CHAPTER 15 –Knowledge Component (KC) Approaches to Learner Modeling

Vincent Aleven and Kenneth R. Koedinger
Carnegie Mellon University - Human-Computer Interaction Institute

Introduction

A distinguishing characteristic of ITSs is that they engage in learner modeling, meaning that they estimate key characteristics of each learner based on interactions with the tutoring system. Although learner modeling is not fundamentally different from other forms of formative assessment, some properties of the learner modeling approaches applied in ITSs are not widely shared with other forms of formative assessment. First, learner modeling methods used in ITSs typically assess learners over time and across many measurement moments. Therefore, they tend to take into account that the learner is learning. Second, the assessment is done primarily to support pedagogical decision making by the tutoring system. Third, learner modeling and instruction go hand in hand; the learners are assessed on the same problems on which they are receiving instruction. Finally, although the term "assessment" is typically associated with a focus on learners' knowledge and skills, across the spectrum of ITSs, learner models hold many different types of information.

In this chapter, we focus on models that represent the knowledge targeted in the instruction and also knowledge that may be used to learn target knowledge (e.g., metacognition or motivational beliefs related to learning). These kinds of learner models capture what a given learner knows (tacitly or explicitly) and does not know yet, so that the system can tailor its instruction accordingly. In particular, we focus on approaches that aim to model the learner's knowledge with sufficient specificity that the learner's performance on future tasks can be accurately predicted. In pursuit of this goal, it has turned out to be useful to model the learner's knowledge as a set of inter-related KCs. The Knowledge-Learning-Instruction (KLI) Framework defines a KC as "an acquired unit of cognitive function or structure that can be inferred from performance on a set of related tasks" (Koedinger, Corbett & Perfetti, 2012). Examples of ITSs that have taken a KC approach to learner modeling are Cognitive Tutors (Anderson, Corbett, Koedinger & Pelletier, 1995; Koedinger & Aleven, 2007), constraint-based tutors (Mitrovic, Mayo, Suraweera & Martin, 2001), and Andes (VanLehn et al., 2005).

The KC approach to learner modeling comes with a number of key challenges. First, in any domain of interest, how can we determine what the KCs are that learners acquire? Put differently, how can the targeted expertise be decomposed into a set of KCs that have psychological reality, as evidenced by the fact that the model can be used to accurately predict student performance across tasks and over time? Second, given a KC model, how can a tutoring system accurately determine, for any learner at any point in time, what that learner's level of mastery is for each of the individual KCs? Third, given an accurate model of an individual learner, how can an ITS tailor its instruction so learners learn more effectively and/or efficiently? In this chapter, we focus primarily on the first question while briefly touching on the other two. A key point in our chapter is that automated approaches to creating and refining KC models can substantially enhance "manual" approaches.

In the KC approach to learner modeling, building an intelligent tutor for a new domain requires a KC model. Since existing KC models are rarely available or "just right," typically, a new model needs to be built. Doing so, however, requires a substantial amount of effort. In a traditional approach to KC modeling, an author creates a model by hand, based on careful cognitive task analysis. Ideally, the author would conduct both theoretical task analysis and empirical task analysis (Lovett, 1998). In theoretical cognitive task analysis, an author elucidates the knowledge demands of a task by carefully analyzing its structure (e.g., its possible goal/subgoal hierarchies). In empirical task analysis, by contrast, an author collects data about problem solving in the given task domain, employing methods such as think-alouds, interviews of experts, analysis of errors novices make on written tests, difficulty factors analysis (DFA), and so forth (Baker, Corbett & Koedinger, 2007).

Given that creating a KC model tends to be labor intensive, ITS authoring tools support aspects of cognitive task analysis and KC modeling. In this chapter, we review how one set of authoring tools (the Cognitive Tutor Authoring Tools or CTAT, for short) support theoretical cognitive task analysis and KC modeling as an integral part of the process of building a tutor, with substantial gains in authoring efficiency (Aleven, McLaren, Sewall & Koedinger, 2009). However, it is becoming clear that manual KC modeling does not always yield optimal KC models. Manual KC modeling efforts have produced multiple cases where KC models created or refined through automated or semi-automated methods did better (e.g., on predicting student performance) than real-world models created by hand. For example, our original assumption in building the cognitive model (a kind of KC model) for story problem solving within Cognitive Tutor Algebra was that story problems are hard because it is challenging to comprehend a story and translate it to an equation. Without collecting data, these hand-crafted models would have been wrong. For beginning algebra students, introductory story problems are not solved using equations and are actually easier to solve than the matched equation (Koedinger & Nathan, 2004). Further, the challenge in translating a story to an equation is more in the difficulties in producing the symbolic equation than in comprehending the English (Koedinger & McLaughlin, 2010). As a second example, in the Geometry Cognitive Tutor, the cognitive model in the fielded tutor did not make a distinction for “decomposition problems” between those where the decomposition was scaffolded and those where it was not. Data mining showed a large and important difference between these two situations, however (Stamper & Koedinger, 2011), and a tutor redesigned based on the data-discovered new model yields more efficient and effective learning (Koedinger, Stamper, McLauglin, Nixon, in press). Third, across 11 datasets, an automated model refinement algorithm called Learning Factors Analysis (LFA) found better cognitive models than the original models and than best data-driven models (Koedinger, McLaughlin & Stamper, 2012).

Part of the problem may be that in practice, the manual approach to KC modeling relies too much on theoretical task analysis and not enough on empirical task analysis. It may be fair to say that the first cut at a KC model is often created without much empirical task analysis. In the process of developing a tutor, it can be difficult to find the time to collect data with students from the target population. A second reason why hand-crafted models can be suboptimal is that even common methods for empirical cognitive task analysis may not be able to address all modeling decisions that arise (often, there are simply too many of them) and may not uncover subtle over-generalizations and under-generalizations that novices make while learning new material. Such discoveries may require larger data sets than those typically collected in empirical task analysis, such as those collected as tutor log data.

As second focus of the chapter, therefore, is on methods and tools for discovering or refining KC models using data on task performance (e.g., tutor log data). Specifically, we concentrate on three data-driven and/or machine learning approaches: (1) use of visualization and statistical modeling tools for analysis and refinement of existing models based student log data for example, by using the DataShop (Koedinger et al., 2010); (2) combining human expertise, machine learning, and log data to automatically discover better KC models (LFA) (Koedinger et al., 2012), again using tutor log data, and (3) using machine-learning-based model of student learning (e.g., SimStudent) to learn an accurate KC model from being tutored in the domain (Li, Matsuda, Cohen & Koedinger, 2011).

Background: What Is KC Modeling?

In this section, we review the notion of a KC as a central element in learning modeling. In the next section, we review some of the key evidence in the ITS literature in support this approach – instances in which a KC-based learner model was used in an ITS to adapt instruction with measurable impact on student learning.

But first, what is a knowledge component? Does any decomposition of the knowledge needed for a particular task result in a KC model? And what does it mean for a KC model to have psychological reality? As mentioned, in the KLI framework (Koedinger et al., 2012), a KC is an acquired unit of cognitive functioning, which (being a latent construct) has to be inferred from learners' performance. This definition encompasses a wide range of knowledge types, not just procedural knowledge. A complex cognitive task is viewed as involving many fine-grained KCs (Anderson, 1993). These KCs are learned separately in the sense that practice with one does not cause improved mastery of another. While there is no possibility of *transfer from one KC to another,* KCs provide a good way of thinking about *transfer between problems or problem steps*. In many domains (e.g., mathematics) different problems require use of overlapping sets of KCs. One expects to see transfer between different problems or problem steps, in the sense that practice on one leads to improved performance on the other, exactly to the extent that there is overlap in the KCs needed for these problems or steps. A key characteristic of the KC approach to learner modeling (which is not shared with other approaches to assessment, even prominent ones such as IRT analysis, cf., Wilson & De Boeck, 2004) is that such transfer relations are explicitly accounted for. The KCs are considered to be the units of transfer.

