# Example-tracing Tutors: Intelligent Tutor Development for Non-Programmers

Vincent Aleven, Bruce M. McLaren, Jonathan Sewall, Martin van Velsen, Octav Popescu, Sandra Demi, Michael Ringenberg, Kenneth R. Koedinger

Human-Computer Interaction Institute
Carnegie Mellon University
5000 Forbes Ave
Pittsburgh, PA 15213

Corresponding author:
Vincent Aleven
aleven@cs.cmu.edu
+1 (412) 268-5475
+1 (412) 268-1266 (fax)

**Keywords:** Intelligent tutoring systems, authoring tools, example-tracing tutors

## Abstract

In 2009, we reported on a new Intelligent Tutoring Systems (ITS) technology, example-tracing tutors, that can be built without programming using the Cognitive Tutor Authoring Tools (CTAT). Creating example-tracing tutors was shown to be 4-8 times as cost-effective as estimates for ITS development from the literature. Since 2009, CTAT and its associated learning management system, the *Tutorshop,* have been extended and have been used for both research and real-world instruction. As evidence that example-tracing tutors are an effective and mature ITS paradigm, CTAT-built tutors have been used by approximately 44,000 students and account for 40% of the data sets in *DataShop*, a large open repository for educational technology data sets. We review 18 example-tracing tutors built since 2009, which have been shown to be effective in helping students learn in real educational settings, often with large pre/post effect sizes. The fact that example-tracing tutors can only handle problems with no more than a moderately-branching solution space is sometimes, though often not, a practical impediment. CTAT and other ITS authoring tools illustrate that non-programmer approaches to building ITS are viable and useful and will likely play a key role in making ITS widespread.

## Introduction

ITS authoring tools aim at making the creation of ITS more efficient and easier to learn. Also, they often aim at lowering the skill level needed to build a tutor and sometimes even at providing design guidance for effective tutors. ITS authoring tools by now have a long history in ITS research. Important early work was presented in Murray's (1999) influential overview paper in IJAIED and in a book edited by Murray, Blessing, Ainsworth (2003). More recent work appeared in a special issue in IJAIED (Koedinger & Mitrovic, 2009) and a series of workshops organized by the Army Research Laboratory and the University of Memphis (Sottilare, Graesser, Hu, & Holden, 2013, 2014; 2015). There are many ITS authoring tools, each with its own underlying ITS technology, such as ASPIRE (Mitrovic et al., 2009), ASTUS (Paquette, Lebeau, Beaulieu, & Mayers, 2015), ASSISTments Builder (Razzaq et al., 2009), AutoTutor tools (Nye, Graesser, & Hu, 2014), CTAT (Aleven, McLaren, Sewall, & Koedinger, 2009; Koedinger, Aleven, Heffernan, McLaren, & Hockenberry, 2004), GIFT (Sottilare, 2012), SDK (Blessing, Gilbert, Ourada, & Ritter, 2009), SimStudent (MacLellan, Koedinger, & Matsuda, 2014; Matsuda, Cohen, & Koedinger, 2015), and xPST (Blessing, Devasani, & Gilbert, 2011; Blessing, Gilbert, Blankenship, & Sanghvi, 2009; Kodavali, Gilbert, & Blessing, 2010). These tools have made ITS development easier and more cost-effective. They have lowered the skill threshold for building tutors. Some have provided estimates of cost savings (Aleven et al., 2009; Razzaq et al., 2009). Some studies have empirically evaluated the authoring efficiency or other aspects of ITS authoring tools' effectiveness (Aleven, McLaren, Sewall, & Koedinger, 2006; Blessing & Gilbert, 2008; Devasani, Gilbert, & Blessing, 2012; Kodaganallur, Weitz, & Rosenthal, 2005; MacLellan et al., 2014; Paquette, Lebeau, & Mayers, 2010; Razzaq et al., 2009).

Although it is hard to generalize from the current crop of ITS authoring tools, one broad trend we see is that non-programmer approaches to tutor building appear to be winning the day. Given that ITS development has traditionally required specialized programming skill that has hampered widespread development and use of tutors, this trend is not surprising. Many tool sets, including ASPIRE (Mitrovic et al., 2009), ASSISTments (Razzaq et al., 2009), CTAT (Aleven et al., 2009), SimStudent (Matsuda et al., 2015), and xPST (Kodavali et al., 2010), do not require advanced programming or any programming at all. While it is clear why easy-to-use and easy-to-learn tools are popular, we do need to ask the question whether non-programmer ITS authoring tools are capable of capturing sophisticated tutoring behaviors that are effective in helping students learn in a wide range of domains.

We focus on a set of authoring tools, CTAT (which stands for Cognitive Tutor Authoring Tools). The project started in 2002 with the key goals of making Cognitive Tutors easier to develop and lowering the skill threshold for building these kinds of tutors. At the time, a substantial number of Cognitive Tutors had been built and were in a use on a regular basis in many schools in the United States. They had been shown to lead to substantial increases in students' learning (Anderson, Corbett, Koedinger, & Pelletier, 1995; Koedinger, Anderson, Hadley, & Mark, 1997). With a few exceptions, subsequent classroom studies have produced similar findings (Koedinger & Aleven, 2007; Ritter, Anderson, Koedinger, & Corbett, 2007), including one conducted by a third party with

17,000 students in 150 schools in which it was found that a year of Cognitive Tutor Algebra led to nearly a doubling of student learning, compared to a "business as usual" control condition (Pane, Griffin, McCaffrey, & Karam, 2013). Although our initial goal was to facilitate development of Cognitive Tutors (Koedinger et al., 2004), along the way the project took an unexpected turn, described in our 2009 article in IJAIED entitled "A new paradigm for intelligent tutoring systems: Example-tracing tutors" (Aleven et al., 2009). We created a novel non-programmer approach to tutor authoring. We called the new tutor type *example-tracing tutors*, by analogy to model-tracing tutors (Anderson et al., 1995). Whereas model-tracing tutors use a generalized rule-based cognitive model to interpret student behavior, example-tracing tutors use generalized examples of problem-solving behavior. This choice sets them apart from many other ITS, as it is more common to use representations of general domain knowledge, such as rules (Anderson et al., 1995; Koedinger & Corbett, 2006) or constraints (Mitrovic & Ohlsson, 1999). No programming is needed and – as we argued in our 2009 IJAIED and continue to maintain today – sophisticated tutoring behaviors can be authored for a broad range of task domains. Further, we presented evidence that even in relatively small tutor-building projects, building example-tracing tutors can be 4-8 times as cost-effective as estimates from tutor building projects reported in the literature. Across projects, we saw roughly a 1:50 to 1:100 ratio of instructional to development time, compared to earlier estimates of 1:200 to 1:300. Additional cost savings come from the fact that tutors can be built without employing expensive programmers. A similar study with the ASSISTments Builder, another non-programmer ITS authoring tool, demonstrated development ratios of 1:28 to 1:40 (Razzaq et al., 2009). (These estimates are not directly comparable across the two tools sets, CTAT and ASSISTments, because the tools support different types of tutors and because the CTAT study estimated the overall tutor development time, whereas the ASSISTments study measured only the time spent with the tools, which leaves out certain development activities.) These studies show that ITS authoring tools for non-programmers can make authoring dramatically more efficient and cost-effective.

In the current article, we first give an overview of example-tracing tutors and of new extensions added to CTAT since 2009. We then revise and bolster the argument first presented in our 2009 IJAIED article that example-tracing tutors are ITS, along the way highlighting the many ways in which they are "adaptive" to learner needs and differences. To illustrate the scope of systems built with CTAT, we present 18 examples of example-tracing tutors, all of which have been used in real educational settings. (Screenshots of these tutors are shown in the Appendix.) We hope that together these arguments present a convincing case that example-tracing tutors are a useful and effective ITS paradigm and, more generally, that non-programmer authoring tools can be used to develop genuinely sophisticated tutors. We believe good authoring tools can make a dramatic difference in making ITS technology widespread.

## Overview of example-tracing tutors and CTAT
CTAT supports two tutor paradigms: example-tracing tutors (Aleven, Sewall, McLaren, & Koedinger, 2006; Aleven et al., 2009; Koedinger et al., 2004) and rule-based Cognitive Tutors (Aleven, 2010; Aleven, McLaren, et al., 2006). In this article, we focus on example-tracing tutors, as they were the focus of our IJAIED 2009 article. Like many

other ITS, this type of tutor provides step-by-step guidance during practice on recurrent problem types. The vast majority of tutors built with CTAT have been example-tracing tutors, because they are easier to author and debug than rule-based Cognitive Tutors. Using CTAT, example-tracing tutors can be built and deployed entirely without programming. Table 1 gives an overview of the functionality supported by CTAT and its associated learning management system, the *Tutorshop*. To give a sense for the scope of this project, we started building CTAT in 2002. Over the years, at least 54 people have contributed to CTAT and the Tutorshop; these two systems together represent approximately 60 man years' worth of work, the large majority of it by professional software engineering staff. Today, the CTAT/Tutorshop code base comprises 750,000 lines of code (including white lines and comments).
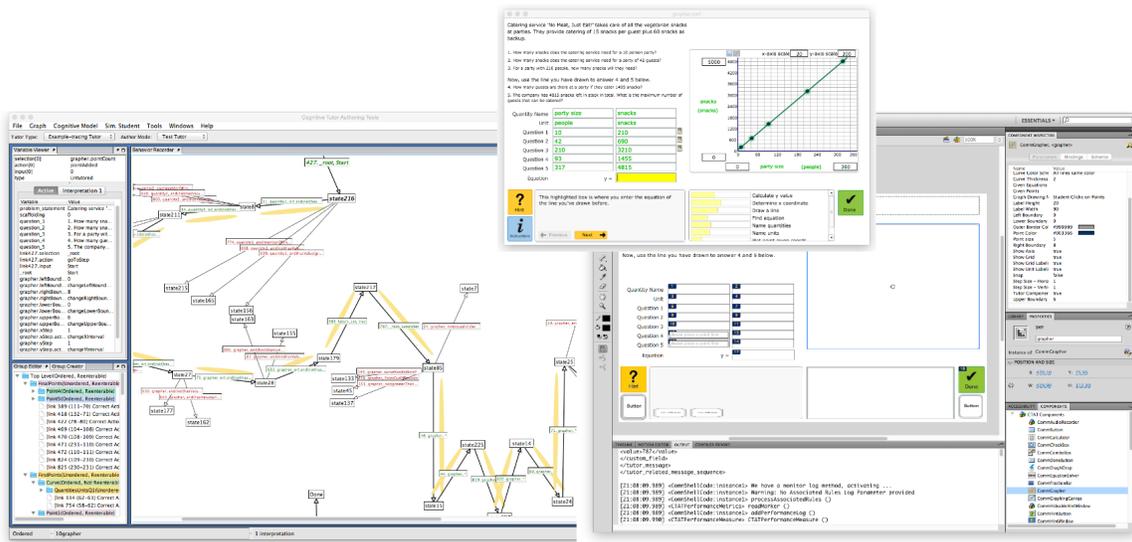


**Figure 1:** Authoring an example-tracing tutor with CTAT

## Authoring process

The process of authoring an example-tracing tutor has six key parts, often carried out in iterative cycles (and not necessarily in sequence). First, the author or authoring team investigates student thinking and learning in the given task domain, using cognitive task analysis (Baker, Corbett, & Koedinger, 2007; Clark, Feldon, van Merriënboer, Yates, & Early, 2008; Lovett, 1998) and (e.g., in the case of tutor redesign) educational data mining methods. Second, informed by the results of the first step, the author designs and creates one or more tutor interfaces. These interfaces tend to be specific to the targeted problem types; they break down complex problem solving into steps. Using CTAT, tutor interfaces can be built through drag-and-drop techniques within an existing interface builder, such as the Flash IDE (Figure 1, right). Other options for the tutor front end are supported as well (see Table 1).

Third, the author creates generalized examples needed by the tutor. She demonstrates problem-solving step on the interface (Figure 1, middle), which CTAT's Behavior

4

**Table 1:** Overview of CTAT/Tutorshop functionality; all functions are supported without programming, except where otherwise indicated

**Front-end options (tutor interface)**
- Java (drag-and-drop interface building)
- Flash/Actionscript (drag-and-drop interface building)
- HTML5/CSS/Javascript (currently requires coding)
- External problem-solving environment or simulator (needs to be integrated using custom programming)

**Inner loop (step loop); within-problem guidance by the example-tracing algorithm**
- Minimal feedback on steps - classified as correct, incorrect, or suboptimal
  - Immediate feedback
  - On-demand feedback
  - No feedback (quiz mode)
- Error-specific feedback
- Success messages on correct steps
- Student-requested hints on the next step (different policies for selecting hints are supported)
- Assessment of knowledge (i.e., maintaining a student model that captures probabilities of mastery of the knowledge components (KCs) targeted in the instruction, updated by Bayesian Knowledge Tracing)
- Skill meter (i.e., an open learner model showing mastery of KCs)
- Multiple solution paths within a problem (major and minor strategy/notational variations)
- Dynamic interfaces (i.e., interfaces that can change, under control of the tutor, as a function of the problem state)
- Collaborative tutors for synchronous, networked collaborative problem solving, with roles and embedded collaboration scripts tied to the problem state
- Backward fading of worked examples
- Input substitution (tutor can replace correct student input with canonical/evaluated form, corrections, etc.)

**Outer loop (task loop); problem selection by the Tutorshop**
- Student picks problem (custom versions exist, not part of the standard installation)
- Fixed problem sequence
- Adaptive problem selection (cognitive mastery based on Bayesian Knowledge Tracing)
- Random problem sequence

**Delivery options**
- Standalone application (Java only)
- Embedded in web pages
- Tutorshop (dedicated ITS-oriented learning managements system)
- Integration in external LMS based on e-learning standards (SCORM, LTI)
- Custom integration in external LMS (OLI, edX by XBlock integration)

**Support for research**
- Datashop logging of student-tutor interactions
- Support for multiple experimental conditions
- Replay of logged interactions

Recorder tool records the in a *behavior graph* (Figure 1, left); the links in the graph represent individual problem-solving steps. During tutoring, the tutor evaluates student actions by comparing them against the graph. The graph is therefore expected to capture *all* solution strategies that are reasonable within the given interface, so that the tutor can recognize all of them as correct student behavior. Different ways of solving the problem are captured as different paths in the graph. Once relevant paths have been recorded, the author *generalizes* the behavior graph, to indicate the range of student behavior that the graph stands for, beyond just literally the paths recorded in the graph with steps in the same order as demonstrated. For example, an author can provide *input matchers* that specify a range of values or notational variations to be accepted for a given step. She also can attach *formulas* (akin to Excel formulas) that specify how steps depend on each other or can define ordered or unordered groups of steps, nested if needed, to indicate what step orders are acceptable. Solution paths that involve the same steps but different values can be captured with a single path with attached formulas, as a way of keeping the graph manageable.

