# Targeting Design-Loop Adaptivity

Stephen E. Fancsali
Hao Li
Michael Sandbothe
Steven Ritter
Carnegie Learning, Inc.
{sfancsali, hli, msandbothe,
sritter}@carnegielearning.com

## ABSTRACT

Recent work describes methods for systematic, data-driven improvement to instructional content and calls for diverse teams of learning engineers to implement and evaluate such improvements. Focusing on an approach called "design-loop adaptivity," we consider the problem of how developers might use data to target or prioritize particular instructional content for improvement processes when faced with large portfolios of content and limited engineering resources to implement improvements. To do so, we consider two data-driven metrics that may capture different facets of how instructional content is "working." The first is a measure of the extent to which learners struggle to master target skills, and the second is a metric based on the difference in prediction performance between deep learning and more "traditional" approaches to knowledge tracing. This second metric may point learning engineers to workspaces that are, effectively, "too easy." We illustrate aspects of the diversity of learning content and variability in learner performance often represented by large educational datasets. We suggest that "monolithic" treatment of such datasets in prediction tasks and other research endeavors may be missing out on important opportunities to drive improved learning within target systems.

## Keywords

Design-loop adaptivity, deep knowledge tracing, Bayesian knowledge tracing, mastery learning, learning engineering.

## 1. INTRODUCTION

Recent work calls on researchers and developers, including teams of learning engineers [14, 26], to focus on "explanatory" models of learners [25] and "design-loop adaptivity" processes [1, 15] to practically improve learning systems. While researchers describe specific examples of how explanatory learner models and design-loop adaptivity can be used to drive improvements to instruction, less (if any) attention has been paid in the literature to the practical problem of how content developers and learning engineers target and prioritize content for improvement.

We focus on cases in which a target system has a large portfolio of content, elements of which must be prioritized and targeted for improvement given finite learning engineering and software development resources. We present a case study using a data set that is among the largest considered in the literature on knowledge tracing and related methods [9, 18, 22], comprised of middle school and high school student work over an academic year on several hundred mathematics topics, each generally completed by thousands of students, generating several hundred million data points tracking student actions. We motivate, describe, and illustrate two approaches to targeting content for improvement within this portfolio, focusing primarily on what Aleven et al. [1] call "design-loop adaptation to student knowledge," relying on large-scale data to find similarities amongst learners we might leverage to redesign instructional content for better learning.

One targeting method is based on a measure of the extent to which learners tend to struggle with particular pieces of content, and we contrast it with an approach based on the relative prediction performance of deep learning models (i.e., Deep Knowledge Tracing; DKT [18, 22]) compared to traditional Bayesian Knowledge Tracing (BKT; [9]) models.

The first method targets content students struggle to learn, relying on measures of knowledge component (KC [19]; or skill) mastery that are internal to the target intelligent tutoring system (ITS). In contrast, the second method is roughly motivated by the idea that identifying content in which there is a large difference in performance between deep learning and traditional Bayesian approaches may suggest areas in which deep learning can leverage statistical regularities in students' performance that could point to improvements in the KC models that are used to drive adaptation with BKT. Such performance differences may suggest a particular focus area for KC model improvements. Relative DKT performance versus BKT performance also provides an instance of a metric that is perhaps less dependent on how the run-time ITS has "set the bar" for success in terms of KC mastery.

In exploring these two approaches, we illustrate the variability in learning content and experiences within widely deployed systems like Carnegie Learning's MATHia (formerly Cognitive Tutor) [23]. While different facets of variation may at times call for different approaches to content improvement (e.g., variation in student motivation could call for redesigns that discourage "gaming the system" [3]), our present work explores how to guide learning engineers' "attention" to particular pieces of content to then consider specific improvements via processes for design-loop adaptivity [1, 15].

Original contributions of this work are two-fold: (1) We describe a novel problem in the literature related to how to target

instructional content improvement or design-loop adaptivity and explore two targeting approaches, and (2) we shed light on opportunities in treating large-scale educational datasets that may be missed by treating such datasets as "monolithic" targets for data-intensive approaches. Treating datasets in a "monolithic" way, though not a universal practice (e.g., [4-5, 10]) may inhibit practical progress in learning engineering.

In addition to considering one of the largest-scale applications of DKT (and BKT) modeling in the literature, we illuminate avenues for research at the intersection of educational data science and learning engineering at scale in a widely-deployed adaptive learning platform for K-12 mathematics. We seek to amplify extant calls for a more nuanced approach to work on performance prediction [15, 25] while illustrating solutions to practical problems in learning engineering and product improvement.

# 2. DESIGN-LOOP ADAPTIVITY

## 2.1 Background

A recent survey of adaptive instructional technologies [1] describes three categories along which learners' experience can be varied, including "step-loop adaptivity," "task-loop adaptivity," and "design-loop adaptivity." Step-loop adaptivity and task-loop adaptivity roughly correspond to "inner" and "outer" loop adaptive functionality in ITSs distinguished by VanLehn (e.g., [28**]**), respectively. We briefly describe step-loop and task-loop adaptivity before considering design-look adaptivity.