We can now state what it means for a KC model to have psychological reality. Put simply, it means the model can be used to make accurate predictions of a given student's performance on future problems based on that student's performance on past problems. Put differently, it means that the transfer predictions that are implied by the model are actually observed in data about student performance, typically, tutor log data (Aleven, 2010). This description leaves open how a KC model can be used to predict future performance. There are multiple ways of doing so; below we describe one often-used method. A key pre-requisite for making accurate predictions is that the model captures knowledge at the right level of generality.

It may be helpful to contrast the KC approach to learner modeling with an alternative but incorrect viewpoint, namely, that "students learn what they spend time on." While that statement may have a ring of truth to it, it is actually misleading. It depends critically on how we categorize student activities. When we look at student activity at the level of problems, we may go wrong. We need to categorize problem-solving steps, not problems, and we need to do so with respect to their cognitive demands. In a study on a tutor unit with geometry area composition problems, model discovery revealed a different skill is needed for unscaffolded decomposition steps than for scaffolded ones (Stamper & Koedinger, 2011). In these decomposition steps, students must realize that they can compute the area of irregular geometric shapes as the sum or difference of the area of two different regular shapes. In the fully-scaffolded tutor, students did not need to plan to decompose the irregular area shape on their own, as the tutor prompted for the areas of the regular shapes and then for their sum or difference. A redesigned tutor included unscaffolded problems where students were not prompted to decompose the figure; they basically had to figure it out themselves, as one big step in the tutor problem. New problems that isolate the decomposition planning step and did not require its execution were also created and used in the redesigned tutor. The students in the redesigned tutor condition learned better (Koedinger, Stamper, McLaughlin & Nixon, in press). All of the problems that both groups solved in this study were composition problems, and the students who worked with the scaffolded tutor spent more time on these problems overall. Thus, by the simple statement above, they should have learned the decomposition skills better. They did not. Students using the redesigned tutor learned the decomposition skill better, because only they had practiced that skill – in the scaffolded group, the tutor's scaffolding took over a critical part of the work. Generally, students learn the KCs they spend time practicing. However, what these KCs are is not obvious. KCs are not directly observable and most are not open to conscious reflection, despite our strong feelings of self-awareness of our own cognition. They can, however, be inferred and discovered from student performance data across multiple tasks via a statistical comparison of alternative categorizations, that is, of alternative KC models.

As another contrasting case, we consider the ALEKS system and its underlying knowledge space algorithm (Falmagne, Koppen, Villano, Doignon & Johannesen, 1990). ALEKS is a commercial system for mathematics practice, focused on individualized problem selection to diagnose a student's knowledge state. It presents problems to students and provides mathematical tools to solve them (e.g., digital version of a compass, on-screen calculator, grapher, etc.) but does not provide the step-by-step guidance (or equivalently, an "inner loop") characteristic of ITSs (VanLehn, 2006, 2011).

The adaptive problem selection technique in ALEKS relies on a "knowledge space" created offline that captures presumed precedence relations among problem types. That problem type A precedes (i.e., is a prerequisite of) type B is determined empirically. In essence, A is assumed to precede B if most students that get B right also get A right and if of those that get B wrong many get A right. In other words, evidence that a student knows B implies they know A. These precedence relations define a partial order among all problem types and this partial ordering is used to diagnose a student's knowledge state, defined as the set of problem types that the student masters (i.e., can solve correctly). From the knowledge state, ALEKS can select an appropriate next problem – one of the problem types that follow those on the edge of what that student knows.

The KC approach described in the current chapter and ALEKS' approach using knowledge spaces share an important goal, namely, learner modeling in support of individualized problem selection. They also share the view that student performance data is a valuable source of information for model building beyond intuition or analysis. A key difference is that the KC approach attempts to explicate what the KCs are, whereas there is no such attempt in the knowledge space approach. Whereas the ALEKS approach uses data *in place* of theory and intuition, the KC approach uses data *in addition to* theory and intuition. To illustrate the difference, the knowledge space algorithm will identify A as a prerequisite of B if A is consistently easier than B (for the same students)[9]. However, unless the KCs used in A are also needed in B (e.g., A is finding factors of a number and B is finding the least common multiple of two numbers, which requires finding factors), the statistical regularity used to infer a prerequisite may be spurious. A second reason why the KC model approach values explication of the KCs is that the resulting explanatory model may not only enhance generalization, but can better be interpreted for use in course redesign (Lovett, Meyer & Thille, 2008) and in tutor redesign (Koedinger & McLaughlin, 2010; Koedinger, Stamper, McLaughlin & Nixon, in press) .

It is worth noting that a knowledge space approach is not incompatible with a KC modeling approach and, indeed, we would recommend analysts using a knowledge space to try to explicate the in-common KCs implied by the links in the prerequisite graph. Links should be eliminated (or not produced) when a plausible KC cannot be generated.

[9] Note that the knowledge space algorithm is more nuanced then this treatment might imply: It is possible for A to be easier than B, but still have a number of students getting B right and A wrong and if that happens enough, a precedent relation from A to B will not be inferred.

Use of KC-Based Learner Models and Evidence of Effectiveness

To motivate the KC approach, we briefly touch on the second and third key challenges identified in the introduction: Once a KC model has been created for a given task domain, how can an ITS assess an individual student's mastery of the KCs in the model? Also, what are good ways of using that assessment to individualize instruction? We review some of the key studies in the literature.

A widely used approach to tracking individual learners' knowledge growth using a KC-based model is BKT (Corbett & Anderson, 1995). In this approach, the posterior probability of mastering a KC is updated each time a student encounters a problem step involving that KC. The posterior probability depends on the performance on that step, the prior probability of mastery, the likelihood of learning from a step, and conditional probabilities that allow for the possibility that the student may guess the step or slip. These parameters are typically set separately for each KC and are sometimes estimated from data (Corbett & Anderson, 1995). As an alternative to BKT, Bayesian Networks are often used to maintain KC-based learner models (Conati, Gertner & Vanlehn, 2002). Further, recent EDM research has developed new methods for tracking individual learners' knowledge growth in terms of KCs (e.g., Gong, Beck & Heffernan, 2011; Lee & Brunskill, 2012). Some of these methods have been shown to have greater predictive accuracy than BKT. To the best of our knowledge, these methods have not yet been used for online learner modeling or online individualization of instruction, although one of them, AFM (described below) has been used offline for KC model and tutor improvement, and there is much ongoing work in this area.