Fourth, an author *annotates* the graph. She can attach hints to links in the graph, which recommend the action on the given link as an appropriate next action to take; hints also typically explain why that action is a good next action, in terms of the domain's problem-solving principles. During tutoring, these hints will be displayed at the student's request. To make the tutor proactively display feedback in response to specific errors, an author can insert *incorrect action links* into the graph, each with a feedback message specific to the given error. To support assessment of knowledge through Bayesian Knowledge Tracing (Corbett & Anderson, 1995), an author may attach knowledge component (KC) labels to the links in the graph. Knowledge components are the smallest units into which the knowledge to be learned can be decomposed (Aleven & Koedinger, 2013; Koedinger, Corbett, & Perfetti, 2012; Koedinger et al., 2010).

Fifth, to make it easier to create multiple practice problems having the same behavior, CTAT supports a template-based feature called "Mass Production." To use it, an author first turns a behavior graph into a template by replacing problem-specific values with variables. She can then define many problems by specifying their values for the template variables in a spreadsheet. A final merge step automatically substitutes the spreadsheet values into the template's variables and generates a behavior graph for each problem.

Finally, when the interface and behavior graph are ready for initial testing with students, the author uploads the tutor to a deployment environment, for example, the *Tutorshop*, described below. After one or more rounds of pilot testing and revision, preferably with students from the target population, and often on a server reserved for testing, the author performs the final edits and uploads the tutor to the final production environment.

**Inner loop: Using a behavior graph to provide tutoring**
At student run time, CTAT's built-in example-tracing algorithm (Aleven, McLaren, & Sewall, 2009) takes care of the tutor's inner loop (a.k.a. step loop; VanLehn, 2006; this issue). That is, it provides step-level guidance within each problem, including step-level correctness feedback, on-demand hints, error-specific feedback messages, and assessment

of knowledge based on Bayesian Knowledge tracing (Corbett & Anderson, 1995). Essentially, the example tracer algorithm evaluates whether the student's solution steps conform to one or more solution paths implied by the generalized graph. CTAT now provides a number of feedback policies, including on-demand feedback and no-feedback (for implementing online tests). At the student's request, the algorithm provides hints by looking for an appropriate choice of next step within the graph, taking into account the solution path(s) the student may be on. CTAT also supports a number of hint selection policies, which give an author some control over how the tutor decides which of multiple possible next steps to recommend in a hint. It provides an error feedback message when the student's action matches one of the incorrect action links in the graph. Finally, CTAT supports input substitution, meaning that the tutor can transform student input in an author-specified way and echoes the transformed version back to the student; this feature is useful for arithmetic calculations, spell checking, etc.

Beyond these basic behaviors CTAT's example-tracing technology supports a range of additional inner loop behaviors (see Table 1). Whereas CTAT originally supported tutors for individual learning only, we recently added support for authoring collaborative example-tracing tutors (Olsen, Belenky, Aleven, Rummel, et al., 2014), described in more detail below. We also added support for gradually fading worked examples, based on research that demonstrated the effectiveness of this way of transitioning from examples to problem solving (e.g., Atkinson, Renkl, & Merrill, 2003; Renkl, Atkinson, & Grosse, 2003; Salden, Aleven, Schwonke, & Renkl, 2010). Further, an author can create dynamic interfaces whose components change depending on the problem state or in response to specific student actions. An author can do so by inserting links in the behavior graph that represent *tutor-performed actions* such as updating, showing, or hiding interface components. This basic functionality can be used to craft a wide range of interface behaviors, such as revealing steps or groups of steps gradually as the student pogresses through the problem, creating dynamically linked representations, or breaking down the problems into steps when the student makes an error (cf. Heffernan & Heffernan, 2014).

**Learner modeling**
As mentioned, CTAT and the Tutorshop support a learner model that records probabilities that the given student masters the KCs targeted in the given problem set. As author defines a KC model for an example-tracing tutor by annotating the links of a behavior graph with KC labels. The learner model is updated (in the inner loop) through Bayesian Knowledge Tracing (Corbett & Anderson, 1995) and is used (as one of the outer loop options) for adaptive problem selection and cognitive mastery. As a form of open learner modeling, CTAT provides an interface component that displays the learner model in the form of a skill meter (see Figure 1, middle, in the bottom right corner of the tutor interface). An author can define an initial KC model based on cognitive task analysis and refine it later based on analysis of tutor log data, for example using tools for learning curve analysis made available in the DataShop (Koedinger et al., 2010). Refined KC models can lead to more effective or efficient instruction (for an overview, see Aleven & Koedinger, 2013). As a way of generalizing CTAT, we are looking into the possibility of supporting the plugging in of custom student models.

**Outer loop**
The Tutorshop supports a number of outer loop (a.k.a. task loop) options. First, it supports cognitive mastery learning based on Bayesian Knowledge Tracing, a form of personalized problem selection that has proven to be very effective (Corbett, McLaughlin, & Scarpinatto, 2000). It also supports randomized problem selection, which is sometimes useful for research purposes. For example, when problems are presented in random order (randomized separately for each student), then in offline analyses of tutor log data, the difficulty of problem steps can be assessed without being confounded by the order in which the problems were presented. One of our goals for the near future is to extend CTAT so it can support the easy plugging in of custom problem selection policies and mastery criteria.

**Tutor front end**
CTAT currently supports three options for the tutor front-end (i.e., the interface in which the students solve problems, with the tutor's guidance), namely, Java, Flash/ActionScript, and HTML5/Javascript. For the first two options, drag-and-drop interface building is supported. For the last of these, HTML5/Javascript, interfaces need to be created by writing code, although we hope to support drag-and-drop interface building in the future. A great amount of our effort in recent years went into CTAT's front-end technology, mainly to keep up with changes in web-based technologies. Since 2009, we have revised or re-implemented our interface technologies three times. First, we updated the look and feel of the tutor interfaces in our ActionScript 2 code base, in line with the design aesthetic for *Mathtutor* (Aleven, McLaren, & Sewall, 2009). Second, we moved to ActionScript 3, a substantial reimplementation effort, as ActionScript 3 provides important enhanced functionality but is not backwards compatible. Finally, we completed an HTML5/JavaScript implementation, which enables us to offer a truly cross-platform ITS approach. We expect this to become the go-to option for CTAT tutors, with Flash not supported on all web client platforms – and on the decline. Keeping up with changing interface technology would not have been possible without a factored architecture maintaining strict tool/tutor separation (Ritter & Koedinger, 1996), described below.

**Delivery and deployment**
Much of our effort since 2009 has also gone into supporting ways to deliver CTAT tutors in a variety of e-learning platforms, including MOOCs. We view this capability as an important goal for ITS authoring tools and for making ITS technology widespread (Aleven et al., 2015b).

So far, the *Tutorshop* has been the go-to platform for deploying CTAT tutors. The *Tutorshop* is a web-based content management and learning management system we created geared specifically for tutor use in classrooms and other settings. It is implemented in Ruby on Rails. *Tutorshop*'s learning management facilities include management of class lists, student and teacher accounts, assignments, and a wide range of reports for students and teachers regarding student progress and learning. *Tutorshop* has

been used in many research projects. Also, it functions as the backbone of the *Mathtutor* website (Aleven, McLaren, & Sewall, 2009) as well as the *Genetics Tutor* (Corbett, Kaufmann, MacLaren, Wagner, & Jones, 2010). We currently host the *Tutorshop* as a service to the research community. For large-scale studies, we sometimes run *Tutorshop* in the Amazon cloud (AWS). Although *Tutorshop* has not been used to deliver non-CTAT tutors, this would be a useful way in which the CTAT architecture could be generalized. We have also made it possible to embed example-tracing tutors in MOOC and e-learning platforms that adhere to the SCORM and LTI e-learning interoperability standards, as many do. To this end, the CTAT/Tutorshop platform now implements the provider/content side of SCORM and LTI. As evidence of this capability, we have embedded CTAT tutors in Moodle (Rice, 2011) and in two edX MOOCs (Aleven et al., 2015b). We have also achieved custom integration with edX through edX's XBlock API and with the Open Learning Initiative (OLI)'s learning management system (http://oli.cmu.edu).

As part of our effort to support tutoring at scale, we have moved the example-tracing tutor engine (which takes care of the tutor's inner loop) from the server to the client, a change of direction since our 2009 IJAIED paper. Although there are a number of advantages to having the tutor engine on the server, discussed in the 2009 paper, doing so complicates the embedding of tutors in external e-learning platforms or learning management systems (LMSs). This is so because with a server-side tutor engine, a tutor is not a fully self-contained learning object. Further, with very large numbers of users, a server-based tutor engine could incur severe server load. Therefore, we now support a variety of options for running the tutor engine on the client, including a Java Web Start option, a Java applet option, and, most recently, a JavaScript version of the example tracer (Aleven et al., 2015b). We expect the latter to become the go-to option for example-tracing tutors in a variety of deployment options.

We see this work as an encouraging first step towards tutoring at scale (e.g., in MOOCs; Aleven et al., 2015b; Cook, Kay, & Kummerfeld, 2015; Kay, Reimann, Diebold, & Kummerfeld, 2013), although work remains to be done. For example, with the current versions of SCORM or LTI, the rich analytics produced by the tutor cannot easily be sent to a learning management system. This information is therefore not displayed in the grade book of the online course or integrated in existing student or instructor dashboards, a lost opportunity to take advantage of some of the advanced capabilities that ITS have to offer. Fortunately, newer versions of SCORM and LTI are moving toward richer data exchange between a tutor and the LMS, so this limitation may be addressed in the near future. Also, work remains to be done to make CTAT's model available within a MOOC, as well as adaptive task selection.


**Support for research**
CTAT tutors, typically running out of the *Tutorshop*, have been used in many dozens of scientific experiments to investigate questions regarding how tutors can most effectively support learning or other desirable educational outcomes. CTAT and *Tutorshop* offer some functionality that facilitates the use of tutors in such research. First, CTAT is fully compatible with *DataShop* (Koedinger et al., 2010)*,* a large, open repository for

educational technology data sets that supports offline analysis of tutor log data. All CTAT tutors log in *DataShop* format, without any additional effort from authors. *DataShop* supports many analyses geared towards data-driven refinement of the KC models underlying tutors (see e.g., Aleven & Koedinger, 2013). As of July, 2015, *DataShop* contains 290 data sets generated by CTAT tutors, approximately 40% of the data sets in *DataShop,* with roughly 48,000,000 transactions by a total of 44,000 students, working for a total of 62,000 hours. Second, *Tutorshop* supports assigning students to experimental conditions (although not automatically at this point in time) and recording experimental conditions in the log data. Finally, we are completing a Log Replayer capability, which will make it possible to replay logs through the tutor that generated them or through an extended version of that tutor that writes new information in the logs. This replay will support new analyses of log data not originally foreseen by the creators of the given tutor, as seems to be commonplace in educational data mining (e.g., Aleven, McLaren, Roll, & Koedinger, 2006; Harpstead, MacLellan, Aleven, & Myers, 2015).

**Architecture**

CTAT is conceived not just as a tool, but as a factored architecture for tutoring, with well-defined components and interfaces between those components (Aleven, McLaren, & Sewall, 2009; Aleven et al., 2009; 2015a; 2015b). In particular, CTAT and the *Tutorshop* enforce a strict separation between "tool" and "tutor." Here "tool" means the problem-solving interface or environment in which the student solves problems; "tutor" means the tutor backend, both the inner and outer loop (Koedinger, Suthers, & Forbus, 1999; Ritter & Koedinger, 1996). In CTAT, the tool and tutor communicate through a message protocol that derives from the work of Ritter and Koedinger (1996) (http://ctat.pact.cs.cmu.edu/index.php?id=tool-tutor). This aspect of the factored architecture has been very valuable. It has made it possible to mix-and-match options for the tutor engine (example-tracing or model-tracing tutor) with options for the tutor interface (Java, Flash/ActionScript, or HTML5/Javascript) or problem-solving environment (e.g., simulators). Also, it has made it possible, critically, to keep up with interface and web technology changes and has helped make it easier to extend the front-end technology with new interface components. Finally, the tool/tutor separation has made it easier to integrate tutors in a wide range of environments, as discussed above.

A second aspect of modularity in the CTAT/Tutorshop architecture is the separation between inner loop (within-problem tutor guidance) and outer loop (between-problem tutor guidance). The inner loop and outer loop communicate strictly through the student model (Aleven et al., 2015a). That is, at the beginning of each problem, the outer loop (currently, running server-side in the Tutorshop) passes the learner model to the inner loop. As the student works on the problem, the inner loop updates the learner model (displaying it as a skill meter, if the author so chooses). At the end of the problem, it passes the updated student model back to the outer loop, so it can be used for adaptive problem selection or stored in the database, if the student decided to log out. This separation facilitates plugging in different options for student modeling and task selection, an area where we are just beginning to gain experience.

We see a number of ways in which the CTAT architecture could be generalized further, so it can be more versatile and support greater interoperability. For example, the Behavior Recorder has a range of functionality that supports cognitive task analysis and tutor testing in a way that is not specific to example-tracing tutors. A generalized behavior recorder API would make this tool available for use with other tutor types (Aleven et al., 2015a). Likewise, the Tutorshop could be useful for other tutors. Also, it would be useful to make it possible to use different tutor engines with the CTAT front end, which to a large degree this is already facilitated by the tool/tutor separation. A promising direction is further to make it possible to plug in different student models, different algorithms for updating the student model, and different policies for task selection.  Finally, additional extensions for supporting research (e.g., A/B testing) would be useful as well.

**ITS authoring tool development philosophy**
In developing CTAT we took, and continue to take, a use-driven design approach. We credit this approach with making CTAT useful and usable. The essence of this approach is that we have made it a high priority to promote and support use of CTAT by others, to learn from users' experiences, and to make sure that what we learned helped shape the tools. We have regularly solicited feature requests and feedback from users. When planning for new releases of CTAT, we invariably prioritized features based on the question: "Who is going to use it?" If we could not identify specific users, the feature would not make it into the release. We provide online documentation and tutorials (http://ctat.pact.cmu.edu). Further, we have made efforts to build a user community, by setting up an online user forum (http://groups.google.com/groups/ctat-users) and by holding yearly summer schools where people can learn about ITS with hands-on work in CTAT. We have also used the tools extensively ourselves to build tutors used in classrooms (Aleven, McLaren, & Sewall, 2009; Forlizzi et al, 2014; Long & Aleven, 2013a; 2013b; McLaren et al., 2012; McLaren, DeLeeuw, & Mayer, 2011a, 2011b; McLaren, Lim, & Koedinger, 2008; McLaren, van Gog, Ganoe, Yaron, & Karabinos, 2014; 2015; McLaren, van Gog, Ganoe, Karabinos, & Yaron, 2016; Olsen, Belenky, Aleven, & Rummel, 2014; Olsen, Belenky, Aleven, Rummel, et al., 2014; Rau, Aleven, & Rummel, 2013; Rau, Aleven, Rummel, & Pardos, 2014; Stampfer & Koedinger, 2013; Waalkens, Aleven, & Taatgen, 2013; Wylie, Sheng, Mitamura, & Koedinger, 2011). "Eating our own dog food" helped spot opportunities for improvement and has driven development of a number of new features. We have always kept the cost-effectiveness of authoring in mind. Before deciding to implement a new feature, we typically ask: How often will this feature be used and how much time will it save authors? Further, to support cost effective authoring, CTAT takes advantage of existing tools, including Flash and Eclipse Window Builder for building interfaces and Microsoft Excel in CTAT's Mass Production process, described above.