Step-loop or inner-loop adaptivity enables an adaptive instructional system or ITS to provide support to learners *within a particular learning task* based on their performance (e.g., providing context-sensitive hints or just-in-time feedback within a math problem based on learner responses). Task-loop or outer-loop adaptivity enable an instructional system to *choose the next appropriate task for a learner* based on a model of student learning and evolving estimates of a learner's mastery of underlying competencies, skills, or KCs [19] based on a learner's performance. Extensive educational data mining (EDM) literature considers, for example, variants of and data-driven parameter optimizations for BKT (e.g., [18]), which can be used to select tasks for learners as their mastery of KCs evolves.

In their recent survey, Aleven and colleagues describe design-loop adaptivity as involving

> data-driven decisions made by course designers before and between iterations of system design, in which a… system is updated based on data about student learning, specifically, data collected with the same system… [1].

They go on to describe goals toward which design-loop adaptations might be made, including adaptations to student knowledge, affect and motivation, student strategies and errors, and self-regulated learning, providing examples of each. Canonical examples of design-loop adaptivity or adaptation to student knowledge, the goal of our present targeting and prioritization endeavor, generally involve situations in which content within tutoring systems or online courses are improved by refining the fine-grained KC models that drive the adaptive experience of learners using a combination of data and human expertise [17, 20, 27].

Design-loop adaptivity for motivation and affect might drive content or system design and redesign to discourage off-task behavior [4] and "gaming the system" [3], wherein students attempt to make progress in a system by taking advantage of system features like hints, rather than making genuine attempts to master content. Aleven et al. [1] suggest that an approach to modeling gaming the system behavior based on a large-scale survey of the extent to which gaming the system [3] manifests across topics (what we will refer to as "workspaces") in an intelligent tutoring system like MATHia provides a foundation for future design-loop adaptivity investigations. One important facet of this work (and related work on off-task behavior [4]) is its appreciation of the extent to which there is variability in how learning occurs across different (types of) content within adaptive instructional systems. Appreciating and surveying this variability is vital to ascertaining where, within large portfolios of content, to target design-loop adaptivity efforts and related data-driven, instructional improvement efforts.

## 2.2 A Process for Design-Loop Adaptivity

Huang et al. [15] describe a systematic approach to design-loop adaptivity or data-driven instructional redesign and improvement. They suggest three general goals for such redesign efforts. For a particular piece of content in an ITS or similar adaptive instructional system with a KC model, the goals are: (1) refine the KC model for the target content, (2) redesign the content, and (3) optimize individualized learning within the content. Existing EDM methods and novel analyses are then described to achieve each of these goals, targeting an "Algebraic Expressions" unit of content within the Mathtutor ITS [2]. For example, KC models can be refined using data-driven, computationally intensive methods like Learning Factors Analysis (LFA; [8]) or a simpler approximation of such an approach that uses regression techniques called "difficulty factor effect analysis" by Huang et al. [15]. Human expertise also plays an important role in such refinements, including in setting up data-driven analyses to produce meaningful results, interpreting these results for inclusion in potential task redesigns, and often in providing suggested refinements for target tasks.

Huang et al. [15] demonstrate that redesigned content improves learning as measured by pre-tests and post-tests. Broadly, these goals align with on-going, data-driven content improvement efforts pursued by learning engineers working with MATHia. Nevertheless, the process of design-loop adaptivity generally requires extensive human and computational resources to be carried out in ways that will drive improved instructional effectiveness. The present work seeks to illustrate how EDM techniques might help improve targeting this process.

# 3. MATHia

## 3.1 Learning Platform

Carnegie Learning's MATHia [23] is an ITS used by hundreds of thousands of learners each year, mostly in middle and high school classrooms as a part of a blended math curriculum that combines collaborative work guided by instructors and Carnegie Learning's *MATHbook* worktexts (60% of instructional time in recommended implementations) with individual work in MATHia (40% of instructional time). Nevertheless, usage of MATHia, contexts in which it is used, and other implementation details vary across a diverse, nationwide user-base.

Grade levels of content in MATHia (e.g., Grade 7, Algebra I) are organized into a series of "modules," each of which is comprised of a series of "units." Units are composed of a series of "workspaces." Workspaces represent the underlying unit of learner progress to mastery in MATHia. Each workspace presents

a set of problems associated with a set of KCs; student progress within the system is determined by students' achievement of mastery of all of the KCs associated with a particular workspace, estimated by MATHia using BKT (see §3.2). Learning experiences vary substantially between workspaces with respect to design patterns, content areas, types of practice and instruction provided, (quality of) KC models intended to practice such content (e.g., some the result of years of iterative refinements, others introduced more recently), BKT parameters, and other parameters that drive task selection and mastery judgment.

Consider the problem solving task illustrated in Figures 1 and 2. Figure 1 illustrates the workspace "Modeling the Constant of Proportionality." In this workspace, students are provided with a word problem and several associated questions (left pane; Figure 1). On the right-hand side of Figure 1, tools are presented to solve the problem's "steps." There is a worksheet or table in which they can provide units of measurement, responses to questions, and fields in which to write expressions to model the problem's scenario. After they have completed entries in the worksheet, students work with a graphing tool. Each problem-step in the ITS can provide context-sensitive hints upon request as well as just-in-time feedback that tracks errors that students often make. Most problem-steps are mapped to KCs, for which MATHia provides an evolving mastery estimate to adapt problem selection to the individual student's needs (see §3.2).