The primary use of KC-based learner models in ITSs is to support individualized problem selection (Corbett, McLaughlin & Scarpinatto, 2000; Corbett & Anderson, 1995; Mitrovic & Martin, 2004). In this approach, the system uses its learner model to select problems that (for the given student, at the given point in time) target unmastered KCs. It continues to do so until the learner model indicates that the learner has reached mastery of the targeted KCs. This approach to individualized problem selection, often called "Cognitive Mastery," is used in Cognitive Tutors, a type of ITS used widely in American mathematics education (Anderson et al., 1995; Koedinger & Aleven, 2007). Individualized problem selection based on a KC model can be very effective in enhancing student learning. In studies with the Lisp Tutor, an early Cognitive Tutor, Corbett et al. (2000) found substantial improvement in student learning outcomes, due to this form of individualization, with on average only a very modest increase in the time spent. Also, a study by Cen et al. (2007) showed that improving the accuracy of learner modeling – by tuning the parameters of the BKT procedure – leads to more efficient learning by students. These studies on individualized problem selection provide key evidence that adaptive individualization by an ITS enhances student learning, an important argument in favor of this advanced learning technology.

KC-based learner models have also been used for individualizing a number of other aspects of tutoring systems. For example, they have been used to individualize ways in which worked examples are used in ITSs. In the SE-COACH by Conati and VanLehn, a KC-based learner model is used to decide what steps in a worked example a particular learner should be prompted to explain, which for early learners was shown to be helpful, compared to a system in which the steps to explain were not selected on an individual basis (Conati & Vanlehn, 2000). In other work, a KC-based learner model was used to select, on an individual basis, suitable examples for analogical comparison (Muldner & Conati, 2007). In a study by Salden and colleagues, a KC-based learner model was used to transition adaptively (KC by KC) from studying worked examples to solving problems. In a lab study and a classroom study, this adaptive technique led to better or more efficient learning (Salden, Aleven, Renkl & Schwonke, 2009). KC-based learner models have also been used as components in models of metacognition, affect, and specific (desirable or undesirable) learning behaviors. For example, in work on modeling adaptive help seeking behavior (a key metacognitive skill), the learner's decision to seek help depends on the student's level of mastery of the given KCs. Help requests are deemed to be adaptive only on steps that involve KCs that are unfamiliar to the student at the given point in time (Aleven, Roll & Koedinger, 2012; Roll, Aleven, McLaren & Koedinger, 2011). Further, in work on creating machine-learned models of various aspects of learners and learner behaviors (e.g., gaming the system, off-task behavior, affective states, etc.) the level of mastery of a KC is one of the features that serves as input to the machine-learned classifier (Baker et al., 2006; Baker, Goldstein & Heffernan, 2011). Finally, KC models are often used as open learner models (OLMs) (Bull & Kay, 2007; Long & Aleven, 2013; Mitrovic & Martin, 2007), which have become a common feature of ITS. The OLM communicates a sense of progress and may support useful self-assessment and reflection. A recent study showed higher post-test scores in a tutor for linear equation solving with an OLM, compared to a version without the OLM (Long & Aleven, 2013).

One way to generalize from this body of empirical work is to say that individualization of instruction based on a KC-based learner model has been shown to be especially effective in the system's outer loop (VanLehn, 2006), as studies on individualized problem selection and example fading indicate, while also being somewhat effective (but not as effective) in the inner loop. This generalization is quite coarse, however – future work may provide a more nuanced perspective.

Tutor Authoring Tools and the KC Approach to Learner Modeling

Let us now return to the first of the three challenges listed in the introduction of the chapter: How can accurate KC models be created? In the remainder of the chapter, we discuss the pros and cons of a variety of approaches, starting with ITS authoring tools that support a "manual" approach to KC modeling. By manual we mean an approach unaided by data mining or machine learning algorithms, although it could involve data collected in the course of doing cognitive task analysis. There are many ways in which authoring tools for ITS can support a manual KC approach to learner modeling. We see two broad areas of functionality. First, tools can provide support for defining/building KC models as part of an iterative tutor development process. This kind of tool support is used offline, when authoring or refining a tutor. Second, authoring tools can provide mechanisms that tutors can use at run time to maintain/update KC-based learner models and tailor aspects of the instruction. While there are many ITS authoring tools (Murray, Blessing & Ainsworth, 2003), we focus on how these two areas of functionality are implemented in CTAT, a tool suite with which they have been involved in the past 10 years (Aleven et al., 2009). Many of our points hold for other ITS authoring tools as well.

CTAT is suite of tools for ITS authoring that has been used to build a wide range of tutors, many of which have been used in actual classrooms or other real educational settings. These tools support KC modeling as an integral part of tutor development. CTAT supports the authoring of two types of tutors: example-tracing tutors and Cognitive Tutors, both of which reflect a KC approach to instruction and learner modeling. Example-tracing tutor rely on behavior graphs to evaluate and interpret student behavior, cognitive tutors rely on a rule-based cognitive model. We view both behavior graphs and rule-based cognitive models as a form of KC models.

A behavior graph is a map of the solution space for a given problem (Koedinger, Aleven, Heffernan, McLaren & Hockenberry, 2004; Newell & Simon, 1972). Creating behavior graphs is a key activity in creating an example-tracing tutor. The tutor uses the graph to interpret student behavior (Aleven et al., 2009). As a first step toward creating such a KC model, an author can use a CTAT tool called the Behavior Recorder to record a behavior graph for a given problem, simply by demonstrating the solution paths on the tutor interface designed for the given problem type. The Behavior Recorder records the demonstrated steps in a graph; the graph may have multiple solution paths corresponding to different ways of solving the problem. The links in the graph represent the steps, and the nodes represent problem-solving states. Behavior graphs are not in and of themselves KC models, however, until the author annotates the links with KC labels. These labels (essentially, names of hypothesized KCs) characterize the knowledge each step is hypothesized to require. Thus, the Behavior Recorder helps an author in coming up with a decomposition of the knowledge in the given task domain, as a form of rational task analysis. A key assumption is that – in creating a KC model – it helps to think at the grain size of problem steps rather than whole problems and it helps to have laid out (and thought about) these steps explicitly. Creating a mapping between steps and KCs can be viewed as implicitly formulating transfer predictions (Koedinger et al., 2004). Annotating two steps with the same KCs implies that one expects to see full transfer between these steps; practice on one improves the other, and vice versa. Conversely, assigning different KC labels to two steps implies that one does not believe such transfer will occur. Partial transfer between steps can be modeled by annotating steps with multiple KCs, some of which are shared between the two steps.

Although behavior graphs labeled with KC names are useful KC models, one limitation is that they do not explicitly state the conditions under which the KCs apply. Put differently, the KCs are defined intensionally, by labeling steps in solution graphs, but not extensionally, by stating applicability conditions. This state of affairs is not ideal, because it means that the exact range of transfer for any given KC is not defined. Rule-based cognitive models (the second kind of KC model an author can create with CTAT) address this limitation (Aleven, 2010). Rule-based cognitive models are computer-runnable simulations of student thinking that can solve tutor problems in the ways that students do. In these models, KCs are expressed in IF-THEN format. The IF-part captures the conditions under which a KC is applicable – in other words, it captures how far the KC transfers. Thus, in creating a rule-based model, an author is forced to think hard about these conditions, more so than when creating example-tracing tutors. CTAT provides a host of tools that help in creating production rule models, including tools for visualizing and debugging production rule models (Aleven, McLaren, Sewall & Koedinger, 2006). Interestingly, in the process of creating a rule-based model, it is often very useful to have behavior graphs with KC labels. These graphs are a specification of how the production rule model should behave. CTAT also supports the use of the behavior graph for testing of rule-based cognitive models.