**Should example-tracing tutors be considered ITS?**
In our 2009 paper, we argued that example-tracing tutors should be viewed as ITS. This argument was based on VanLehn's (2006; 2011) criterion that what distinguishes ITSs from other forms of computer tutors is that they have an inner loop (i.e., provide step-

level guidance to learners, rather than feedback only at the end of each problem).[1] We update and bolster this argument for two reasons: First, a recent chapter by Pavlik, Brawner, Olney, and Mitrovic (2013) questioned whether example-tracing tutors should be viewed as intelligent tutors, on the grounds that they might be similar to "programmed instruction." Second, although VanLehn's (2006) criterion has much going for it, we now more clearly recognize the limitations of this criterion. The issue is important, because how we position our systems vis-à-vis other kinds of learning technologies may influence public perception, acceptance, and eventual widespread adoption of ITS.

Although experts do not agree about how to define ITS (Woolf, 2009, p. 21), the crux may be how *adaptive* to student needs and student differences (e.g., in knowledge or motivation) the tutoring system is in ways that enhance student learning, motivation, or other desirable outcomes.  In defining adaptivity, we believe it is appropriate to look at the system's *behavior* and the effect that the system has on the student experience, in line with how Newell and Simon (1976) view intelligence:

> "By 'general intelligent action' we wish to indicate … that in any real situation behavior appropriate to the ends of the system and adaptive to the demands of the environment can occur … "

What we like about VanLehn's (2006; 2011) criterion (i.e., that the system has an inner loop) is that it emphasizes adaptive behavior as a hallmark of intelligence, in line with Newell and Simon (1976). Also, it aligns with key empirical evidence, namely, that systems with an inner loop tend to have a stronger positive effect on student learning than systems without (VanLehn, 2011). On the other hand, this criterion is not without its shortcomings. Step-based guidance may not be very adaptive if the tutor can recognize only one particular set of steps for each problem. Also, certain desirable forms of adaptivity may not easily be viewed as step-level support, such as reacting to student affect or adaptively selecting problems in the system's outer loop. Furthermore, a number of systems that have a legitimate claim to being adaptive and intelligent do not have a very elaborate inner loop, for example, ASSISTments (Heffernan & Heffernan, 2014) and Wayang Outpost/Mathsprings (Arroyo et al., 2014). Instead they have other features that warrant viewing them as ITS, such as being designed with a fundamental and sound understanding of student learning and the specific difficulties that students face in the given task domain, or that their outer loop is adaptive to student metacognition and affect (Arroyo et al., 2014).

Therefore, we offer an alternative definition (Aleven, 2015; Aleven, Beal, & Graesser, 2013; Aleven, McLaughlin, Glenn, & Koedinger, forthcoming):

---

[1] In a more recent framing, VanLehn (this issue) has generalized his conception of the inner and outer loop, calling them step loop and task loop, respectively. VanLehn now views the inner and outer loop of an ITS as instances of so-called "regulative loops" that repeatedly compare a student's performance to a gold standard. This generalization allows for including, for instance, certain types of computer-supported collaborative learning (CSCL) and other forms of intelligent learning support.  It does not affect our argument made here.

A learning environment is adaptive to the degree that:
a. Its design is grounded in a thorough empirical understanding of learners in the given task domain;
b. it takes into account, in its pedagogical decision making, how individual learners measure up along different psychological dimensions; and
c. it is appropriately interactive and responsive to learner actions.

Specific to the concerns of the field of AIED, the first part of this definition emphasizes (implicitly) the use of cognitive task analysis and data mining to guide tutor design and cyclical improvement of tutors (Aleven, 2010; Aleven & Koedinger, 2013; Baker et al., 2007; Clark et al., 2008; Lovett, 1998). The second part emphasizes adaptive individualization across a host of learner variables both in the inner (or step) loop and in the outer (or task) loop, consistent with Woolf's (2009) emphasis on having a student model and using it to adapt instruction. The third part of the definition emphasizes interactivity, consistent with VanLehn's inner loop or step loop.

Example-tracing tutor technology supports the building of tutors that exhibit all three factors that make up this definition. Regarding the first factor, example-tracing tutors vary in the depth of cognitive task analysis and other forms of data analysis that underlies their design. While the degree of cognitive task analysis depends ultimately on the efforts and procedures of the designers and developers, we note that CTAT's Behavior Recorder tool is particularly suited to support this activity, for it permits designers to model and visualize a solution space and to rapidly prototype different tutor behaviors (Aleven et al., 2015a). Furthermore, example-tracing tutors are an offshoot of Cognitive Tutors, which have always been grounded in cognitive task analysis and cognitive modeling. Therefore, example-tracing tutors can meet the first requirement. They also meet the second factor: As mentioned, example-tracing tutors and the *Tutorshop* support individualized problem selection based on Bayesian Knowledge Tracing (Corbett & Anderson, 1995; Anderson, Conrad, & Corbett, 1989; Corbett et al., 2000; VanLehn, 2011).

To see how example-tracing tutors address the third factor, we briefly review how they can be adaptive in their inner loop. Example-tracing tutors support basic inner loop (or step loop) functionality (VanLehn, 2006) with next-step hints and correctness feedback on steps and error feedback messages for common errors. Step-level feedback is strongly supported in the empirical ITS and education literature as enhancing student learning (Arroyo, Beck, Woolf, Beal, & Schultz, 2000; Kleij, Feskens, & Eggen, 2015; McKendree, 1990; VanLehn, 2011). Beyond basic inner loop functionality, example-tracing tutors can follow students within a problem with respect to multiple strategies, regardless of which strategy the student decides to follow. They do so by virtue of having multiple paths in their behavior graph (Waalkens et al., 2013) or by virtue of using formulas to capture commonalities among solution paths. By contrast, the simplest approaches for building interactive e-learning software (e.g., authoring questions one at a time, independent of each other) do not capture dependencies among steps (i.e., do not capture multiple paths). Capturing multiple solution paths also supports other adaptive behaviors. For example, a multi-path example-tracing tutor has the ability to respond differently to the same student input, depending on context, a hallmark of adaptive,

intelligent behavior. That is, whether a certain student step is marked correct depends on which path(s) the student is deemed to be on, based on the student's prior steps in the problem. Similarly, the tutor's hints can be sensitive to what solution path the student is on, with different hints being given for the same step, dependent on the student's prior path through the problem. An example-tracing tutor may change the hint it gives regarding what to do for a certain step, as more information about the student's solution becomes available, a form of flexible, adaptive behavior. It is also possible to create a tutor whose hints depend not only on the solution strategy the student is following, but also on the particular steps within the given strategy the student has completed already, as a way of making hints even more context sensitive. In sum, many adaptive behaviors are possible in an example-tracing tutor's inner loop. All behaviors described above emerge from the basic example-tracing algorithm.

In spite of the many adaptive behaviors that example-tracing tutors can support, Pavlik et al. (2013) question whether this type of tutor ought to be considered ITS. They view example-tutors as analogues to "programmed instruction" or "branched instruction" and state: "Perhaps these systems [i.e., example-tracing tutors] do not qualify as ITS, considering the student state space is small and that pedagogical options are few." The analogy between behavior graphs and branched instruction is problematic, however. Both types of graphs are used to provide instruction, but behavior graphs do not represent instructional sequences or pedagogical decisions, as do the branches of branched instruction. Rather, behavior graphs capture problem-solving processes and their variations (Newell & Simon, 1972). More broadly, we reject the notion that only tutoring systems capable of handling elaborate solution spaces should be viewed as adaptive or intelligent. (This would also rule out ASSISTments or Mathsprings, for example.) In principle, even within a small solution space, many pedagogical decisions are possible, so there is room for adaptivity to be effective. Perhaps more fundamentally, even a limited amount of adaptivity within a small solution space might be just the right amount in order to support an effective and efficient student experience. Pavlik et al. (2013) may look at the system's *internal structures* as a key criterion for intelligence or adaptivity, while overlooking the many *adaptive behaviors* that behavior graphs enable.

We do see a number of limitations in example-tracing tutors. First, example-tracing tutors currently do not have specific features for responding adaptively to student self-regulation, metacognition, or affect, although individual projects have added various forms of adaptive support for these aspects, including machine-learned detectors for help avoidance and for fast actions (Corbett, personal communication), simple support for self-explanation (Rau, Aleven, & Rummel, 2015; Wylie et al., 2011), self-assessment (Long & Aleven, 2013b; Roll, Aleven, McLaren, & Koedinger, 2011), and (with custom-programmed software modifications), shared student/system control over problem selection (Long & Aleven, 2013b). Also, one project (AdaptErrEx) added an external module for adaptively responding to student misconceptions in the outer loop through custom modifications (Goguadze, Sosnovsky, Isotani, & McLaren, 2011; McLaren et al., 2012). Other limitations discussed in our 2009 IJAIED paper (and still apropos) are that the example-tracing technology is not particularly efficient when it comes to authoring simpler interactive items (e.g., multiple-choice questions) and is not geared toward

supporting tutorial interactions in natural language or towards building systems with large domain ontologies or that draw on large stores of factual or conceptual knowledge, or to integrate (without programming) animated pedagogical agents. Additionally, example-tracing tutors have been proven only to a limited degree in open-ended domains, where each problem tends to have its own structure (but see Ogan, Aleven, & Jones, 2009).

## Evidence of the utility of example-tracing tutors

To consider whether example-tracing tutors are a generally useful ITS paradigm and to illustrate the level of maturity that CTAT has reached, we look at example-tracing tutors built with CTAT since our 2009 IJAIED paper. In that paper, we reported that example-tracing tutors had been used in 26 research studies in real educational settings. The domains for which example-tracing tutors had been built included mathematics (at the elementary, middle, and high-school levels), science (chemistry, genetics), engineering (thermodynamics), language learning (Chinese, French, and English as a Second Language), and learning of intercultural competence (references provided in the original paper). Since 2009, a substantial number of new tutors have been built with CTAT. We review 18 such tutors in this section, all of which were used in real educational settings, most of them in research studies. For this review we informally clustered these tutors based on key aspects of their pedagogy. We label these clusters as problem-solving tutors, tutors that use worked examples or erroneous examples, tutors that emphasize the use of interactive graphical representations, tutors that use pedagogical approaches other than standard tutored problem solving, and (in a singleton cluster), a tutor for language learning. As we discuss these clusters, we focus on (a) the degree to which the tutors could be built entirely "within" the tools, (b) use in real educational settings, (c) evidence that students learned from the tutors, when they were used in such settings, and (d) applicability and limitations of example-tracing tutors. Screenshots of the 18 tutors are shown in the Appendix.

## Problem-solving tutors

A number of example-tracing tutors built with CTAT can be viewed as "traditional" problem-solving tutors, meaning they provide step-by-step guidance as students practice the solving of recurrent complex problems. The two most comprehensive are *Mathtutor* and the *Genetics Tutor. Mathtutor* covers mathematics topics for grades 6 through 8 in the American school system (https://mathtutor.web.cmu.edu) (Aleven, McLaren, & Sewall, 2009). It is a re-implementation, as example-tracing tutors, of a set of Cognitive Tutors for middle-school mathematics that were created in our lab prior to CTAT (Baker et al., 2007; Koedinger, 2002; Koedinger & Terao, 2002; Rittle-Johnson & Koedinger, 2005). Each of these tutors had seen multiple rounds of classroom use. The *Genetics Tutor* supports problem-solving and reasoning tasks for high school and college level genetics (Corbett et al., 2010). It has more than 25 units covering topics in Mendelian Transmission, Pedigree Analysis, Gene Mapping, Population Genetics and Genetic Pathways Analysis. Various tutor units have been evaluated in 15 colleges and universities and in 4 high schools. In a total of 45 single-unit in-course evaluations, pretest-to-posttest learning gains averaged about 18 percentage points (equivalent to

almost 2 letter grades) across topics at both the post-secondary and high school levels. Both *Mathtutor* and the *Genetics Tutor* were originally implemented as rule-based Cognitive Tutors and were reimplemented as example-tracing tutors so as to make them available over the Internet. (In the mean time, CTAT has progressed so that rule-based tutors built in CTAT can also be deployed on the Internet, with a server-side model-tracing engine.)

In addition to these two tutors, several tutors with smaller domain scopes have been implemented that we also categorize as problem-solving tutors. *Lynnette* is a tutor for basic equation solving (Long & Aleven, 2013a; 2013b; Waalkens et al., 2013). It was originally implemented as an example-tracing tutor, and later re-implemented (also using CTAT) as a rule-based Cognitive Tutor (Long & Aleven, 2014), so as to be more flexible in recognizing students' major and minor solution variations. Thus, it followed the opposite trajectory of the two tutors just discussed. In four classroom studies with a total 487 students in grades 6 through 8, the example-tracing version of *Lynnette* led to pre/post gains in basic equation solving skill with medium to large effect sizes ($d = .69$, $d = 1.65$, and $d = 1.17$; in one experiment, the gains were not significant, due to a ceiling effect.

Our final example of a problem-solving tutor is the *Tuning Tutor,* an example-tracing tutor developed by Carolyn Rosé for use in her course at Carnegie Mellon University entitled "Applied Machine Learning." This tutor teaches students how to apply the general principles of avoiding overfitting in cross-validation to the case where parameters of a model need to be tuned. It was used during two semesters and was well received by students, many of whom completed more than the required number of tutor problems. Informally, students expressed their appreciation for the opportunity to practice with feedback and suggested that other concepts in the course include CTAT exercises as well. The incidence of students attending office hours during the unit on tuning, which used to be the most difficult unit in the second half of the course, dropped to nearly zero. These examples confirm that example-tracing tutors can effectively support learning at a variety of educational levels, including advanced college courses.

