent *learner effort-outcomes paradigm (LEOPARD)* [15] offers a

[1] Note that recent work [16, 47] still conducts single-skill knowledge tracing on coarse-grained skill levels and treats multiple fined-grained skills (subskills) as features.

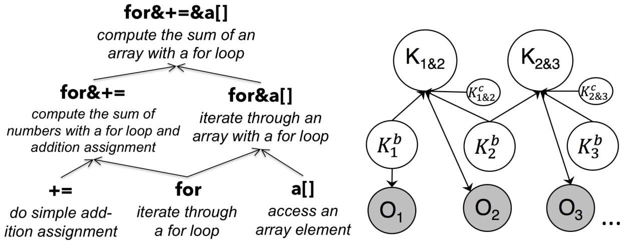
general framework for empirically evaluating learner models for adaptive tutoring. However, this framework is limited to single-skill practice learner models. Our evaluation framework presented below combines empirical evaluation with data-driven evaluation and extends both of our previous frameworks, LEOPARD and Polygon [15, 20], to the evaluation of complex skill practices.

2.3 Patterns in Programming Expertise

Experts in the area of psychology of programming have long argued that programming patterns form an important part of programming expertise [13, 41]. Most actively, all kinds of programming patterns, such as plans, techniques, templates, and "cliches", were used by researchers in the area of intelligent tutoring systems to support intelligent analysis of student programs [22, 30, 36, 46]. While such intelligent debuggers are able to both recognize and diagnose pattern errors, they do not maintain a model of student knowledge at the pattern level. Learner models on the level of patterns were first introduced by Brusilovsky [4], who used expert-suggested construct pairs as skills for problem sequencing; Weber [45] applied larger programming "episodes" as skills for adaptive recommendation of programming examples [46]. The more advanced episodic model has never been expanded or ported to another language, due to its complexity and high demand for knowledge engineering. In contrast, the simpler pair-based approach has been used in a few follow-up projects [27]. This paper continues to explore the simple pair-based representation, but applies modern probabilistic approach to conduct learner modeling, which also has the flexibility to incorporate more complex representations.

3 MODELING SKILL INTEGRATION

In this section, we introduce our two innovations for modeling skill integration: a new type of knowledge graph, and the Bayesian network built based on such a knowledge graph.


for&+=&a[]
compute the sum of an array with a for loop
for&+=
compute the sum of numbers with a for loop and addition assignment
for&a[]
iterate through an array with a for loop
+=
do simple addition assignment
for
iterate through a for loop
a[]
access an array element
$K_{1\&2}$
$K_{2\&3}$
$K^c_{1\&2}$
$K^c_{2\&3}$
K^b_1
K^b_2
K^b_3
O_1
O_2
O_3
...

(a) Integration Graph (b) CKM-HI

Figure 2: Integration graph and CKM-HI Bayesian network. O nodes represent binary observed student performance and K nodes represent binary latent skill knowledge levels.

3.1 Integration Graph

We propose an *integration graph* (Figure 2a) to represent the integration relationships among skills. The lowest level consists of basic component skills and the higher level consists of integration skills requiring integrating lower levels' skills. For example, in the Java programming domain in which we are working, experts identified an integration skill of "computing the sum of numbers with a for loop and addition assignment" based on more basic component skills, such as "doing simple addition assignment" and "iterating through a for loop". Nodes and edges are correspondingly created in the graph. Simplified symbolic notations are used to label skills in the graph. For example, "for&+=" is used as the label for the integration skill. The labeling (coding) schema for the skill model, the notation of the skill nodes, and the depth of the graph can be designed according to the characteristics of the domain. In this work, we demonstrate one successful example of constructing and utilizing such a graph.

3.2 Learner Model Structure and Parameters

A Bayesian network is a natural way to use an integration graph that maintains its structure and each node's meaning. We propose a Bayesian network that we call *conjunctive knowledge modeling with hierarchical integration skills (CKM-HI)* based on an integration graph. Figure 2b shows the network structure of CKM-HI for modeling pair-based integration skills. Built based on CKM, CKM-HI also uses noisy-AND gates to model the relation between skills and items. This choice is suitable when each problem only has one solution that requires students know all of the underlying skills, and is necessary when each problem requires many skills (e.g., more than 3) to reduce computational complexity. This is the case for our dataset and many other (programming) tutoring systems. However, the core of CKM-HI is less about the probability distribution that we choose to model the skill to item relationship, but more about how we represent integration skills, the skill to skill and the skill to item relationships, which are explained as follows:

- Basic component skills and integration skills are represented by different nodes, so that the target of remediation can be clearly identified. Basic component skill nodes model the basic understanding and application of a component skill. For example, in Figure 2b, K_2^b represents the basic understanding and application of simple iteration through a for loop. Integration skill nodes represent the level of integrating component skills. For example, $K_{2\&3}$ represents the skill of integrating *for* and *a[]* for iterating through an array with a for loop. This is different from traditional WKT or CKM models, where these two kinds of skills can't be differentiated, and effectiveness of remediation could be reduced.
- Latent skills are organized in a hierarchical way. The lowest level consists of basic component skills, and the higher levels consist of integration skills requiring integrating lower-level skills. This hierarchical structure allows efficiency and accuracy in inference: once a student has mastered an integration skill, they should already have mastered its component skills. This avoids tedious assessment and the over-practicing of basic component skills.
- Integration skills are directly connected to items, and edges from basic component skills to items are removed if their integration skills are required. For example, O_2 requires the integration skill $K_{1\&2}$, so the edges from K_1^b to O_2, from K_2^b to O_2 are removed. In this way, remediation can directly operate at integration skill levels. This is different from granularity-based networks [8, 9, 32] including competency-based networks [33, 35] where higher level nodes represent aggregation (not integration) of lower level skills and aren't directly connected to items. As a result, remediation can't directly operate at their higher levels.