Contrast the learning experience of the problem in Figure 1 with that of Figure 2. "Modeling the Constant of Proportionality" (Figure 1) involves substantive reading, modeling the problem scenario via algebraic expressions, working through concrete instances of these expressions, and using a graphing tool. Figure 2 illustrates problem-solving in a menu-based equation "solver" workspace, "Solving with the Distributive Property Over Multiplication." Here the student is tasked with solving for $x$ in the equation $65 = 10\,(x + 6)$. There is little reading and no context provided for the equation, but hints and just-in-time feedback are available. Learners' progress toward mastery is tracked for a different set of KCs. The menu-based solver constrains possible student actions at various points in the equation-solving process compared to the typed-in input that students provide in the worksheet in Figure 1. Far from an exhaustive list, we seek to illustrate a few from among substantial differences in types of content provided, design patterns, interaction modalities, underlying KC models, and tools available, even within the relatively constrained domain of math, any of which may have important impacts on inferences that might be drawn from data or the ability of different methods to predict performance and learning within such content. While any of the features in these examples might reasonably be refined as a part of the design-loop adaptivity or content improvement process, we leave to future work the data-driven targeting of specific improvements within a workspace. We consider how to target specific "workspaces" for design-loop adaptivity improvements.

## 3.2 Knowledge Tracing & Mastery Learning

BKT [9] posits a binary (i.e., "mastered" or "unmastered") knowledge state for each independently modeled KC and can be formalized as a four-parameter hidden Markov model. One parameter represents the probability that a learner has already mastered a KC before their first opportunity to practice it. A second parameter represents the probability that a learner transitions from the unmastered to the mastered state at any particular KC practice opportunity. Two parameters link the knowledge state to observable outcomes at any KC practice opportunity: the probability that a student is in the unmastered state and responds correctly ("guessing") and the probability that a student is in the mastered state and answers incorrectly ("slipping"). Extensive EDM literature has explored the data-driven fitting of BKT parameters as well as individualized (e.g., [30]) and more sophisticated variants of this approach (e.g., [18]).

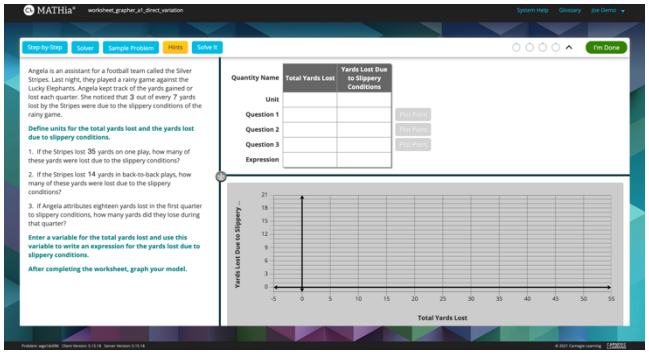


**Figure 1. Problem-solving screenshot from a MATHia workspace called "Modeling the Constant of Proportionality."**

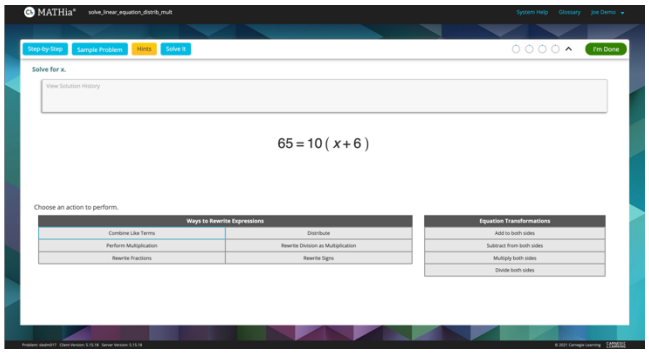


**Figure 2. Screenshot from the MATHia workspace "Solving with the Distributive Property Over Multiplication."**

Based on parameter settings and performance data collected as a student practices each KC, the system can use BKT to infer and update estimates of the probability that a student is in the "mastered" state for any particular KC. Typically, systems set a threshold for mastery (often 0.95, as in MATHia); if the system's estimate that the probability a student has mastered a particular KC is above the threshold, then the system considers the KC mastered for that student.

Relying on evolving estimates of learner KC mastery, instructional systems can use knowledge tracing frameworks like BKT to drive "task-loop" (or "outer loop") adaptivity [1, 28] and mastery learning [7, 24]. After a student completes a problem (or task; like the problems illustrated in Figures 1-2), the system can select the next problem based on KCs that a student has yet to master. In this way, systems can adapt to the student's evolving mastery of KCs, providing (ideally) just enough practice for students to master KCs and avoiding cases in which the system provides too little or too much practice.