In addition to supporting the development of KC models, CTAT supports the use of such models within running tutors for purposes of assessment, learner modeling, and individualization. Both example-tracing tutors and Cognitive Tutors built with CTAT use their KC model in the inner loop to assess student problem solving and interpret it in terms of KCs. Both types of tutors are capable of tracking learners' knowledge (i.e., their mastery of the KCs in the tutor's KC model) over time using BKT, briefly described above (Corbett & Anderson, 1995). They also provide individualized problem selection based on Corbett's cognitive mastery mechanism, also described above (Corbett et al., 2000). Thus, as one would expect from an ITS authoring tool, CTAT supports use of the learner model for outer loop decision making, namely, problem selection (VanLehn, 2006). It does not support use of the learner model in the inner loop to individualize instruction, given the limited empirical evidence that inner loop adaptivity leads to improved student learning. Nonetheless, it will be helpful if ITS authoring tools can support use of a learner model to individualize various aspect of the inner loop (e.g., deciding what type of hints to give (cf. Arroyo Beck, Woolf, Beal & Schultz, 2000), if only so that more studies can be done to evaluate what kinds of inner loop individualization are most effective.

Returning to an issue raised in the introduction, the CTAT tools for KC modeling primarily support theoretical cognitive task analysis. Behavior graphs and production rule models help in thinking about what knowledge may be needed to solve particular problems, how that knowledge might be decomposed to capture distinctions that students might make, and how widely specific KCs will transfer. It is left up to the author however whether or not to do so with the aid of empirical task analysis. The tools themselves do not directly support empirical task analysis or the use of data to help create KC models, with one exception: a tool called "novice bootstrapping" supports a process in which behavior graphs are created directly from log data of novice problem solving, rather than from expert demonstrations (Harrer, McLaren, Walker, Bollen & Sewall, 2006). This capability is though to be useful especially for problems with large solution spaces or in ill-defined domains. While CTAT proper provides limited support for using data in tutor or KC model development, it is fully integrated with the DataShop (Koedinger et al., 2010), which provides many tools for this purpose.

Tools for Data-Driven KC Model Validation and Refinement

Once a tutor has been created, log data from the tutor can be used in a variety of data-driven methods to evaluate how well the tutor's KC model captures the psychological reality of student problem solving in the given task domain and how it might be refined/improved. We look at a set of tools we have developed called the DataShop. The DataShop is geared toward supporting a KC-based approach to learner modeling. The DataShop is a large, open, online repository for process data from ITSs and other advanced learning technologies. In addition to being a repository with data sets available for secondary analysis, the DataShop supports KC model analysis, discovery, refinement, and testing. It also offers a standardized format for tutor log data. Given that the DataShop offers an extensive set of tools for KC modeling, logging in DataShop format is a useful feature for any ITS authoring tool. CTAT is fully compatible with the DataShop in this sense. That is, all tutors built with CTAT write detailed logs of student-tutor interactions in DataShop format, without extra work by the author.

Using DataShop tools, an author can perform the following:

- Visually inspect learning curves for individual KCs or groups of KCs extracted from log data. Analysis of a KC model often starts with visual inspection of the learning curves. The learning curves may be based on an existing KC model, such as the one that the given tutor is using, or a new model created by hand. Learning curves capture the performance of a group of students on successive opportunities to apply the given KC or KCs. The learning curves show how challenging the KC is for the target population and the rate at which it is learned. Learning curves are useful tools for visualizing student learning with an ITS, both for researchers and developers. The visual appearance of the learning curves can indicate whether the KC model is psychologically real, with smooth, gradually declining curves suggesting that it is, and ragged or flat curves suggesting it might not be.

- Evaluate or validate a KC model by testing how well it predicts performance in the given log data set. As described below, the DataShop uses a logistic regression model called AFM to predict student performance based on a given KC model (Cen et al., 2006; Spada & McGaw, 1985).

- Search for explanations and newly hypothesized model features when learning curves fail to have their theoretically predicted shape (e.g., are ragged) or when a KC model has poor fit with the data (i.e., leads to inaccurate performance predictions using AFM). An author may try to pinpoint where in the tutor's problem set deviations from ideal learning curves occur, so as to come up with hypotheses for how the KC model might be improved. For example, an author might look at problem steps that are notably easier or notably more difficult than the theoretically predicted value. This type of analysis is supported by a DataShop tool called the Performance Profiler.

- Compare learning curves or learning rates (obtained from fitting AFM) across conditions in an experiment, e.g., does one condition lead to a more efficient or more effective learning *process* than another? (cf., Mathan & Koedinger, 2005).

- Compare how well different KC models fit the same data set (using the AFM). This type of analysis is a way of exploring transfer or lack thereof or equivalently, of exploring the level of generality/specificity at which students learn skills or acquire knowledge. Comparing alternative KC models may help an author understand student learning and may be a key step towards finding a better KC model. An author may create alternative models by hand and upload them into the DataShop. The DataShop presents a "score board" ranking the different KC models for a given data set in terms of their predictive accuracy.

- Analyze what errors are frequent in the data set, for purposes of both tutor improvement (e.g., an author might add error feedback messages for the most common or the most confusing errors) as well as model exploration focused on errors.

- Automatically refine a KC model by running an automated best-first search procedure over a space of model variations; this procedure, called LFA, is described below.

Now let us turn to the question how KC models can be evaluated beyond visual inspection of learning curves, a fundamental concern in the KC modeling approach. Above we said that KC models should be judged by their ability to support accurate predictions of student learning across tasks and over time, but we left open the question how exactly a model can be used to make predictions. Although there is not a single right way of doing so, a good way is to use a logistic regression model known as the AFM, shown in Figure 15-1 (Cen et al., 2006; Spada & McGaw, 1985). This model is used in the DataShop.

$$log \frac{p_{ij}}{1- p_{ij}} = \theta_i + \sum_k \beta_k Q_{kj} + \sum_k Q_{kj} \gamma_k T_{ik}$$

Given:

p_{ij}	(0 or 1) probability that student *i* gets step *j* correct
Q_{kj}	(0 or 1) whether KC k is needed for step *j*
T_{ik}	number of opportunities student *i* has had to practice KC *j*

Estimated:

θ_i	proficiency of student *i*
β_k	ease of KC *j*
γ_k	gain for each opportunity to practice KC *j*

Figure 15-1. AFM built into the DataShop for using a KC model to make predictions about student performance

AFM can be used to estimate the probability that the student will get a step in a tutor problem right, based on that and other students' history on problem steps that involve the same KCs. Each step is assumed to involve one or more KCs. Thus, to use the model it is necessary to have a mapping between tutor problem steps and KCs. In a running ITS, this mapping is typically created in the system's inner loop (VanLehn, 2006), using the tutor's KC model; the mapping is then recorded in the tutor log data. In offline approaches to KC model validation, a static "Q matrix" is often used, which specifies the mapping between steps and KCs. This mapping is essentially what is meant by a KC model in this context. A mapping created dynamically by an ITS can be more flexible than a Q matrix, in the sense that the KCs in a step can depend on the particular solution path the student takes through the problem (Aleven, 2010).