- Each integration skill node has its own parent node (e.g., $K^c_{2\&3}$ for $K_{2\&3}$), which denotes the level of cognitive load (or familiarity) that is required to conduct the integration. This means that the level of integration depends not only on the levels of basic component skills, but also on the cognitive load (or familiarity) in each specific integration. If we remove such nodes, the level of integration will be fully determined by the levels of basic component skills, which results in the definition of basic component skills encompassing integration. This increases the difficulty in diagnosing whether students should improve their basic component skills or their integration skills.
- Multinomial (here, binomial) distributions are used for integration skill nodes, where basic component skill nodes and cognitive load nodes are specified as parents to denote prerequisite relations. One can argue that noisy-AND gates could be used, yet multinomial distributions allow component skills to have different importance to integration, and have generated a better model than noisy-AND gates by our cross-validation experiments. However, for highly complex integration skills requiring many components, we can switch to noisy-AND gates for tractability.

To fully determine the network structure, we need to specify skill-to-item and skill-to-skill mappings. Two experts in Java programming performed the labeling following the same schema (with conflicts resolved), which was validated further in Section 5.1.2.

We learn all parameters from the data. We use an expectation maximization algorithm, since the network involves latent variables, and the standard junction-tree algorithm to conduct inference.

Admittedly, there are alternate ways to formulate the model; but in this work, we don't primarily aim to find the best model for modeling integration skills. Instead, we demonstrate the feasibility and value by using a reasonable way of modeling.

3.3 Performance Prediction and Dynamic Knowledge Update

After learning parameters from data (given the network structure), we apply the network for predicting problem performance and infer the knowledge level of each skill at each practice opportunity. For each student's first practice, the network uses the same prior probabilities (obtained by *init* parameters) for latent skill nodes (and cognitive load nodes) to predict their performance and update their levels of knowledge; after observing different students' practice sequences, the network starts to differentiate among students by maintaining different up-to-date knowledge estimates. In order to achieve this, CKM-HI follows the same dynamic BN roll-up mechanism as in [9]: it uses posterior knowledge probabilities conditioned on historical observations as the priors for the next time steps. Currently, CKM-HI doesn't model the transition probabilities of latent skills between time steps. Similar to [9, 32], we argue that the change in knowledge estimates is mainly determined by the new evidence (i.e., observed performance), since the learning gain from each practice would be ultimately translated into an observed performance that serves as the evidence for updating knowledge beliefs. Such a mechanism indeed can achieve good performance, as shown in the latter results section. We leave the incorporation of learning dynamics, which is a non-trivial task for such a network with hierarchical structure among latent variables, for future studies.

4 MULTIFACETED EVALUATION

In this section, we introduce our multifaceted evaluation for learner models. There is growing concern of using performance prediction metrics as the only evaluation approach [15, 20]. Our multifaceted evaluation also examines the parameters and knowledge inference quality, under both data-driven and user study settings.

4.1 Performance Prediction

We report two popular prediction metrics used in evaluating learner (skill) models, root mean squared error (RMSE) and area under the receiver operating characteristic curve (AUC), based on a suggestion from a recent paper [39] that raised a concern in using only AUC for evaluation learner (skill) models[2].

4.2 Parameter Plausibility

Parameter plausibility has become an important aspect of examining learner models. It determines the accuracy and reliability of the latent knowledge inference. The foundational assumption behind knowledge-tracing learner models is that knowing the required skills generally leads to correct answers, and that not knowing the required skills generally leads to incorrect answers [3]. However, since the data usually contains noise (uncertainty), guess and slip parameters are introduced to tolerate exceptions where students still succeed, even if they are in an unlearned state, and where they fail, even if they are in a learned state. Such parameters should be relatively low; otherwise, they contradict our foundational assumptions and will generate inaccurate knowledge estimations [3, 38, 42]. One primary source of high guess or slip parameters is from an improper skill model. For example, if a skill model fails to identify several difficult skills of an item and students mostly reach a high knowledge level of the identified easier skills when facing this item, the learner model will use high slip parameters to explain the high ratio of incorrect performance that is observed in the data. Here, we compute the average value of guess or slip across skills or items for each model as the metrics that indicate parameter plausibility and prefer smaller values. These two metrics extend the parameter plausibility metrics proposed in our Polygon framework [20].

4.3 Expected Instructional Effectiveness

The ultimate goal of a learner model is to improve instructional effectiveness, which mainly consists of two aspects: 1) whether students can reach high knowledge levels for the targeted skills; and 2) how much effort students need to exert in order to reach the desired knowledge levels. The above-mentioned performance prediction and parameter plausibility metrics fail to give direct information on these two aspects. So, we propose a new evaluation dimension extending our recent *Learner Effort-Outcomes Paradigm (LEOPARD)* [15] from single skill to multiple-skill learner model evaluation. The details are explained as follows.

[2] When these two metrics result in a contradictory selection of models, we primarily focus on RMSE, according to [39].