The *Mathtutor, Genetics Tutor,* and *Lynnette* projects help us better understand the practical import of the fact that example-tracing tutors can handle only problem types with a limited number of structurally dissimilar solution paths. As described, both *Mathtutor* (Aleven, McLaren, & Sewall, 2009) and the *Genetics Tutor* (Corbett et al., 2010) were originally developed as Cognitive Tutors (i.e., having a rule-based cognitive model) (Aleven, 2010) and later re-implemented as example-tracing tutors. These example-tracing tutors have been used extensively in schools and (in the case of the *Genetics Tutor*) in colleges, evidence that they are *bona fide,* real-world tutors. In both projects, the problem types, tutor interfaces, and tutoring behavior were kept largely the same when the tutors were re-implemented as example-tracing tutors. In both projects, the authors were able to capture, as example-tracing tutors, a large proportion of the problem sets of the original tutors, without simplifying their solution spaces or tutoring behavior (roughly 95% of the problem sets in both projects). On the other hand, in both projects, there was a small residue of problem types whose solution space was too large

to be feasibly implemented as example-tracing tutors (namely, a unit on abductive problem solving in genetics and units on equation solving in middle-school mathematics). This finding was confirmed in the *Lynnette* project, in which a tutor for basic equation solving initially implemented as an example-tracing tutor (Long & Aleven, 2013a; 2013b; Waalkens et al., 2013) was re-implemented as a rule-based Cognitive Tutor (Long & Aleven, 2014), in spite of the example-tracing tutor's proven effectiveness in multiple classroom studies (see above). The purpose of the reimplementation was to make it easier to create new problem types for the tutor but also to have greater flexibility in recognizing solution variations within each problem. These three projects thus provide key evidence that example-tracing tutors are often an excellent option for tutor development. The requirement of having a moderately-branching solution space turned out to sometimes be an obstacle, but only infrequently so. This finding is interesting especially if one considers that the problem types that were transitioned from rule-based Cognitive Tutors to example-tracing tutors were not created or selected with example-tracing tutors in mind. Evidently, in many domains, good practice problems need not have widely branching solution spaces.


## Tutors that use worked-out examples or erroneous examples

Several CTAT-built tutors have been created and used as platforms for research that investigates potential benefits of worked examples and/or erroneous examples in ITS, a topic that has seen increased interest since 2009 (e.g., Salden, Koedinger, Renkl, Aleven, & McLaren, 2010; McLaren, Adams, & Mayer, 2015).

A number of projects used example-tracing tutors to implement worked examples or erroneous examples. The *Stoichiometry Tutor* (McLaren et al., 2014; 2015; 2016) was extended so that students watch the step-by-step narrated playback of worked and erroneous examples in the tutor's problem-solving interface. They are prompted to explain the solutions, in the case of standard worked examples, or fix the errors, in the case of erroneous examples. In two classroom studies (McLaren et al., 2014; 2015), involving 295 10th and 11th grade students across the studies, erroneous examples and worked examples yielded the same learning outcomes as tutored and untutored problems but were more efficient in terms of time (*d* ranging between 1.76 and 3.31) and mental effort (*d* ranging between 0.89 and 1.04) (McLaren et al., 2014). The *AdaptErrEx* project used erroneous examples to help students learn decimals. Working with this tutor, students find, explain, and fix errors in decimal problems. In two studies, one with 208 subjects (Adams et al., 2014) and another with 390 subjects (McLaren, Adams, & Mayer, 2015), students who worked with erroneous examples performed significantly better on a delayed test (*d* = .62 and *d* = .33, respectively) than students in a tutored problem-solving condition with explanation steps. Building on this work and on the example-tracing technology, McLaren and colleagues created a suite of educational games, *Decimal Point*, unified by an amusement park metaphor, that use erroneous examples as the core instructional technique (Forlizzi et al., 2014). A considerable amount of special-purpose Flash code was required and some CTAT interface components had to be modified. Nevertheless, CTAT provided a valuable foundation. As a final project that employed worked examples in an example-tracing tutor, the *Proportional Reasoning Tutor* (Earnshaw, 2014) was created and used in a classroom study with 143 middle-school

students that examined effects of worked examples and tutored problems. Learners in the worked example condition took less time and scored higher on the post-test than learners in the two other conditions.

With more and more studies showing that worked examples enhance learning with an ITS or, at minimum, make it more efficient, we expect to see worked examples become a staple of ITS, combined with self-explanation prompts. We see a need for more research that reconciles the findings of studies focused on worked examples in the ITS literature with those in other literatures (e.g., van Gog & Rummel, 2010; Renkl, 2013). For example, it would be useful to test whether the typical expertise reversal effect (studying examples is more effective early on, problem solving is more effective later on) (Kalyuga, 2007; Kalyuga, Ayres, Chandler, & Sweller, 2003) occurs in the context of ITS.

As the projects reviewed above illustrate, worked-out examples or erroneous examples can often (as in the *Proportional Reasoning Tutor*) be authored entirely within CTAT (i.e., without programming). The same can be said about prompts for self-explanation, which often accompany worked examples (Booth, Lange, Koedinger, & Newton, 2013; McLaren et al., 2014; 2015; Rau, Aleven, & Rummel, 2015; see also Conati & VanLehn, 2000; Renkl, 2013). At other times, tool extensions were needed to support examples (e.g., the *Stoichiometry Tutor* and *AdaptErrEx*). Regarding the outer loop, that is, how examples and problems can be sequenced in a problem set, as mentioned, CTAT now supports the gradual backward fading of worked examples, shown to be an effective method of transitioning from worked examples to problems (Atkinson et al., 2003; Salden et al., 2010). This functionality is used in all problem sets of *Mathtutor* (Aleven, McLaren, & Sewall, 2009). As an alternative strategy, an author could decide to interleave fully worked-out examples and fully open problems (e.g., Paas & van Merriënboer, 1994), which she could do using the standard way of ordering problems within a problem set in the *Tutorshop*. A third strategy is fixed or adaptive fading of example steps at the knowledge component level, which was shown to be effective in one (non-CTAT) project (Salden, et al., 2010). CTAT provides building blocks for tutor authors to implement adaptive example fading in this manner, though we do not know of any CTAT tutors using this capability.

## Tutors with interactive graphical representations

Since 2009, a number of example-tracing tutors have been created with CTAT that feature the use of interactive graphical representations of learning materials. This work shows that interactive graphical representations can be a key way of designing effective tutors and leveraging ITS technology.

For some of these projects, special-purpose graphical interface components were developed (which required programming). For example, the *Fractions Tutor* (https://fractions.cs.cmu.edu) focuses on conceptual learning with multiple, interactive graphical representations including number lines, fractions circles, and rectangles (Rau, Aleven, & Rummel, 2013, 2015; Rau et al., 2014). This tutor was used in five classroom studies with over 3,000 students. In the last study, learning gains, up from the pre-test,

were $d = .40$ for the immediate post-test and $d = .60$ for a delayed post-test (Rau, Aleven, Rummel, & Rohrbach, 2012). The *Fractions Tutor* project illustrates that ITS, and example-tracing tutors specifically, can be effective with elementary school students. The only other ITS work we know of at the elementary school level is a set of studies by Stankov, Rosić, Žitko, and Grubišić (2007). To build the *Fractions Tutor*, special interface components for the interactive graphical representations of fractions were developed and added to CTAT's component class hierarchy. Once they were, they could be used, for example, to create dynamically linked representations (i.e., where the student's interactions with one representation are mirrored, automatically, in a second representation). They also became part of the standard CTAT release package and were used in other tutors. For example, in a different project, also dealing with elementary school fractions learning (Stampfer & Koedinger, 2013; Wiese & Koedinger, 2015), they were used to implement an instructional approach called *grounded feedback* (Mathan & Koedinger, 2005; Nathan, 1998). Given that CTAT provided interface components for interactive fraction representations, the grounded feedback tutor could be built entirely within the tools, using tutor-performed actions to make interface changes in the concrete representation in response to student entries in the abstract representation.

*Chem Tutor* (Rau, Michaelis, & Fay, 2015) features domain-specific interactive graphical representations, such as Lewis structures, Bohr models, and energy diagrams. Students receive step-by-step guidance for planning and constructing representations and are prompted to self-explain differences between different representations so as to reflect on the limitations of either representation. New interface components for these interactive representations were built first. *Chem Tutor* led to large learning gains in a field study with 74 undergraduate students enrolled in an introductory course for science majors ($d = 1.44$) (Rau et al., 2015) and in a lab experiment with 117 undergraduates ($d = .78$) (Rau & Wu, 2015). *Chem Tutor* has been used as a research platform to investigate support for representational competencies (Rau & Wu, 2015), effects of students' spatial abilities on their interactions with graphical representations (Rau, 2015), and visual attention behaviors (Peterson et al., 2015; Rau et al., 2015).

Some projects built interactive graphical representations using standard CTAT interface components, bypassing the need to first create new custom interface components; these include the *Genetics Tutor* (discussed above) and the *RedBlackTree Tutor*. The latter is an example-tracing tutor that aims to help students in a college level introductory data structures courses understand (i.e., "hand simulate") a key algorithm for red-black trees (Liew & Xhakaj, 2015; Xhakaj, 2015; Xhakaj & Liew, 2015), a data structure with many applications (e.g., Weiss, 2010). Interactive red-black tree structures in the tutor interface were built out of standard CTAT components such as text boxes, radio buttons and drop-down components, combined with standard Flash elements and a small amount of custom ActionScript. In two small evaluation studies (Liew & Xhakaj, 2015; Xhakaj & Liew, 2015; Xhakaj, 2015, approximately one hour's work with the *RedBlackTree Tutor* led to large learning gains ($d = 1.66$ and $d = 3.06$, respectively).

Thus, CTAT offers various ways of supporting interactive graphical representations. In the best-case scenario (as in the *Grounded Feedback Tutor*), CTAT already offers an

interactive interface component for the given representation, so the tutor can be built without programming. Sometimes, a new interactive graphical representation can be assembled entirely from standard CTAT-supported interface components (as in the *Genetics Tutor*), sometimes with a small amount of custom ActonScript (as in the *RedBlackTree Tutor*). Sometimes, when moving into a new domain, it is necessary to create new interactive interface components for the given graphical representations (e.g., *Mathtutor*, the *Fractions Tutor*, and *Chem Tutor*), which can then become part of the tool set. This experience illustrates that it is important that an ITS authoring tool is "open" so that it is easy to extend its collection of interface components, or to use custom programming to extend what tutors built with the tool can do.


**Tutors that support other pedagogical approaches**
A number of tutors have been built that use pedagogical approaches other than tutored problem solving: educational games (example discussed above), collaborative learning, learning with an external problem-solving environment, activities that target sense making and fluency building, and guided invention activities.

CTAT's example-tracing tutor technology has been extended so that it now supports the authoring of tutors with simple support for *collaborative* learning (Olsen, Belenky, Aleven, Rummel, et al., 2014), similar to earlier work done with constraint-based tutors (Baghaei, Mitrovic, & Irwin, 2007). Using CTAT, an author can build tutor activities that combine tutored problem solving with embedded simple collaboration scripts. In these activities, networked small teams of students work synchronously on tutor problems. The students can have a shared but – if the author so chooses – differentiated view of a joint problem and can have different actions available, so collaborating students can have different roles. The phases of the collaboration script can be tied to the problem-solving state, for example so students may switch roles dynamically within a given problem. This form of collaboration support is quite flexible; different forms of collaboration support can be authored. It has a number of limitations, however. For example, it is not currently straightforward within CTAT to provide feedback on how students are collaborating (e.g., Rummel, Walker, & Aleven, under review; Walker, Rummel, & Koedinger, 2014). Further, collaboration scripts have to be built from low-level interface components and there is no provision for reacting to dialogue between students (e.g., Adamson, Dyke, Jang, & Rosé, 2014).

CTAT's collaborative features were used to author collaboration scripts that support proven ways of scripting collaboration such as roles, cognitive group awareness (Janssen & Bodemer, 2013), and individual accountability (Slavin, 1996). In a pull-out study conducted in schools with 56 students, these tutors were shown to help elementary students learn collaboratively (Olsen, Belenky, Aleven, & Rummel, 2014), with gains no different from equivalent tutors used individually.  In a second study, a classroom study with 189 participating students, there again were no differences in learning gains between students working individually and collaboratively, but students working collaboratively spent less time on the tutor (Olsen, Rummel, & Aleven, under review). This line of work may help as a bridge between ITS and research in Computer-Supported Collaborative

Learning (CSCL) (see also Baghaei et al., 2007; Kumar & Kim, 2014; Rummel, Walker, & Aleven, under review; VanLehn, this issue).

CTAT has also been used to provide tutoring within external problem-solving environments including simulators for thermodynamics and chemistry (Aleven, Sewall, et al., 2006; Borek, McLaren, Karabinos, & Yaron, 2009). In a new project by McLaren and colleagues embeds a CTAT example-tracing tutor within Google Sheets. This tutor guides college students in their work on business modeling problems, represented on spreadsheets, using replayed solutions as worked examples and tutors developed in CTAT. This work builds on work on plug-in tutoring agents that pre-dates CTAT (Koedinger, Suthers, & Forbus, 1999; Mathan & Koedinger, 2005; Ritter & Koedinger, 1996). Although hooking up an external problem-solving environment can be challenging (it typically requires extensions to the external problem-solving environment), it is facilitated by CTAT's factored architecture, which, conforming to Ritter and Koedinger's (1996) notion, strictly separates interface and tutor functionality (http://ctat.pact.cs.cmu.edu/index.php?id=tool-tutor). This notion is also being addressed in the xPST authoring tools project (Blessing et al., 2009; Kodavali et al., 2010).

A new project with the *Fractions Tutor* studies whether an ITS can be made more effective by adaptively targeting a broader range of learning mechanisms than ITS typically do. The work is grounded in the Knowledge-Learning-Instruction (KLI) framework (Koedinger et al., 2012), which links cognitive theory and instructional design. The tutor aims to support all three key KLI learning mechanisms, by supporting not just induction and refinement (IR) processes as ITS typically do, but also verbal sense-making (SM) processes and fluency-building (F) processes. SM activities in the *Fractions Tutor* include instructional videos that explain fractions topics, interleaved with brief supporting problem-solving exercises and opportunities to self-explain via drag-and-drop fill-in-the-blank questions. F activities emphasize procedural practice with design features that may encourage students to work more quickly (e.g., bigger problem steps, short hints, and on-screen timers). In a classroom study with 1068 fourth and fifth grade students across 12 schools, we found that the tutor leads to significant pre- to post-test learning gains ($d = .47$) (Doroudi, Holstein, Aleven, & Brunskill, 2015); students who did relatively more fluency-building activities learned more. Ongoing work uses machine learning techniques to create policies for adaptive activity selection.

CTAT has also been used to implement tutors for guided invention activities, in which the goal is for students to develop a quantitative method that captures a mathematical or physics concept, guided by carefully designed sets of contrasting examples. Prior research shows that these kinds of activities prepare students well to learn and transfer from a more traditional lesson (Schwartz & Martin, 2004; Schwartz, Chase, Oppezzo, & Chin, 2011). Over the years, three different tutors for guided invention activities have been built with CTAT. The first one, by Roll, Aleven, and Koedinger (2010) relied on rules and constraints (i.e., was not an example tracing tutor). A second tutor was an example-tracing tutor that provided domain-general guidance during invention activities (Holmes, Day, Park, Bonn, & Roll, 2014). In an evaluation study, in which 87 undergraduate students in a first-year physics lab course at the University of British

Columbia used the system for two activities, roughly 30 minutes each (Holmes et al., 2014), domain-general guidance during invention activities enhanced students' conceptual understanding. In addition, the system was used as part of the regular physics instruction at the University of British Columbia until last year. A third tutor for invention activities, for middle-school physics, is currently under development and also builds on example-tracing tutors, although with substantial custom programming (Chase, Marks, Bernett, & Aleven, 2015; Chase, Marks, Bernett, Bradley, & Aleven, 2015).