AFM estimates the log odds of the probability that the student will get the step right as the sum of three key quantities: the proficiency of the student (i.e., how good the student is in general, across all KCs, or θ_i), the ease of each KC involved in the step (β_k), and how much the student has learned on prior opportunities to practice each of these KCs ($\gamma_k T_{ik}$). Thus, the model captures effects of practice over time and takes into account that a student is learning, essential characteristics of learner modeling approaches in ITS. Further, the model assumes that practice with a given KC does not have any direct effect on other KCs, a fundamental assumption in KC modeling discussed above. This regression model also makes a number of simplifying assumptions. First, it assumes that all students learn the same KCs. Second, it assumes that a KC that is more difficult than other KCs is so for all students. That is, the model does not allow for the possibility that a particular KC is harder than other KCs for one student, but easier than other KCs for other students (i.e., the model has no student × KC interaction term). Also, the model assumes that the effect of past practice opportunities is measured accurately solely by the number of such opportunities, and that there is no additional information to be gained for example from how successful these past opportunities were. In spite of these assumptions, the model has the virtue of (relative) simplicity and works well in practice. Addressing some of these assumptions leads to more complex models, as has been explored in later work by various researchers (e.g., Gong, Beck & Heffernan, 2011).

In order to use the model to predict student performance, it is necessary to estimate the values for the parameters in the model, specifically, the proficiency of each student (θ_i), the ease of each KC (β_k), and the learning rate for each KC (γ_k), as shown in Figure 15-1. These parameters can be estimated through standard logistic regression modeling techniques, such as generalized linear modeling (in R, the glm function with family= binomial). With such parameter estimates in hand, it is possible to calculate the predictive accuracy of the model on held-out data (i.e., cross-validation). Alternatively, predictive fit indices such as Akaike information criterion (AIC) and Bayesian information criterion (BIC) can be used, especially when the focus is on comparing alternative KC models for the same data set.

How does a KC model's psychological reality relate to its predictive ability? The short answer is, if a KC model is incorrect in the sense that it does not capture the true KCs that students acquire, then the parameter estimates obtained through logistic regression process described above will be inaccurate and therefore lead to inaccurate predictions. To illustrate, consider a model that models KCs at too high a level of abstraction, meaning that it misses distinctions that students actually make. This model models as one KC what for students are two separate KCs. To make the example concrete, consider a model for geometry problem solving that uses one KC for steps that use the circle area formula. Let us assume, however, that in reality, using the circle area formula when the radius is given is different for students from using it when the area of the circle is given – going from area to radius involves a square root, a difficult mathematical operation. As a consequence, the β_k (ease of KC) estimate for the overly abstract KC in our model would be a rough average of the β_k's of the two true KCs, which may have different true β_k's. Similarly, the learning rate parameter (γ_k) for the overly abstract KC might also not accurately reflect those for the true KCs. Similar comments can be made for a model with overly specific KCs. In a model with overly specific KCs (i.e., a model that misses abstractions that students actually make), the practice opportunities are not assigned correctly to the true KCs, which will lead to inaccurate parameter estimates.

Automated Methods for KC Modeling Learning Factors Analysis (LFA)

While tools for data-driven KC model analysis and validation such as those provided by the DataShop are very helpful, a model author still needs to do a substantial amount of work to come up with model variations and test whether they have greater predictive accuracy than the model from which they were derived. In any realistic KC modeling effort, there may be many modeling decisions about which there is uncertainty. As a result, the space of models that might potentially do a better job than an initial model created by hand is often quite large. It would be difficult for an author to optimize a model by searching this large space by going through the manual (though tool-supported) model refinement process described in the previous section. Therefore, EDM research has started to produce automated methods for discovering and refining KC models.

LFA is one such method (Cen et al., 2006). It automates the process of hypothesizing alternative cognitive models and testing them against data. This method searches a space of KC models, given (1) an initial model, (2) information from which model variations can be generated, and (3) a data set with step-level problem-solving data from a group of students on relevant problems (a typical tutor log data set, in other words, such as those captured in the DataShop). The method does a best-first search, using AFM as described above to evaluate each model variation and using the AIC metric for relative model fit.

To use LFA for model refinement, an author must provide information that the LFA algorithm uses to derive new KCs, item b in the list above. To this end, an author needs to identify factors in the given problem set that are hypothesized to make a difference in how students view or approach problems and therefore in the KCs that they acquire when they work through these problems. For each of these hypothesized difficulty factors, the author also needs to specify in which problems that factor occurs, a so-called "P matrix." The LFA algorithm uses this information to split KCs into more specific KCs, based on the difficulty factors, resulting in a finer-grained model. The algorithm also is capable of merging two KCs under appropriate circumstances, resulting in a coarser-grained model. The new model (with the split and/or merged KCs) is evaluated using the AFM described above and it is retained in the best-first search process if the AIC criterion shows improved fit over the model from which it was derived. The LFA algorithm can be run, on request, on any DataShop dataset and has been run on tens of datasets.

As a hypothetical example, a KC model for geometry problem solving might have a KC for applying the isosceles triangle theorem to infer the measure of one of the two angles opposite the congruent sides, given the measure of the other such angle. (The two angles are congruent.) A model author might wonder whether it makes a difference whether the isosceles triangle is acute or obtuse – students in the target population may be more familiar with acute triangles and recognize them more readily as being isosceles. Similarly, an author might wonder whether the triangle's orientation in the diagram matters – the third, non-congruent side may be at the bottom (fir tree configuration) or at the top (ice cream cone) or one of the congruent sides may be at the bottom (lying on the side). Perhaps the fir tree configuration is more familiar and recognizable for students. Our author would therefore define two difficulty factors, one capturing the obtuse/acute distinction, the other the tree/cone/on-the-side distinction. These two factors together would yield 12 possible ways of refining the KC model for the isosceles triangle skill, all more fine-grained than the original model. The LFA search would test if any of the more fine-grained KC models better accounts for the student data that was observed, based on its AIC score. The example illustrates how the space of models can be quite large, which is a key reason why automated methods are useful.

In a study by Koedinger, McLaughlin, and Stamper (2012), the LFA method was applied to 11 DataShop data sets from a variety of Cognitive Tutors and an educational game in a variety of domains. Models generated by LFA were found to improve upon handmade models in all 11 cases, demonstrating the potential of the method for automated model generation. Among the improved models was one for problem solving with the geometry area formula. As described above, the method discovered that for circle area problems, it mattered whether the radius of the circle or the area of the circle was given, whereas for other formulas, it did not make a difference whether the formula was applied forwards or backwards – only the circle formula required use of the square root when applied backwards.

A limitation of LFA is that it needs an initial KC model and therefore does not obviate the need for manual KC modeling. Some attempt at indicating categories of problem (steps), as simple as topics or learning objectives, is necessary. More desirable is to use LFA as supplement to cognitive task analysis done in the early stages of tutor building. Further, LFA cannot compensate for limitations in the breadth of the problem-solving tasks used in a tutor. Task difficulty factors that have a large effect on student performance can be recognized with just one or a few problems involving them. For LFA to detect smaller effects, however, a larger number of problems with and without a difficulty factor are needed. This frequency and variety may or may not occur when a problem set has not been designed with these difficulty factors in mind. Finally, better results can be achieved if there is at least some randomness in the order in which problems are assigned to students, which is not always the case in tutor problem sets.

Use of Mixed-Initiative Machine Learning for KC Modeling

As a final tool for KC modeling, we look at a machine-learning approach to rule-based KC modeling. As mentioned, advantages of rule-based models over other types of KC models are that transfer predictions are made explicit in the conditions of the rules (i.e., in their IF part, given they are written in IF-THEN format) and that they deal well with problems that have a large solution space, as occurs for example in algebraic problem solving (Waalkens, Aleven & Taatgen, 2013). However, writing rule-based KC models requires a substantial amount of work and technical expertise (i.e., AI programming skill). The machine learning approach described in this section helps address this shortcoming.