Effort. This metric empirically quantifies the expected number of practices when the tutor stops instruction. It is computed by counting the number of practice attempts a student needs in order to reach a given mastery threshold on real data. This is similar to the original metric [15], but has one major difference, in that the basic component skill to item links are removed from the mapping when the integration skill is required, so that effort considered in integration skills won't be repeatedly counted for basic component skills (while effort solely for basic component skills is maintained). The following formulas explain the computation of *Effort* for mastering the set of skills Q on a dataset, given a mastery threshold R, based on computing effort for a single student u on a single skill q:

$$\begin{aligned} \text{Effort(u, q)} &= \sum_{1 \le t \le |\mathbf{O}_{\text{u,q}}|} \prod_{1 \le t' \le t} \mathbf{I}(K_{u,q,t'} < R) \\ \text{Effort} &= \frac{1}{|U|} \sum_{u \in U} \sum_{q \in Q} \text{Effort(u, q)} \end{aligned} \tag{1}$$

where t denotes the index of the observation sequence $\mathbf{O}_{u,q}$ of a student u on a skill q, $K_{u,q,t'}$ denotes the inferred knowledge at t^{th} observation , and $\mathbf{I}$ denotes an indicator function.

Score. This metric empirically quantifies the expected performance of students when the tutor stops instruction. It is computed by the actual ratio of correct performance on real data where the learner model asserts that a student reaches a given mastery threshold for all of the required skills of the current item. The original metric only applies when each item maps to a single skill by simply examining the performance sequence of each skill. which is not applicable to multiple-skill practices, since the responsibility of each skill for the performance is not clear. To address this, we jointly examine knowledge states for multiple skills. Following formula explains the computation of *Score* for mastering the set of skills Q on a dataset, given a mastery threshold of R:

$$\text{Score} = \frac{\sum_{1 \le t \le |\mathbf{O}|} \prod_{q \in Q_{O_t}} \mathbf{I}(K_{q,t} \ge R) \cdot \mathbf{I}(O_t = 1)}{\sum_{1 \le t \le |\mathbf{O}|} \prod_{q \in Q_{O_t}} \mathbf{I}(K_{q,t} \ge R)} \tag{2}$$

where $\mathbf{O}$ denotes the overall observations, Q_{O_t} denotes the set of direct parent skills of the item corresponding to O_t, $K_{q,t}$ denotes the inferred knowledge level of a skill q at t^{th} observation.

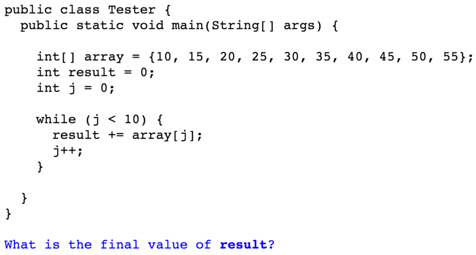
Joint examination of Effort and Score across thresholds. Prior work examining expected effort has ignored examining expected performance [29]. Consider a learner model that tends to overestimate students' knowledge levels. Although the expected effort will be low, the expected performance will also be low, resulting in under-practicing. So we consider Effort and Score jointly in this work. Moreover, there are two important differences with the original LEOPARD framework: 1) different mastery thresholds are considered, since there is no ground truth of what mastery threshold should be used; and 2) we avoid imputation when mastery is not reached, since it could distort the original distribution, and focus on thresholds with sufficient data (i.e., with at least 20% of the complete data available to compute the metrics and at least 85% of skills with at least one student reaching mastery).

4.4 Real-World Recommendation Helpfulness

In addition to the data-driven evaluation, we also design a user study to examine the helpfulness of the remedial recommendations that are generated by learner models. Admittedly, this is an indirect assessment of leaner models; yet, we argue that directly collecting users' feedback on learner models' knowledge inference involves non-trivial complication, and that the effectiveness of a learner model is ultimately reflected by remediation. The main task is to solve Java problems and rank recommended *subproblems* according their helpfulness for each participant in solving the original problem. A *subproblem* is an easier version of an original problem that primarily remediates one skill. Thus, a learner model's diagnosis can be "translated" into recommended subproblems. Subproblems are created systematically: first, skills in a problem are ordered by estimated difficulties (computed by the average success rate of problems requiring the current skill); then, for each skill, a subproblem is created by removing harder skills (if the remediated skill is the hardest, then the 2^{nd} hardest skill will be removed). Since different skill models specify different skills for a problem, subproblems can be classified into those that address integration skills and those that don't. For two subproblems with the same basic component skills, we try to make sure the only difference is whether the basic component skills are integrated or just sequentially put together.

We compare recommendations generated from CKM-HI, CKM, WKT, and a *distractor*, which randomly picks an irrelevant subproblem. All learner models employ the same recommendation strategy: after a student makes an attempt, the learner model picks a subproblem that addresses the weakest skill and another that addresses the second weakest skill. We expect that recommendation strategies can have a non-trivial impact on learner models' effectiveness, so we examine two strategies: *MaxDiff* maximizes the difference of CKM-HI with baseline models by disallowing WKT and CKM to recommend subproblems that address integration skills, and *MinDiff* minimizes the difference of CKM-HI with baseline models by allowing WKT and CKM to have a 50% chance to pick subproblems that address integration skills if any basic component skill of the integration skill is identified to be remediated. We employ only the MaxDiff strategy in our user study and use the collected data to conduct simulation for the MinDiff strategy: whenever CKM or WKT picks a subproblem that addresses an integration skill, if its ranking is available from the same user on the same problem, then this ranking is assigned to the subproblem; otherwise, it will be assigned the best ranking. In doing so, we examine the upper and lower bounds of the difference among CKM-HI and the baselines.