**A tutor for language learning**
The remaining cluster comprises a single tutor, the Article Tutor, an example-tracing tutor that teaches students the English article system (when to use *a*, *an*, *the*, or no article). Other tutors for language learning have been built with CTAT as well (e.g., Guan, Liu, Chan, Ye, & Perfetti, 2011; Liu et al., 2011; Ogan et al., 2009). The Article Tutor was built to be part of a course for English as a Second Language (ESL). It was used in research to investigate whether the use of self-explanation can be effective as a learning strategy for ESL. In total, six tutor versions were built, including an adaptive tutor that prompted students to self-explain only if they got the question wrong. This form of adaptivity could be authored entirely within CTAT's non-programmer tutor authoring paradigm. In four classroom studies (390 students total) all conditions learned from the tutor but the practice-only (no self-explanation) condition consistently was the more efficient form of instruction (Wylie, Koedinger, & Mitamura, 2009; 2010a; 2010b; Wylie et al., 2011). The work refines the conditions under which self-explanation is understood to be effective (e.g., Koedinger et al., 2012).

**Discussion and conclusions**
The main contributions highlighted in our IJAIED 2009 paper were: First, the CTAT project pioneered a non-programmer paradigm for ITS authoring that involves (a) generalized examples of problem-solving behavior as the tutor's representation of domain knowledge, (b) tools for creating, without programming, tutors that use these generalized examples to provide within-problem, step-level guidance to students, and (c) an algorithm for flexibly using generalized examples to interpret student behavior and provide step-based tutoring. A second intellectual contribution claimed at the time was a demonstration, across a range of tutor research projects, that this paradigm can be widely useful and effective. A third contribution was evidence of substantial cost savings: Building example-tracing tutors was shown to be 4-8 times as cost-effective, compared to estimates in prior literature.

We see six novel scientific contributions of our project since 2009. First, we update and bolster our argument that example-tracing tutors should be viewed as first-class citizens in the world of ITS. We now focus on adaptive behavior as a hallmark of intelligence, following Newell and Simon (1976). We provide a definition for adaptivity based on three factors, (cf. Aleven, 2015; Aleven et al., 2013; forthcoming) and highlight many elements of adaptivity in the behavior of example-tracing tutors.

Second, we provide additional evidence that example-tracing tutors are a useful and mature paradigm for developing intelligent tutors. We describe 18 example-tracing tutors built since 2009 and used in real educational settings, many with statistically significant pre/post learning gains. Most of these tutors were for STEM domains (science, technology, engineering, and mathematics) including computer programming, but we also see tutors for business modeling and language learning. The tutors were used by students ranging from late elementary school to university undergraduate and graduate students. As evidence of widespread use, CTAT-built tutors were used by 44,000 students and account for 40% of the data sets in *DataShop*. This work thus supports the notion that a non-programmer approach to ITS authoring can yield effective tutors. It also illustrates that CTAT has reached a state of maturity in which tutors built with CTAT routinely withstand the rigors of classroom use and even use within MOOCs.

Third, the 18 reviewed example-tracing tutors illustrate a range of pedagogical approaches, including (standard) tutored problem solving, the use of worked-out and erroneous examples, interactive graphical representations, and collaborative learning. CTAT supports these features to a substantial degree; however, some amount of programming is sometimes necessary. Thus, we see that an ITS authoring tool, in the hands of creative authors, can be used in unanticipated ways. We also see that an ITS authoring environment, even one that supports a non-programmer approach to authoring, should be easily extensible and accommodate custom programming.

Fourth, the strengths and limitations of example-tracing tutors are now better understood. In particular, we better understand the extent to which it is limiting that example-tracing tutors support only problems that have no more than a moderately-branching solution space (unless many branches are isomorphic so that an author can collapse them into a small number of branches using formulas). The experience across a range of tutor development projects suggests that occasionally this limitation precludes use of the example-tracing paradigm, but more frequently, example-tracing tutors are a viable approach. Other limitations are that adaptivity in response to affect, metacognition, and motivation is not currently supported and that the number of example tracing tutors that have been demonstrated in domains outside of STEM domains remains relatively small.

Fifth, we have learned how an ITS architecture can be factored so it supports the flexible re-use of tutor components. First, the notion of separating tool and tutor pre-dates CTAT (Ritter & Koedinger, 1996), but CTAT demonstrates some advantages of it that have not been demonstrated before. Most importantly, the changes in web technologies that forced us twice to revamp our tutor front-end technology would have spelled doom for CTAT had it not been for the strict tool/tutor separation. The tool/tutor separation has also made it possible to mix-and-match tutor engines and interface technology and makes it easier to extend the tutor interface by creating new components (rather than project-specific interfaces). We recommend this separation for any ITS project. The second key way of factoring is to separate inner loop and outer loop, with the student model as the sole means of communication between the two loops. This separation has proven to be useful (e.g., it has been relatively easy to plug in an alternative student model and outer loop). It will be interesting to see if this separation holds up when we extend the range of student

models and task selection policies. Sixth and finally, we have created ways of embedding CTAT tutors in a range of e-learning environments, and continue to work on both extending the range of environments in which these tutors can be embedded, and the range of advanced tutoring functionality (including learning analytics and student modeling) that can be made available to these environments.

We see interesting days ahead for ITS authoring tools. Our ongoing work focuses on generalizing CTAT and supporting tutoring at scale. For ITS technology to spread, it is critical that authoring tools not only support cost-effective authoring of sophisticated tutor behaviors, but also that tools created with these tutors interface with popular e-learning and MOOC environments across the range of popular client platforms.

## Appendix: Screenshots of 18 example-tracing tutors

We provide screenshots of the 18 example-tracing tutors built with CTAT since 2009 that are discussed in the body of the document.

### Mathtutor





*Mathtutor* (Aleven, McLaren, & Sewall, 2009) is a comprehensive web-based tutoring system for mathematics in grades 6 through 8.

## Genetics Tutor

1. Determine the dominance and linkage of the disease shown and then determine the probabilities that the labeled individuals are carriers of the disease.

**1. Determine the dominance/linkage for the pedigree.**

The disease allele for this trait is recessive.

The trait is autosomal.

**2. Enter the probability the selected individuals are carriers in the fields at the right.**

**3. Finally, enter the probability VI-1 is affected.**

0.25*0.5*1/12*0.5

The probability of inheriting the disease allele from V-4 is 1/4 * 1/2=1/8.

The probability of inheriting the disease allele from V-5 is 1/12 * 1/2=1/24.

What is the probability that VI-1 will inherit the allele from both parents?

[Previous] [Next →]

[? Hint] [✓ Done]

2. You have already shown (Problem 1) that very strong selection against the homozygous recessive does not significantly change the frequency of the recessive allele when the allele is at VERY LOW FREQUENCY in the population. In this problem, you will examine how such very strong selection against the homozygous recessive changes the frequency of the recessive allele when the recessive allele is at VERY HIGH FREQUENCY.

1.) You have only one data point ('cc' frequency = 0.6) and one tool, the HW equation ($p^2 + 2pq + q^2 = 1$).
Consider how the genotype frequencies in this population change in one generation.
2.) There is an intense selection ( w(cc) = 0.0 ) against the affected homozygous recessives.
How does this selection change the numbers of each genotype in two successive generations (population of 10,000) when the recessive allele is very frequent?

Table 1: Generation 1 (Population Size = 10,000)

| Genotype | Frequency in Generation 1 | Numbers before Selection | Numbers after Selection |
|---|---|---|---|
| CC Homozygote | 0.36 | 3600 | 3600 |
| Cc Heterozygote | 0.48 | 4800 | 4800 |
| cc Homozygote | 0.16 | 1600 | 0 |
| Totals | 1 | 10000 | 8400 |

Table 2: Generation 2 (Population Size = 10,000)

| Genotype | Frequency in Generation 2 | Numbers before Selection | Numbers after Selection |
|---|---|---|---|
| CC Homozygote | 0.5102 | 5102 | 5102 |
| Cc Heterozygote | 0.4082 | 4082 | 4082 |
| cc Homozygote | 0.0816 | 816 | 0 |
| Totals | 1 | 10000 | 9184 |

[? Hint] [✓ Done]

Table 3: Allele Frequencies

| | Generation 1 | Generation 1 Survivors | Generation 2 Survivors |
|---|---|---|---|
| C (dominant): | 0.6 | 0.7143 | 0.7778 |
| c (recessive): | 0.4 | 0.2857 | 0.2222 |

Recessive Allele (c) Frequency Over 75 Generations

— Prob2: f(c)=0.4; w(cc)=0.0
— Prob1: f(c)=0.02; w(cc)=0.0

[← Previous] [Next →]

1. In problem 1, recessive allele (c) frequency decreased from 0.02 to 0.0192. How much do the allele frequencies change in this problem when there is selection against the homozygous recessives as before, but RECESSIVE ALLELES ARE MUCH MORE FREQUENT?

Conclusion: Recessive allele frequency decreases a lot MORE THAN WHEN THE RECESSIVE ALLELE IS INFREQUENT.

Explanation: When recessive alleles are frequent, homozygotes carry a large proportion of those alleles.

2. Examine the graph. What long-term evolutionary outcome is expected under these circumstances compared to problem 1?

Observation: Loss of the recessive allele is fast initially but continually slows down.

Summary: Loss of recessive alleles is much faster when homozygous recessive frequency is much higher.

The *Genetics Tutor* (Corbett et al., 2010) covers a wide range of problem-solving activities in high-school and college-level genetics.

## Lynnette – Basic Equation Solving



*Lynnette*, a tutor for basic equation solving for grades 6, 7, and 8, originally implemented as an example-tracing version (top) (Long & Aleven 2013a; 2013b; Waalkens et al., 2013), later re-implemented, also with CTAT, as a rule-based Cognitive Tutor (bottom)(Long & Aleven, 2014)

# The Tuning Tutor – Parameter Fitting in Machine Learning

In the first stage we selected the setting we would use to build a tuned model over the whole data set (which we would do in stage 2). Now we want to estimate what that model's performance would be on new data (stage 3). We do that using an embedded cross validation.

In the table below, the top 5 rows represent the inner loop of the cross validation and the bottom 5 rows represent the outer loop. For the outer loop we just divide each fold into a training set and a testing set, as we did for the simple cross validation. The only exception here is that we may select a different setting on each fold based on the inner loop. For the inner loop, instead of a cross validation on the training data for each fold, we divide each fold into train, validation, and hold out. The hold out is the test set, which we will not consult on the inner loop. The train set is what we train on and the validation set is what we test on. Based on this train/test split for a fold, we will make a selection of a setting for that fold, which we will then use in the outer loop.

| Training | Validation | Hold out | A | B | C | D |
|---|---|---|---|---|---|---|
| {Y3,Y4,Y5} | {Y2} | {Y1} | 0.58 | 0.73 | 0.94 | 0.71 |
| {Y1,Y3,Y5} | {Y4} | {Y2} | 0.45 | 0.59 | 0.52 | 0.67 |
| {Y1,Y2,Y4} | {Y5} | {Y3} | 0.74 | 0.67 | 0.56 | 0.68 |
| {Y1,Y2,Y5} | {Y3} | {Y4} | 0.78 | 0.79 | 0.67 | 0.5 |
| {Y2,Y3,Y4} | {Y1} | {Y5} | 0.69 | 0.75 | 0.76 | 0.43 |
| {Y2,Y3,Y4,Y5} | {Y1} | - | 0.59 | 0.58 | 0.62 | 0.57 |
| {Y1,Y3,Y4,Y5} | {Y2} | - | 0.61 | 0.69 | 0.62 | 0.82 |
| {Y1,Y2,Y4,Y5} | {Y3} | - | 0.81 | 0.37 | 0.79 | 0.76 |
| {Y1,Y2,Y3,Y5} | {Y4} | - | 0.71 | 0.8 | 0.53 | 0.57 |
| {Y1,Y2,Y3,Y4} | {Y5} | - | 0.59 | 0.68 | 0.52 | 0.64 |

We compare the baseline model with the tuned model to see whether the tuning process makes significant improvement.

| Fold | Baseline Model Performance | Tuned Model Performance |
|---|---|---|
| {Y1} | 0.59 | 0.62 |
| {Y2} | 0.61 | 0.82 |
| {Y3} | 0.81 | |
| {Y4} | 0.71 | |
| {Y5} | 0.59 | |
| Average | 0.662 | |

1. Please click on the highest performance value for fold 3 on the inner loop and then click on the corresponding performance value for the outer loop.

The performance value you have selected on the inner loop is not the highest performance value.

**?** Hint

✓ Done

← Previous    Next →

The *Tuning Tutor* helps graduate students and advanced undergraduate students learn to use cross validation to avoid overfitting when tuning model parameters. It was used at Carnegie Mellon University in a course for graduate students and advanced undergrads called "Applied Machine Learning" by Carolyn Rosé.

**Stoichiometry Tutor**





The *Stoichiometry Tutor* (McLaren et al., 2014; 2015; 2016) supports the narrated replay of example solutions. As the steps of the problem are replayed, a flashing yellow box draws the student's attention to the next step of the worked example (top). After the worked example plays back, the student is prompted to fill out the reasons for every step, and then their solution is evaluated (i.e., delayed feedback; bottom).

**AdaptErrEx – Erroneous Examples**

Allison has two ribbons. One ribbon is 0.125 inches long and the other is 0.83 inches long. Allison's friend asks her to choose the longest ribbon.

Allison said this incorrect answer: I placed the numbers on a number line to see that 0.125 is the largest, so I want the 0.125 inch long ribbon.

0.83  0.125

0                                                      1
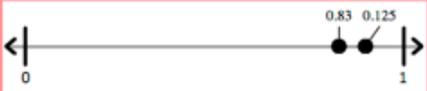
What did Allison do wrong?

She thinks that _____.

○ longer decimals are smaller

○ shorter decimals are smaller than zero

○ shorter decimals are larger

◉ longer decimals are larger

Click on the line where the incorrect point should go to fix Allison's error.

0.125                                   0.83

0                                                      1

Looking at the corrected number line, which ribbon is longer?

◉ 0.83 inches because it is closer to 1

○ 0.125 inches because it is closer to 0

What advice would you give to Allison to solve the problem right next time?

Allison, to find the longest ribbon you should pay attention to which decimal _____.

○ is the shortest

○ is the longest

○ has the smallest number in the tenths place

◉ has the largest number in the tenths place

**Message Window**

You've got it. Well done.

← Previous        Next →

Done

*AdaptErrEx* (Adams et al., 2014; McLaren, Adams, & Mayer, 2015) is an example-tracing tutor for learning decimals (part of 6th-grade mathematics) that has students identify, correct, and explain incorrect steps in worked-out problem solutions

*Decimal Point:* **Educational Games for Learning Decimals**



*Decimal Point* (Forlizzi et al., 2014), built using CTAT as foundation, supports game-based learning with erroneous examples to help middle-school students learn decimals

**Proportional Reasoning Tutor**

3. Bill's Hometown Furniture Store creates custom-ordered furniture. Last week, Bill, the owner, received an order for 12 identical kitchen cabinets. Bill hired 4 carpenters who were able to make 7 cabinets in 5 days. Unfortunately, over the weekend, one of the carpenters broke his arm and will be unable to help finish the order. If Bill has 3 healthy carpenters complete the remaining cabinets, how long will it take them to finish the job?