Implementing self-paced mastery learning [7, 24], MATHia provides practice to a student until they have either mastered all KCs associated with a particular workspace or they have reached the maximum number of problems that designers have specified for a particular workspace. Once the student masters all of the KCs in a particular workspace (or reaches the max number of problems), they are moved on to the next workspace in an

assigned content sequence. Teachers are alerted when students reach the max number of problems in a workspace without reaching mastery. Setting a max number of problems ensures that students do not endlessly struggle unproductively within a piece of content [11].

## 3.3 Data

We consider data from 252,036 learners who used MATHia during the 2018-19 academic year and completed at least one of 308 workspaces that track KC mastery across math content for Grades 6-8, Algebra I, Algebra II, and Geometry. These data account for approximately 3.8 million workspace completions. Models are learned over subsets of 267,419,999 student actions (i.e., first-attempts, including hint requests) at problem-steps mapped to KCs. Over the 308 workspaces, MATHia tracks 2,152 KCs. Table 1 provides summary statistics.

**Table 1. Summary statistics for 308 MATHia workspaces in 2018-2019; "KCs" = # KCs tracked; "Comps." = # student-workspace-completions; "Actions" = sum across all students completing workspace of count of first attempts (including hint requests) at problem-steps within workspace problems.**

| | Min. | Q1 | Med. | Q3 | Max. |
|---|---|---|---|---|---|
| **KCs** | 2 | 5 | 6 | 9 | 15 |
| **Comps.** | 167 | 4275 | 9414 | 18801 | 51097 |
| **Actions** | 5530 | 197757 | 489159 | 1278325 | 7191034 |

When working with large, complex datasets, it is essential to focus learning engineering efforts on the portions of the system for which improvements can be most impactful. Rather than consider such a broad dataset as a single monolithic target, especially for performance prediction modeling in §4.2, we learn models for each workspace within the dataset; input data are sequences of correctness labels for learner actions (e.g., binary correct or incorrect, where incorrect includes both errors and hint requests) and labels for KCs mapped to each action.

# 4. METRICS FOR TARGETING IMPROVEMENTS

As illustrated in Figures 1 and 2, workspace-to-workspace variability in learning experiences is substantial. Types of practice vary (e.g., equation solving, graphing, etc.), and developers make a plethora of design choices in creating content. Some workspaces require more reading; KC models vary in complexity, and some have been iteratively refined over the course of nearly two decades while others are newly deployed in a given year. Given this variation and the nature of grade-level content standards, there is also variability in the extent to which learners find particular content difficult.

Learner difficulties manifest at the problem-step level in the form of problem-solving errors and hint requests and at the workspace level in at least two ways: (1) that some learners require a greater number of problems to achieve mastery of all KCs, and (2) that some learners reach the maximum number of problems set by designers without having achieved mastery of all KCs. These latter students are moved along within their curriculum sequence without mastery. Teachers are alerted of this failure to reach mastery via reporting analytics available to them as well as in the LiveLab teacher companion app to MATHia. Some students fail to reach mastery in a workspace because of genuine difficulty with presented math content, but relatively frequent instances of such failure to reach mastery often indicate that content improvements (i.e., design-loop adaptivity) is called for to enhance experiences for learners.

Prior research considers MATHia's workspace level as a unit of analysis. Researchers have focused on associations between characteristics of Cognitive Tutor "lessons" (MATHia's workspaces) and learners' affective states like confusion and frustration [10] as well as the extent to which students go off-task [4] and game the system [5]. In what follows, we adopt an approach similar in spirit to this literature by considering a large corpus of MATHia data as broken down into workspaces rather than treating the entire dataset in a monolithic fashion.

The first metric we consider helps identify content that is instructionally ineffective in ways that manifest as difficulty for learners to successfully complete the content. In considering the second metric, we explore one example where the metric may be providing some insights into places where content is not "difficult" (i.e., measures of difficulty do not "raise flags" about improvement needs) but where design-loop adaptivity improvements might drastically improve student learning.

## 4.1 Proportion of Failures to Reach Mastery

The first design-loop adaptivity targeting metric we consider is the proportion of learners who fail to reach mastery of at least one of the KCs associated with a workspace before reaching the maximum number of problems set by content designers. Figure 3 provides a histogram showing the overall distribution of this proportion across workspaces. The median workspace has 4.3% of students fail to reach mastery of all its KCs (minimum = 0%; Q1 = .7%; Q3 = 12.1%; maximum = 77.7%).

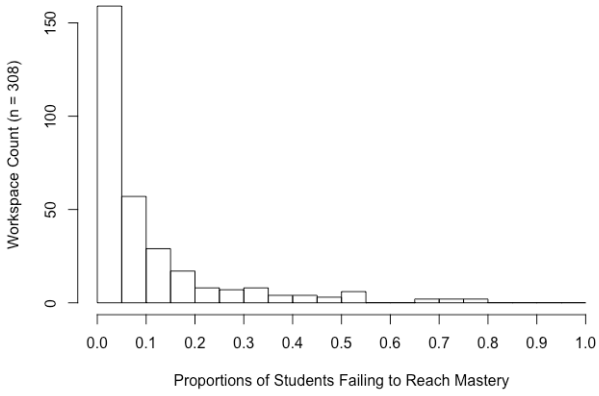


**Figure 3. Histogram illustrating the distribution of the proportion of students failing to reach mastery of all KCs associated with 308 workspaces in the 2018-19 academic year.**

Fancsali et al. [11] argue that students' failure to achieve mastery at a level of aggregation like that of a workspace is an important outcome for predictive modeling, mostly overlooked in the literature on so-called "wheel spinning" (e.g., [6]), which tends to develop models to predict whether students will master particular KCs in a tutoring system, ignoring other elements of how instructional content is presented. Fancsali et al. argue that, given the clustering of KCs within problems, the clustering of problems within workspaces, and the fact that workspaces are the unit at which learners make progress in ITSs like MATHia, reporting outcomes like the count and percentage of KCs that student fail to master (a la Beck and Gong [6]) is of dubious practical value.