Participants receive mixed, permutated recommendations from different models at the same time and are asked to give a non-repeated ranking to each subproblem. They are presented with the original problem and the selected subproblem side-by-side. They are asked (but not forced) to attempt subproblems before ranking, and can adjust rankings any time before attempting next problem. We design the study to focus on students with a basic understanding of component skills and on problems requiring integration, since CKM-HI is primarily designed for monitoring and remediating integration skills compared with its simpler counterparts. In fact, CKM-HI maintains knowledge inference accuracy for basic component skills and increases prediction accuracy for non-integration problems, since it avoids over-penalizing basic component skills. Our latter data-driven evaluation demonstrates CKM-HI's advantage on the overall dataset that includes all students and items.



```
public class Tester {
  public static void main(String[] args) {

    int[] array = {10, 15, 20, 25, 30, 35, 40, 45, 50, 55};
    int result = 0;
    int j = 0;

    while (j < 10) {
      result += array[j];
      j++;
    }

  }
}

What is the final value of result?
```

Figure 3: An example of QuizJET problem in *Arrays* topic.

Such a focused user study allows us to draw more powerful conclusions, given the limited number of participants. We choose seven problems from *Loops* and *Arrays* topics, each of which covers 1 to 5 integration skills, and recruit students who have some prior experience in Java. But since we can't fully guarantee participants' levels, we design a *pretrain* session with problems and examples testing and teaching basic component skills, and ask students to solve such problems before moving on to the next step. The study consists of four parts: 1) a background survey; 2) a pretrain session; 3) a pretest on all integration skills, where each problem targets one integration skill; and 4) the main task.

5 EXPERIMENTS

In this section, we report both data-driven and user study evaluations as we compare two popular multiple-skill knowledge tracing models WKT and CKM with our proposed model CKM-HI.

5.1 Data-Driven Evaluation

5.1.1 Dataset and Experimental Setup. We used a Java programming dataset collected through classroom studies between fall 2013 and fall 2015 at the University of Pittsburgh, from the system QuizJET [19]. Students are requested to give the output or the final value of a variable by comprehending a program (Figure 3). Only one answer is accepted. Each problem is generated by a template, and students can make multiple attempts, where each attempt corresponds to a new instantiation changing the values of some variables. For each problem, students need to apply multiple skills at the same time in order to succeed. The system (only) provides correctness (0/1) information for each attempt. Students decide whether to try a problem again or to move on to another problem. Problems are grouped by topics (e.g., *For Loops*, *ArrayList*). To reduce the complexity for analysis, we removed the two most complex topics (*Interface* and *Inheritance*), which resulted in 91 items. The final dataset contains 25,988 observations (including all attempts) from 347 students, with an average success rate of 67%. For all experiments, we conducted a 10-fold student stratified cross-validation; i.e., in each fold, we trained on 90% of students, and conducted prediction and inference for the remaining 10% of students. For training CKM and CKM-HI models, we compressed multiple attempts per item into a single attempt by computing the average success rate across attempts at the same item, since the network doesn't model the dynamics across attempts of the same item. Since students often fail at their first attempts and finally succeed in their last attempts (learning merely from correctness feedback), keeping only the first or last attempt will risk either overestimating or underestimating the difficulty of skills. For training WKT, we kept the original multiple-attempt sequence, since WKT contains learning rate parameters. However, during the prediction phrase we kept all attempts in the test sets for all models, since all models perform dynamic knowledge updates and predictions at each time step, conditioned on historical performance.

For each metric, we conducted a two-sided paired t-test test (after confirming that normality wasn't violated) with a Bonferroni correction. We reported the common Cohen's d_{av} [28] for the effect size. We used the SMILE [12] toolkit to construct all the models. For each model, we initialized all root skill nodes' *init* parameters (the probability of initially knowing the skill) by the average success rate of problems that require this skill. We initialized cognitive load nodes $K^c_{i\&j}$ in CKM-HI in the same way. We initialized the *learn* parameter for WKT as 0.15, and all models' *guess* and *slip* parameters as 0.3. For each integration skill node $K_{i\&j}$ in CKM-HI, we initialized the CPT given the values of its parents K^b_i, K^b_j and $K^c_{i\&j}$ by setting { P(T|TTT)=0.99, P(T|TTF)=0.6, P(T|TFT)=0.6, P(T|TFF)=0.25, P(T|FTT)=0.6, P(T|FTF)=0.25, P(T|FFT)=0.4, P(T|FFF)=0.01 }.

5.1.2 Basic Component Skill to Item Mapping Validation. Individual basic component skill to item mapping provides a strong foundation for adding integration skills. In our Java programming tutor, one problem can easily require multiple skills. For example, the problem in Figure 3 requires the understanding and application of *WhileStatement*, *ArrayElement*, and *AddAssignment* (among others). We compared three sets of available mappings: 1) one from an automatic Java parser [18], which indexes an item with all concepts that appear in the code; 2) one from experts' dense labeling that considers only the important prerequisite and outcome concepts (i.e., those taught in the current topic), which is less dense than the previous one; and 3) one from experts' sparse labeling that considers only the important outcome concepts. We ran WKT with these three sets of skill models and compared the performance prediction metric RMSE through a 10-fold cross-validation. We found that the third mapping achieved significantly better prediction performance (p<.0001) with a large effect size (d_{av}>1) in each pairwise comparison. This skill model maps 4 basic component skills per item on average (ranging from 1 to 8) with a total of 72. This mapping was directly used in WKT and CKM. For integration skill to item mapping, we chose a sparse labeling from experts and didn't conduct further validation. One reason is due to the implementation (rather than theoretical) limitation of the toolkit being unable to run a more dense mapping of integration skills. Another reason, as we stated before, is that we primarily focus on demonstrating one successful way to model integration skills and leave finding the best skill model for future work. The final integration skill to item mapping indexes 2 integration skills per item on average (ranging from 1 to 5) with a total of 43. 47 (out of 91) items having at least one integration skill. This mapping was used to modify the chosen basic component skill to item mapping, and was used in CKM-HI.