Li, Matsuda, Cohen, and Koedinger (2011) developed an approach that automatically discovers student models using a state-of-art machine-learning agent, SimStudent. They show that the discovered model is of higher quality than human-generated models and demonstrate how the discovered model can be used to improve a tutoring system's instruction strategy. SimStudent learns a rule-based KC model, of the kind that CTAT authors can create by hand, by being tutored. An author (e.g., a subject matter expert without programming expertise) interacts with SimStudent by first presenting it with a problem using a CTAT interface (e.g., an equation to solve like 100–4x=40). SimStudent uses any productions it has learned so far to try to generate a first step in solving the problem. If none apply to the current state, SimStudent asks the author to demonstrate the next step. It then uses multiple inductive learning mechanisms to learn the IF-part (information retrieval and conditions of applicability) and THEN-part (a function sequence) of a production rule that can reproduce the demonstrated step and generalize to other similar steps. If SimStudent has a production that applies (its IF-part matches the current state), it displays the resulting action (by executing the THEN-part) in the CTAT interface to be checked by the author. If the author confirms the step, this action is added as a positive example of the production. If the author rejects the step, the action is added as a negative example and the productions IF-part will be refined (e.g., by adding a condition that stops the rule from firing in states like this one in the future). SimStudent tries again, perhaps grounding out in asking for a demonstration of the next step.

The author may (but need not) provide guidance to SimStudent about what steps to try to generalize across by providing the same skill label to these steps, similar to labeling steps in behavior graphs. Even when a label is provided for a set of steps, SimStudent may not find a single general production that works across all same-labeled steps. In those cases it creates multiple productions and these, then, are the key basis for theoretical discovery of a better cognitive model (these are an automated generation of the equivalent of a factor used to perform a split in LFA). For example, in Li et al. (2011), equation-solving steps involving dividing by a coefficient (e.g., 3x=12, 15=–5x, or –x=8) were all given the same label (meaning that the author hypothesized that these steps involve the same KC). However, SimStudent learned separate productions for steps where there is number before the x (e.g., 3x=12 or 15=–5x) vs. steps where there is just a "–" sign before the x (e.g., –x=8 or –13=–x). In other words, SimStudent, at its own initiative, refined the KC model proposed by the author, by splitting the author's hypothesized KC. The KC model created by SimStudent (it made other novel distinctions besides this one) was compared (using AFM) with the best existing hand-created model. It led to better prediction fit as measured both by AIC and root-mean-square error (RMSE) on the test set in cross-validation. In particular, the cognitive model SimStudent learned better captured that human students find the second category of problems, with no explicit numerical coefficient (e.g., –x=8 or –13=–x), to be much more difficult than the first category (60% vs. 30% error rate). This kind of cognitive model discovery via SimStudent has now been tested across multiple domains: algebra equation solving, fraction addition, chemistry stoichiometry problems, and English article choices.

In some cases (e.g., algebra equation solving), the SimStudent-discovered cognitive model beat the prior best hand model and the prior best LFA model. However, adding the SimStudent discovered factors as input to LFA has so far always produced a better model (e.g., in fraction addition) than SimStudent or LFA alone.

Discussion

In this chapter, we focused on a key challenge in the KC approach to learner modeling, namely, that of accurately determining what the KCs are that learners acquire in a given task domain. While up until now, manual approaches to KC modeling have been prevalent, data-driven approaches have progressed to the point that they can at least be a useful supplement and perhaps even a substitute. The ways in which manual models based on intuition fall short provide interesting feedback on our intuitions about learning. Perhaps the examples given in the chapter illustrate that even experienced KC modelers tend to be over-optimistic about transfer. Perhaps the exercise of refining handmade models through data-driven, automated methods will eventually lead to a better general understanding when transfer is and is not likely to occur. Interestingly, the educational impact of KC modeling approaches will reach well beyond ITS development, as lessons learned through KC modeling can be applied to the design of a wide range of instructional materials, not just advanced learning technologies and ITSs. For example, many or all of the examples of discoveries about student learning given in the chapter could be applied to textbook and (non-tutored) homework assignments.

Given that we have described a range of tools for KC modeling, each with pros and cons, it may be useful to reflect on how these tools might feature in a realistic tutor development scenario. In an ideal scenario, an author might first build a tutor based on manual KC modeling. For example, the author might use an ITS authoring tool that supports KC modeling as an integral part of tutor development, such as CTAT. The author might build an example-tracing tutor, hand-write a production rule cognitive model, or automatically develop a production rule cognitive model by tutoring SimStudent. Ideally, the author would not rely on theoretical cognitive task analysis only but also do a substantial empirical task analysis (e.g., think-alouds and difficulty factors assessments), so as to increase the quality of the initial KC model.

However, even with empirical cognitive task analysis up front, there is still room for improvement through data-driven discovery from search over a larger space of tasks and models. After deploying the initial version of the tutor in a real educational setting, an author might use the data-driven tools described in this chapter to analyze the tutor log data. Specifically, the author evaluates how effective the tutor is in helping students learn the targeted KCs and explores ways in which the KC model can be refined using DataShop tools, using its learning curves display, fit indices for the AFM predictions, and the DataShop's performance profiler. For example, the author might investigate the causes of ragged learning curves using the performance profiler and try out a few model extensions by hand to see if AFM fit improves. In a more systematic approach, the author may use LFA to find a better model. Having found a better KC model (i.e., one that better matches the data about student learning), the author updates the tutor to improve its effectiveness ("closing the loop"). We emphasize that although the data-driven approaches such as use of DataShop tools, LFA, or SimStudent are helpful, in our opinion, they do not do away with the need for a careful cognitive task analysis at least to interpret the resulting data, but better yet toward creating a good KC model for use in the first version of the tutor.

Implications for GIFT

In this final section, we discuss implications for GIFT and other ITS authoring tools. We started the chapter by identifying three key challenges related to KC approaches to learner modeling: identifying psychologically accurate KCs, modeling the knowledge of individual students, and using KC models to individualize instruction. To support a KC approach to learner modeling, an ITS authoring tool would ideally address all three challenges. We illustrated how CTAT, in combination with the DataShop, successfully addresses these three challenges.

How could new authoring tools, such as GIFT, address these same challenges? Could they borrow from the CTAT/DataShop approach? We see some low-hanging fruit in particular with respect to the first challenge (creation/discovery/refinement of KC models). A good start is integration with the DataShop (Koedinger et al., 2010; Stamper & Koedinger, 2011). All that is needed is that tutors log student-tutor interactions in DataShop format (see http://pslcdatashop.org/dtd/). This integration is a good first step even before the authoring tool supports any kind of KC model. Even in the absence of any KC model within the tutoring system itself, the DataShop can be used for offline KC modeling based on tutor log data, which can lead to recommendations for improving the tutoring system. DataShop logging requires that the student interactions are divided into steps, which can involve successful attempts, unsuccessful attempts, and hint requests. VanLehn (2006) illustrates how interactions from many different tutoring systems can naturally be thought of as involving steps.

In addition, behavior graphs may be useful in GIFT and other authoring tools. Mapping out the solution space for tutored problems (as behavior graphs do) is useful for purposes of theoretical task analysis, as illustrated in the current chapter, and is compatible with a range of different types KC models.