Since design-loop adaptivity improvements are likely to often involve redesign of instructional content, we similarly contend that measures closely aligned to instructional delivery are likely to be helpful in targeting this process. Large proportions of students failing to master instructional content are likely to be important in determining what learning content to improve with limited resources. This metric serves as a foil to a second approach.

## 4.2 DKT vs. BKT Prediction Performance

Extensive recent literature (e.g., [12, 18, 22]) considers deep learning approaches to the problem of predicting student performance at fine-grained opportunities to demonstrate mastery of KCs in learning systems like ASSISTments [13]. DKT [22] has been compared (e.g., [12, 18] to BKT and logistic regression approaches to the same type of prediction task [21]. Early work demonstrated that DKT generally had superior prediction performance compared to BKT [22], but subsequent literature also suggests that variations of BKT (e.g., modeling "forgetting") and logistic regression approaches can bridge some, if not most, of the gap in prediction performance (e.g., [18, 29]).

Nevertheless, we seek to better understand the extent to which DKT out-performs BKT when considered workspace-by-workspace across a large dataset from MATHia, which presents a wide variety of learning experiences. We find that, for a variety of workspaces, classic BKT's performance is often comparable to DKT even without accoutrements added in the work of Khajah et al. [18]. Further, in keeping with our primary concern in the present work, we explore the extent to which observed differences in performance between the two approaches, especially examples of DKT's far superior prediction performance, might serve as a metric for targeting improvement work for MATHia workspaces, possibly indicating an especially flawed KC model.

### *4.2.1 Modeling Approach*

We rely on the Khajah et al. [18] implementation of DKT with long short-term memory (LSTM) recurrent units.[1] We use Yudelson's *hmm-scalable*[2] implementation of classic BKT parameter fitting using expectation maximization [30]. We learn DKT and BKT models for each of the 308 workspaces, splitting the data for each workspace into training and test sets with a 80%-20% student-level split and calculate the AUC (area under the receiver-operating characteristic curve) on the test set following methods in Khajah et al. [18]. BKT and DKT models are trained and tested on the same datasets. AUC is a measure of the extent to which a model can "discriminate" between or predict students' correct and incorrect responses in the held-out test set. An AUC value of 0.5 indicates "chance" ability to discriminate between two classes; a value of 1.0 indicates perfect discrimination.

### *4.2.2 Results*

Table 2 provides summary statistics for AUC performance for DKT, BKT, and AUC differences of these methods over all workspaces. As expected, DKT generally provides superior prediction performance to classic BKT over the 308 workspaces. However, there is substantial variability, with classic BKT in some cases, albeit many (but not all) with relatively small sample sizes, even out-performing DKT. While there is a modest, statistically significant positive correlation between the AUC difference in DKT and BKT and sample size (i.e., the number of student-sequences available for training and testing) ($r = .2$; $p < .001$), BKT performs comparably to DKT on a number of workspaces with tens of thousands of students' data, and BKT only underperforms DKT by approximately .07 AUC units for the median workspace. The Q1 value for this difference (the greatest difference over 77 workspaces) is approximately in line, in terms of AUC units, with a value (.03 AUC units) declared comparable by Khajah et al. [18] for BKT "variants" compared to DKT.

The difference in AUC between DKT and BKT is uncorrelated with the proportion of students who fail to reach mastery ($r = -.05$; $p = .4$) and is thus not an indicator of the relative difficulty of particular workspaces, regardless of the source of difficulty.

**Table 2. Summary statistics for AUC performance over 308 workspaces of DKT and BKT models and of the difference between DKT and BKT performance (Δ); negative minimum value indicates better BKT performance for some workspaces.**

| AUC | Min. | Q1 | Med. | Q3 | Max. |
|---|---|---|---|---|---|
| **DKT** | .5852 | .7839 | .8331 | .8783 | .9763 |
| **BKT** | .5150 | .7045 | .7456 | .7854 | .9563 |
| **Δ** | -.0802 | .0361 | .0676 | .1281 | .3073 |

### 4.2.3 *Practical Promise*

We consider two observations relating to workspace design patterns that emerge from considering workspaces with the largest differences in terms of DKT's (generally better) prediction performance compared to BKT. First, we consider the design of a particular workspace as a prime target for design-loop adaptivity to student knowledge, motivation, and affect. Second, we consider more general design patterns in workspaces on which DKT and BKT performance differences are greatest, suggesting more "macro-level" design-loop adaptivity that may affect broader categories of workspaces.