5.1.3 Performance Prediction and Parameter Plausibility. Table 1 and Table 2 summarize the comparison of both performance prediction and parameter plausibility. We also report prediction on the first attempts of items, since they are usually important when conducting remediation. Both CKM-HI and CKM significantly beat WKT in all 6 metrics with a large effect size, with CKM-HI beating

Table 1: Comparison of performance prediction and parameter plausibility metrics, computed by averaging across 10 folds. The best result is denoted in bold.

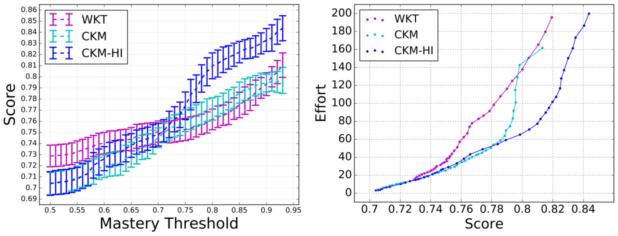
Models	RMSE	AUC	RMSE (1st att.)	AUC (1st att.)	Guess	Slip
WKT	.4494	.6873	.4433	.7001	.4239	.2836
CKM	.4446	.7273	.4073	.7945	.3806	.2093
CKM-HI	**.4437**	**.7283**	**.4064**	**.7958**	**.3625**	**.1860**

Table 2: Statistical test p values and effect sizes for prediction performance and parameter plausibility comparison.

Models	RMSE	AUC	RMSE (1st att.)	AUC (1st att.)	Guess	Slip
CKM vs. WKT	*+	***+	***+	***+	***+	***+
CKM-HI vs. WKT	**+	***+	***+	***+	***+	***+
CKM-HI vs. CKM	**	*	***	*	***+	***+

* sig. at 0.05/3=0.017, ** sig. at 0.01/3=0.0033, *** sig. at 0.001/3=0.00033.
+ effect size ≥ 1 (large).

WKT with smaller p values and larger effect sizes. Further, CKM-HI also significantly outperforms CKM in all prediction metrics. Though it has a small effect size (d_{av}<.2), it is able to significantly outperform CKM in parameter plausibility with a large effect size. Clearly, we see an advantage in modeling integration skills over the two popular multiple-skill knowledge tracing models in both performance prediction and parameter plausibility. Admittedly, CKM-HI only achieves a small effect size prediction improvement over CKM; yet the advantage in parameter plausibility is non-trivial, which should significantly increase the accuracy of latent knowledge inference or diagnosis. Other evaluation aspects shown in latter sections will further reveal the advantage of CKM-HI over CKM. We hypothesize that this small effect size prediction gain can be due to CKM using less plausible parameters to fit the data, and that a larger prediction gain should be revealed if we impose parameter constraints. Due to the space limit, our experiments proving this hypothesis will be reported elsewhere in the future.



(a) (b)

Figure 4: Comparison of CKM-HI, CKM and WKT on expected instructional effectiveness.

5.1.4 Expected Instructional Effectiveness. Figure 4a shows Scores plotted against mastery thresholds (with a 95% confidence interval across 10 folds). Figure 4b shows the combined Effort vs. Score graph with them connected by matching mastery thresholds. We consider a broad range of thresholds with enough data, as mentioned in Section 4.3: [0.5, 0.93]. Comparing CKM-HI with WKT on Scores, CKM-HI has worse Scores in low mastery thresholds, but much better Scores in high thresholds; when examining Effort and Score jointly, CKM-HI requires much less Effort to reach the same Score across almost all thresholds. When comparing CKM-HI with CKM on Scores, CKM-HI has similar Scores in low mastery thresholds and much better Scores in high thresholds; when examining Effort and Score jointly, CKM-HI requires much less Effort to reach the same Score in most of the thresholds. These metrics clearly demonstrate that to reach the same expected performance, students who are guided by the CKM-HI model are expected to exert the least amount of effort, and that by using the same amount of effort, students guided by CKM-HI are expected to have higher performance, as compared with CKM and WKT. Surprisingly, although CKM significantly outperforms WKT in prediction, it requires similar Effort given same Scores in high thresholds.

5.2 User Study Evaluation

The main goal of our user study is to examine the real-world recommendation helpfulness of learner models. The user study was conducted with 20 students pursuing undergraduate or master's degrees in information science at the University of Pittsburgh. The study lasted for around 1.5 hours on average. All of the problems are of the same type as QuizJET. We deployed the same learner models that were constructed during data-driven evaluation, in order to make the comparison compatible. All participants reported that they had some prior experience with Java. The mean score for first and last attempts in pretraining (which tests on and teaches students about basic component skills) is 0.836 and 0.997. The mean score for first and last attempts in the pretest (where each problem tests students on one integration skill) is 0.893 and 0.907. The mean score for the first and last attempts in the main problems (which tests and teaches multiple integration skills at the same time) is 0.676 and 0.949. As the statistics show, participants generally know basic component skills, but still have some difficulty integrating them (particularly when multiple integration skills are required together). We report results by answering different research questions, which are shown as follows.