Step 1.

$$\frac{4}{7} = \frac{3}{x}$$

Step 2.

$$\boxed{4} \cdot x = \boxed{7} \cdot \boxed{3}$$

$$\boxed{4x} = \boxed{21}$$

Step 3.

$$x = \frac{\boxed{4}}{\boxed{\phantom{xxxx}}}$$

$$x = \boxed{\phantom{xxxx}}$$

Please enter '21' in the highlighted field.

**?** Hint

← Previous    Next →

✔ Done

The *Proportional Reasoning Tutor* (Earnshaw, 2014) supports worked examples and tutored problems in middle-school mathematics.

**Fractions Tutor**



**Making Fractions**

A. **Let's make a fraction to compare it to another!**

This is the unit. Let's make $\frac{4}{5}$.

1. Into how many equal sections must you partition the unit? [5]

2. Drag one section into the empty circle.

3. The blue section is $\frac{1}{5}$ of the gray circle.

4. To show $\frac{4}{5}$, you need [4] sections.

B. **Let's make a second fraction to compare it to the first!**

This is the unit. Let's make $\frac{4}{9}$.

1. Into how many equal sections must you partition the unit? [9]

2. Drag one section into the empty circle.

3. The purple section is $\frac{1}{9}$ of the gray circle.

4. To show $\frac{4}{9}$, you need [4] sections.

C. **Which fraction is bigger?**

1. How many total sections are in the blue circle? [5]
   How many total sections are in the purple circle? [9]

2. The blue sections are [larger than ▼] the purple sections because there are [fewer ▼] sections in the blue circle.

3. Since both circles have [4] colored sections, $\frac{4}{5}$ [larger than ▼] $\frac{4}{9}$

**? Hint**

**continue**
**Wonderful!**

**Equivalent Fractions**

A. **Let's review a rectangle as an example to find equivalent fractions!**

The rectangles below should all show the same amounts. $= \frac{4}{5}$

$= \frac{8}{10}$ [Ok]

$\frac{12}{15}$ [Ok]

$\frac{16}{20}$ [Ok]

1. Type in the fraction shown in the pink rectangle.

B. **Let's partition number lines to make equivalent fractions!**

All number lines below show the same amounts. $= \frac{4}{5}$

$= \frac{8}{10}$ [Ok]

$\frac{12}{15}$ [Ok]

$\frac{16}{20}$ [Ok]

1. Partition each number line into differently sized sections that remain equivalent to each other. Then, type in the fraction that each number line shows.

**? Hint**

**continue**
**You did it!**

C. **What did we learn about the rectangle and the number line?**

1. Multiplying the numerator and the denominator by the same number is like partitioning the areas into more sections [without ▼] changing the amount.

2. Rectangles and number lines show [the same ▼] amount with [different ▼] numbers of sections show equivalent fractions.

The *Fractions Tutor* (Rau, Aleven, & Rummel, 2013, 2015; Rau et al., 2014) supports conceptual learning of fractions in grades 4 and 5 using multiple interactive graphical representations of fractions

**Grounded Feedback Tutor**

$$\frac{2}{7} \;+\; \frac{3}{5} \;=\; ?$$

☐ I need to convert this fraction   ☐ I need to convert this fraction

$$\frac{2}{7} + \frac{3}{5} = \frac{5}{12}$$

**? Hint**

← Previous   Next →

Done

$$\frac{2}{7} \;+\; \frac{3}{5} \;=\; ?$$

2 × 5 = 10
7 × 5 = 35

3 × 7 = 21
5 × 7 = 35

$$\frac{10}{35} + \frac{21}{35} = \frac{31}{35}$$

**? Hint**

← Previous   Next →

Done

*Grounded Feedback Tutor* (Stampfer & Koedinger, 2015; Wiese & Koedinger, 2015) for elementary school fractions learning; instead of providing explicit correctness feedback, as the student enters a solution using numeric symbols, the fraction bars (except those representing the given fractions) are updated by the system to reflect the student input

## Chem Tutor

**Bonding**

Let's make the Lewis structure for carbon dioxide!

? Hint
Periodic Table

1  The chemical formula for carbon dioxide is $CO_2$. That means carbon dioxide has [1 carbon atom ▾] and [2 oxygen atoms ▾].

2  The carbon atom has [4] **valence electrons**, and each oxygen has [6] **valence electron**.

3  The carbon atom has **single bonds** with [no oxygen atom ▾], and **double bonds** with [2 oxygen atoms ▾].

4  Use the tool box on the left to draw the **Lewis structure** for carbon dioxide.

5  Now, draw the **dipole moment vector** for each carbon-oxygen bond in carbon dioxide.

You're close! You drew the right number of lone pairs, but you did not arrange them correctly. Remember, valence electrons that do not participate in bonding occur in pairs.

← Previous          Next →

→ Identify properties of the molecule

→ Plan features of the representation

→ Receive feedback on interactions

→ Construct representations with an interactive tool

In *Chem Tutor* (Rau, Michaelis, & Fay, 2015), designed for introductory undergraduate chemistry learning, students plan and construct a graphical representation (Lewis structures), with feedback from the system

**Atoms and Electrons**

A  Let's revisit the Bohr model for chlorine!

B  Let's revisit the energy diagram for chlorine!

? Hint
Periodic Table

Energy

3p  ↑↓ ↑ | x  ↑↓ | y  ↑↓ | z  ↑↓
3s  ↑↓

2p  ↑↓ | x  ↑↓ | y  ↑↓ | z  ↑↓
2s  ↑↓

1s  ↑↓

Check

No, this is not correct. Electrons in the same orbital must have opposite spin.

← Previous          Next

→ Construct a different representation of the same atom

→ Receive immediate, error-specific feedback

→ Reflect on differences and limitations of representations

C  Let's look at the differences between these diagrams!

1  Regarding the **electrons**, the Bohr model [shows electrons as dots ▾], whereas the energy diagram [shows electrons as arrows ▾].

2  The Bohr model shows the **relative energy levels** of [shells only ▾], whereas, the energy diagram shows the **relative energy levels** of [orbitals and shells ▾].

3  Regarding the **spin states of electrons**, the Bohr model shows [nothing about the spin state ▾], whereas the energy diagram shows [spin with the direction of the arrow ▾].

Given one representation, students construct a different kind of graphical representation of the same atom and are prompted to reflect on the differences and limitations of the two visual representations.

**RedBlackTree Tutor**



Example problem in the *RedBlackTree Tutor* (Liew & Xhakaj, 2015; Xhakaj, 2015; Xhakaj & Liew, 2015), which helps explain how red-black trees are built, a common data structure in computer science.

**Tutor for collaborative learning of fractions**

### Equivalent Fractions

A **Match the equivalent fractions.**

1 Match an equivalent fraction for each fraction shown below. Make sure the circles show the same amounts before hitting submit.
You and your partner can each move only half of the fractions. Discuss with your partner what the correct answers are.

| $\frac{8}{10}$ | $\frac{5}{9}$ | $\frac{2}{7}$ | | $\frac{6}{7}$ | $\frac{16}{24}$ | $\frac{1}{4}$ |

$\frac{3}{4} = \frac{9}{12}$ | $\frac{3}{7} = \frac{6}{14}$ | $\frac{4}{6} =$ [ ]

? Hint

← Previous    Next →

Submit

### Equivalent Fractions

A **Match the equivalent fractions.**

1 Match an equivalent fraction for each fraction shown below. Make sure the circles show the same amounts before hitting submit.
You and your partner can each move only half of the fractions. Discuss with your partner what the correct answers are.

| $\frac{8}{10}$ | $\frac{5}{9}$ | $\frac{2}{7}$ | | $\frac{6}{7}$ | $\frac{16}{24}$ | $\frac{1}{4}$ |

$\frac{3}{4} = \frac{9}{12}$ | $\frac{3}{7} = \frac{6}{14}$ | $\frac{4}{6} =$ [ ]

? Hint

← Previous    Next →

Submit

Elementary school students (grades 4 and 5) use the *Collaborative Fractions Tutor* with a partner; each partner has a different role, with a different view of the problem and different available actions (Olsen, Belenky, Aleven, & Rummel, 2014; Olsen, Belenky, Aleven, Rummel, et al., 2014; Olsen, Rummel, & Aleven, under review)

**Tutor for Business Modeling with Google Sheets**



An example-tracing tutor build by McLaren and colleagues, embedded within Google Sheets, provides guidance with business modeling problems

**Fractions Tutor version that supports Sense Making, Induction/Refinement, and Fluency Building**



A new version of the *Fractions Tutor* (Doroudi et al., 2015) has activities targeting each of the three Knowledge-Learning-Instruction (KLI; Koedinger et al., 2012) learning mechanisms: induction and refinement (IR) (left), sense-making (SM) (center), and fluency (F) (right)

## Tutors for Guided Invention activities



Tutor for guided invention activities by Roll et al. (2010)



A tutor for guided invention activities with a Wizard of Oz interface (Chase et al., 2015); CTAT's collaborative tutoring facility enabled separate roles and capabilities for student and wizard.

**The Article Tutor**



Version of the *Article Tutor* (Wylie et al., 2011). that supports self-explanation

# References

Adams, D. M., McLaren, B. M., Durkin, K., Mayer, R. E., Rittle-Johnson, B., Isotani, S., & Velsen, M. V. (2014). Using erroneous examples to improve mathematics learning with a web-based tutoring system. *Computers in Human Behavior*, *36*, 401 - 411. doi:10.1016/j.chb.2014.03.053

Adamson, D., Dyke, G., Jang, H., & Rosé, C. P. (2014). Towards an agile approach to adapting dynamic collaboration support to student needs. *International Journal of Artificial Intelligence in Education*, *24*(1), 92-124. doi:10.1007/s40593-013-0012-6

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. (2015). A is for Adaptivity, but what is Adaptivity? Re-defining the field of AIED. In K. Porayska-Pomsta, G. McCalla, & B. du Boulay (Eds.), Proceedings of the Workshops at the 17th International Conference on Artificial Intelligence in Education AIED 2015 **(**Vol. 4, Workshop on Les Contes du Mariage: Should AI stay married to Ed? A workshop examining the current and future identity of the AIED field). Available from: http://adenu.ia.uned.es/publications/aied2015ws/procs/v2/

Aleven, V., Beal, C. R., & Graesser, A. C. (2013). Introduction to the special issue on advanced learning technologies. *Journal of Educational Psychology*, *105*(4), 929. doi:10.1037/a0034155

Aleven, V., & Koedinger, K. R. (2013). Knowledge component approaches to learner modeling. In R. Sottilare, A. Graesser, X. Hu, & H. Holden (Eds.), *Design recommendations for adaptive intelligent tutoring systems* (Vol. I, Learner Modeling, pp. 165-182). Orlando, FL: US Army Research Laboratory.

Aleven, V., McLaren, B. M., & Sewall, J. (2009). Scaling up programming by demonstration for intelligent tutoring systems development: An open-access web site for middle school mathematics learning. *IEEE Transactions on Learning Technologies*, *2*(2), 64-78.

Aleven, V., McLaren, B. M., Roll, I., & Koedinger, K. R. (2006). Toward meta-cognitive tutoring: A model of help seeking with a cognitive tutor. *International Journal of Artificial Intelligence in Education*, *16*, 101-128.

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., McLaughlin, E. A., Glenn, R. A., & Koedinger, K. R. (forthcoming). Instruction based on adaptive learning technologies. In R. E. Mayer & P. Alexander (Eds.), *Handbook of research on learning and instruction.* Routledge.

Aleven, V., Sewall, J., McLaren, B. M., & Koedinger, K. R. (2006). Rapid authoring of intelligent tutors for real-world and experimental use. In Kinshuk, R. Koper, P. Kommers, P. Kirschner, D. G. Sampson, & W. Didderen (Eds.), *Proceedings of the 6th IEEE International Conference on advanced learning technologies (ICALT 2006)* (pp. 847-851). Los Alamitos, CA: IEEE Computer Society.

Aleven, V., Sewall, J., Popescu, O., van Velsen, M., Demi, S., & Leber, B. (2015a). Reflecting on twelve years of ITS authoring tools research with CTAT. In R. Sottilare, A. Graesser, X. Hu, & K. Brawner (Eds.), *Design recommendations for adaptive intelligent tutoring systems* (Vol. III, Authoring Tools, pp. 263-283). Orlando, FL: US Army Research Laboratory.

Aleven, V., Sewall, J., Popescu, O., Xhakaj, F., Chand, D., Baker, R. S., . . . Gasevic, D. (2015b). The beginning of a beautiful friendship? Intelligent tutoring systems and MOOCs. In C. Conati, N.

Heffernan, A. Mitrovic, & M. F. Verdejo (Eds.), *Proceedings of the 17th International Conference on AI in Education, AIED 2015* (pp. 525-528). New York: Springer. doi:10.1007/978-3-319-19773-9_53

Anderson, J. R., Conrad, F. G., & Corbett, A. T. (1989). Skill acquisition and the LISP tutor. *Cognitive Science*, *13*(4), 467 - 505. doi: 10.1016/0364-0213(89)90021-9

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 differences 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.

Arroyo, I., Woolf, B. P., Burleson, W., Muldner, K., Rai, D., & Tai, M. (2014). A multimedia adaptive tutoring system for mathematics that addresses cognition, metacognition and affect. *International Journal of Artificial Intelligence in Education*, *24*(4), 387-426. doi:10.1007/s40593-014-0023-y

Atkinson, R. K., Renkl, A., & Merrill, M. M. (2003). Transitioning from studying examples to solving problems: Effects of self-explanation prompts and fading worked-out steps. *Journal of Educational Psychology, 95*(4), 774-783.

Baghaei, N., Mitrovic, A., & Irwin, W. (2007). Supporting collaborative learning and problem-solving in a constraint-based CSCL environment for UML class diagrams. *International Journal of Computer-Supported Collaborative Learning, 2*(2-3), 159-190. doi:10.1007/s11412-007-9018-0

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 Artificial Intelligence and Education*, *17*(4), 341-369.

Blessing, S., Devasani, S., & Gilbert, S. (2011). Evaluation of WebxPST: A browser-based authoring tool for problem-specific tutors. In G. Biswas, S. Bull, J. Kay, & A. Mitrovic (Eds.), *Proceedings of the 15th International Conference on Artificial Intelligence in Education (AIED 2011)* (pp. 423-425). Berlin: Springer.