#### *4.2.3.1 Example Workspace*

The second greatest observed difference in AUC occurred for the workspace "Checking Solutions to Linear Equations" (DKT AUC = .968; BKT AUC = .684). A mere 0.2% of students fail to master all KCs in this workspace, suggesting that it may not be "flagged" for design-loop adaptivity improvements based on difficulty. Nevertheless, careful inspection of the workspace yields several areas for improvement.

This workspace presents students with problems (See Figure 4) like: "Jordan solved the equation $-3u - 8 = 10$. She calculated $u = -6$. Use the Solver to check Jordan's solution." The student is then presented with a menu-based equation solver. Work with the equation solver should involve the student substituting in the solution value from the problem presentation and checking whether the result is a balanced equation. After choosing "Substitute for variable" from the menu, the student then must input a value on the left-hand side of the equation (see Figure 5).

Problems in this workspace present both correct and incorrect cases, but the KC model does not distinguish between correct and incorrect cases, making problems with a correct solution targets for possible gaming the system. For example, in the problem in Figure 5, the student might enter $10$ to complete "$10 = 10$." This response may not reflect having correctly carried out the variable substitution to arrive at this solution. KCs in this workspace are also not currently mapped to work in the solver; the solver provides hints and just-in-time feedback on errors, but it is not instrumented to track KC mastery. Once the student has entered

[1] https://github.com/mmkhajah/dkt

[2] https://github.com/myudelson/hmm-scalable

the appropriate value, two questions appear on the left of the screen (see Figure 6), and student responses to these questions trigger updates to two KCs at a time.

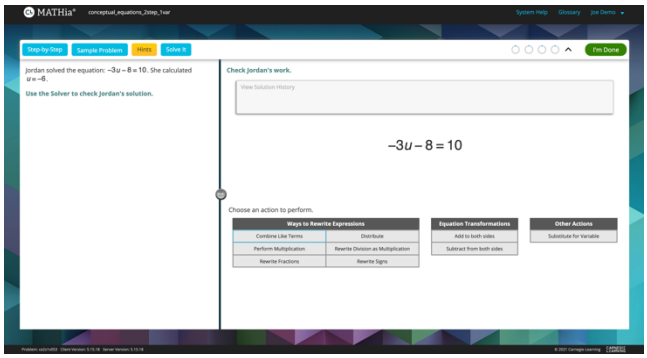


**Figure 4. Screenshot in the MATHia workspace "Checking Solutions to Linear Equations."**

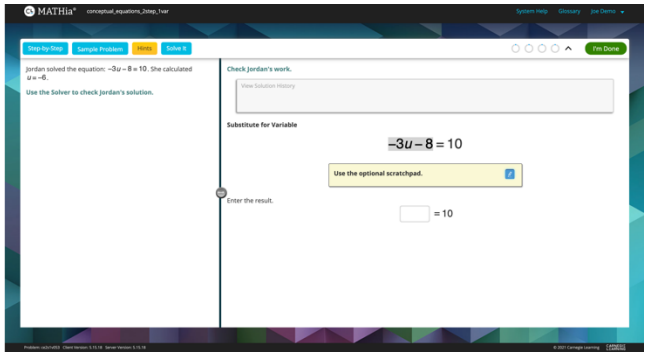


**Figure 5. Screenshot after the student has selected "Substitute for variable" from the equation solving menu (see Figure 4).**

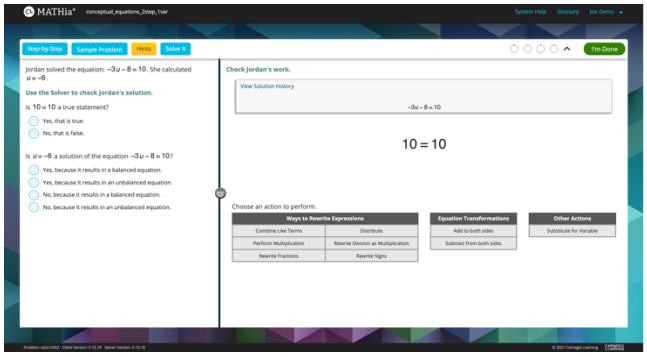


**Figure 6. Screenshot after the student has entered the value 10 on the left-hand side of the equation (see Figure 5).**

Following design-loop adaptivity steps laid out by Huang et al. [15], we have identified (1) where the KC model can be refined, and (2) areas for task redesign. The third step would involve fitting a BKT model using the hypothetical re-mapping of KCs to steps within problems in existing data to determine whether the hypothesized, refined KC model fits the data better than the existing KC model. Future work will experimentally test workspace redesigns to "close the loop" (cf. [16, 25]) between this data-driven approach and empirical learning outcomes.

### *4.2.3.2 Prominent Design Patterns*

Patterns emerge in comparing the performance of DKT to BKT over 308 workspaces. In the top twenty workspaces in which DKT outperforms BKT, differences in AUC units range from .307 to .212, and all provide constrained input mechanisms relative to broader MATHia content. Fourteen workspaces (70%) involve equation solving, and the others are split between those that involve placing values on a number line and those in which problem input is provided via drop-down menus.