Does CKM-HI receive the highest ranking? We analyzed the ranking data in two common methods for both MaxDiff and MinDiff strategies: treating ranking as a continuous score or treating ranking as an ordinal score. For the first method, we first computed the aggregated score that a model receives from a student by averaging the scores across the 7 main problems (we found out that, on average, the relative ranking among models are persistent across the 7 problems, so computing an average shouldn't affect the conclusion). Since normality is violated and we have repeated measurements per participant, we conducted a two-sided Wilcoxon signed rank test and computed its effect size (r=z/$\sqrt{N}$) [40]. For the second method, we first kept only the recommendations where students gave the best rank, and then for each model for each student, we counted the number of times that a best-ranked recommendation was generated from the current model across the 7 main problems. Since normality is not violated in this case, we conducted a paired t-test and its effect size d_{av} [28]. Table 3 and Table 4 report the average values and statistical test results. We draw similar conclusions from two kinds of analysis for both strategies. CKM-HI beats both WKT and CKM significantly with a large effect size. Surprisingly, although CKM shows a significant prediction gain over WKT, its ranking

Table 3: Ranking result comparison averaging across participants (best results denoted in bold).

	Avg. rank		Avg. count	
	MaxDiff	MinDiff	MaxDiff	MinDiff
Distr.	4.30	4.30	0.4	0.4
WKT	3.27	2.99	0.95	3.2
CKM	3.36	3.02	0.85	3.3
CKM-HI	**2.11**	**2.11**	**5.6**	**5.6**

Table 4: Statistical test p values (with a Bonferroni correction) and effect sizes for ranking comparison.

	Avg. rank		Avg. count	
	MaxDiff	MinDiff	MaxDiff	MinDiff
WKT vs. Distr.	*+	**+		***+
CKM vs. Distr.	**+	**+		***+
CKM-HI vs. Distr.	**+	**+	***+	***+
CKM vs. WKT				
CKM-HI vs. WKT	**+	**+	***+	***+
CKM-HI vs. CKM	**+	*+	***+	**+

* sig. at 0.05/6=0.0083, ** sig. at 0.01/6=0.0017, *** sig. at 0.001/6=0.00017.
+ effect size ≥ 0.5 (Wilcoxon signed rank test) and ≥ 1 (paired t-test).

is not significantly different from that of WKT. Considering all recommendations generated by each model (by the first continuous rank analysis), all learner models significantly outperform Distractor with a large effect size. Considering only recommendations received the best ranking (by the second count analysis), CKM-HI still significantly outperforms Distractor, but CKM and WKT can’t maintain a similar significant effect under MaxDiff.

Does the higher ranking of CKM-HI come from recommending subproblems that address integration skills? We compared the average ranking of subproblems that address integration skills with those that only address basic component skills (after removing subproblems generated by Distractor). Under MinDiff, integration subproblems receive an average ranking of 1.9, as compared with 3.18 of the non-integration ones; under MaxDiff, integration subproblems receive an average ranking of 2.01, as compared with 3.28 of the non-integration ones. Both differences are significant by a two-sided Wilcoxon signed ranked test (since normality is violated) (p<.001) with large effect size (r>.5). We conclude that students indeed favor subproblems with integration skills during the remediation. Is CKM-HI more able to recommend such subproblems? We found out that among the recommended subproblems of a learner model, the percentile of those addressing integration skills are 84%, 13%, 10%, and 9% for CKM-HI, CKM, WKT, and Distractor under MinDiff, and 0% for CKM and WKT under MaxDiff (CKM-HI and Distractor remain the same). As a result, we conclude that the higher ranking of CKM-HI indeed comes from recommending subproblems that address integration skills.

Can we trust students’ subjective rankings? Admittedly, it is a concern that the ranking data are subjective measurements. While we can’t fully eliminate such noise, we tried to identify evidence from our collected data that could increase the trustability of these subjective rankings. First, we were able to demonstrate that all learner models receive significantly higher rankings than the Distractor, when considering all recommendations and the best ranked recommendations under MinDiff strategy. Second, we analyzed the post-test questionnaire that asked about their ranking strategies (at most, two choices) and found that the highest two strategies aligned well with our ranking requirements; namely, they assigned a higher ranking to the subproblem that contains most of the concepts in the original problems, or that they used key concepts in a similar way to the original problem, rather than preferring a subproblem that contains more concepts. In future work, we plan to conduct a large-span and long-scale study to collect objective measurements.

6 CONCLUSION

In this paper, we advocate for the importance of modeling integration skills and have clearly demonstrated the feasibility and value of learner modeling for integration skills. Using a combination of analytical studies based on a Java programming dataset and a user study, we demonstrated that our proposed learner model, CKM-HI, offers significant improvements over two popular multiple-skill knowledge tracing models, WKT and CKM, over a range of aspects that are considered by our multifaceted evaluation framework: performance prediction accuracy, parameter plausibility, expected instructional effectiveness, and recommendation helpfulness. A combination of analytical and empirical approaches has enabled us to make some interesting observations about the limitations of performance prediction evaluation. By examining expected instructional effectiveness and recommendation helpfulness, we found out that a small (effect size) performance prediction gain can still lead to significant improvement in adaptive tutoring (CKM-HI vs. CKM); and surprisingly, a significant prediction gain can result in almost no improvement in adaptive tutoring (CKM vs. WKT).

Altogether, our paper brings three major contributions to the field of learner modeling and adaptive educational systems. First, we introduce a new type of knowledge graph that we call an integration graph, which shows how basic component skills progressively integrate and form new skills that are essential to describe domain expertise. Second, we create a novel integration-level learner model based on an integration graph, which outperforms traditional multiple-skill knowledge tracing models. Third, we introduce a multifaceted learner modeling evaluation framework over a range of aspects, including analytical evaluation and user-study evaluation. The evaluation component of this paper could be considered as an example of the application of this evaluation framework.