Beyond this low-hanging fruit, the main challenge we see in supporting a KC approach in an authoring tool such as GIFT is in representing a particular type of KC model. Ideally, the authoring tool also makes it considerably easier for an author to build KC models of the targeted type in a range of application domains. A key related challenge is to the use of the KC model in the system's inner loop to interpret and assess student behavior. The commitment to support a particular type of KC model influences many aspects of the authoring tool and should be made early on in the tool's development. Different authoring tools address this challenge in their own way. CTAT supports two types of KC models, rule-based models and annotated behavior graphs. ASPIRE supports the authoring of constraints as key KCs with various tools (Mitrovic, Martin, Suraweera, Zakharov, et al., 2009).

With respect to the second and third challenges (using a KC model to track individual students' knowledge growth and individualizing instruction), innovative ITS authoring tools such as GIFT may include mechanisms or algorithms that enable tutors built with the tools to track over time how well individual learners do in terms of a given KC model. Also, they may offer a range of different ways of adapting the instruction based on such models of individual learners (the third challenge). By supporting easy customizability with respect to a range of options, authoring tools can do much to advance research into the benefits of different ways of individualizing instruction, a forte of ITSs.

References

Aleven, V. (2010). Rule-Based cognitive modeling for intelligent tutoring systems. In R. Nkambou, J. Bourdeau & R. Mizoguchi (Eds.), *Studies in Computational Intelligence: Vol. 308. Advances in intelligent tutoring systems* (pp. 33-62). Berlin, Heidelberg: Springer. doi:10.1007/978-3-642-14363-2_3

Aleven, V., McLaren, B. M., Sewall, J. & Koedinger, K. R. (2006). The cognitive tutor authoring tools (CTAT): Preliminary evaluation of efficiency gains. In M. Ikeda, K. D. Ashley & T. W. Chan (Eds.), *Proceedings of the 8th International Conference on Intelligent Tutoring Systems (ITS 2006)* (pp. 61-70). Berlin, Heidelberg: Springer-Verlag. doi:10.1007/11774303_7

Aleven, V., McLaren, B. M., Sewall, J. & Koedinger, K. R. (2009). A new paradigm for intelligent tutoring systems: Example-Tracing tutors. *International Journal of Artificial Intelligence in Education*, *19*(2), 105-154.

Aleven, V., Roll, I. & Koedinger, K. R. (2012). Progress in assessment and tutoring of lifelong learning skills: An intelligent tutor agent that helps students become better help seekers. In P. J. Durlach & A. M. Lesgold (Eds.), *Adaptive technologies for training and education* (pp. 69-95). New York: Cambridge University Press.

Anderson, J. R. (1993). *Rules of the mind*. Hillsdale, NJ: Lawrence Erlbaum Associates.

Anderson, J. R., Corbett, A. T., Koedinger, K. R. & Pelletier, R. (1995). Cognitive tutors: Lessons learned. *The Journal of the Learning Sciences*, *4*(2), 167-207.

Arroyo, I., Beck, J., Woolf, B. P., Beal, C. R. & Schultz, K. (2000). Macroadapting animalwatch to gender and cognitive differnces with respect to hint interactivity and symbolism. In G. Gauthier, C. Frasson & K. VanLehn (Eds.), *Proceedings of the 5th International Conference on Intelligent Tutoring Systems* (ITS 2000) (pp. 574-583). Berlin: Springer Verlag.

Baker, R. S. J. d., Corbett, A. T. & Koedinger, K. R. (2007). The difficulty factors approach to the design of lessons in intelligent tutor curricula. *International Journal of Artificifial Intelligence and Education*, *17*(4), 341-369.

Baker, R. S. J. D., Corbett, A. T., Koedinger, K. R., Evenson, S., Roll, I., Wagner, A. Z., . . . Beck, J. E. (2006). Adapting to when students game an intelligent tutoring system. In *ITS'06: Proceedings of the 8th International Conference on Intelligent Tutoring Systems (ITS 2006)* (pp. 392-401). Berlin, Heidelberg: Springer-Verlag. doi:10.1007/11774303_39

Baker, R. S. J. D., Goldstein, A. B. & Heffernan, N. T. (2011). Detecting learning moment-by-moment. *International Journal of Artificial Intelligence in Education*, *21*(1-2), 5-25. doi:10.3233/JAI-2011-015

Bull, S. & Kay, J. (2007). Student models that invite the learner in: The SMILI: () Open learner modelling framework. *International Journal of Artificial Intelligence in Education*, *17*(2), 89 - 120. Retrieved from http://search.ebscohost.com/login.aspx?direct=true&db=psyh&AN=2008-01222-001&site=ehost-live

Cen, H., Koedinger, K. & Junker, B. (2006). Learning factors analysis--a general method for cognitive model evaluation and improvement. In M. Ikeda, K. D. Ashley & T. W. Chan (Eds.), *Proceedings of the 8th International Conference on Intelligent Tutoring Systems (ITS 2006)* (pp. 164-175). Berlin: Springer.

Cen, H., Koedinger, K. R. & Junker, B. (2007). Is over practice necessary?-improving learning efficiency with the cognitive tutor through educational data mining. In R. Luckin, K. R. Koedinger & J. Greer (Eds.), *Proceedings of the 13th International Conference on Artificial Intelligence in Education* (pp. 511-518). Amsterdam: IOS Press.

Conati, C. & Vanlehn, K. (2000). Toward computer-based support of meta-cognitive skills: A computational framework to coach self-explanation. *International Journal of Artificial Intelligence in Education*, *11*(4), 389-415. Retrieved from Google Scholar.

Conati, C., Gertner, A. & Vanlehn, K. (2002). Using bayesian networks to manage uncertainty in student modeling. *User Modeling and User-adapted Interaction*, *12*(4), 371-417.

Corbett, A., McLaughlin, M. & Scarpinatto, K. C. (2000). Modeling student knowledge: Cognitive tutors in high school and college. *User Modeling and User-Adapted Interaction*, *10*, 81-108.

Corbett, A. T. & Anderson, J. R. (1995). Knowledge tracing: Modeling the acquisition of procedural knowledge. *User Modeling and User-Adapted Interaction*, *4*(4), 253-278.

Falmagne, J. -C., Koppen, M., Villano, M., Doignon, J. -P. & Johannesen, L. (1990). Introduction to knowledge spaces: How to build, test and search them. *Psychological Review*, *97*(2), 201-224.

Y. Gong, J. E. Beck & N. T. Heffernan (2011). How to construct more accurate student models: Comparing and optimizing knowledge tracing and performance factor analysis. *International Journal of Artificial Intelligence in Education, 21*(1-2), 27–46.

Harrer, A., McLaren, B. M., Walker, E., Bollen, L. & Sewall, J. (2006). Creating cognitive tutors for collaborative learning: Steps toward realization. *User Modeling and User-Adapted Interaction*, *16*(3-4), 175-209. doi:10.1007/s11257-006-9007-4

Koedinger, K. R. & Aleven, V. (2007). Exploring the assistance dilemma in experiments with cognitive tutors. *Educational Psychology Review*, *19*(3), 239-264.

Koedinger, K. R., Aleven, V., Heffernan, N., McLaren, B. & Hockenberry, M. (2004). Opening the door to non-programmers: Authoring intelligent tutor behavior by demonstration. In J. C. Lester, R. M. Vicario & F. Paraguaçu (Eds.), *Proceedings of Seventh International Conference on Intelligent Tutoring Systems, ITS 2004* (pp. 162-174). Berlin: Springer.