Blessing, S. B., & Gilbert, S. (2008). Evaluating an authoring tool for model-tracing intelligent tutoring systems. In B. Woolf, E. Aimeur, R. Nkambou, & S. Lajoie (Eds.), *ITS '08: Proceedings of the 9th International Conference on Intelligent Tutoring Systems* (pp. 204-215). Berlin, Heidelberg: Springer-Verlag. doi:10.1007/978-3-540-69132-7_25

Blessing, S. B., Gilbert, S. B., Blankenship, L., & Sanghvi, B. (2009). From SDK to XPSt: A new way to overlay a tutor on existing software. In *Proceedings of the 2009 FLAIRS Conference* (pp. 466-467). Association for the Advancement of Artificial Intelligence.

Blessing, S. B., Gilbert, S. B., Ourada, S., & Ritter, S. (2009). Authoring model-tracing cognitive tutors. *International Journal of Artificial Intelligence in Education*, *19*(2), 189-210.

Booth, J. L., Lange, K. E., Koedinger, K. R., & Newton, K. J. (2013). Using example problems to improve student learning in algebra: Differentiating between correct and incorrect examples. *Learning and Instruction, 25*, 24-34. doi:10.1016/j.learninstruc.2012.11.002

Borek, A., McLaren, B. M., Karabinos, M., & Yaron, D. (2009). How much assistance is helpful to students in discovery learning? In U. Cress, V. Dimitrova, & M. Specht (Eds.), *Proceedings 4th European Conference on Technology-Enhanced Learning, EC-TEL 2009* (pp. 391-404). Springer Berlin Heidelberg. doi:10.1007/978-3-642-04636-0_38

Chase, C., Marks, J., Bernett, D., Bradley, M., & Aleven, V. (2015). Towards the development of the Invention Coach: A naturalistic study of teacher guidance for an exploratory learning task. In C. Conati, N. Heffernan, A. Mitrovic, & M. F. Verdejo (Eds*.), Proceedings of the 17th International Conference on Artificial Intelligence in Education (AIED 2015)* (pp. 558-561). New York: Springer. doi:10.1007/978-3-319-19773-9_6

Chase, C., Marks, J., Bernett, D., & Aleven, V. (2015). The design of an exploratory learning environment to support Invention. Paper presened during the Workshop on Intelligent Support in Exploratory and Open-Ended Learning Environments, held as part of the 17th International Conference on Artificial Intelligence in Education (AIED 2015).

Clark, R. E., Feldon, D., van Merriënboer, J. J. G., Yates, K., & Early, S. (2008). Cognitive task analysis. In J. M. Spector, M. D. Merrill, J. J. G. van Merriënboer, & M. P. Driscoll (Eds.). *Handbook of research on educational communications and technology* (3rd ed.) (pp. 577-593). New York: Macmillan/Gale.

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.

Cook, R., Kay, J., & Kummerfeld, B. (2015). MOOClm: User modelling for moocs. In F. Ricci, K. Bontcheva, O. Conlan, & S. Lawless (Eds.), *Proceedings of the 23rd International Conference on User Modeling, Adaptation and Personalization, UMAP 2015* (pp. 80-91). Springer International Publishing. doi:10.1007/978-3-319-20267-9_7

Corbett, A., Kauffman, L., MacLaren, B., Wagner, A., & Jones, E. (2010). A Cognitive Tutor for genetics problem solving: Learning gains and student modeling. *Journal of Educational Computing Research*, *42*(2), 219-239.

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.

Devasani, S., Gilbert, S. B., & Blessing, S. B. (2012). Evaluation of two intelligent tutoring system authoring tool paradigms: Graphical user interface-based and text-based. In *Proceedings 21st Annual Conference on Behavior Representation in Modeling and Simulation 2012 (BRiMS 2012)* (pp. 51-58). *R*ed Hook, NY: Curran Associates, Inc.

Doroudi, S., Holstein, K., Aleven, V., & Brunskill, E. (2015). Towards understanding how to leverage sense-making, induction/refinement and fluency to improve robust learning. In O. C. Santos, J. G. Boticario, C. Romero, M. Pechenizkiy, A. Merceron, et al. (Eds.) *Proceedings of the 8th International Conference on Educational Data Mining, EDM 2015* (pp. 376-379). Worcester, MA: International Educational Data Mining Society.

Earnshaw, Y. (2014). Effects of levels of instructional assistance on learning and mental effort in an intelligent tutoring system: Proportional reasoning and middle school students. Doctoral dissertation. Available from ProQuest Dissertations and Theses database. (UMI No. 3637974)

Forlizzi, J., McLaren, B. M., Ganoe, C., McLaren, P. B., Kihumba, G., & Lister, K. (2014). Decimal point: Designing and developing an educational game to teach decimals to middle school students. In C. Busch (Ed.), *8th European Conference on Games-Based Learning: ECGBL2014* (pp. 128-135). Reading, UK: Academic Conferences and Publishing International.

Goguadze, G., Sosnovsky, S., Isotani, S., & McLaren, B.M. (2011). Evaluating a Bayesian student model of decimal misconceptions. In Pechenizkiy, M., Calders, T., Conati, C., Ventura, S., Romero , C., & Stamper, J. (Eds.), *Proceedings of the 4th International Conference on Educational Data Mining (EDM 2011)* (pp. 301-306). Worcester, MA: International Data Mining Society.

Guan, C. Q., Liu, Y., Chan, D. H. L., Ye, F., & Perfetti, C. A. (2011). Writing strengthens orthography and alphabetic-coding strengthens phonology in learning to read Chinese. Journal of Educational Psychology, 103(3), 509-522. doi:10.1037/a0023730

Harpstead, E., MacLellan, C. J., Aleven, V., & Myers, B. A. (2015). Replay analysis in open-ended educational games. In C. S. Loh, Y. Sheng, & D. Ifenthaler (Eds.), *Serious games analytics* (pp. 381-399). New York: Springer International Publishing. doi:10.1007/978-3-319-05834-4_17

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. *International Journal of Artificial Intelligence in Education*, *24*(4), 470-497. doi:10.1007/s40593-014-0024-x

Holmes, N. G., Day, J., Park, A. H., Bonn, D. A., & Roll, I. (2014). Making the failure more productive: scaffolding the invention process to improve inquiry behaviours and outcomes in productive failure activities. *Instructional Science, 42*(4), 523-538. doi:10.1007/s11251-013-9300-7.

Janssen, J., & Bodemer, D. (2013). Coordinated computer-supported collaborative learning: Awareness and awareness tools. *Educational Psychologist, 48*(1), 40-55.

Kalyuga, S. (2007). Expertise reversal effect and its implications for learner-tailored instruction. *Educational Psychology Review*, *19*(4), 509-539. doi:10.1007/s10648-007-9054-3

Kalyuga, S., Ayres, P., Chandler, P., & Sweller, J. (2003). The expertise reversal effect. *Educational Psychologist*, *38*(1), 23-31. doi:10.1207/S15326985EP3801_4

Kay, J., Reimann, P., Diebold, E., & Kummerfeld, B. (2013). MOOCs: So many learners, so much potential . *IEEE Intelligent Systems*, (3), 70-77. doi:10.1109/MIS.2013.66

Kleij, F. M. V. D., Feskens, R. C. W., & Eggen, T. J. H. M. (2015). Effects of feedback in a computer-based learning environment on students' learning outcomes. *Review of Educational Research.* doi:10.3102/0034654314564881

Kodaganallur, V., Weitz, R., & Rosenthal, D. (2005). A comparison of model-tracing and constraint-based intelligent tutoring paradigms. *International Journal of Artificial Intelligence in Education*, *15*(1), 117-144.

Kodavali, S. K., Gilbert, S., & Blessing, S. (2010). Expansion of the xpst framework to enable non-programmers to create intelligent tutoring systems in 3D game environments. In V. Aleven, J. Kay, & J. Mostow (Eds.), *Lecture Notes in Computer Science: Proceedings of the 10th International Conference on Intelligent Tutoring Systems, ITS 2010, vol. 2* (Vol. 6095, pp. 365-367). Berlin: Springer.

Koedinger, K. R. (2002). Toward evidence for instructional design principles: Examples from cognitive tutor math 6. Invited paper. In *Proceedings of PME-NA XXXIII (the North American Chapter of the International Group for the Psychology of Mathematics Education* (pp. 21-49).

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., Anderson, J. R., Hadley, W. H., & Mark, M. A. (1997). Intelligent tutoring goes to school in the big city. *International Journal of Artificial Intelligence in Education*, *8*(1), 30-43.

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* (pp. 43–55). Boca Raton, FL: CRC Press.

Koedinger, K. R., & Corbett, A. T. (2006). Cognitive Tutors: Technology bringing learning sciences to the classroom. In R. K. Sawyer (Ed.), *The Cambridge handbook of the learning sciences* (pp. 61-78). New York: Cambridge University 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., & Mitrovic, A. (2009). Authoring intelligent tutoring systems: Preface for special issue on authoring intelligent tutoring systems. *International Journal of Artificial Intelligence in Education*, *19*(2), 103-104.

Koedinger, K. R., Suthers, D. D., & Forbus, K. D. (1999). Component-based construction of a science learning space. *International Journal of Artificial Intelligence in Education*, *10*(3), 292-313.

Koedinger, K. R., & Terao, A. (2002). A cognitive task analysis of using pictures to support pre-algebraic reasoning. In W. Gray & C. D. Schunn (Eds.), *Proceedings of the twenty-fourth annual Conference of the Cognitive Science Society* (pp. 542-547). Taylor & Francis Group.

Kumar, R., & Kim, J. (2014). Preface to the special issue on intelligent support for learning in groups. *International Journal of Artificial Intelligence in Education*, *24*(1), 1-7. doi:10.1007/s40593-013-0013-5

Liew, C. W., &Xhakaj, F. (2015). Teaching a complex process: Insertion in red black trees. In C. Conati, N. Heffernan, A. Mitrovic, & M. F. Verdejo (Eds.), *Proceedings of the 17th International Conference on Artificial Intelligence in Education (AIED* 2015 (pp. 698-701). New York: Springer International Publishing. doi:10.1007/978-3-319-19773-9_95

Long, Y., & Aleven, V. (2014). Gamification of joint student/system control over problem selection in a linear equation tutor. In S. Trausan-Matu, K. E. Boyer, M. Crosby, & K. Panourgia (Eds.), *Proceedings of the 12th International Conference on Intelligent Tutoring Systems, ITS 2014* (pp. 378-387). New York: Springer. doi:10.1007/978-3-319-07221-0_47.

Long, Y., & Aleven, V. (2013a). Active learners: Redesigning an intelligent tutoring system to support self-regulated learning. In D. Hernández-Leo, T. Ley, R. Klamma, & A. Harrer (Eds.), *Scaling up Learning for Sustained Impact, Proceedings of the Eighth European Conference on Technology Enhanced Learning (EC-TEL 2013),* (pp. 490-495). Berlin, Heidelberg: Springer. doi: 10.1007/978-3-642-40814-4_44. Short Paper.

Long, Y., & Aleven, V. (2013b). Supporting students' self-regulated learning with an open learner model in a linear equation tutor. In H. C. Lane, K. Yacef, J. Mostow, & P. Pavlik (Eds.), *Proceedings of the 16th International Conference on Artificial Intelligence in Education (AIED 2013)* (pp. 249-258). Berlin: Springer.

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.

Liu, Y., Wang, M., Perfetti, C. A., Brubaker, B., Wu, S., & MacWhinney, B. (2011). Learning a tonal language by attending to the tone: An in vivo experiment. *Language Learning, 61*(4), 1119-1141. doi:10.1111/j.1467-9922.2011.00673.x

MacLellan, C., Koedinger, K. R., & Matsuda, N. (2014). Authoring tutors with SimStudent: An evaluation of efficiency and model quality. In S. Trausan-Matu, K. E. Boyer, M. Crosby, & K. Panourgia (Eds.), *Proceedings of the 12th International Conference on Intelligent Tutoring Systems, ITS 2014* (pp. 551-560). Berlin: Springer. doi:10.1007/978-3-319-07221-0_66

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

Matsuda, N., Cohen, W. W., & Koedinger, K. R. (2015). Teaching the teacher: Tutoring SimStudent leads to more effective Cognitive Tutor authoring. *International Journal of Artificial Intelligence in Education*, *25*(1), 1-34. doi:10.1007/s40593-014-0020-1

McKendree, J. (1990). Effective feedback content for tutoring complex skills. *Human-Computer. Interaction*, *5*(4), 381-413. doi:10.1207/s15327051hci0504_2

McLaren, B. M., Adams, D. M.,  & Mayer, R.E. (2015). Delayed learning effects with erroneous examples: A study of learning decimals with a web-based tutor. *International Journal of Artificial Intelligence in Education*, *25*(4), 520-542.

McLaren, B. M., Adams, D., Durkin, K., Goguadze, G., Mayer, R. E., Rittle-Johnson, B., . . . Velsen, M. V. (2012). To err is human, to explain and correct is divine: A study of interactive erroneous examples with middle school math students. In A. Ravenscroft, S. Lindstaedt, C. Delgado Kloos, & D. Hernández-Leo (Eds.), *21$^{st}$-Century Learning for 21$^{st}$-Century Skills: 7th European Conference on Technology-Enhanced Learning, EC-TEL 2012* (pp. 222-235). Berlin, Heidelberg: Springer. doi:10.1007/978-3-642-33263-0_18

McLaren, B. M., DeLeeuw, K. E., & Mayer, R. E. (2011a). A politeness effect in learning with web-based intelligent tutors. *International Journal of Human Computer Studies*, *69*(1-2), 70-79. doi:10.1016/j.ijhcs.2010.09.001

McLaren, B. M., DeLeeuw, K. E., & Mayer, R. E. (2011b). Polite web-based intelligent tutors: Can they improve learning in classrooms? *Computers & Education*, *56*(3), 574-584.

McLaren, B. M., Lim, S. J., & Koedinger, K. R. (2008). When and how often should worked examples be given to students? New results and a summary of the current state of research. In B. C. Love, K. McRae, & V. M. Sloutsky (Eds.), *Proceedings of the 30th Annual Meeting of the Cognitive Science Society* (pp. 2176-2181). Austin, TX: Cognitive Science Society.

McLaren, B. M., van Gog, T., Ganoe, C., Yaron, D., & Karabinos, M. (2014). Exploring the assistance dilemma: Comparing instructional support in examples and problems. In S. Trausan-Matu, K. E. Boyer, M. Crosby, & K. Panourgia (Eds.), *Proceedings of the 12th International Conference on Intelligent Tutoring Systems, ITS 2014* (pp. 354-361). Berlin: Springer. doi:10.1007/978-3-319-07221-0_66

McLaren, B. M., van Gog, T., Ganoe, C., Yaron, D., & Karabinos, M. (2015). Worked Examples are more efficient for learning than high-assistance instructional software. In C. Conati, N. Heffernan, A. Mitrovic, & M. F. Verdejo (Eds.), *Proceedings of the 17th International Conference on AI in Education, AIED 2015* (pp. 710-713). Berlin: Springer. doi: 10.1007/978-3-319-19773-9_98

McLaren, B.M., van Gog, T., Ganoe, C., Karabinos, M., & Yaron, D. (2016). The efficiency of worked examples compared to erroneous examples, tutored problem solving, and problem solving in classroom experiments. *Computers in Human Behavior*, *55,* 87-99.