In future work, we plan to explore skill integration beyond the single context reported in this paper, while continuing to contribute to best practices in evaluating adaptive educational systems. In particular, we plan to explore automated methods for extracting integration skills that advance our preliminary approach [21].

ACKNOWLEDGMENTS

This research was supported by Andrew Mellon Predoctoral Fellowship from University of Pittsburgh. The learner model was constructed based on the SMILE Engine from BayesFusion, LLC (https://www.bayesfusion.com/). We also thank Rosta Farzan, Igor Labutov for giving valuable suggestions.

REFERENCES

[1] Susan A Ambrose, Michael W Bridges, Michele DiPietro, Marsha C Lovett, and Marie K Norman. 2010. *How learning works: Seven research-based principles for smart teaching.* John Wiley & Sons.

[2] John R Anderson, Albert T Corbett, Kenneth R Koedinger, and Ray Pelletier. 1995. Cognitive tutors: Lessons learned. *The journal of the learning sciences* 4, 2 (1995), 167–207.

[3] Ryan Baker, Albert Corbett, and Vincent Aleven. 2008. More Accurate Student Modeling through Contextual Estimation of Slip and Guess Probabilities in Bayesian Knowledge Tracing. In *International Conference on Intelligent Tutoring Systems*. Springer, 406–415.

[4] Peter Brusilovsky. 1992. Intelligent Tutor, Environment and Manual for Introductory Programming. *Educational and Training Technology International* 29, 1 (1992), 26–34.

[5] Peter Brusilovsky, Charalampos Karagiannidis, and Demetrios Sampson. 2004. Layered evaluation of adaptive learning systems. *International Journal of Continuing Engineering Education and Lifelong Learning* 14, 4/5 (2004), 402 – 421.

[6] Cristina Carmona, Eva Millán, José-Luis Pérez-de-la Cruz, Mónica Trella, and Ricardo Conejo. 2005. Introducing prerequisite relations in a multi-layered Bayesian student model. In *International Conference on User Modeling*. Springer, 347–356.

[7] David Chin. 2001. Empirical Evaluations of User Models and User-Adapted Systems. *User Modeling and User-Adapted Interaction* 11, 1-2 (2001), 181–194.

[8] Jason Collins, Jim Greer, and Sherman Huang. 1996. Adaptive assessment using granularity hierarchies and Bayesian nets. In *Intelligent Tutoring Systems*. Springer, 569–577.

[9] Cristina Conati, Abigail Gertner, and Kurt Vanlehn. 2002. Using Bayesian Networks to Manage Uncertainty in Student Modeling. *User Modeling and User-Adapted Interaction* 12, 4 (2002), 371–417.

[10] Albert T Corbett and John R Anderson. 1995. Knowledge tracing: Modeling the acquisition of procedural knowledge. *User Modeling and User-Adapted Interaction* 4, 4 (1995), 253–278.

[11] Michel C Desmarais and Ryan S Baker. 2012. A review of recent advances in learner and skill modeling in intelligent learning environments. *User Modeling and User-Adapted Interaction* 22, 1-2 (2012), 9–38.

[12] Marek J Druzdzel. 1999. SMILE: Structural Modeling, Inference, and Learning Engine and GeNIe: a development environment for graphical decision-theoretic models. In *Proceedings of the Sixteenth National Conference on Artificial Intelligence (AAAI-99)*. 902–903.

[13] D. J. Gilmore and T. R. G. Green. 1988. Programming plans and programming expertise. *The Quarterly Journal of Experimental Psychology Section A* 40, 3 (1988), 423–442.

[14] Yue Gong, Joseph E Beck, and Neil T Heffernan. 2010. Comparing knowledge tracing and performance factor analysis by using multiple model fitting procedures. In *Intelligent Tutoring Systems*. Springer, 35–44.

[15] José P 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 *Proc. 8th Intl. Conf. Educational Data Mining*. 187–194.

[16] José P González-Brenes, Yun Huang, and Peter Brusilovsky. 2014. General features in knowledge tracing: Applications to multiple subskills, temporal item response theory, and expert knowledge. In *Proc. 7th Int. Conf. Educational Data Mining*. 84–91.

[17] Neil T. Heffernan and Kenneth R. Koedinger. 1997. The composition effect in symbolizing: The role of symbol production vs. text comprehension. In *the Nineteenth Annual Conference of the Cognitive Science Society*. Lawrence Erlbaum Associates, 307–312.

[18] Roya Hosseini and Peter Brusilovsky. 2013. JavaParser: A Fine-Grained Concept Indexing Tool for Java Problems. In *The First Workshop on AI-supported Education for Computer Science (AIEDCS)*.

[19] I-H Hsiao, Sergey Sosnovsky, and Peter Brusilovsky. 2010. Guiding students to the right questions: adaptive navigation support in an E-Learning system for Java programming. *Journal of Computer Assisted Learning* 26, 4 (2010), 270–283.

[20] Yun Huang, José P González-Brenes, Rohit Kumar, and Peter Brusilovsky. 2015. A framework for multifaceted evaluation of student models. In *Proc. 8th Int. Conf. Educational Data Mining*. 203–210.

[21] Yun Huang, Julio Guerra, and Peter Brusilovsky. 2016. Modeling skill combination patterns for deeper knowledge tracing. In *Proceedings of the 6th Workshop on Personalization Approaches in Learning Environments (PALE 2016)*.