Koedinger, K. R., Baker, R. S. J. d., Cunningham, K., Skogsholm, A., Leber, B. & Stamper, J. (2010). A data repository for the EDM community: The PSLC datashop. In S. Ventura, C. Romero, M. Pechenizkiy & R. S. J. d. Baker (Eds.), *Handbook of educational data mining*. Boca Raton, FL: CRC Press.

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. doi:10.1111/j.1551-6709.2012.01245.x

Koedinger, K.R. & McLaughlin, E.A. (2010). Seeing language learning inside the math: Cognitive analysis yields transfer. In S. Ohlsson & R. Catrambone (Eds.), *Proceedings of the 32nd Annual Conference of the Cognitive Science Society* (pp. 471-476). Austin, TX: Cognitive Science Society.

Koedinger, K. R., McLaughlin, E. A. & Stamper, J. C. (2012). Automated student model improvement. In K. Yacef, O. Zaïane, A. Hershkovitz, M. Yudelson & J. Stamper (Eds.), *Proceedings of the 5th International Conference on Educational Data Mining* (pp. 17-24). International Educational Data Mining Society, www.educationaldatamining.org.

Koedinger, K. R. & Nathan, M. J. (2004). The real story behind story problems: Effects of representations on quantitative reasoning. *The Journal of the Learning Sciences*, *13*(2), 129-164.

Koedinger, K.R., Stamper, J.C., McLaughlin, E.A. & Nixon, T. (in press). Using data-driven discovery of better student models to improve student learning. To appear in *Proceedings of the 16th International Conference on Artificial Intelligence in Education*.

Lee, J. I. & Brunskill, E. (2012). The impact on individualizing student models on necessary practice opportunities. In K.Yacef, O.R.Zaïane, A. Hershkovitz, M. Yudelson & J. C. Stamper (Eds.), *Proceedings of the 5th International Conference on Educational Data Mining,* (pp. 118–125). International Educational Data Mining Society, www.educationaldatamining.org.

Li, N., Matsuda, N., Cohen, W. W. & Koedinger, K. R. (2011). A machine learning approach for automatic student model discovery. In M. Pechenizkiy, T. Calders, C. Conati, S. Ventura, C. Romero & J. Stamper (Eds.), *Proceedings of the 4th International Conference on Educational Data Mining* (pp. 31-40). International Educational Data Mining Society, www.educationaldatamining.org.

Long, Y. & Aleven, V. (2013). Supporting students' self-regulated learning with an open learner model in a linear equation tutor. To appear in *the Proceedings of the 16th International Conference on Artificial Intelligence in Education (AIED 2013)*.

Lovett, M., Meyer, O. & Thille, C. (2008). JIME-The open learning initiative: Measuring the effectiveness of the OLI statistics course in accelerating student learning. *Journal of Interactive Media in Education*, *2008*(1).

Lovett, M. C. (1998). Cognitive task analysis in the service of intelligent tutoring system design: A case study in statistics. In B. P. Goettle, H. M. Halff, C. L. Redfield & V. J. Shute (Eds.), *Intelligent Tutoring Systems, Proceedings of the Fourth International Conference* (pp. 234-243). Berlin: Springer Verlag.

Mathan, S. A. & Koedinger, K. R. (2005). Fostering the intelligent novice: Learning from errors with metacognitive tutoring. *Educational Psychologist*, *40*(4), 257-265.

Mitrovic, A. & Martin, B. (2004). Evaluating adaptive problem selection. In P. M. E. de Bra & W. Nejdl (Eds.), *Proceedings of the Third International Conference on Adaptive Hypermedia and Adaptive Web-based Systems, AH 2004* (Vol. 3137, pp. 185-194). New York : Springer-Verlag .

Mitrovic, A. & Martin, B. (2007). Evaluating the effect of open student models on self-assessment. *International Journal of Artificial Intelligence in Education*, *17*(2), 121-144.

Mitrovic, A., Martin, B., Suraweera, P., Zakharov, K., Milik, N., Holland, J. & Mcguigan, N. (2009). ASPIRE: An authoring system and deployment environment for constraint-based tutors. *International Journal of Artificial Intelligence in Education, 19*(2), 155-188

Mitrovic, A., Mayo, M., Suraweera, P. & Martin, B. (2001). Constraint-based tutors: A success story. In L. Monostori, J. Váncza & M. Ali (Eds.), *Proceedings 14th International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems (IEA/AIE 2001)* (pp. 931-940). Berlin Heidelberg: Springer.

Muldner, K. & Conati, C. (2007). Evaluating a decision-theoretic approach to tailored example selection. In M. Veloso (Ed.), *Proceedings of the 20th International Joint Conference on Artificial Intelligence, IJCAI 2007* (pp. 483-488). San Francisco, CA: Morgan Kaufmann.

Murray, T., Blessing, S. & Ainsworth, S. (2003). *Authoring tools for advanced technology learning environments: Toward cost-effective adaptive, interactive and intelligent educational software*. Amsterdam: Kluwer Academic Publishers.

Newell, A. & Simon, H. A. (1972). *Human problem solving*. Englewood Cliffs, NJ: Prentice-Hall.

Roll, I., Aleven, V., McLaren, B. M. & Koedinger, K. R. (2011). Improving students' help-seeking skills using metacognitive feedback in an intelligent tutoring system. *Learning and Instruction*, *21*(2), 267-280.

Salden, R. J. C. M., Aleven, V. A., Renkl, A. & Schwonke, R. (2009). Worked examples and tutored problem solving: Redundant or synergistic forms of support? *Topics in Cognitive Science*, *1*(1), 203-213.

Spada, H. & McGaw, B. (1985). The assessment of learning effects with linear logistic test models. In S. E. Embretson (Ed.), *Test design: Developments in psychology and psychometrics* (pp. 169-194). Orlando, FL: Academic Press.

Stamper, J. C. & Koedinger, K. R. (2011). Human-machine student model discovery and improvement using datashop. In G. Biswas, S. Bull, J. Kay & T. Mitrovic (Eds.), *Proceedings of the 15th International Conference on Artificial Intelligence in Education (AIED 2011)* (pp. 353-360). Berlin, Heidelberg: Springer-Verlag.

VanLehn, K. (2006). The behavior of tutoring systems. *International Journal of Artificial Intelligence in Education*, *16*(3), 227-265.

VanLehn, K. (2011). The relative effectiveness of human tutoring, intelligent tutoring systems, and other tutoring systems. *Educational Psychologist*, *46*(4), 197-221.

VanLehn, K., Lynch, C., Schulze, K., Shapiro, J. A., Shelby, R., Taylor, L., . . . Wintersgill, M. (2005). The Andes physics tutoring system: Lessons learned. *International Journal of Artificial Intelligence in Education*, *15*(1), 147-204.

Waalkens, M., Aleven, V. & Taatgen, N. (2013). Does supporting multiple student strategies lead to greater learning and motivation? Investigating a source of complexity in the architecture of intelligent tutoring systems. *Computers & Education*, *60*(1), 159-171. doi:10.1016/j.compedu.2012.07.016

Wilson, M. & De Boeck, P. (2004). Descriptive and explanatory item response models. In P. De Boeck & M. Wilson, (Eds.) *Explanatory item response models: A generalized linear and nonlinear approach* (pp. 43-74). New York: Springer-Verlag.