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., & Ohlsson, S. (1999). Evaluation of a constraint-based tutor for a database language. *International Journal of Artificial Intelligence in Education, 10*(3-4), 238-256.

Murray, T. (1999). Authoring intelligent tutoring systems: An analysis of the state of the art. *International Journal of Artificial Intelligence in Education*, *10*(1), 98-129.

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.

Nathan, M. J. (1998). Knowledge and situational feedback in a learning environment for algebra story problem solving. *Interactive Learning Environments, 5*(1), 135-159.

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

Newell, A., & Simon, H. A. (1976). Computer science as empirical inquiry: Symbols and search. *Communications of the ACM, 19*(3), 113-126. doi:10.1145/360018.360022

Nye, B. D., Graesser, A. C., & Hu, X. (2014). AutoTutor and family: A review of 17 years of natural language tutoring. *International Journal of Artificial Intelligence in Education*, *24*(4), 427-469. doi:10.1007/s40593-014-0029-5

Ogan, A., Aleven, V., & Jones, C. (2009). Advancing development of intercultural competence through supporting predictions in narrative video. *International Journal of Artificial Intelligence in Education, 19*(3), 267-288.

Olsen, J. K., Belenky, D. M., Aleven, V., & Rummel, N. (2014). Using an intelligent tutoring system to support collaborative as well as individual learning. In S. Trausan-Matu, K. E. Boyer, M. Crosby, & K. Panourgia (Eds.), *Proceedings of the 12th International Conference on Intelligent Tutoring Systems, ITS 2014* (pp. 134-143). Berlin: Springer. doi:10.1007/978-3-319-07221-0_66

Olsen, J. K., Belenky, D. M., Aleven, V., Rummel, N., Sewall, J., & Ringenberg, M. (2014). Authoring tools for collaborative intelligent tutoring system environments. In S. Trausan-Matu, K. E. Boyer, M. Crosby, & K. Panourgia (Eds.), *Proceedings of the 12th International Conference on Intelligent Tutoring Systems, ITS 2014* (pp. 523-528). Berlin: Springer. doi:10.1007/978-3-319-07221-0_66

Olsen, J. K., Rummel, N., & Aleven, V. (under review). Investigating effects of embedding collaboration in an intelligent tutoring system for elementary school students.

Paas, F. G. W. C., & Van Merriënboer, J. J. G. (1994). Variability of worked examples and transfer of geometrical problem-solving skills: A cognitive-load approach. *Journal of Educational Psychology, 86*(1), 122-133.

Pane, J. F., Griffin, B. A., McCaffrey, D. F., & Karam, R. (2013). Effectiveness of Cognitive Tutor Algebra I at scale. *Educational Evaluation and Policy Analysis*, 0162373713507480. doi:10.3102/0162373713507480

Paquette, L., Lebeau, J. -F., & Mayers, A. (2010). Authoring problem-solving tutors: A comparison between ASTUS and CTAT. In R. Nkambou, J. Bourdeau, & R. Mizoguchi (Eds.), *Advances in intelligent tutoring systems* (pp. 377-405). Berlin, Heidelberg: Springer. doi:10.1007/978-3-642-14363-2_19

Paquette, L., Lebeau, J. -F., Beaulieu, G., & Mayers, A. (2015). Designing a knowledge representation approach for the generation of pedagogical interventions by MTTs. *International Journal of Artificial Intelligence in Education*, *25*(1), 118-156. doi:10.1007%2Fs40593-014-0030-z

Pavlik, P. I., Brawner, K., Olney, A., & Mitrovic, A. (2013). A review of student models used in intelligent tutoring systems. In R. Sottilare, A. Graesser, X. Hu, & H. Holden (Eds.), *Design recommendations for adaptive intelligent tutoring systems* (Vol. I, Learner Modeling, pp. 39-68). Orlando, FL: US Army Research Laboratory.

Peterson, J., Pardos, Z., Rau, M. A., Swigart, A., Gerber, C., & McKinsey, J. (2015). Understanding student success in chemistry using gaze tracking and pupillometry. In C. Conati, N. Heffernan, A. Mitrovic, & M. F. Verdejo (Eds.), *Proceedings of the 17th International Conference on AI in Education, AIED 2015* (pp. 358-366). New York: Springer International Publishing. doi:10.1007/978-3-319-19773-9_36

Rau, M. A. (2015). Why do the rich get richer? A structural equation model to test how spatial skills affect learning with representations. In O. C. Santos, J. G. Boticario, C. Romero, M. Pechenizkiy, A. Merceron, et al. (Eds.) *Proceedings of the 8th International Conference on Educational Data Mining, EDM 2015* (pp. 350-359). Worcester, MA: International Educational Data Mining Society.

Rau, M., Aleven, V., & Rummel, N. (2013). Interleaved practice in multi-dimensional learning tasks: Which dimension should we interleave? *Learning and Instruction*, *23*, 98-114. doi:learninstruc.2012.07.003

Rau, M. A., Aleven, V., & Rummel, N. (2015). Successful learning with multiple graphical representations and self-explanation prompts. *Journal of Educational Psychology, 107*(1), 30-46. doi:10.1037/a0037211

Rau, M. A., Aleven, V., Rummel, N., & Pardos, Z. (2014). How should intelligent tutoring systems sequence multiple graphical representations of fractions? A multi-methods study. *International Journal of Artificial Intelligence in Education*, *24*(1), 125-161. doi: 10.1007/s40593-013-0011-7

Rau, M. A., Aleven, V., Rummel, N., & Rohrbach, S. (2012). Sense making alone doesn't do it: Fluency matters too! ITS support for robust learning with multiple representations. In S. A. Cerri, W. J. Clancey, G. Papadourakis, & K. Panourgia (Eds.), *Proceedings of the 11th International Conference on Intelligent Tutoring Systems, ITS 2012* (pp. 174-184). Berlin: Springer.

Rau, M. A., Michaelis, J. E., & Fay, N. (2015). Connection making between multiple graphical representations: A multi-methods approach for domain-specific grounding of an intelligent tutoring system for chemistry. *Computers and Education, 82,* 460-485. doi: 10.1016/j.compedu.2014.12.009

Rau, M. A., & Wu, S. P. W. (2015). ITS support for conceptual and perceptual processes in learning with multiple graphical representations. In C. Conati, N. Heffernan, A. Mitrovic, & M. F. Verdejo (Eds.), *Proceedings of the 17th International Conference on AI in education, AIED 2015* (pp. 398-407). New York: Springer International Publishing. doi:10.1007/978-3-319-19773-9_40

Razzaq, L., Patvarczki, J., Almeida, S. F., Vartak, M., Feng, M., Heffernan, N. T., & Koedinger, K. R. (2009). The Assistment Builder: Supporting the life cycle of tutoring system content creation. *IEEE Transactions on Learning Technologies*, *2*(2), 157-166.

Renkl, A. (2013). Toward an instructionally oriented theory of example-based learning. *Cognitive Science*, *38*(1), 1-37. doi:10.1111/cogs.12086

Renkl, A., Atkinson, R. K., & Grosse, C. S. (2003). How fading worked solution steps works – A cognitive load perspective. *Instructional Science, 32,* 1-24

Rice, W. (2011). *Moodle 2.0 e-learning course development: A complete guide to successful learning using Moodle*. Birmingham, UK: Packt Publishing Ltd.

Ritter, S., & Koedinger, K. R. (1996). An architecture for plug-in tutor agents. *International Journal of Artificial Intelligence in Education*, *7*(3-4), 315-347.

Ritter, S., Anderson, J. R., Koedinger, K. R., & Corbett, A. (2007). Cognitive Tutor: Applied research in mathematics education. *Psychonomic Bulletin & Review*, *14*(2), 249-55.

Rittle-Johnson, B., & Koedinger, K. R. (2005). Designing knowledge scaffolds to support mathematical problem solving. *Cognition and Instruction*, *23*(3), 313-349.

Roll, I., Aleven, V., & Koedinger, K. R. (2010). The Invention Lab: Using a hybrid of model tracing and constraint-based modeling to offer intelligent support in inquiry environments. In V. Aleven, J. Kay, & J. Mostow (Eds.), *Lecture Notes in Computer Science: Proceedings of the 10th International Conference on Intelligent Tutoring Systems, ITS 2010,* (Vol. I, pp. 115-124). Berlin: Springer.

Roll, I., Aleven, V., McLaren, B. M., & Koedinger, K. R. (2011). Metacognitive practice makes perfect: Improving students' self-assessment skills with an intelligent tutoring system. In G. Biswas, S. Bull, J. Kay, & A. Mitrovic (Eds.), *Proceedings of the 15th international conference on artificial intelligence in education, AIED 2011* (pp. 288-295). Berlin, Heidelberg: Springer. doi:10.1007/978-3-642-21869-9_38

Rummel, N., Walker, E., & Aleven, V. (under review). Different futures of adaptive collaborative learning support. Manuscript submitted for publication.

Salden, R. J. C. M., Aleven, V., Schwonke, R., & Renkl, A. (2010). The expertise reversal effect and worked examples in tutored problem solving. *Instructional Science, 38*(3), 289-307. doi:10.1007/s11251-009-9107-8

Salden, R. J. C. M., Koedinger, K. R., Renkl, A., Aleven, V., & McLaren, B. M. (2010). Accounting for beneficial effects of worked examples in tutored problem solving. *Educational Psychology Review, 22*(4), 379-392. doi:10.1007/s10648-010-9143-6

Schwartz, D. L., Chase, C. C., Oppezzo, M. A., & Chin, D. B. (2011). Practicing versus inventing with contrasting cases: The effects of telling first on learning and transfer. *Journal of Educational Psychology*, *103*(4), 759-775. doi:10.1037/a0025140

Schwartz, D. L., & Martin, T. (2004). Inventing to prepare for future learning: The hidden efficiency of encouraging original student production in statistics instruction. *Cognition and Instruction, 22*(2), 129-184.

Slavin, R. E. (1996). Research on cooperative learning and achievement: What we know, what we need to know. *Contemporary Educational Psychology, 21*(1), 43-69.

Sottilare, R. (2012). Considerations in the development of an ontology for a generalized intelligent framework for tutoring. In *I3M defense and homeland security simulation Conference (DHSS 2012)*.

Sottilare, R., Graesser, A., Hu, X., & Holden, H. (Eds.) (2013). *Design recommendations for adaptive intelligent tutoring systems* (Vol. I, Learner Modeling). Orlando, FL: US Army Research Laboratory.

Sottilare, R., Graesser, A., Hu, X., & Holden, H. (Eds.) (2014). *Design recommendations for adaptive intelligent tutoring systems* (Vol. II - Instructional Management). Orlando, FL: US Army Research Laboratory.

Sottilare, R., Graesser, A., Hu, X., & Brawner, K. (Eds.) (2015). Design recommendations for adaptive intelligent tutoring systems (Vol. III - Authoring Tools and Expert Modeling Techniques). Orlando, FL: US Army Research Laboratory.

Stampfer, E., & Koedinger, K. R. (2013). When seeing isn't believing: Influences of prior conceptions and misconceptions. In M. Knauff, M. Pauen, N. Sebanz, & I. Wachsmuth (Eds.), *Proceedings of the 35th Annual Conference of the Cognitive Science Society* (pp. 916-919). Berlin, Heidelberg: Springer. doi:10.1007/978-3-642-39112-5_145

Stankov, S., Rosić, M., Žitko, B., & Grubišić, A. (2007). TEx-Sys model for building intelligent tutoring systems. *Computers & Education*, *51*(3), 1017–1036. doi:10.1016/j.compedu.2007.10.002

Van Gog, T., & Rummel, N. (2010). Example-based learning: Integrating cognitive and social-cognitive research perspectives. *Educational Psychology Review*, *22*(2), 155-174. doi:10.1007/s10648-010-9134-7

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. (this issue). Regulative loops, step loops and task loops. *International Journal of Artificial Intelligence in Education, 26*(1).

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

Walker, E., Rummel, N., & Koedinger, K. R. (2014). Adaptive intelligent support to improve peer tutoring in algebra. *International Journal of Artificial Intelligence in Education, 24*(1), 33-61. doi:10.1007/s40593-013-0001-9

Weiss, M. A. (2010). Data structures and problem solving using Java (4th Edition). New York, Pearson Education, Inc.

Wiese, E. S., & Koedinger, K.R. (2015). Grounded feedback in a fraction addition tutor. Paper presented as part of the Symposium Multiple Representations and Multimedia: Student Learning and Instruction at the 2015 Annual Meeting of the American Educational Research Association (AERA). Chicago, IL.

Woolf, B P. Building Intelligent Interactive Tutors: *Student-centered Strategies for Revolutionizing E-learning.* Burlington, MA: Morgan Kaufmann, 2009.

Wylie, R., Koedinger, K. R, & Mitamura, T. (2009). Is self-explanation always better? The effects of adding self-explanation prompts to an English grammar tutor. In N. A. Taatgen, & H. van Rijn (Eds.), *Proceedings of the 31st Annual Conference of the Cognitive Science Society* (pp. 1300-1305). Austin, TX: Cognitive Science Society.

Wylie, R., Koedinger, K. R, & Mitamura, T. (2010a). Analogies, explanations, and practice: Examining how task types affect second language grammar learning. In V. Aleven, J. Kay, & J. Mostow (Eds.), Lecture Notes in Computer Science: *Proceedings of the 10th International Conference on Intelligent Tutoring Systems, ITS 2010* (Vol. 6094, pp. 214-223). Berlin: Springer.  doi: 10.1007/978-3-642-13388-6_26

Wylie, R., Koedinger, K. R, & Mitamura, T. (2010b). Extending the self-explanation effect to second language grammar learning. In K. Gomez, L. Lyons, & J. Radinsky, J. (Eds.), *Learning in the Disciplines: Proceedings of the 9th International Conference of the Learning Sciences (ICLS 2010)* (Vol. 1, pp. 57-64). International Society of the Learning Sciences: Chicago IL.

Wylie, R., Sheng, M., Mitamura, T., & Koedinger, K. (2011). Effects of adaptive prompted self-explanation on robust learning of second language grammar. In G. Biswas, S. Bull, J. Kay, & A. Mitrovic (Eds.), *Lecture Notes in Computer Science: Artificial intelligence in Education* (Vol. 6738, pp. 588-590). Springer Berlin Heidelberg. doi:10.1007/978-3-642-21869-9_110

Xhakaj, F. (2015). Intelligent tutors and granularity a new approach to red black trees. Unpublished senior thesis, Department of Computer Science, Lafayette College, Easton Pennsylvania.USA.

Xhakaj, F., & Liew, C. W. (2015). A new approach to teaching red black tree. In V. Dagienė, C. Schulte, & T. Jevsikova (Eds.), *Proceedings of the 20th ACM Annual Conference on Innovation and Technology in Computer Science Education, ITiCSE '15* (pp. 278-283). New York: ACM. doi:10.1145/2729094.2742624