[22] W. Lewis Johnson and Elliot Soloway. 1985. PROUST: Knowledge-Based Program Understanding. *IEEE Transactions on Software Engineering* 11, 3 (1985), 267–275.

[23] Brian W Junker and Klaas Sijtsma. 2001. Cognitive assessment models with few assumptions, and connections with nonparametric item response theory. *Applied Psychological Measurement* 25, 3 (2001), 258–272.

[24] Tanja Käser, Severin Klingler, Alexander Gerhard Schwing, and Markus Gross. 2014. Beyond knowledge tracing: Modeling skill topologies with bayesian networks. In *International Conference on Intelligent Tutoring Systems*. Springer, 188–198.

[25] Kenneth R Koedinger, Albert T Corbett, and Charles Perfetti. 2012. The Knowledge-Learning-Instruction framework: Bridging the science-practice chasm to enhance robust student learning. *Cognitive science* 36, 5 (2012), 757–798.

[26] Kenneth R Koedinger, Philip I Pavlik Jr, John C Stamper, Tristan Nixon, and Steven Ritter. 2011. Avoiding Problem Selection Thrashing with Conjunctive Knowledge Tracing. In *Educational Data Mining*. 91–100.

[27] Amruth N. Kumar. A Scalable Solution for Adaptive Problem Sequencing and its Evaluation. In *4th International Conference on Adaptive Hypermedia and Adaptive Web-Based Systems (AH'2006)*. Springer Verlag, 161–171.

[28] Daniël Lakens. 2013. Calculating and reporting effect sizes to facilitate cumulative science: a practical primer for t-tests and ANOVAs. *Frontiers in psychology* 4 (2013), 863.

[29] Jung In Lee and Emma Brunskill. 2012. The Impact on Individualizing Student Models on Necessary Practice Opportunities. In *Proceedings of the 5th International Conference on Educational Data Mining*. www.educationaldatamining.org, 118–125.

[30] Chee-Kit Looi. 1991. Automatic debugging of Prolog programs in a Prolog intelligent tutoring system. *Instructional Science* 20, 2 (1991), 215–263.

[31] Michael Mayo and Antonija Mitrovic. 2001. Optimising ITS behaviour with Bayesian networks and decision theory. *International Journal of Artificial Intelligence in Education* (2001), 124–153.

[32] Eva Millán and José Luis Pérez-De-La-Cruz. 2002. A Bayesian diagnostic algorithm for student modeling and its evaluation. *User Modeling and User-Adapted Interaction* 12, 2-3 (2002), 281–330.

[33] Robert J Mislevy and Drew H Gitomer. 1995. The role of probability-based inference in an intelligent tutoring system. *ETS Research Report Series* 1995, 2 (1995).

[34] Antonija Mitrovic, Michael Mayo, Pramuditha Suraweera, and Brent Martin. 2001. Constraint-based tutors: a success story. In *International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems*. Springer, 931–940.

[35] Rafael Morales, Nicolas Van Labeke, and Paul Brna. 2006. Approximate modelling of the multi-dimensional learner. In *Intelligent Tutoring Systems*. Springer, 555–564.

[36] William R. Murray. 1985. Heuristic and formal methods in automatic program debugging. In *9-th International Joint Conference on Artificial Intelligence*. 15–19.

[37] Alexandros Paramythis and Stephan Weibelzahl. 2005. A Decomposition Model for the Layered Evaluation of Interactive Adaptive Systems. In *10th International User Modeling Conference*, Vol. 3538. Springer Verlag, 438–442.

[38] Zachary A Pardos and Neil T Heffernan. 2010. Navigating the parameter space of Bayesian Knowledge Tracing models: Visualizations of the convergence of the Expectation Maximization algorithm. *Educational Data Mining* 2010 (2010), 161–170.

[39] Radek Pelanek. 2015. Metrics for Evaluation of Student Models. *Journal of Educational Data Mining* 7, 2 (2015), 1–19.

[40] Robert Rosenthal, H Cooper, and LV Hedges. 1994. Parametric measures of effect size. *The handbook of research synthesis* (1994), 231–244.

[41] Elliot Soloway and Kate Ehrlich. 1984. Empirical Studies of Programming Knowledge. *IEEE Trans. Software Engineering* SE-10, 5 (1984), 595–609.

[42] Brett van De Sande. 2013. Properties of the bayesian knowledge tracing model. *JEDM-Journal of Educational Data Mining* 5, 2 (2013), 1–10.

[43] Kurt Vanlehn, Collin Lynch, Kay Schulze, Joel A Shapiro, Robert Shelby, Linwood Taylor, Don Treacy, Anders Weinstein, and Mary Wintersgill. 2005. The Andes physics tutoring system: Lessons learned. *International Journal of Artificial Intelligence in Education* 15, 3 (2005), 147–204.

[44] Kurt VanLehn, Zhendong Niu, Stephanie Siler, and Abigail S Gertner. 1998. Student modeling from conventional test data: A Bayesian approach without priors. In *International Conference on Intelligent Tutoring Systems*. Springer, 434–443.

[45] Gerhard Weber. 1996. Episodic learner modeling. *Cognitive Science* 20, 2 (1996), 195–236.

[46] Gerhard Weber. 1996. Individual selection of examples in an intelligent learning environment. *Journal of Artificial Intelligence in Education* 7, 1 (1996), 3–31.

[47] Yanbo Xu and Jack Mostow. 2012. Comparison of methods to trace multiple subskills: Is LR-DBN best?. In *Proc. 5th Intl. Conf. Educational Data Mining*. 41–48.