Gervet et al. raise questions about explanations for observed properties of DKT in predicting student performance. Can DKT, for example, "better pick up on local patterns of student behavior like gaming the systems" [12]? While far from conclusive, DKT's performance for the workspace "Checking Solutions to Linear Equations" could exemplify this phenomenon. Workspaces with more constrained inputs may provide examples where DKT "picks up" on local patterns that BKT does not. Future work ought to investigate whether these particular types of relatively constrained input mechanisms are easy to "game" or whether and how DKT learns local performance patterns.

Equation solver and number line workspaces are widespread in the top workspaces in which DKT outperforms BKT. "Checking Solutions to Linear Equations" has readily apparent flaws, suggesting that our approach may be promising in targeting instructional improvement work. Systematic review of these results remains future work.

## 5. DISCUSSION

There are numerous questions for future research. That differences in AUC between DKT and BKT are uncorrelated with an important measure of instructional ineffectiveness, combined with DKT's ability to find regularities in data that are not found by BKT suggests that this difference may be signaling important workspace characteristics. Analysis of a particular workspace (§4.2.3.1) suggests that DKT-BKT differences may signal inadequacies in the KC model. These findings can be compared to the results of data-driven search for better KC models [8]. Improvements can be made to the workspace, and A/B tests can "close the loop" and establish more effective approaches.

Systematic analysis of instructional content and prediction performance differences in DKT and BKT might follow work that explores a space of properties and features of particular tutor "lessons" to determine which predict students' affect, gaming the system, and off-task behavior [4-5, 10]. Comparisons to logistic regression methods (e.g., [12]) are also needed.

Naïve learning engineering may focus on reducing students' mastery failures. Such an approach could lead to "over-simplified" tasks that don't produce failure because they don't require much knowledge. Large differences between DKT and BKT may help identify over-simplified workspaces that provide opportunities for students to game the system [3, 12]. To what extent do gaps in modeling techniques' performance indicate unproductive patterns of "local" behavior in particular workspaces? What else drives differences? What other behavior patterns indicate ways to target improvement?

Methodologically, our "non-monolithic" analysis of a large educational data set treats component instructional experiences as units for analysis. Such analytical decomposition is vital to practical learning engineering to improve instructional systems and large portfolios of content used by learners every day.

## 6. ACKNOWLEDGMENTS

This research was sponsored by the National Science Foundation under the award The Learner Data Institute (Award #1934745). The opinions, findings, and results are solely those of the authors and do not reflect those of the National Science Foundation.

## 7. REFERENCES


[1] Aleven, V., McLaughlin, E.A., Glenn, R.A., Koedinger, K.R. 2017. Instruction based on adaptive learning technologies. In *Handbook of Research on Learning and Instruction*, 2nd Ed., Routledge, New York, 522–560.

[2] Aleven, V., Sewall, J. 2016. The frequency of tutor behaviors: a case study. In *ITS 2016*. LNCS, vol. 9684, Springer, Cham, 396–401. https://doi.org/10.1007/978-3-319-39583-8_47

[3] Baker, R.S., Corbett, A.T., Koedinger, K.R., Wagner, A.Z. 2004. Off-task behavior in the Cognitive Tutor classroom: When students "game the system." In *Proceedings of ACM CHI 2004: Computer-Human Interaction*, 383-390.

[4] Baker, R.S.J.d. 2009. Differences between intelligent tutor lessons, and the choice to go off-task. In *Proceedings of the 2nd International Conference on Educational Data Mining*, 11-20.

[5] Baker, R.S.J.d., de Carvalho, A.M.J.A., Raspat, J., Aleven, V., Corbett, A.T., Koedinger, K.R. 2009. Educational software features that encourage and discourage "gaming the system". In *Proceedings of the 14th International Conference on Artificial Intelligence in Education*, 475-482.

[6] Beck J.E., Gong Y. 2013. Wheel-Spinning: Students Who Fail to Master a Skill. In *Artificial Intelligence in Education.* AIED 2013. LNCS, vol 7926. Springer, Berlin/Heidelberg. https://doi.org/10.1007/978-3-642-39112-5_44

[7] Bloom, B. S. 1968. Learning for mastery. *Evaluation Comment* 1(2). Los Angeles: University of California at Los Angeles, Center for the Study of Evaluation of Instructional Programs.

[8] Cen, H., Koedinger, K.R., Junker, B. 2006. Learning factors analysis: A general method for cognitive model evaluation and improvement. In *Proceedings of the 8th International Conference on Intelligent Tutoring Systems* Springer-Verlag, Berlin, 164–175.

[9] Corbett, A.T., Anderson, J.R. 1994. Knowledge tracing: Modeling the acquisition of procedural knowledge. *User Modeling and User-Adapted Interaction* 4, 253– 278.

[10] Doddannara L.S., Gowda S.M., Baker R.S.J.., Gowda S.M., de Carvalho A.M.J.B. 2013. Exploring the relationships between design, students' affective states, and disengaged behaviors within an ITS. In *Artificial Intelligence in Education.* AIED 2013. LNCS, vol 7926. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-39112-5_4

[11] Fancsali S.E., Holstein K., Sandbothe M., Ritter S., McLaren B.M., Aleven V. 2020. Towards practical detection of unproductive struggle. In *Artificial Intelligence in Education*. AIED 2020. LNCS, vol 12164. Springer, Cham. https://doi.org/10.1007/978-3-030-52240-7_17

[12] Gervet, T., Koedinger, K., Schneider, J., & Mitchell, T. (2020). When is deep learning the best approach to knowledge tracing? *Journal of Educational Data Mining* 12(3), 31-54. https://doi.org/10.5281/zenodo.4143614

[13] Heffernan, N.T., Heffernan, C.L. 2014.The ASSISTments ecosystem: building a platform that brings scientists and teachers together for minimally invasive research on human learning and teaching. *Int. J. Artif. Intell. Educ.* 24(4), 470–497.

[14] Hess, F., Saxberg, B. 2014. *Breakthrough leadership in the digital age: Using learning science to reboot schooling*. Thousand Oaks, CA: Corwin.

[15] Huang Y., Aleven V., McLaughlin E., Koedinger K. 2020. A general multi-method approach to design-loop adaptivity in intelligent tutoring systems. In *Artificial Intelligence in Education.* AIED 2020. LNCS, vol 12164. Springer, Cham, 124-129. https://doi.org/10.1007/978-3-030-52240-7_23

[16] Liu, R., & Koedinger, K.R. (2017). Closing the loop: Automated data-driven cognitive model discoveries lead to improved instruction and learning gains. *Journal of Educational Data Mining* 9(1), 25–41.

[17] Lovett, M., Meyer, O., Thille, C. 2008. The open learning initiative: Measuring the effectiveness of the OLI statistics course in accelerating student learning. *Journal of Interactive Media Education*, 14.

[18] Khajah, M., Lindsey, R.V., Mozer, M.C. 2016. How deep is knowledge tracing? In *Proceedings of the 9th International Conference on Educational Data Mining (Jun 29 - Jul 2, 2016, Raleigh, NC, USA)*. EDM 2016. International Educational Data Mining Society, 94-101.

[19] Koedinger, K.R., Corbett, A.T., Perfetti, C. 2012. The knowledge-learning-instruction framework: Bridging the science-practice chasm to enhance robust student learning. *Cognitive Science* 36, 5, 757–798.

[20] Koedinger, K.R. & McLaughlin, E.A. 2010. Seeing language learning inside the math: Cognitive analysis yields transfer. In *Proceedings of the 32nd Annual Conference of the Cognitive Science Society*. Austin, TX, Cognitive Science Society, 471-476.

[21] Pavlik, P.I., Cen, H., Koedinger, K.R. 2009. Performance factors analysis – a new alternative to knowledge tracing. In *Proceedings of the 2009 Conference on Artificial Intelligence in Education*. IOS Press, 531–538.

[22] Piech, C., Bassen, J., Huang, J., Ganguli, S., Sahami, M., Guibas, L.J., Sohl-Dickstein, J. 2015. Deep knowledge tracing. In *Advances in Neural Information Processing Systems*, 505–513.

[23] Ritter, S., Anderson, J.R., Koedinger, K.R., and Corbett, A.T. 2007. Cognitive Tutor: applied research in mathematics education. *Psychon. B. Rev.* 14, 249-255.

[24] Ritter, S., Yudelson, M., Fancsali, S.E., Berman, S.R. 2016. How mastery learning works at scale. In *Proceedings of the 3rd Annual ACM Conference on Learning at Scale* (Apr 25 - 26, 2016, Edinburgh, UK). L@S 2016. ACM, New York, NY, 71-79.

[25] Rosé, C.P., McLaughlin, E.A., Liu, R., Koedinger, K.R. 2019. Explanatory learner models: Why machine learning (alone) is not the answer. *Br J Educ Technol* 50, 2943-2958. https://doi.org/10.1111/bjet.12858

[26] Simon, H.A. 1967. The job of a college president. *Educational Record* 48(Winter), 68–78.

[27] Stamper, J. & Koedinger, K.R. 2011. Human-machine student model discovery and improvement using data. In *Proceedings of the 15th International Conference on Artificial Intelligence in Education*. Springer, Berlin/Heidelberg, 353-360.

[28] Vanlehn, K. 2006. The behavior of tutoring systems. *International Journal of Artificial Intelligence in Education*, 16(3), 227–265.

[29] Wilson, K.H., Xiong, X., Khajah, M., Lindsey, R.V., Zhao, S., Karklin, Y., Van Inwegen, E.G., Han, B., Ekanadham, C., Beck, J.E., Heffernan, N., Mozer, M.C. 2016. Estimating student proficiency: Deep learning is not the panacea. In *Proceedings of the Workshop on Machine Learning for Education at the 30th Conference on Neural Information Processing Systems (NIPS 2016).*

[30] Yudelson, M., Koedinger, K., Gordon, G. 2013. Individualized Bayesian Knowledge Tracing Models. In *Proceedings of 16th International Conference on Artificial Intelligence in Education* (Memphis, TN) AIED 2013. LNCS vol. 7926, Springer-Verlag, Berlin/Heidelberg, 171-180.